INSTRUCTIONS FOR USE

PLC Automation

Automation Builder, AC500

Automation Builder 2.4.0, AC500 V3
Table of contents

1 PLC Automation with V3 CPUs
   1.1 Getting started
      1.1.1 Introduction
      1.1.2 Cyber security in AC500 V3 products
         1.1.2.1 Defense in depth
         1.1.2.2 Secure operation
      1.1.3 Installing Automation Builder
      1.1.4 Licensing procedure
         1.1.4.1 Activating Automation Builder license
         1.1.4.2 Activation of Automation Builder licensing
         1.1.4.3 Online activation
         1.1.4.4 Offline activation
         1.1.4.5 Activating PLC application library license
         1.1.4.6 Installing an additional license
         1.1.4.7 Activating the stand-alone tool
      1.1.5 Set-up communication parameters in windows
      1.1.6 Information on the installed version
      1.1.7 PLC runtime and demo licensing
      1.1.8 Menues, views, windows
         1.1.8.1 Start page and menus
         1.1.8.2 'All Messages' window
      1.1.9 Device repository
      1.1.10 Creating and configuring projects
      1.1.11 Handling of AC500 projects
      1.1.12 Connection of devices
         1.1.12.1 Configuring devices
         1.1.12.2 Symbolic names for variables, inputs and outputs
         1.1.12.3 Update of AC500 devices
         1.1.12.4 Comparing objects
      1.1.13 Connection of serial interfaces
         1.1.13.1 Programming of applications
      1.1.14 I/O mapping
      1.1.15 AC500 PLC configuration
      1.1.16 Convert an AC500 V2 project to an AC500 V3 project
      1.1.17 Example projects
         1.1.17.1 Hardware AC500 V3
         1.1.17.2 Example project for central I/O expansion
            1.1.17.2.1 Preconditions
            1.1.17.2.2 Create, set-up and save your AC500 V3 project
            1.1.17.2.3 Configure the I/O module
            1.1.17.2.4 Programming and compiling
            1.1.17.2.5 Set-up the communication gateway
            1.1.17.2.6 AC500 V3 firmware installation and update
            1.1.17.2.7 Log-in to CPU and download the program
# Table of contents

1.1.17.2.8 Test the program................................................................. 62
1.1.17.2.9 Set-up visualization............................................................ 64
1.1.17.2.10 Create visualization......................................................... 68
1.1.17.2.11 Enable web visualization................................................ 75
1.1.17.2.12 Reset the CPU................................................................. 80
1.1.17.3 Example project for remote I/O expansion with PROFINET.............................................................................. 81
1.1.17.3.1 Preconditions.................................................................. 81
1.1.17.3.2 Set-up PROFINET controller........................................... 82
1.1.17.3.3 Set-up PROFINET device................................................ 84
1.1.17.3.4 Add remote I/O expansion to project......................... 88
1.1.17.3.5 Test the program.............................................................. 91
1.1.17.3.6 Reset the CPU................................................................. 92
1.2 Automation Builder installation manager.......................................... 94
1.2.1 Installing customer specific package..................................... 95
1.2.2 Adding or removing installed software packages................. 96
1.2.3 Checking for updates.............................................................. 97
1.2.4 Uninstalling Automation Builder.......................................... 98
1.3 Programming with IEC 61131-3 editor.............................................. 99
  1.3.1 CODESYS Development System............................................. 99
    1.3.1.1 Configuring CODESYS.................................................. 102
    1.3.1.1.1 Setting CODESYS options...................................... 102
    1.3.1.1.2 Customizing the user interface............................... 102
    1.3.1.2 Your first CODESYS program.................................... 108
    1.3.1.3 Creating and configuring a project................................. 124
    1.3.1.3.1 Creating standard projects.................................. 124
    1.3.1.3.2 Adding objects....................................................... 125
    1.3.1.3.3 Changing the compiler version............................... 126
    1.3.1.3.4 Opening a V3 project............................................ 127
    1.3.1.3.5 Opening a V2.3 project.......................................... 128
    1.3.1.3.6 Configuring a project............................................ 131
    1.3.1.4 Exporting and transferring a project............................. 135
    1.3.1.4.1 Exporting and importing projects.......................... 135
    1.3.1.4.2 Transferring projects............................................ 136
    1.3.1.5 Comparing projects....................................................... 137
    1.3.1.5.1 Creating a comparison view.................................. 137
    1.3.1.5.2 Opening the Detailed Compare View............... 138
    1.3.1.6 Protecting and saving a project..................................... 139
    1.3.1.6.1 Setting up write protection.................................. 142
    1.3.1.6.2 Assigning passwords............................................. 143
    1.3.1.6.3 Protecting projects using a dongle.......................... 144
    1.3.1.6.4 Setting up a user management................................ 145
    1.3.1.6.5 Protecting objects in the project by access rights..... 146
    1.3.1.6.6 Logging in via user account and password manager... 147
    1.3.1.6.7 Encrypting projects with certificates..................... 149
    1.3.1.6.8 Save project....................................................... 150
    1.3.1.6.9 Saving/Sending the project archive......................... 152
    1.3.1.6.10 Linking a project to the source control system..... 152
    1.3.1.7 Localizing projects...................................................... 153
    1.3.1.8 Configuring I/O links................................................... 155
      1.3.1.8.1 Device tree and device editor............................ 155
      1.3.1.8.2 Mapping a hardware structure in the device tree..... 159

2020/12/10 3ADR010583, 1, en_US 3
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.3.1.8.3</td>
<td>Configuring devices and I/O mapping</td>
<td>162</td>
</tr>
<tr>
<td>1.3.1.9</td>
<td>Programming of applications</td>
<td>171</td>
</tr>
<tr>
<td>1.3.1.9.1</td>
<td>Designating identifiers</td>
<td>171</td>
</tr>
<tr>
<td>1.3.1.9.2</td>
<td>Declaration of variables</td>
<td>172</td>
</tr>
<tr>
<td>1.3.1.9.3</td>
<td>Creating source code in IEC</td>
<td>182</td>
</tr>
<tr>
<td>1.3.1.9.4</td>
<td>Configuring the memory reserve for an online change</td>
<td>206</td>
</tr>
<tr>
<td>1.3.1.9.5</td>
<td>Function block — Calling functions or methods with external implementation</td>
<td>208</td>
</tr>
<tr>
<td>1.3.1.9.6</td>
<td>Using input assistance</td>
<td>208</td>
</tr>
<tr>
<td>1.3.1.9.7</td>
<td>Using pragmas</td>
<td>211</td>
</tr>
<tr>
<td>1.3.1.9.8</td>
<td>Using library POUs</td>
<td>213</td>
</tr>
<tr>
<td>1.3.1.9.9</td>
<td>Managing text in text lists</td>
<td>213</td>
</tr>
<tr>
<td>1.3.1.10</td>
<td>Using image pools</td>
<td>221</td>
</tr>
<tr>
<td>1.3.1.10.1</td>
<td>Integrating C Modules</td>
<td>222</td>
</tr>
<tr>
<td>1.3.1.10.12</td>
<td>Programmatic access to I/Os</td>
<td>226</td>
</tr>
<tr>
<td>1.3.1.10.13</td>
<td>Checking syntax and analyzing code</td>
<td>230</td>
</tr>
<tr>
<td>1.3.1.10.14</td>
<td>Orientation and navigation</td>
<td>232</td>
</tr>
<tr>
<td>1.3.1.10.15</td>
<td>Searching and replacing in the entire project</td>
<td>236</td>
</tr>
<tr>
<td>1.3.1.10.16</td>
<td>Refactoring</td>
<td>236</td>
</tr>
<tr>
<td>1.3.1.10.17</td>
<td>Task configuration</td>
<td>239</td>
</tr>
<tr>
<td>1.3.1.10.18</td>
<td>Encrypting an application</td>
<td>242</td>
</tr>
<tr>
<td>1.3.1.10.19</td>
<td>Unit conversion</td>
<td>246</td>
</tr>
<tr>
<td>1.3.1.10.20</td>
<td>Data persistence</td>
<td>249</td>
</tr>
<tr>
<td>1.3.1.10.21</td>
<td>Alarm management</td>
<td>257</td>
</tr>
<tr>
<td>1.3.1.10.22</td>
<td>Using POUs for implicit checks</td>
<td>262</td>
</tr>
<tr>
<td>1.3.1.10.23</td>
<td>Object-oriented programming</td>
<td>263</td>
</tr>
<tr>
<td>1.3.1.11</td>
<td>Working with control networks</td>
<td>270</td>
</tr>
<tr>
<td>1.3.1.11.1</td>
<td>Network and addressing</td>
<td>270</td>
</tr>
<tr>
<td>1.3.1.11.2</td>
<td>Symbol configuration</td>
<td>275</td>
</tr>
<tr>
<td>1.3.1.11.3</td>
<td>Network variables</td>
<td>277</td>
</tr>
<tr>
<td>1.3.1.11.4</td>
<td>Data source manager</td>
<td>280</td>
</tr>
<tr>
<td>1.3.1.11.5</td>
<td>Subordinate safety controller</td>
<td>294</td>
</tr>
<tr>
<td>1.3.1.11.6</td>
<td>Transferring an application to the PLC</td>
<td>295</td>
</tr>
<tr>
<td>1.3.1.11.11.1</td>
<td>Configuring the connection to the PLC</td>
<td>295</td>
</tr>
<tr>
<td>1.3.1.11.11.2</td>
<td>Encrypting communication, changing security settings</td>
<td>296</td>
</tr>
<tr>
<td>1.3.1.11.11.3</td>
<td>Handling of device user management</td>
<td>300</td>
</tr>
<tr>
<td>1.3.1.11.11.4</td>
<td>Generating application code</td>
<td>303</td>
</tr>
<tr>
<td>1.3.1.11.11.5</td>
<td>Downloading the application code, logging in, and starting the PLC</td>
<td>304</td>
</tr>
<tr>
<td>1.3.1.11.11.6</td>
<td>Generating boot applications</td>
<td>305</td>
</tr>
<tr>
<td>1.3.1.11.11.7</td>
<td>Downloading source code to and from the PLC</td>
<td>307</td>
</tr>
<tr>
<td>1.3.1.11.11.8</td>
<td>Distributing application code to different memory areas</td>
<td>308</td>
</tr>
<tr>
<td>1.3.1.12</td>
<td>Testing and debugging</td>
<td>309</td>
</tr>
<tr>
<td>1.3.1.12.1</td>
<td>Testing in simulation mode</td>
<td>309</td>
</tr>
<tr>
<td>1.3.1.12.2</td>
<td>Using breakpoints</td>
<td>310</td>
</tr>
<tr>
<td>1.3.1.12.3</td>
<td>Stepping through a program</td>
<td>313</td>
</tr>
<tr>
<td>1.3.1.12.4</td>
<td>Forcing and writing variables</td>
<td>315</td>
</tr>
<tr>
<td>1.3.1.12.5</td>
<td>Resetting applications</td>
<td>318</td>
</tr>
<tr>
<td>1.3.1.12.6</td>
<td>Flow control</td>
<td>319</td>
</tr>
<tr>
<td>1.3.1.12.7</td>
<td>Determining the current processing position with the call stack</td>
<td>322</td>
</tr>
<tr>
<td>1.3.1.12.8</td>
<td>Checking the task deployment</td>
<td>322</td>
</tr>
<tr>
<td>1.3.1.13</td>
<td>Application at runtime</td>
<td>322</td>
</tr>
<tr>
<td>1.3.1.13.1</td>
<td>Monitoring of values</td>
<td>323</td>
</tr>
<tr>
<td>Section</td>
<td>Title</td>
<td>Page</td>
</tr>
<tr>
<td>---------</td>
<td>-------</td>
<td>------</td>
</tr>
<tr>
<td>1.3.1.13.2</td>
<td>Changing values with recipes</td>
<td>331</td>
</tr>
<tr>
<td>1.3.1.13.3</td>
<td>Data recording with trace</td>
<td>335</td>
</tr>
<tr>
<td>1.3.1.13.4</td>
<td>Data recording with trend</td>
<td>344</td>
</tr>
<tr>
<td>1.3.1.13.5</td>
<td>Monitoring tasks</td>
<td>349</td>
</tr>
<tr>
<td>1.3.1.13.6</td>
<td>Reading the PLC log</td>
<td>349</td>
</tr>
<tr>
<td>1.3.1.13.7</td>
<td>Analysing errors with core dump</td>
<td>349</td>
</tr>
<tr>
<td>1.3.1.13.8</td>
<td>Using PLC shell for requesting information</td>
<td>351</td>
</tr>
<tr>
<td>1.3.1.13.9</td>
<td>PLC operation control via system variables</td>
<td>352</td>
</tr>
<tr>
<td>1.3.1.13.10</td>
<td>Backup and restore</td>
<td>353</td>
</tr>
<tr>
<td>1.3.1.14</td>
<td>Updating an application on the PLC</td>
<td>354</td>
</tr>
<tr>
<td>1.3.1.14.1</td>
<td>Executing the online change</td>
<td>355</td>
</tr>
<tr>
<td>1.3.1.14.2</td>
<td>Execution of a download</td>
<td>356</td>
</tr>
<tr>
<td>1.3.1.15</td>
<td>Copying files to/from PLC</td>
<td>357</td>
</tr>
<tr>
<td>1.3.1.16</td>
<td>Using device applications</td>
<td>358</td>
</tr>
<tr>
<td>1.3.1.16.1</td>
<td>Creating device applications</td>
<td>359</td>
</tr>
<tr>
<td>1.3.1.17</td>
<td>Using redundant PLC devices, CODESYS Redundancy</td>
<td>362</td>
</tr>
<tr>
<td>1.3.1.17.1</td>
<td>Configuring redundant PLC devices</td>
<td>363</td>
</tr>
<tr>
<td>1.3.1.17.2</td>
<td>Restrictions</td>
<td>365</td>
</tr>
<tr>
<td>1.3.1.18</td>
<td>Using the command-line interface</td>
<td>366</td>
</tr>
<tr>
<td>1.3.1.19</td>
<td>Using libraries</td>
<td>371</td>
</tr>
<tr>
<td>1.3.1.19.1</td>
<td>Information for library developers</td>
<td>372</td>
</tr>
<tr>
<td>1.3.1.19.2</td>
<td>Adding a library to the application</td>
<td>373</td>
</tr>
<tr>
<td>1.3.1.19.3</td>
<td>Adding a library to the repository</td>
<td>373</td>
</tr>
<tr>
<td>1.3.1.19.4</td>
<td>Exporting library files</td>
<td>374</td>
</tr>
<tr>
<td>1.3.1.20</td>
<td>Managing devices</td>
<td>374</td>
</tr>
<tr>
<td>1.3.1.20.1</td>
<td>Installing devices</td>
<td>374</td>
</tr>
<tr>
<td>1.3.1.21</td>
<td>Managing packages and licenses</td>
<td>375</td>
</tr>
<tr>
<td>1.3.1.21.1</td>
<td>Installing/Uninstalling a package</td>
<td>377</td>
</tr>
<tr>
<td>1.3.1.21.2</td>
<td>Licensing of products</td>
<td>377</td>
</tr>
<tr>
<td>1.3.1.22</td>
<td>Using scripts</td>
<td>388</td>
</tr>
<tr>
<td>1.3.1.22.1</td>
<td>Executing a script</td>
<td>390</td>
</tr>
<tr>
<td>1.3.1.22.2</td>
<td>Creating a python script</td>
<td>396</td>
</tr>
<tr>
<td>1.3.1.23</td>
<td>Security</td>
<td>421</td>
</tr>
<tr>
<td>1.3.1.23.1</td>
<td>General information</td>
<td>421</td>
</tr>
<tr>
<td>1.3.1.23.2</td>
<td>Security for the development system</td>
<td>423</td>
</tr>
<tr>
<td>1.3.1.23.3</td>
<td>Security for the runtime system / PLC</td>
<td>423</td>
</tr>
<tr>
<td>1.3.1.23.4</td>
<td>FAQ</td>
<td>424</td>
</tr>
<tr>
<td>1.3.1.24</td>
<td>Reference, Programming</td>
<td>428</td>
</tr>
<tr>
<td>1.3.1.24.1</td>
<td>Programming languages and editors</td>
<td>428</td>
</tr>
<tr>
<td>1.3.1.24.2</td>
<td>Variables</td>
<td>489</td>
</tr>
<tr>
<td>1.3.1.24.3</td>
<td>Operators</td>
<td>505</td>
</tr>
<tr>
<td>1.3.1.24.4</td>
<td>Operands</td>
<td>557</td>
</tr>
<tr>
<td>1.3.1.24.5</td>
<td>Data types</td>
<td>569</td>
</tr>
<tr>
<td>1.3.1.24.6</td>
<td>Pragmas</td>
<td>599</td>
</tr>
<tr>
<td>1.3.1.24.7</td>
<td>Identifiers</td>
<td>644</td>
</tr>
<tr>
<td>1.3.1.24.8</td>
<td>Shadowing rules</td>
<td>650</td>
</tr>
<tr>
<td>1.3.1.24.9</td>
<td>Keywords</td>
<td>652</td>
</tr>
<tr>
<td>1.3.1.24.10</td>
<td>Methods 'FB_Init', 'FB_Reinit', and 'FB_Exit'</td>
<td>653</td>
</tr>
<tr>
<td>1.3.1.24.11</td>
<td>Error messages</td>
<td>658</td>
</tr>
<tr>
<td>1.3.1.25</td>
<td>Reference, User Interface</td>
<td>710</td>
</tr>
<tr>
<td>1.3.1.25.1</td>
<td>Objects</td>
<td>710</td>
</tr>
</tbody>
</table>
Table of contents

1.3.1.25.2 Menu Commands................................................................. 852
1.3.1.25.3 Dialogs........................................................................ 1039
1.3.2 Fieldbus Support................................................................. 1099
  1.3.2.1 Device diagnostics............................................................ 1099
1.3.3 Runtime Systems, OPC UA Server......................................... 1100
  1.3.3.1 OPC UA server............................................................... 1100
1.3.4 Libraries.............................................................................. 1105
  1.3.4.1 Guidelines for creating libraries......................................... 1105
1.3.5 CODESYS Visualization....................................................... 1106
  1.3.5.1 Preparing CODESYS and projects..................................... 1108
  1.3.5.2 Limitation of the number of usable web pages on AC500 V3 PLCs... 1110
1.3.5.3 Designing a visualization with elements................................. 1111
   1.3.5.3.1 Selecting an element.................................................... 1112
   1.3.5.3.2 Positioning elements, adapting size and layer.................. 1113
   1.3.5.3.3 Assigning a color............................................................ 1115
   1.3.5.3.4 Using texts.................................................................. 1117
   1.3.5.3.5 How to display variable values in the visualization......... 1123
   1.3.5.3.6 How to change variable values via the visualization...... 1123
   1.3.5.3.7 Designing a background.............................................. 1124
   1.3.5.4 Configuring user inputs.................................................. 1124
   1.3.5.4.1 Configuring user inputs for visualization elements........ 1125
   1.3.5.4.2 Configuring gesture recognition.................................... 1126
   1.3.5.4.3 Configuring text input with the virtual keyboard........... 1128
   1.3.5.4.4 Configuring keyboard shortcuts..................................... 1131
   1.3.5.4.5 Capturing user input events.......................................... 1135
   1.3.5.5 Setting up user management............................................ 1139
   1.3.5.5.1 Setting up user management for visualizations.............. 1139
   1.3.5.5.2 Configuring users and groups........................................ 1140
   1.3.5.5.3 Editing and selecting user management dialogs.............. 1141
   1.3.5.5.4 Configuring permissions for groups............................ 1142
   1.3.5.6 Setting up multiple languages......................................... 1143
   1.3.5.7 Visualizing alarm management........................................ 1146
   1.3.5.8 Animating visualization elements...................................... 1150
   1.3.5.8.1 Configuring rotations and offsets................................ 1150
   1.3.5.8.2 Animating a text display............................................. 1152
   1.3.5.8.3 Animating a color display............................................. 1152
   1.3.5.9 Displaying data arrays in tables...................................... 1155
   1.3.5.9.1 Displaying structured variable values in tables.............. 1155
   1.3.5.9.2 Configuring and multiplying visualization elements as templates... 1156
   1.3.5.10 Displaying data curve with trace.................................... 1162
     1.3.5.10.1 Getting started with trace........................................ 1163
   1.3.5.11 Displaying data curve with trend..................................... 1165
     1.3.5.11.1 Getting started with trend visualization..................... 1165
     1.3.5.11.2 Programming a trend.............................................. 1168
   1.3.5.12 Displaying and editing text files..................................... 1171
     1.3.5.12.1 Configuring the display of a text file....................... 1171
     1.3.5.12.2 Configuring the editing of a text file....................... 1174
   1.3.5.13 Configuring a variable assignment with unit conversion......... 1176
   1.3.5.14 Using recipes in visualization elements............................ 1176
   1.3.5.15 Creating a structured user interface................................. 1177
     1.3.5.15.1 Displaying multiple visualizations in one visualization... 1178
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.4.2.6.2 CAA_CaIlBack.Library</td>
<td>2121</td>
</tr>
<tr>
<td>1.4.2.6.3 CAA_CiA405.library</td>
<td>2134</td>
</tr>
<tr>
<td>1.4.2.6.4 CAA_DTUtil.library</td>
<td>2159</td>
</tr>
<tr>
<td>1.4.2.6.5 CAA_File.library</td>
<td>2169</td>
</tr>
<tr>
<td>1.4.2.7 MQTT client library</td>
<td>2196</td>
</tr>
<tr>
<td>1.4.2.7.1 Function blocks</td>
<td>2196</td>
</tr>
<tr>
<td>1.4.2.7.2 Structures and enumerations</td>
<td>2216</td>
</tr>
<tr>
<td>1.4.2.7.3 Global variables</td>
<td>2220</td>
</tr>
<tr>
<td>1.5 PLC integration</td>
<td>2220</td>
</tr>
<tr>
<td>1.5.1 Product overview and comparison</td>
<td>2220</td>
</tr>
<tr>
<td>1.5.1.1 Comparison of AC500 V3 terminal bases</td>
<td>2220</td>
</tr>
<tr>
<td>1.5.1.2 Comparison of features and protocols</td>
<td>2223</td>
</tr>
<tr>
<td>1.5.1.3 Ethernet protocols and ports for AC500 V3 products</td>
<td>2224</td>
</tr>
<tr>
<td>1.5.1.3.1 Default open Ethernet ports of PM56xx-2ETH</td>
<td>2226</td>
</tr>
<tr>
<td>1.5.1.3.2 Overview of protocols, sockets and ports</td>
<td>2226</td>
</tr>
<tr>
<td>1.5.1.3.3 Limitation of connections per protocol</td>
<td>2228</td>
</tr>
<tr>
<td>1.5.1.3.4 Ethernet configuration</td>
<td>2229</td>
</tr>
<tr>
<td>1.5.1.3.5 Online access</td>
<td>2230</td>
</tr>
<tr>
<td>1.5.2 PLC introduction</td>
<td>2230</td>
</tr>
<tr>
<td>1.5.2.1 Safety instructions</td>
<td>2230</td>
</tr>
<tr>
<td>1.5.2.2 Cyber security in AC500 V3 products</td>
<td>2233</td>
</tr>
<tr>
<td>1.5.2.2.1 Defense in depth</td>
<td>2234</td>
</tr>
<tr>
<td>1.5.2.2.2 Secure operation</td>
<td>2235</td>
</tr>
<tr>
<td>1.5.2.3 License and third party information</td>
<td>2238</td>
</tr>
<tr>
<td>1.5.2.4 Regulations</td>
<td>2238</td>
</tr>
<tr>
<td>1.5.2.5 Standardization</td>
<td>2239</td>
</tr>
<tr>
<td>1.5.2.6 Terms and abbreviations</td>
<td>2240</td>
</tr>
<tr>
<td>1.5.2.7 Definitions: PLC system start-up</td>
<td>2242</td>
</tr>
<tr>
<td>1.5.2.8 Device lists</td>
<td>2244</td>
</tr>
<tr>
<td>1.5.2.8.1 Device list: Terminal bases</td>
<td>2244</td>
</tr>
<tr>
<td>1.5.2.8.2 Device list: Processor modules (CPUs)</td>
<td>2245</td>
</tr>
<tr>
<td>1.5.2.8.3 Device list: Communication modules</td>
<td>2246</td>
</tr>
<tr>
<td>1.5.2.8.4 Device list: Terminal units</td>
<td>2247</td>
</tr>
<tr>
<td>1.5.2.8.5 Device list: S500-eCo I/O modules</td>
<td>2249</td>
</tr>
<tr>
<td>1.5.2.8.6 Device list: S500 I/O modules</td>
<td>2250</td>
</tr>
<tr>
<td>1.5.2.8.7 Device list: Communication interface modules</td>
<td>2252</td>
</tr>
<tr>
<td>1.5.2.8.8 Device list: Accessories</td>
<td>2254</td>
</tr>
<tr>
<td>1.5.2.9 Your tasks - Documentation from the user's point of view</td>
<td>2254</td>
</tr>
<tr>
<td>1.5.2.9.1 Consulting engineers</td>
<td>2254</td>
</tr>
<tr>
<td>1.5.2.9.2 Sales and export people</td>
<td>2255</td>
</tr>
<tr>
<td>1.5.2.9.3 Switch-gear cabinet manufacturers, design engineers</td>
<td>2255</td>
</tr>
<tr>
<td>1.5.2.9.4 Commissioning people</td>
<td>2255</td>
</tr>
<tr>
<td>1.5.2.9.5 Programmers</td>
<td>2255</td>
</tr>
<tr>
<td>1.5.2.9.6 Operators, service, maintenance</td>
<td>2256</td>
</tr>
<tr>
<td>1.5.2.9.7 Education, training, plant training</td>
<td>2256</td>
</tr>
<tr>
<td>1.5.2.9.8 Call center, help line</td>
<td>2256</td>
</tr>
<tr>
<td>1.5.2.10 PLC system description</td>
<td>2256</td>
</tr>
<tr>
<td>1.5.2.10.1 AC500 product family</td>
<td>2256</td>
</tr>
<tr>
<td>1.5.2.10.2 AC500/S500 system structure</td>
<td>2257</td>
</tr>
<tr>
<td>1.5.2.10.3 AC500/S500: Short description hardware</td>
<td>2259</td>
</tr>
<tr>
<td>1.5.2.10.4 Short description software</td>
<td>2263</td>
</tr>
</tbody>
</table>
Table of contents

1.5.2.10.5 Control panels (HMI)........................................................................................................ 2264
1.5.2.11 AC500-S.................................................................................................................................. 2264
1.5.2.12 Convert an AC500 V2 project to an AC500 V3 project............................................................ 2264
1.5.3 Device specifications................................................................................................................. 2265
1.5.3.1 Status LEDs, display and control elements............................................................................ 2265
1.5.3.2 Terminal bases (AC500 standard)......................................................................................... 2265
1.5.3.2.1 TB56xx for AC500 V3 products......................................................................................... 2265
1.5.3.3 Processor modules............................................................................................................... 2274
1.5.3.3.1 AC500 (standard)........................................................................................................... 2274
1.5.3.4 Communication modules (AC500 standard)...................................................................... 2286
1.5.3.4.1 Overview....................................................................................................................... 2286
1.5.3.4.2 Compatibility of communication modules and communication interface modules..... 2289
1.5.3.4.3 CANopen...................................................................................................................... 2290
1.5.3.4.4 EtherCAT....................................................................................................................... 2297
1.5.3.4.5 PROFINET...................................................................................................................... 2301
1.5.3.5 Terminal units (AC500 standard).......................................................................................... 2307
1.5.3.5.1 TU507-ETH and TU508-ETH for Ethernet communication interface modules............ 2307
1.5.3.5.2 TU515, TU516, TU541 and TU542 for I/O modules......................................................... 2311
1.5.3.5.3 TU517 and TU518 for communication interface modules............................................. 2317
1.5.3.5.4 TU531 and TU532 for I/O modules................................................................................ 2320
1.5.3.6 I/O modules......................................................................................................................... 2327
1.5.3.6.1 Digital I/O modules....................................................................................................... 2327
1.5.3.6.2 Analog I/O modules...................................................................................................... 2543
1.5.3.6.3 Digital/Analog I/O modules........................................................................................... 2718
1.5.3.7 Communication interface modules (S500)......................................................................... 2785
1.5.3.7.1 Compatibility of communication modules and communication interface modules........ 2785
1.5.3.7.2 CANopen...................................................................................................................... 2786
1.5.3.7.3 EtherCAT....................................................................................................................... 2848
1.5.3.7.4 Modbus........................................................................................................................ 2898
1.5.3.7.5 PROFINET...................................................................................................................... 2962
1.5.3.8 Accessories......................................................................................................................... 3028
1.5.3.8.1 AC500 (standard)........................................................................................................... 3028
1.5.3.8.2 S500 (standard)............................................................................................................. 3035
1.5.4 System assembly, construction and connection....................................................................... 3039
1.5.4.1 Introduction......................................................................................................................... 3039
1.5.4.2 Regulations......................................................................................................................... 3040
1.5.4.3 Safety instructions............................................................................................................... 3040
1.5.4.4 Overall information (valid for complete AC500 product family)........................................ 3044
1.5.4.4.1 Serial I/O bus............................................................................................................... 3044
1.5.4.4.2 Mechanical encoding.................................................................................................... 3046
1.5.4.4.3 Earthing concept (Block diagrams)............................................................................... 3049
1.5.4.4.4 EMC-conforming assembly and construction.............................................................. 3051
1.5.4.4.5 Power consumption of an entire station........................................................................ 3054
1.5.4.4.6 Decommissioning.......................................................................................................... 3057
1.5.4.4.7 Recycling and disposal.................................................................................................. 3057
1.5.4.5 AC500 (Standard)............................................................................................................ 3058
1.5.4.5.1 System data AC500....................................................................................................... 3058
1.5.4.5.2 Mechanical dimensions............................................................................................... 3062
1.5.4.5.3 Mounting and demounting.......................................................................................... 3068
1.5.4.5.4 Connection and wiring................................................................................................. 3076
1.5.4.5.5 Handling of accessories............................................................................................... 3088
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.5.4.6</td>
<td>AC500-XC</td>
<td>3099</td>
</tr>
<tr>
<td>1.5.4.6.1</td>
<td>System data AC500-XC</td>
<td>3099</td>
</tr>
<tr>
<td>1.5.4.7</td>
<td>AC500-S</td>
<td>3103</td>
</tr>
<tr>
<td>1.5.5</td>
<td>System technology for AC500 V3 products</td>
<td>3104</td>
</tr>
<tr>
<td>1.5.5.1</td>
<td>System technology of CPU and overall system</td>
<td>3105</td>
</tr>
<tr>
<td>1.5.5.1.1</td>
<td>Handling of remanent variables for AC500 V3 products</td>
<td>3105</td>
</tr>
<tr>
<td>1.5.5.1.2</td>
<td>System processing</td>
<td>3110</td>
</tr>
<tr>
<td>1.5.5.1.3</td>
<td>Real-time clock and battery</td>
<td>3115</td>
</tr>
<tr>
<td>1.5.5.1.4</td>
<td>LEDs, display and function keys on the front panel</td>
<td>3118</td>
</tr>
<tr>
<td>1.5.5.1.5</td>
<td>Onboard technologies</td>
<td>3140</td>
</tr>
<tr>
<td>1.5.5.1.6</td>
<td>Hot swap</td>
<td>3148</td>
</tr>
<tr>
<td>1.5.5.1.7</td>
<td>KNX IP integration</td>
<td>3151</td>
</tr>
<tr>
<td>1.5.5.1.8</td>
<td>Communication with Modbus RTU</td>
<td>3166</td>
</tr>
<tr>
<td>1.5.5.1.9</td>
<td>Communication with Modbus TCP/IP</td>
<td>3182</td>
</tr>
<tr>
<td>1.5.5.1.10</td>
<td>Fast counters</td>
<td>3194</td>
</tr>
<tr>
<td>1.5.5.2</td>
<td>System technology of the AC500 communication modules</td>
<td>3197</td>
</tr>
<tr>
<td>1.5.5.2.1</td>
<td>CANopen communication modules</td>
<td>3197</td>
</tr>
<tr>
<td>1.5.5.3</td>
<td>System technology of the communication interface modules</td>
<td>3200</td>
</tr>
<tr>
<td>1.5.5.3.1</td>
<td>Modbus communication interface module</td>
<td>3200</td>
</tr>
<tr>
<td>1.5.5.3.2</td>
<td>PROFINET communication interface module</td>
<td>3226</td>
</tr>
<tr>
<td>1.5.6</td>
<td>Configuration in Automation Builder for AC500 V3 products</td>
<td>3228</td>
</tr>
<tr>
<td>1.5.6.1</td>
<td>General settings</td>
<td>3228</td>
</tr>
<tr>
<td>1.5.6.1.1</td>
<td>Project handling</td>
<td>3228</td>
</tr>
<tr>
<td>1.5.6.1.2</td>
<td>User and access rights management</td>
<td>3240</td>
</tr>
<tr>
<td>1.5.6.1.3</td>
<td>Later change-over of target system</td>
<td>3245</td>
</tr>
<tr>
<td>1.5.6.1.4</td>
<td>Firmware identification and update</td>
<td>3245</td>
</tr>
<tr>
<td>1.5.6.1.5</td>
<td>Migration of third party devices</td>
<td>3250</td>
</tr>
<tr>
<td>1.5.6.1.6</td>
<td>Advanced IO device handling</td>
<td>3252</td>
</tr>
<tr>
<td>1.5.6.2</td>
<td>PLC devices and components</td>
<td>3254</td>
</tr>
<tr>
<td>1.5.6.2.1</td>
<td>Device repository</td>
<td>3254</td>
</tr>
<tr>
<td>1.5.6.2.2</td>
<td>PLC start-up</td>
<td>3257</td>
</tr>
<tr>
<td>1.5.6.2.3</td>
<td>Processor modules</td>
<td>3274</td>
</tr>
<tr>
<td>1.5.6.2.4</td>
<td>Terminal bases</td>
<td>3278</td>
</tr>
<tr>
<td>1.5.6.2.5</td>
<td>Onboard Ethernet configuration</td>
<td>3278</td>
</tr>
<tr>
<td>1.5.6.2.6</td>
<td>Onboard CAN configuration</td>
<td>3285</td>
</tr>
<tr>
<td>1.5.6.2.7</td>
<td>Communication modules</td>
<td>3286</td>
</tr>
<tr>
<td>1.5.6.2.8</td>
<td>Communication interface modules</td>
<td>3301</td>
</tr>
<tr>
<td>1.5.6.2.9</td>
<td>I/O-bus and I/O modules</td>
<td>3306</td>
</tr>
<tr>
<td>1.5.6.2.10</td>
<td>Serial interface</td>
<td>3325</td>
</tr>
<tr>
<td>1.5.6.2.11</td>
<td>Gateway configuration</td>
<td>3331</td>
</tr>
<tr>
<td>1.5.6.2.12</td>
<td>CAN-based fieldbuses</td>
<td>3332</td>
</tr>
<tr>
<td>1.5.6.2.13</td>
<td>EtherCAT configurator</td>
<td>3347</td>
</tr>
<tr>
<td>1.5.6.2.14</td>
<td>PROFINET IO configurator</td>
<td>3363</td>
</tr>
<tr>
<td>1.5.6.3</td>
<td>Protocols and special servers</td>
<td>3384</td>
</tr>
<tr>
<td>1.5.6.3.1</td>
<td>IEC60870-5-104 (Telecontrol)</td>
<td>3384</td>
</tr>
<tr>
<td>1.5.6.3.2</td>
<td>Modbus protocol</td>
<td>3422</td>
</tr>
<tr>
<td>1.5.6.3.3</td>
<td>SNTP protocol</td>
<td>3424</td>
</tr>
<tr>
<td>1.5.6.3.4</td>
<td>FTP server</td>
<td>3428</td>
</tr>
<tr>
<td>1.5.6.3.5</td>
<td>MQTT client protocol</td>
<td>3429</td>
</tr>
<tr>
<td>1.5.6.3.6</td>
<td>AC500 V3 secure protocols</td>
<td>3431</td>
</tr>
<tr>
<td>1.5.6.3.7</td>
<td>KNX configurator</td>
<td>3436</td>
</tr>
<tr>
<td>Section</td>
<td>Page</td>
<td></td>
</tr>
<tr>
<td>------------------------------------------------------------------------</td>
<td>------</td>
<td></td>
</tr>
<tr>
<td>1.6.2.6 Communication module and fieldbus diagnosis</td>
<td>3560</td>
<td></td>
</tr>
<tr>
<td>1.6.2.6.1 Fieldbus commissioning</td>
<td>3560</td>
<td></td>
</tr>
<tr>
<td>1.6.2.6.2 CI52x Modbus diagnosis</td>
<td>3564</td>
<td></td>
</tr>
<tr>
<td>1.6.3 Diagnosis messages</td>
<td>3565</td>
<td></td>
</tr>
<tr>
<td>1.6.3.1 CPU diagnosis</td>
<td>3565</td>
<td></td>
</tr>
<tr>
<td>1.6.3.2 I/O bus diagnosis</td>
<td>3567</td>
<td></td>
</tr>
<tr>
<td>1.6.3.3 S500 I/O modules diagnosis</td>
<td>3569</td>
<td></td>
</tr>
<tr>
<td>1.6.3.4 Communication modules diagnosis</td>
<td>3578</td>
<td></td>
</tr>
<tr>
<td>1.6.3.4.1 AC500-S: errors from safety CPU and safety I/O modules</td>
<td>3578</td>
<td></td>
</tr>
<tr>
<td>1.6.3.4.2 CM579-PNIO – PROFINET I/O controller diagnosis</td>
<td>3582</td>
<td></td>
</tr>
<tr>
<td>1.7 Engineering interfaces and tools</td>
<td>3587</td>
<td></td>
</tr>
<tr>
<td>1.7.1 Export and import interfaces</td>
<td>3587</td>
<td></td>
</tr>
<tr>
<td>1.7.1.1 Exporting and importing ECAD data (PBF)</td>
<td>3587</td>
<td></td>
</tr>
<tr>
<td>1.7.1.1.1 Requirements on EPLAN electric P8</td>
<td>3588</td>
<td></td>
</tr>
<tr>
<td>1.7.1.1.2 Importing PLC data from the ECAD tool</td>
<td>3588</td>
<td></td>
</tr>
<tr>
<td>1.7.1.1.3 Importing third party devices</td>
<td>3589</td>
<td></td>
</tr>
<tr>
<td>1.7.1.1.4 Exporting PLC data to ECAD tool</td>
<td>3589</td>
<td></td>
</tr>
<tr>
<td>1.7.1.1.5 Exporting third party devices</td>
<td>3590</td>
<td></td>
</tr>
<tr>
<td>1.7.1.1.6 Importing ECAD PLC data to existing AB project</td>
<td>3590</td>
<td></td>
</tr>
<tr>
<td>1.7.1.1.7 Arrange or map devices imported to the device pool</td>
<td>3590</td>
<td></td>
</tr>
<tr>
<td>1.7.1.1.8 Limitations</td>
<td>3591</td>
<td></td>
</tr>
<tr>
<td>1.7.1.2 Exporting and importing I/O mapping (CSV)</td>
<td>3591</td>
<td></td>
</tr>
<tr>
<td>1.7.1.2.1 Exporting IO mapping data to CSV</td>
<td>3592</td>
<td></td>
</tr>
<tr>
<td>1.7.1.2.2 Importing I/O mapping data from CSV</td>
<td>3593</td>
<td></td>
</tr>
<tr>
<td>1.7.1.3 Exporting and importing device list (CSV)</td>
<td>3593</td>
<td></td>
</tr>
<tr>
<td>1.7.1.3.1 Exporting device list to CSV</td>
<td>3593</td>
<td></td>
</tr>
<tr>
<td>1.7.1.3.2 Creating CSV device list</td>
<td>3594</td>
<td></td>
</tr>
<tr>
<td>1.7.1.3.3 Importing a device list from CSV</td>
<td>3595</td>
<td></td>
</tr>
<tr>
<td>1.7.1.3.4 Renaming devices</td>
<td>3595</td>
<td></td>
</tr>
<tr>
<td>1.7.2 CODESYS Security Agent</td>
<td>3597</td>
<td></td>
</tr>
<tr>
<td>1.7.2.1 Integration in CODESYS Development System</td>
<td>3597</td>
<td></td>
</tr>
<tr>
<td>1.7.2.2 Encrypted communication with devices via controller certificates</td>
<td>3597</td>
<td></td>
</tr>
<tr>
<td>1.7.2.3 Encrypting boot applications with certificates</td>
<td>3598</td>
<td></td>
</tr>
<tr>
<td>1.7.2.4 Reference User Interface</td>
<td>3599</td>
<td></td>
</tr>
<tr>
<td>1.7.2.4.1 'View 'Security-Screen' - 'Devices'</td>
<td>3599</td>
<td></td>
</tr>
<tr>
<td>1.7.3 CODESYS Static Analysis</td>
<td>3601</td>
<td></td>
</tr>
<tr>
<td>1.7.3.1 Configuring and Running Static Analysis</td>
<td>3601</td>
<td></td>
</tr>
<tr>
<td>1.7.3.2 Reference, User Interface</td>
<td>3604</td>
<td></td>
</tr>
<tr>
<td>1.7.3.2.1 Menu Commands</td>
<td>3605</td>
<td></td>
</tr>
<tr>
<td>1.7.3.2.2 Dialogs</td>
<td>3606</td>
<td></td>
</tr>
<tr>
<td>1.7.3.3 Reference, Programming</td>
<td>3611</td>
<td></td>
</tr>
<tr>
<td>1.7.3.3.1 Pragmas and Attributes</td>
<td>3611</td>
<td></td>
</tr>
<tr>
<td>1.7.3.3.2 Rules</td>
<td>3616</td>
<td></td>
</tr>
<tr>
<td>1.7.4 CODESYS Profiler</td>
<td>3688</td>
<td></td>
</tr>
<tr>
<td>1.7.4.1 Overview</td>
<td>3688</td>
<td></td>
</tr>
<tr>
<td>1.7.4.2 Executing the Profiling Measurements</td>
<td>3689</td>
<td></td>
</tr>
<tr>
<td>1.7.4.2.1 Profiling by Code Instrumentation</td>
<td>3690</td>
<td></td>
</tr>
<tr>
<td>1.7.4.2.2 Using the Profiler Watch List</td>
<td>3692</td>
<td></td>
</tr>
<tr>
<td>1.7.4.2.3 Profiling by Sampling</td>
<td>3693</td>
<td></td>
</tr>
<tr>
<td>1.7.4.3 Measuring Code Coverage</td>
<td>3693</td>
<td></td>
</tr>
<tr>
<td>1.7.4.4 Creating Snapshots</td>
<td>3694</td>
<td></td>
</tr>
</tbody>
</table>
1.7.4.5  Reference, User Interface ................................................................. 3695
1.7.4.5.1  Objects ......................................................................................... 3695
1.7.4.5.2  Menu Commands ........................................................................ 3702
1.7.5  Drive composer pro integration ................................................................. 3706
1.7.6  CODESYS SVN .................................................................................. 3710
1.7.6.1  Getting Started ................................................................. 3711
1.7.6.2  Version control ........................................................................... 3711
1.7.6.3  Using an SVN Repository ............................................................. 3711
1.7.6.4  Using Working Copies .................................................................. 3713
1.7.6.5  Reference, User Interface ............................................................... 3714
1.7.6.5.1  Overlay Icons ............................................................................... 3714
1.7.6.5.2  Commands ................................................................................... 3716
1.7.6.5.3  Dialogs ......................................................................................... 3742
1.7.6.5.4  Objects ........................................................................................ 3749
1.7.7  Subversion ......................................................................................... 3750
1.7.7.1  Project Version Control with Subversion ........................................ 3750
1.7.7.1.1  Preconditions .............................................................................. 3751
1.7.7.1.2  Working with Project Version Control ........................................... 3751
1.7.7.3  Recommendations on Working with Project Version Control ........ 3752
1.7.7.1.4  Known Issues and Troubleshooting ............................................... 3752
1.7.7.2  SVN Support Examples ................................................................. 3753
1.7.7.2.1  Importing Automation Builder Project to SVN Repository ........... 3753
1.7.7.2.2  Logging in User2 ......................................................................... 3753
1.7.7.2.3  Examples ..................................................................................... 3754
1.7.8  Python .............................................................................................. 3755
1.8  Human machine interface ........................................................................ 3756
1.8.1  Panel Builder interface ...................................................................... 3756
1.8.1.1  Adding desired AC500 PLC to the project ........................................... 3756
1.8.1.2  Creating a Panel Builder project ..................................................... 3757
1.8.1.3  Configuring Panel Builder ............................................................. 3760
1.8.2  SCADA Integration ............................................................................ 3763
1.8.2.1  Creating Workspace and Project ................................................... 3763
1.8.2.2  Loading existing Workspace and Project ........................................ 3765
1.8.2.2.3  Checking the Gateway Settings in a Zenon Project ..................... 3765
1.8.2.4  Generating a Symbol File .............................................................. 3766
1.8.2.5  Updating Standard Data Types ..................................................... 3766
1.8.2.6  Creating Data Types ..................................................................... 3767
1.8.2.7  Importing Data Types in zenon Editor ........................................... 3767
1.9  Contact ABB ......................................................................................... 3767
1.9.1  ABB contact for PLC and panel builder products ............................... 3768
2  Index ....................................................................................................... 3769
1 PLC Automation with V3 CPUs
1.1 Getting started
1.1.1 Introduction

ABB Automation Builder is the integrated software suite for machine builders and system integrators wanting to automate their machines and systems in a productive way. Combining the tools required for configuring, programming, debugging and maintaining automation projects from a common intuitive interface, Automation Builder addresses the largest single cost element of most of today's industrial automation projects: software. ABB Automation Builder covers the engineering of ABB PLCs, Safety PLCs, control panels, drives, motion and robots.

Before starting Automation Builder configuration read the version specific information provided in the Automation Builder ReadMe file. It describes new features and functions as well as workarounds on known problems. The ReadMe file is stored in the installation directory of Automation Builder, however can be downloaded as well from ABB website http://new.abb.com/plc/automationbuilder.

1.1.1.1 Cyber security in Automation Builder

This product is designed to be connected to and to communicate information and data via a network interface. It is your sole responsibility to provide and continuously ensure a secure connection between the product and your network or any other network (as the case may be). You shall establish and maintain any appropriate measures (such as but not limited to the installation of firewalls, application of authentication measures, encryption of data, installation of anti-virus programs, etc.) to protect the product, the network, its system and the interface against any kind of security breaches, unauthorized access, interference, intrusion, leakage and/or theft of data or information. ABB Ltd and its affiliates are not liable for damages and/or losses related to such security breaches, any unauthorized access, interference, intrusion, leakage and/or theft of data or information.

Although ABB provides functionality testing on the products and updates that we release, you should institute your own testing program for any product updates or other major system updates (to include but not limited to code changes, configuration file changes, third party software updates or patches, hardware exchanges, etc.) to ensure that the security measures that you have implemented have not been compromised and system functionality in your environment is as expected. This also applies to the operating system. Security measures (such as but not limited to the installation of latest patches, installation of firewalls, application of authentication measures, installation of anti-virus programs, etc.) are in your responsibility. You have to be aware that operating systems provide a considerable number of open ports that should be monitored carefully for any threats.

It has to be considered that online connections to any devices are not secured. It is your responsibility to assure that connections are established to the correct device (and e.g. not to an unknown device pretending to be a known device type). Furthermore you have to take care that confidential data exchanged with the PLC is either compiled or encrypted.

1.1.1.2 Cyber security in AC500 V3 products

This product is designed to be connected to and to communicate information and data via a network interface. It is your sole responsibility to provide and continuously ensure a secure connection between the product and your network or any other network (as the case may be). You shall establish and maintain any appropriate measures (such as but not limited to the installation of firewalls, application of authentication measures, encryption of data, installation of anti-virus programs, etc.) to protect the product, the network, its system and the interface against any kind of security breaches, unauthorized access, interference, intrusion, leakage and/or theft of data or information. ABB Ltd and its affiliates are not liable for damages and/or losses related to such security breaches, any unauthorized access, interference, intrusion, leakage and/or theft of data or information.

Although ABB provides functionality testing on the products and updates that we release, you should institute your own testing program for any product updates or other major system updates (to include but not limited to code changes, configuration file changes, third party software updates or patches, hardware exchanges, etc.) to ensure that the security measures that you have implemented have not been compromised and system functionality in your environment is as expected. This also applies to the operating system. Security measures (such as but not limited to the installation of latest patches, installation of firewalls, application of authentication measures, installation of anti-virus programs, etc.) are in your responsibility. You have to be aware that operating systems provide a considerable number of open ports that should be monitored carefully for any threats.

It has to be considered that online connections to any devices are not secured. It is your responsibility to assure that connections are established to the correct device (and e.g. not to an unknown device pretending to be a known device type). Furthermore you have to take care that confidential data exchanged with the PLC is either compiled or encrypted.
programs, etc.) to protect the product, the network, its system and the interface against any kind of security breaches, unauthorized access, interference, intrusion, leakage and/or theft of data or information. ABB Ltd and its affiliates are not liable for damages and/or losses related to such security breaches, any unauthorized access, interference, intrusion, leakage and/or theft of data or information.

The firmware update files for the AC500 V3 PLC are digitally signed releases by ABB. During the update process, these signatures are validated by a hardware security component in the PLC. This way, the AC500 V3 PLC will only update with valid, authentic firmware, signed by ABB.

The AC500 V3 PLC comes with minimal services opened by default. Only the services needed for initial setup and programming are open before any user application is downloaded.

Details

Chapter 1.5.1.3 “Ethernet protocols and ports for AC500 V3 products” on page 2224.

The AC500 V3 PLC contains a secure shell service to access core logging data in case of problems which need a deeper analysis. This service is inactive by default, which means that no one can access this privileged shell in the normal operating state.

To activate this service, local access to the PLC is necessary and activation is only valid until the next power cycle of the PLC. Once activated, the service run on TCP port 22. Each PLC also protects the secure shell access by an individual password.

1.1.1.2.1 Defense in depth

The defense in depth approach implements multi-layer IT security measures. Each layer provides its special security measures. All deployed security mechanisms in the system must be updated regularly. It is also important to follow the system vendor’s recommendations on how to configure and use these mechanisms. As a basis, the components must include security functions such as:

- Virus protection
- Firewall protection
- Strong and regularly changed passwords
- User management
- Using VPN tunnels for connections between networks

Additional security components such as routers and switches with integrated firewalls should be available. A defined user and rights concept managing access to the controllers and their networks is mandatory. Finally, the manufacturer of the components should be able to quickly discover weaknesses and provide patches.

References: CODESYS Security Whitepaper

Security zones

IT resources vary in the extent to which they can be trusted. A common security architecture is therefore based on a layered approach that uses zones of trust to provide increasing levels of security according to increasing security needs. Less-trusted zones contain more-trusted zones and connections between the zones are only possible through secure interconnections such as firewalls. Fig. 1. All resources in the same zone must have the same minimum level of trust. The inner layers, where communication interaction needs to flow freely between nodes, must have the highest level of trust. This is the approach described in the IEC 62443 series of standards.
Firewalls, gateways, and proxies are used to control network traffic between zones of different security levels, and to filter out any undesirable or dangerous material. Traffic that is allowed to pass between zones should be limited to what is absolutely necessary because each type of service call or information exchange translates into a possible route that an intruder may be able to exploit. Different types of services represent different risks. Internet access, incoming e-mail and instant messaging, for example, represent very high risks.

![Cooperate network](image)

**Fig. 1: Security zones**

Fig. 1 shows three security zones, but the number of zones does not have to be as many or as few as three. The use of multiple zones allows access between zones of different trust levels to be controlled to protect a trusted resource from attack by a less trusted one.

High-security zones should be kept small and independent. They need to be physically protected, i.e. physical access to computers, network equipment and network cables must be limited by physical means to authorized persons only. A high-security zone should obviously not depend on resources in a less secure zone for its security. Therefore, it should form its own domain that is administered from the inside, and not depend on, e.g., a domain controller in a less secure network.

Even if a network zone is regarded as trusted, an attack is still possible: by a user or compromised resource that is inside the trusted zone, or by an outside user or resource that succeeds to penetrate the secure interconnection. Trust therefore depends also upon the types of measures taken to detect and prevent compromise of resources and violation of the security policy.

References: *Security for Industrial Automation and Control Systems*

### 1.1.2 Secure operation

The controller must be located in a protected environment in order to avoid accidental or intended access to the controller or the application.

A protected environment can be:

- Locked control cabinets without connection from outside
- No direct internet connection
- Use firewalls and VPN to separate different networks
- Separate different production areas with different access controls

To increase security, physical access protection measures such as fences, turnstiles, cameras or card readers can be added.
Follow these rules for the protected environment:

- Keep the trusted network as small as possible and independent from other networks.
- Protect the cross-communication of controllers and the communication between controllers and field devices via standard communication protocols (fieldbus systems) using appropriate measures.
- Protect such networks from unauthorized physical access.
- Use fieldbus systems only in protected environments. They are not protected by additional measures, such as encryption. Open physical or data access to fieldbus systems and their components is a serious security risk.
- Physically protect all equipment, i.e., ensure that physical access to computers, network equipment and cables, controllers, I/O systems, power supplies, etc., is limited to authorized persons.
- When connecting a trusted network zone to outer networks, make sure that all connections are through properly configured secure interconnections only, such as a firewall or a system of firewalls, which is configured for “deny by default”, i.e., blocks everything except traffic that is explicitly needed to fulfill operational requirements.
- Allow only authorized users to log on to the system, and enforce strong passwords that are changed regularly.
- Continuously maintain the definitions of authorized users, user groups, and access rights, to properly reflect the current authorities and responsibilities of all individuals at all times. Users should not have more privileges than they need to do their job.
- Do not use the system for e-mail, instant messaging, or internet browsing. Use separate computers and networks for these functions if they are needed.
- Do not allow installation of any unauthorized software in the system.
- Restrict temporary connection of portable computers, USB memory sticks and other removable data carriers. Computers that can be physically accessed by regular users should have ports for removable data carriers disabled.
- If portable computers need to be connected, e.g., for service or maintenance purposes, they should be carefully scanned for viruses immediately before connection.
- All CDs, DVDs, USB memory sticks and other removable data carriers, and files with software or software updates, should also be checked for viruses before being introduced into the trusted zone.
- Continuously monitor the system for intrusion attempts.
- Define and maintain plans for incident response, including how to recover from potential disasters.
- Regularly review the organization as well as technical systems and installations with respect to compliance with security policies, procedures and practices.

A protected local control cabinet could look like in picture 2, page 17. This network is not connected to any external network. Security is primarily a matter of physically protecting the automation system and preventing unauthorized users from accessing the system and from connecting or installing unauthorized hardware and software.

![Diagram of Trusted Zone and Isolated Automation Network]

**Fig. 2: Isolated automation system**
Servers and workplaces that are not directly involved in the control and supervision of the process should preferably be connected to a subnet that is separated from the automation system network by means of a router/firewall. This makes it possible to better control the network load and to limit access to certain servers on the automation system network. Note that servers and workplaces on this subnet are part of the trusted zone and thus need to be subject to the same security precautions as the nodes on the automation system network.

Fig. 3: Plant information network connected to an automation system

For the purposes of process control security, a general-purpose information system (IS) network should not be considered a trusted network, not the least since such networks are normally further connected to the Internet or other external networks. The IS network is therefore a different lower-security zone, and it should be separated from the automation system by means of a firewall. The IS and automation system networks should form separate domains.
1.1.2 Installing Automation Builder

**Preconditions**  You must have administrator rights on your PC to install Automation Builder.

**Installation**

1. Go to [abb.com/automationbuilder](http://abb.com/automationbuilder) to access the homepage of Automation Builder.
2. In the “Downloads” section, select “Download Automation Builder”.
3. In the “Latest Automation Builder” section, select “Automation Builder x.x. Download” (x.x = latest version). This downloads the installer on your PC.
4. Open the downloaded installer and follow the instructions of the installation manager.
5. Keep the default type of installation to “Premium Edition”.
6. Select software packages to be installed:
   Enable the check box “PLC - AC500 V3” to activate installation of all options for AC500 V3.
7. Click “Download and install” and follow the instructions of the setup.

1.1.3 Licensing procedure

1.1.3.1 Activating Automation Builder license

1. Automation Builder software must be installed successfully.
2. Start Automation Builder.
   A licensing wizard starts and guides you through the licensing procedure.
3. Enter user information and select [OK].
   In case of future support requests, your registration details enable ABB support team to handle your questions quickly.
4. If you purchase a license that is locked to a flash drive, plug in the device and select [Next].
**Fig. 5: Activation mode**

- **Activation key**: Enable this option and enter the key. For further information on how to receive this key, contact ABB sales team.
- **Trial license**: You can test Automation Builder software with a trial version in Premium Edition. In this case no activation key is required.
- **Basic license**: You can install Automation Builder software in Basic Edition. In this case no activation key is required.
- **Offline activation**: This is required in case of unavailability of internet on the PC. See Chapter 1.1.3.4 “Offline activation” on page 23.

4. Choose an activation mode and select [Next].
5. Select the license lock mode according to your purchased license and select [Next].
   - If you hold a single PC license, choose PC locked option to license and lock Automation Builder software to the current PC.
   - If you hold a license for portable Automation Builder software, plug a USB drive, enable Portable option and choose your USB drive from the drop-down list. With this, a portable Automation Builder license is created, which allows to operate Automation Builder software on any PC.

6. Select the activation mode and select [Next].
   - Select “Online Activation” in Chapter 1.1.3.3 “Online activation” on page 23 if the PC contains internet access or select “Offline Activation” in Chapter 1.1.3.4 “Offline activation” on page 23.
1.1.3.2 Automation Builder licensing

Automation Builder requires a valid license. Depending on the required functionality, Automation Builder can be licensed in Basic, Standard or Premium edition.

For license validation, use CodeMeter licensing server from WIBUS system. Generally, licensing mechanism is automatic without further user action. However, you can set up and use your own licensing server. For further information on setting up this server, see the manufacturer documentation (http://www.wibu.com/en/manuals-guides.html).

Part of Automation Builder licensing is registration and activation (usually with an activation key).

For testing purpose, Automation Builder can be installed as a trial version (Premium Edition). The trial version expires after 30 days and then the Automation Builder requires an appropriate license.

1.1.3.3 Online activation

The activation data can be sent to ABB License server if the current PC contains internet access.

1. Enable “Online Activation” and select [Next].
   - Activation procedure starts. A successfully ended licensing procedure ends with a success message.
2. Select [OK] to end the wizard.
   - Automation Builder license is activated and starts.

1.1.3.4 Offline activation

The activation data cannot be sent to the licensing server if the PC does not have internet access. In this case, the offline activation file is created. This file can be used to transfer activation data to the licensing server from another PC with internet access.

1. Enable “Offline Activation” and select [Next].
2. From the file system choose a shareable location (e.g., a shared folder or a USB drive).
   ● Enter a file name and select [Save]. An activation file is created and stored to the selected directory.
   ● Do not close (Cancel) or proceed (Next) the wizard. The license activation is made from a PC with internet availability.
3. Upload the activation file to the licensing website (http://lc.codemeter.com/32838/depot).
   - Follow the instructions.
   - Default language can be changed with the drop-down menu on the right side of the screen.
     ⇒ You will receive an activation response file.

4. Store the file to a shareable location (e.g. a shared folder or a USB drive).

5. Return to your original PC.

6. Select [Browse] and select the activation response file from the shareable location.
   ⇒ Due to security reasons an activation receipt file is created.

7. Enter a file name and store the activation receipt file to a shareable location (e.g. a shared folder or a USB drive).

8. Move to a PC with internet availability and upload the activation receipt file from the shareable location to the registration website (http://lc.codemeter.com/32838/depot).


### 1.1.3.5 Activating PLC application library license

The PLC application library can be activated using Automation Builder License Manager in online and offline modes.

- “Automation Builder License Manager” is installed.

1. Start “Automation Builder License Manager”.
   - Go to “Start ➔ All Programs ➔ ABB ➔ Automation Builder ➔ Automation Builder License Manager”.

---

**Fig. 8: Licensing website**

- Upload activation request
- Download activation response
- Upload activation receipt

Activating your licenses offline - First step “Upload activation request”:
1. Create an activation request file with the Automation Builder Activation Wizard.
2. Pick the created activation request file.
3. Click “Upload request and continue”.

Pick activation request file (*.xml)

Choose File  No file chosen

Upload request and continue

---

PLC Automation with V3 CPUs
Getting started > Licensing procedure
2. Choose an activation mode and click “Next”.
   - **Activation key**: For information on how to receive an activation key, contact the ABB sales team.
   - **Offline activation**: This is required in case of unavailability of internet access.

3. Follow step 5 and 6 in Chapter 1.1.3.1 “Activating Automation Builder license” on page 20 to proceed further.

For an online or offline activation of a PLC application library license, see Chapter 1.1.3.3 “Online activation” on page 23 and Chapter 1.1.3.4 “Offline activation” on page 23.

### 1.1.3.6 Installing an additional license

The additional feature or software which need additional license other than Automation Builder license can be activated in an easy way. You can activate additional license using online and offline modes.

1. In Automation Builder, go to “Tools ⇒ Install additional license”.
2. Choose activation mode and select [Next].
   - **Online activation**: Activate the license in online mode using activation key Chapter 1.1.3.3 “Online activation” on page 23.
   - **Offline activation**: Activate the license in offline mode using activation file Chapter 1.1.3.4 “Offline activation” on page 23.

   For information on how to receive online activation key and offline activation file, contact ABB sales team.

3. After installing additional license, restart Automation Builder to enable the new feature or software.

### 1.1.3.7 Activating the stand-alone tool

Automation Builder License Manager is a stand-alone tool which is used for registration and license activation.

In case, if the user need to install stand-alone tool using **Install Additional Tools** without Automation Builder installation, the stand-alone tool must be activated using Automation Builder license key. The user can launch the tool manually from start menu and do the registration and license activation same as Automation Builder license activation. For further information, see section Chapter 1.1.3.1 “Activating Automation Builder license” on page 20.

### 1.1.4 Set-up communication parameters in windows

To set-up the communication between the PC and the PLC, e.g., for downloading the compiled program, you have to set-up the communication parameters.

The IP address of your PC must be in the same class as the IP address of the CPU.

The factory setting of the IP address of the CPU is 192.168.0.10.

The IP address of your PC should be 192.168.0.X. Avoid X = 10 in order to prevent an IP conflict with the CPU.

Subnet mask should be 255.255.255.0.
Change the IP address

1. Open Windows Control Panel. Click “Network and Internet ➔ Network and Sharing Center”.

2. Click Change adapter settings.

   If using existing network with several devices, please pay attention on given network rules or contact your system administrator.

3. Right-click Local Area Connection (Ethernet) and select Properties.

5. Enter your desired IP address and subnet mask.

1.1.5 Information on the installed version

Further information on the installed Automation Builder version such as installed packages or license terms can be found on the "About" page (help menu) and in startupsettings. Safety Version is visible if safety option is installed. Safety Version Information shows the versions of all safety components.
It is possible to either continue working with a project on an older Automation Builder version or to update a project to the latest Automation Builder version (see also startupsettings).

### 1.1.6 PLC runtime and demo licensing

The use of some libraries and devices require the PLC to have a runtime license. Further it is possible to try out device features or library features by using a demo license \ Chapter 1.5.6.2.2.2 “PLC runtime licensing” on page 3258. See PLC Runtime Licensing.

### 1.1.7 Create log files for support

Professional support requires some information about the project and the devices. To collect this information proceed as follows:
1. Click “Help ➔ About” in the main menu of Automation Builder.

2. Click [Create package for support] and wait until a list of log files is displayed.

3. Click [Save package] to store the zipped log files to your disk, or click [Send package] to send the zipped log files to ABB support.

4. Click [OK].

1.1.8 Menues, views, windows

Ensure the full display of Automation Builder editors by choosing the option Smaller - 100 % (default) in “Start ➔ Control Panel ➔ Appearance and Personalization ➔ Display”. 
1.1.8.1 Start page and menus

After start-up of Automation Builder software the start page is displayed.

All items of the Automation Builder user interface are described in the CODESYS documentation:

- **Commands** ➧ Chapter 1.3.1.25.2 “Menu Commands” on page 852
- **Objects** ➧ Chapter 1.3.1.25.1 “Objects” on page 710
- **Dialog Boxes** ➧ Chapter 1.3.1.25.3 “Dialogs” on page 1039

1.1.8.2 'All Messages' window

Errors, warning and success messages are written to the “All messages” window:
1.1.9 Device repository

The Device Repository of Automation Builder manages the pool of devices that can be used in the PLC configuration.

You install or uninstall devices in the “Device Repository” dialog box. The system installs a device by reading the device description files, which define the device properties for configurability, programmability, and possible connections to other devices.

You can use the devices provided in the device repository by adding them to the device tree of your project.
Dialog device repository

1. Click "Tools ➔ Device Repository".
   ➔ The “Device Repository” dialog box opens.

   ![Device Repository dialog box]

   - [Edit Locations]: Changes the default repository location. The devices can be managed at different locations.
   - [Install] / [Uninstall]: Installs or uninstalls devices.
   - [Renew device repository]: Updates the device list, e.g. after uninstallation of a device.
   - [Details]: Provides technical details on the selected device.

2. Select the install location. “System Repository” is set by default.

Installing devices

The device repository cannot be changed manually, e.g. by copying or deleting files. Use always the Device Repository dialog to add or remove devices.
1. Click [Install] and select the appropriate file format.
   ⇒ The “Install Device Description” dialog box opens.

2. Select the file path of the device description.
3. Select the file type filter of the required device description.
   ⇒ All device descriptions of the selected file type are listed.
4. Select the required device description and click “Open”.
   ⇒ Automation Builder adds the device description to the matching category of your device repository.

   If errors occur during installation (for example, missing files that are referenced by the device description), then Automation Builder displays them in the lower part of the device repository dialog box.

   During the installation the device description files and all additional files referenced by that description will be copied to an internal location. Altering the original files will have no further effects to an internal location.

   The changes take only effect after reinstalling the corresponding device(s). The version number shown in the information section of the device should be verified.

Uninstalling devices
Select the device you want to remove and click [Uninstall].
The device is removed from the list.

Uninstalled devices which are used in existing projects are indicated by the symbol 🝦. The device will not be configured properly.
1.1.10 Creating and configuring projects

What is a project?

- A project contains the objects that are necessary for the creation of a control program ('application'):
  - Pure POUs, for example programs, function blocks, functions, GVLs.
  - Objects that are additionally required in order to be able to execute the application on a PLC. For example, task configuration, library manager, symbol configuration, device configuration, visualizations, external files.
- You can program several applications and connect several control devices in a project.
- CODESYS manages device-specific and application-specific POUs in the "Devices" view ('Device tree'), project-wide available POUs in the "POUs" view.
- For the creation of projects there are templates that already contain certain objects.
- Basic configurations and information for the project are defined in the "Project Settings" and the "Project Information". For example:
  - Compiler settings
  - User management
  - Author
  - Data for the project file.

There are settings for the version compatibility of the project in the configuration dialog boxes for the "Project Environment".

- You save a project as a file in the file system. Optionally you can pack it together with project-relevant files and information in a project archive. Saving in a source control system such as SVN is also possible
- Each project contains the information regarding the CODESYS version with which it was created. If you open it in another version, CODESYS will draw your attention to possible or necessary updates with regard to file format, library versions, etc.
- You can compare, export or import projects and create documentation for them.
- You can protect a project against change and completely, i.e. also against reading. By using a user management system you can purposefully control access to the project and even to individual objects in the project.

Handling of AC500 projects such as project creation, export/import, comparison of projects etc. is described in the sections for AC500 V3 products Chapter 1.5.6.1.1 "Project handling" on page 3228

1.1.11 Handling of AC500 projects

Handling of AC500 projects such as project creation, export/import, comparison of projects etc. is described in the sections for AC500 V3 products Chapter 1.5.6.1.1 "Project handling" on page 3228

Copy-and-paste from one project to another project in two different Automation Builder instances is possible. After copying parts of a project to a higher Automation Builder version the copied components have to be updated.

It is not possible to downgrade a project to an earlier Automation Builderversion.

- Import of export files is only allowed in the same profile version.
- Copy-and-paste of configurations must not be used to copy objects to an earlier version.
Automation Builder performs an integrity check for the PLC configuration before generating the configuration.

Project archive
Automation Builder supports the creation and the import of project archive files. Archive files contain all relevant project data including the PLC configuration, the CODESYS project files and all device descriptions. This allows exchanging Automation Builder projects without taking care of the target environment General Settings % Chapter 1.5.6.1 “General settings” on page 3228.

User and access rights of a project
The ‘User Management’ provides functions for defining user accounts and configure the access rights within a project. The rights to access project objects via specified actions are assigned only to user groups, not to a single user account. So each user must be member of a group General Settings % Chapter 1.5.6.1 “General settings” on page 3228.

1.1.12 Connection of devices
1.1.12.1 Configuring devices
Modify your Automation Builder project by adding device objects. Preset items can be replaced in the same way.

1. In the device tree, right-click an item node. Select “Add object”.

2. Select the desired object and click [Add object].
3. Double-click the new object in the device tree to configure the device settings. Depending on the selected item different configuration tabs are available.

1.1.12.2 Symbolic names for variables, inputs and outputs

The IEC naming rules are not checked during input in Automation Builder.

Input and output mapping

Devices with I/Os provide an I/O Mapping tab in their configuration editor where the available I/O channels can directly be mapped to a global variable.

The corresponding variable declarations are automatically available in the project.

All available I/O channels can easily be assigned to a variable.

<table>
<thead>
<tr>
<th>Variable</th>
<th>Mapping</th>
<th>Channel</th>
<th>Address</th>
<th>Type</th>
<th>Default Value</th>
<th>Unit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>b_Input_180</td>
<td></td>
<td>Digital inputs 10 - 17</td>
<td>%I0.0</td>
<td>BYTE</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>x_Input_180_I00</td>
<td></td>
<td>Digital input 10</td>
<td>%I0.0</td>
<td>BOOL</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>x_Input_180_I01</td>
<td></td>
<td>Digital input 11</td>
<td>%I0.1</td>
<td>BOOL</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>x_Input_180_I02</td>
<td></td>
<td>Digital input 12</td>
<td>%I0.2</td>
<td>BOOL</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>x_Input_180_I03</td>
<td></td>
<td>Digital input 13</td>
<td>%I0.3</td>
<td>BOOL</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>x_Input_180_I04</td>
<td></td>
<td>Digital input 14</td>
<td>%I0.4</td>
<td>BOOL</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>x_Input_180_I05</td>
<td></td>
<td>Digital input 15</td>
<td>%I0.5</td>
<td>BOOL</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>x_Input_180_I06</td>
<td></td>
<td>Digital input 16</td>
<td>%I0.6</td>
<td>BOOL</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>x_Input_180_I07</td>
<td></td>
<td>Digital input 17</td>
<td>%I0.7</td>
<td>BOOL</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Digital inputs 10 - 115</td>
<td>%I1</td>
<td>BYTE</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Digital inputs 116 - 123</td>
<td>%I2</td>
<td>BYTE</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Digital inputs 124 - 131</td>
<td>%I3</td>
<td>BYTE</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

AC500 uses Intel Byte Order (Little Endian).

Only entries with a data type set in column "Type" can be mapped. These entries can be expanded to show the available I/O channels.

If the project has been imported from a previous Automation Builder version, all variables should be checked to avoid inconsistencies concerning the I/O mapping.
1.1.12.3 Update of AC500 devices

Perform a **firmware update** to update AC500 V3 devices. Chapter 1.1.17.2.6 “AC500 V3 firmware installation and update” on page 59.

1.1.12.4 Comparing objects

To compare similar objects within a project (such as the project configuration) select both objects. Right-click and select **Compare Objects** to see the differences.

![Image of compare objects](image)

1.1.13 Connection of serial interfaces

Depending on the device type, the configuration of serial interfaces is different.

AC500 V3 Products: **Configuration of serial interfaces** Chapter 1.5.6.2 “PLC devices and components” on page 3254

1.1.13.1 Programming of applications

For creating an application program which is executable on the PLC, you fill POUs with declarations and implementation code (source code), you define the mapping of the hardware I/Os on application variables, and you configure the task handling. After having checked and debugged your implementation, the CODESYS compiler will generate the application code which can be downloaded to the PLC.

The programming of the application POUs will be supported by the programming language editors and further features like for example textlists, image pools, alarm configuration, pragmas, refactoring and usage of ready modules provided by CODESYS Development System or by libraries.
1.1.14 I/O mapping

For all connected I/O devices perform an I/O Mapping.

AC500 V3 Products § Chapter 1.5.6.2.9.7 “I/O mapping list” on page 3309

1.1.15 AC500 PLC configuration

See Getting Started for AC500 V3 Products § Chapter 1.5.6.2.2 “PLC start-up” on page 3257.

1.1.16 Convert an AC500 V2 project to an AC500 V3 project

Instructions on how to convert a V2 project to a V3 project and differences between V2 and V3.

Keywords: Migration, conversion, V2, V3, differences between V2 and V3, application example, checkbox “Change to AC500 V3 PLC”

1.1.17 Example projects

1.1.17.1 Hardware AC500 V3

1.1.17.1.1 Configuration for example projects

The example projects require a small PLC configuration with I/O devices, e.g., as available in the training case TA5450-CASE. https://to.abb/AfO9-ftT

Table 1: Modules for example projects to get started with AC500 V3 PLC

<table>
<thead>
<tr>
<th>Product name</th>
<th>Type</th>
<th>First project § Chapter 1.1.17.2 “Example project for central I/O expansion” on page 40</th>
<th>Second project § Chapter 1.1.17.3 “Example project for remote I/O expansion with PROFINET” on page 81</th>
</tr>
</thead>
<tbody>
<tr>
<td>PM590-ETH</td>
<td>AC500 V3 CPU</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>TB5620-2ETH</td>
<td>terminal base for CPU</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>DA501</td>
<td>analog/digital mixed input/output (I/O) module</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>TU516-H</td>
<td>terminal unit for I/O module</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>CM579-PNIO</td>
<td>PROFINET communication module</td>
<td>--</td>
<td>x</td>
</tr>
<tr>
<td>CI502-PNIO</td>
<td>PROFINET communication interface module</td>
<td>--</td>
<td>x</td>
</tr>
<tr>
<td>TU508-ETH</td>
<td>terminal unit for communication interface module</td>
<td>--</td>
<td>x</td>
</tr>
<tr>
<td>TA524</td>
<td>blind cap for terminal base</td>
<td>x</td>
<td>x</td>
</tr>
</tbody>
</table>
Electrical connection

In the training case, the control panel CP6607 is included. A control panel is not needed for the example projects.

For testing the example project some inputs require to be connected as follows:

Fig. 9: Training case TA5450

Fig. 10: Wiring of training case
For the example projects, not all input switches and none of the potentiometers included in training case are necessary.

You will need switch I1 for the example project for central I/O expansion.

You will need switch I5 for the example project for remote I/O expansion.

### 1.1.17.2 System assembly, construction and connection

**NOTICE!**

**Avoidance of electrostatic charging**

PLC devices and equipment is sensitive to electrostatic discharge, which can cause internal damage and affect normal operation. Observe the following rules when handling the system:

- Touch a grounded object to discharge potential static.
- Wear an approved grounding wrist strap.
- Do not touch connectors or pins on component boards.
- Do not touch circuit components inside the equipment.
- If available, use a static-safe workstation.
- When not in use, store the equipment in appropriate static-safe packaging.

You can mount AC500 PLC either to DIN rail or to a metal plate. Chapter 1.5.4.5.3 “Mounting and demounting” on page 3068. Here, we recommend to mount on DIN rail.

1. Snap the terminal base onto DIN rail.
2. Snap the additional terminal units for I/O modules onto DIN rail.
3. Make the sensor/actuator wire connections according to the dedicated electronic module you want to use. Provide external process power supply as required.
4. If required, make the fieldbus connections according to the dedicated master communication module you want to use.
5. Plug the appropriate electronic and I/O modules in the correct locations (processor module, communication modules on terminal base, and eventually also communication interface modules and I/O modules onto dedicated terminal units).
6. Connect a programming cable (Ethernet cable between ETH port of CPU and PC with engineering software).

### 1.1.17.2 Example project for central I/O expansion

The following steps show how to set-up an application project and configure the hardware. A simple logic is used as example to introduce in programming and commissioning of the PLC. The workflow for creation of a visualization is explained, as well as how to set-up a webserver for visualization.

#### 1.1.17.2.1 Preconditions

- Automation Builder is installed and licensed as, at least, basic edition.
- AC500 V3 CPU is assembled and connected to the PC.
1.1.17.2 Create, set-up and save your AC500 V3 project

Create a project

1. Launch Automation Builder either out of the desktop icon or out of the Windows menu.

2. Select “New Project” or go to menu “File ➔ New Project”.

3. Select “Projects”.
4. Select “AC500 project”.
5. Fill in project name and choose a location to save the project to.
6. Select “OK”.

7. Select “PLC - AC500 V3”.
8. Select the CPU according to your hardware set-up.
9. Select “Add PLC” to add the CPU to your application.
Configure your CPU

1. Double-click “PLC_AC500_V3”.
   - A tab opens in the editor view.
2. Select “CPU-Parameters Parameters”.
3. Under parameter “Check battery”, choose the value “Off” since there is no battery present inside the CPU module.
4. Keep the default values for all other parameters.

Create folders in the device tree

To optimize the project readability, you will create different folders to group similar objects. The folder names are exemplary. Because the device tree view follows an alphabetical order, we use number prefixes to determine the order.
1. Right-click “Application”.
2. Select “Add Folder”.
3. Type in "10 POUs". This is a name example. Here, the intension is to see this folder as a last one.
   The folder "10 POUs" is for program organization units (POU). POUs are objects of type program, function or function block that are used to create a user program.

Save the project

Select menu “File ➔ Save Project”.
Alternatively, select the "save" icon in the tool bar.
Alternatively, press [STRG] + [S].

1.1.17.3 Configure the I/O module

- The types and order of modules in the Automation Builder project must match the real hardware configuration.
- The position of the modules in the device tree can be changed by drag and drop.
Add an I/O bus module

1. Right-click "IO_Bus" in the device tree.
2. Select "Add object".
3. Select “S500 I/O modules”.
5. Select “Add object” to add the module to the I/O bus.
DA501 variable mapping

1. Double-click “DA501” in the device tree.
   ⇒ A tab opens in the editor view.
2. Select “DA501 I/O Mapping”.
   ⇒ Here, you will map variable names (symbols) for the channels you will need in the program.

The suggested name convention is based on "Hungarian notation". A name prefix is describing variable type: e.g., "x" = variable of type BOOL, "w" = WORD, "i" = INT (integer) etc. This increases the code readability and is helpful for program analysis.

Handle the digital input variables

1. Open the list of the digital inputs.
2. Fill in the variable names:

<table>
<thead>
<tr>
<th>Channel</th>
<th>Type</th>
<th>Variable</th>
</tr>
</thead>
<tbody>
<tr>
<td>Digital input DI8</td>
<td>BOOL</td>
<td>xDI_08_DA501_I1</td>
</tr>
</tbody>
</table>
Handle the digital output variables

1. Open the list of the digital outputs.
2. Fill in the variable names:

<table>
<thead>
<tr>
<th>Channel</th>
<th>Type</th>
<th>Variable</th>
</tr>
</thead>
<tbody>
<tr>
<td>Digital output DC16</td>
<td>BOOL</td>
<td>xStartDrilling1</td>
</tr>
</tbody>
</table>

1.17.2.4 Programming and compiling

Task configuration

A task is a time unit in the processing of a user program (IEC application), which defines by parameters the way and the speed the CPU is executing the user program.

In the device tree, you see the objects “Task configuration” and “Task”. Both created automatically with the project.

Double-click “Task” in the device tree.

⇒ A tab opens in the editor view.

For this project you will use only one cyclic task. Keep the default settings for the task.

Priority

This is how the CPU prioritizes the task, when more than one task is defined. Priority 0...15 = realtime tasks, priority 16 = non-realtime task.

Type

In the CPU you can run tasks dependent on the demands of the process

Interval

For cyclic tasks you can set the cyclical execution time. It is usually set in milliseconds with IEC time syntax

Watchdog

To keep track of the time it takes to complete the task

Calls

You can call in one or more program POUs in one single task
Main program PLC_PRG

In the default task configuration \( \infty \) (shown in chapter 1.1.17.2.4.1 Task configuration on page 48), there is one call of a POU (program organization unit) i.e. "PLC_PRG". In your project the "PLC_PRG" will become a main program containing calls to other programs (POUs) which you will create one by one.

![Diagram of PLC_PRG POU](image)

The PLC_PRG POU has been defined by default in ST (structured text) editor. Keep this setting because of good visibility of the instructions at a glance and good handling for troubleshooting.

To optimize the project readability, you will work with the previously created folder "10 POUs" and add the created subroutines (POUs) to this folder. The subroutines will be created in FBD (function block diagram) editor.

Boolean logic "NOT"

Application example "driller"

Recognizing of a driller by a photo sensor. "TRUE" input signal from sensor indicates that a driller is broken. If driller has been found correct, then start drilling.

![Diagram of driller](image)

Table 2: Required behavior

<table>
<thead>
<tr>
<th>Signal from photo sensor</th>
<th>Required signal of motor ON</th>
</tr>
</thead>
<tbody>
<tr>
<td>FALSE</td>
<td>TRUE</td>
</tr>
<tr>
<td>TRUE</td>
<td>FALSE</td>
</tr>
</tbody>
</table>
Table 3: Hardware set-up

<table>
<thead>
<tr>
<th>Element</th>
<th>HW channel</th>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Switch I1</td>
<td>DA501 DI8</td>
<td>xDI_08_DA501_I1</td>
<td>Photo sensor</td>
</tr>
<tr>
<td>LED output DC16</td>
<td>DA501 DC16</td>
<td>xStartDrilling1</td>
<td>Motor on</td>
</tr>
</tbody>
</table>

Implementation

Create a new program POU in the project

1. Right-click “10 POUs”.
2. Select “Add object”.
3. Select “POU”.
4. Select “Add object”.

PLC Automation with V3 CPUs
Getting started > Example projects

2020/12/10 ADR010583, 1, en_US
5. Fill in "_01_Assignment_NOT".
6. Select "Program".
7. Select "Function Block Diagram (FBD)".
8. Select [Add] to add the POU.

Assign the hardware DI signals to local variables
1. Double-click POU "_01_Assignment_NOT" in the device tree.
2. Select "Assignment" from the ToolBox.
3. Drag and drop "Assignment" into the "Start here" field in network "1".
4. Select “???” on the left side of the assignment, then select “…”

5. Open the “Io Config_Globals_Mapping” mapping list and select “xDI_08_DA501_I1”.

6. Select [OK] to add this variable to the left side of the assignment connector.

7. Select “???” on the right side of the assignment connector and mark the “???”.

8. Create a new local variable by typing in "xDrillerBroken1" which will replace the “???”.

9. Press [Enter].

   “Auto Declare” opens.

   You see the written variable name and the data type BOOL. The scope is "VAR". It means it is a local variable within this POU.

10. Select [OK] to accept the entries.
11. Drag and drop “Network” from the ToolBox to the down-arrow of network 1.
   ⇒ You added a network “2” below network 1.

Add assignments and a Boolean NOT to the DO signals

1. Add an assingment from the ToolBox.
2. Type in or copy & paste "xDrillerBroken1" to the left side of the instruction line.
3. Select “???” on the right side of the instruction line, then select “...”.
   ⇒ “Input Assistant” opens.

4. In the “IoConfig_Globals_Mapping” variable list, select “xStartDrilling1”.
5. Select [OK] to close the dialog.

6. Right-click the center of assignment pin.

7. Select “Negation” to add a negation to the assignment.
Call the POU in the PLC_PRG

1. Double-click “PLC_PRG”.
2. Select the first line in “PLC_PRG” and press [F2].
   → “Input Assistant” opens.
3. Select “Module Calls”.
4. Open “Application”.
5. Open “10 POUs” and select “_01_Assignment_NOT”.

![Image of PLC_PRG call process](image-url)
6. Select [OK] to close the dialog.

Compile the project

Before logging-in to the CPU, you need to compile the complete code without any errors.

Select menu “Build ➔ Generate code”.

The result of the compiling is shown in the “Messages” field at the bottom of the screen.

If you skip the compiling and select “Login”, the Automation Builder will automatically trigger compiling in advance to logging-in.

Save the project

Select menu “File ➔ Save Project”.

Alternatively, select the "save” icon in the tool bar.

Alternatively, press [STRG] + [S].

1.1.17.2.5 Set-up the communication gateway

IP is configured properly ➔ Chapter 1.1.4 “Set-up communication parameters in windows” on page 25.
CPU and PC are connected with an Ethernet cable.

1. Right-click “PLC_AC500_V3” in the device tree.
2. Select “Communication Settings”.
3. Keep the default value in the IP address of the CPU or type in the current IP address, if differs.

The standard (default) IP address of the port ETH1 is: 192.168.0.10
The standard (default) IP address of the port ETH2 is: 192.168.1.10
4. Select “OK” to implement the IP address.

**Network scan**

If you need to scan the network for the CPU or if you have multiple CPUs on the same network.

1. Right-click “PLC_AC500_V3” in the device tree.
2. Select “Communication Settings”.

   ![Communication Settings](image)

3. Select “...”.
   - “Pick IP Address for “PLC_AC500_V3”” opens.

   ![Pick IP Address](image)

The automatic scan runs.

The results will appear in this field.

4. Select the CPU in the field and select [OK] to implement the needed communications gateway.

**Check communication settings**

If you need to check the communications settings or if you want to see more information about the current selected CPU.
1. Double-click “PLC_AC500_V3” in the device tree.
2. Select “Communication Settings”.
   - The selected IP address is shown.
3. If the IP address is not visible, enter the IP address manually.
4. To test the connection and/or to see the CPU information press [Enter] or click on the black dot next to the PLC picture.

### 1.17.2.6 AC500 V3 firmware installation and update

The PLC firmware can be updated via Automation Builder.

> This is also necessary for commissioning V3 CPUs.

A very new CPU has no pre-installed firmware. To guarantee the authenticity of delivered AC500 firmware, V3 CPUs are delivered with a boot loader only. You need to download a valid firmware to the CPU. After download, the functionality of the CPU is given.

- An Automation Builder project with an AC500 V3 CPU is open.
- CPU is in "stop" mode or shows uPdAtE (update) on the display.
- After update the CPU shows either donE or StoP on the display.
For new modules: IP address is set. (The default IP address is 192.168.0.10)

1. Double-click CPU “PLC_AC500_V3”.
2. Select “Version information”.
3. Select [Update Firmware].
   - While the update process is running, the RUN and ERR LEDs are toggling, i.e., they are flashing alternating.
4. Wait for the PLC to finish the update.
   A completed update is indicated by a message on the display. Either done, or Stop.

**NOTICE!**
Do not disconnect the power supply during the update process! The PLC could be damaged.

- Stop indicates a restart has been performed by the CPU. When done is displayed sometimes it is necessary to re-boot the CPU manually, e.g., by powering-off. Manual re-boot might be, e.g., for some older CPU versions or if downgrading to an older firmware version according to application settings.

The CPU display shows "stop" after re-boot. The update process is finished.
5. If necessary, refresh the version information by switching to another tab and back.

⇒ Successful firmware update:

<table>
<thead>
<tr>
<th>LED</th>
<th>LED flashes</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>RUN and ERR</td>
<td>Toggling</td>
<td>Update pending</td>
</tr>
<tr>
<td>RUN</td>
<td>Flashing slow</td>
<td>Done successful</td>
</tr>
<tr>
<td>ERR</td>
<td>Flashing slow</td>
<td>Done failed</td>
</tr>
</tbody>
</table>

**Behavior of LEDs during firmware update**

**Log-in to CPU and download the program**

Logging-in to the CPU will load the project into the AC500 V3 CPU. The first log-in will also load the hardware set-up.
1. Select menu “Online → Login [PLC_AC500_V3]”.  
   ➤ A pop-up will appear.
2. Select “Yes” to download the application to the AC500V3 CPU.
   ➤ PLC is in "stop" mode.
3. Start the PLC  
   Chapter 1.1.17.2.8.1 “Start the program execution” on page 62.

By default, a download generates following actions in the CPU:

- The project is stored in the RAM memory.
- The project is stored in the flash EEPROM, if boot application was created.

**1.1.17.2.8 Test the program**

**Start the program execution**

- You are logged in the CPU.
- An executable project is loaded to the CPU.
The CPU is in "stop" mode.

Select menu ""Debug ➔ Start [PLC_AC500_V3]"". Alternatively, select the "start" icon in the tool bar. Alternatively, press [F5].

Test the function

Operate the switch I1 and observe:
- The LEDs of the relevant DA501 inputs and outputs.
- The online status of inputs and outputs within the POU.

Stop the program execution

You are logged in the CPU.
☑️ An executable project is loaded to the CPU.
☑️ The CPU is in "run" mode.

Select menu ""Online ➔ Stop [PLC_AC500_V3]""
Select menu "Debug ➔ Stop [PLC_AC500_V3]"
Alternatively, select the "stop" icon in the tool bar.
Alternatively, press [Shift] + [F8].
1.1.17.2.9 Set-up visualization

Add the VisualizationManager

1. Right-click “Application” in the device tree.
2. Select “Add object”.
3. Select “VisualizationManager”.
4. Select [Add object] to add the VisualizationManager to the project.
   ➞ Dialog “Add Visualization Manager” opens.
5. Select [Add].

⇒ You added the objects “VisualizationManager” and “VISU-TASK” to the device tree.
Set-up the VisualizationManager

1. Double-click VisualizationManager in the device tree.
   ⇒ A tab opens in the editor view.

2. Select “Settings”.

3. Open the drop-down menu “Selected style”.

4. Select “Default, x.x.x” (exemplary).
5. Open the drop-down menu “Selected language”.

8. Keep the file transfer to enable the visualization on the PLC (mandatory for web server function © Chapter 1.1.17.2.11 “Enable web visualization” on page 75).
Save the project

[Diagram of project saving process]

- Select menu “File ➔ Save Project”.
- Alternatively, select the "save" icon in the tool bar.
- Alternatively, press [STRG] + [S].

1.1.17.2.10 Create visualization

Add a folder for visualization screens

1. Right-click “Application” in the device tree.
2. Select “Add Folder”.
3. Type in "02 VISUs".
4. Select [OK] to add the folder.
Add a screen for "_01_Assignment_NOT" POU

1. Right-click "02 VISUs".
2. Select "Add object".
3. Select object "Visualization".
4. Select [OK].
5. Type in "PLC_VISU".
6. Select [Add].

⇒ A tab opens in the editor view.
The name "PLC_VISU" has been chosen, because it is the default name for a home screen in a web visualization.

If you have more than one visualization object in your project, it will be useful to choose another name, e.g. "_01_Assignment_NOT_v". And to choose "PLC_VISU" as a home screen to access all available visualization screens.

The name of a visualization object can be modified afterwards.

Creating and configuring of visualization

Change background color

1. Double-click “PLC_VISU” in the device tree.
   ⇒ A tab opens in the editor view.

2. Right-click anywhere on the "PLC_VISU" editor page.

3. Select "Background".

4. Enable the check box “Use Color”.
   ⇒ This enables the drop-down menu.

5. Select a color, e.g., “Lightgray”.

6. Select [OK] to add the color to "PLC_VISU".
Add a screen title

1. Double-click on “PLC_VISU” in the device tree.

2. Select “ToolBox”.

3. Select “Common controls”.

4. Drag and drop “Label” to the page.

5. Type in "Start drilling condition".

Further lines and labels

1. Double-click on “PLC_VISU” in the device tree.

2. Select “ToolBox”.

3. Select “Basic”.

4. Drag and drop the line. Then drag the line to the needed length.
5. Follow the same procedure to create the other shapes and labels.

**Lamp element for signal indication**

1. Double-click on “PLC_VISU” in the device tree.
2. Select “ToolBox”.
3. Select “Lamps/Switches/Bitmaps”.
4. Drag and drop “Lamp” to the screen.
5. Adapt the size, if required.

7. Double-click on “Variable” and select “...” to select a variable from the list.

8. Under “IoConfig_Globals_Mapping”, select “xStartDrilling1”.

9. Select [OK].

**Compile the project**

Before logging-in to the CPU, you need to compile the complete code without any errors.
Select menu “Build ➔ Generate code”.

The result of the compiling is shown in the “Messages” field at the bottom of the screen.

If you skip the compiling and select “Login”, the Automation Builder will automatically trigger compiling in advance to logging-in.

Save the project

Select menu “File ➔ Save Project”. Alternatively, select the “save” icon in the tool bar. Alternatively, press [STRG] + [S].

Loading the project to the CPU

1. Download the project to the CPU as described in Chapter 1.1.17.2.7, on page 61.
2. Check the notification window at the end of the download. In case of message "Boot parameters were changed. These changes will be applied after reboot", a reboot of the CPU is required after creation of the boot project.
Test the program

1.1.17.2.11 Enable web visualization

Add a web server object to the device tree

Both Ethernet ports can be configured for web server protocol each. This description deals with ETH1 configuration for the web server.
1. Right-click “Protocols” in the device tree.
   Right-click “ETH1” in the device tree.
2. Select “Add object”.
3. Select “Web Server”.
4. Select “Add object”.
   ⇒ You added and activated a web server on Ethernet port 1 on the AC500 V3 CPU.

**Set-up the web server**

1. Double-click “WebVisu” in the device tree.
2. Under “Start Visualization”, select “…”.
   ⇒ A list opens.

3. Select the “PLC_VISU” screen from the list.

4. Keep all further settings with default values.
5. Select the link “Show used visualizations”.

The VisualizationManager editor and there the tab “Visualizations” opens. All screens and dialog elements created in the project are visible. Here, you can select which screens are enabled or disabled for web visualization.

If you want to select another screen as a start visualization, you must modify the adequate parameter in the webvisu.htm file:

```html
<param name="STARTVISU" value="PLC_VISU">
```

Compile the project

Before logging-in to the CPU, you need to compile the complete code without any errors.

Select menu “Build ➔ Generate code”.

The result of the compiling is shown in the “Messages” field at the bottom of the screen.

If you skip the compiling and select “Login”, the Automation Builder will automatically trigger compiling in advance to logging-in.
Save the project

Select menu “File ➔ Save Project”. Alternatively, select the "save" icon in the tool bar. Alternatively, press [STRG] + [S].

Loading the project to the CPU

1. Download the project to the CPU as described in Chapter 1.1.17.2.7, on page 61.
2. Check the notification window at the end of the download. In case of message "Boot parameters were changed. These changes will be applied after reboot", a reboot of the CPU is required after creation of the boot project.

Create a boot project

By default, after project download, the boot project is created automatically.

Rebooting the CPU

Reboot the CPU by switching OFF and ON the power supply. (The parameter for webserver activation is a boot parameter which is loaded during boot of the CPU)

Test the web visualization

☑ You have downloaded the project and created the boot project.
☑ The CPU has been rebooted.
☑ You are logged in.
☑ CPU is in "stop" mode.
1. Start the project execution, e.g., from the tool bar.
2. Launch an internet browser.
   192.168.0.10 is the IP address of CPU’s ETH1 port.
   /webvisu.htm is the default htm file.
   Web visualization will be loaded.

   The start screen “PLC_VISU” is displayed in a responsive view.

4. Test the function by operating switch I1.
5. Test the results for responsive view by changing the web browser window size.

1.17.2.12  Reset the CPU

**Reset values and parameters**
In some cases, it could be required to do a CPU reset, e.g., for resetting of counter values, parameters etc.

![Reset commands in “Online” menu](image)

*Fig. 11: Reset commands in “Online” menu*

- **Reset warm**
  - All variables are reset, except RETAIN PERSISTENT variables.

- **Reset cold**
  - Causes initialization of all variables, except PERSISTENT variables. By recommended creation of remanent variables always with both properties: PERSISTENT and RETAIN, this command resets all variables, except PERSISTENT RETAIN variables.

- **Reset origin**
  - All variables and the application project are reset.
Table 4: Behavior of variables of type VAR (local or global) and variables of type PERSISTENT RETAIN

<table>
<thead>
<tr>
<th>After online command</th>
<th>VAR</th>
<th>VAR PERSISTENT RETAIN</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Online change”</td>
<td>no change</td>
<td>no change</td>
</tr>
<tr>
<td>“Download”</td>
<td>initialization</td>
<td>no change</td>
</tr>
<tr>
<td>“Reset warm”</td>
<td>initialization</td>
<td>no change</td>
</tr>
<tr>
<td>“Reset cold”</td>
<td>initialization</td>
<td>no change</td>
</tr>
<tr>
<td>“Reset origin”</td>
<td>initialization</td>
<td>initialization</td>
</tr>
<tr>
<td>After power supply off</td>
<td>initialization</td>
<td>no change</td>
</tr>
</tbody>
</table>

Complete reset of the CPU

To do a complete reset of the CPU thereby erasing the application from the RAM and flash EEPROM do the following.

1. Right-click the station object “PLC_AC500_V3” in the device tree.
2. Select “Reset origin device [station name]”.  
   ⇒ The application is completely erased from the CPU (complete project from all memory areas).

1.1.17.3 Example project for remote I/O expansion with PROFINET

This example introduces the configuration of the PLC with remote I/O. The use of I/O channels in a program and commissioning of the configuration is shown.

1.1.17.3.1 Preconditions

- Automation Builder is installed and licensed as, at least, standard edition.
- AC500 V3 CPU is assembled and connected to the PC.
● Configuration and programming of this example project will be made in the existing example project for central I/O expansion "Chapter 1.1.17.2 “Example project for central I/O expansion” on page 40.

● CM579-PNIO communication module is inserted in terminal base and connected to the PLC.

● CI502-PNIO communication interface module is inserted in terminal unit and connected to the PLC.

1.1.17.3.2 Set-up PROFINET controller

Add the CM579-PNIO to the device tree

1. Under “Extension_Bus”, right-click “Slot_1” in the device tree.
2. Select “Add object”.
3. Select “CM579-PNIO”.
4. Select “Replace object” to add the CM579-PNIO.

Set-up the general behavior


   ⇒ A tab opens in the editor view.
2. Select “CM579-PNIO Parameters”.

Run on configuration fault
This parameter will prohibit the PLC from running if the CM579-PNIO has a configuration error/fault.

Bus behavior
This parameter sets how the data from the bus flows in/out of the CM579-PNIO.

3. Select “Status”.
⇒ This opens the bus controller status and gives a basic status overview.

4. Select “Information”.
⇒ This page contains general information about the CM579-PNIO.

5. For the example project, you can keep the default settings.

Set-up the PROFINET IO controller

To edit settings for the controller, you must not be logged-in to the PLC.

⇒ A tab opens in the editor view.

2. Select “PROFINET IO CONTROLLER”
3. Select "General".

4. Here, you can set-up the way, IP addresses are distributed out to the industrial bus network. You can even set, what IP-address and DNS name (station name) the PROFINET controller has.

   For the example project, keep the default settings.

### 1.1.17.3.3 Set-up PROFINET device

#### Hardware preparation

1. Switch off the power supply of your PLC.

2. Use a screw driver to set the CI502 module address to "02" by positioning of the upper rotary switch to "0" and lower switch to "2". Note, that the numbers have hexadecimal format.

3. Switch on the power supply.
Add the CI502-PNIO to the device tree

1. Right-click “PNIO_Controller” in the device tree.
2. Select “Add object”.
3. Select “CI502-PNIO-Device”.
4. Select “Add object” to add the device.

Configure the CI502-PNIO device

Configure the CI502-PNIO PROFINET IO device

1. Double-click “CI502_PNIO_Device”.
   ⇒ A tab opens in the editor view.
2. Select "PROFINET IO Device".
   Select “General”.

   ![Image of PROFINET IO Device configuration panel]

   Station name: Default station name
   IP Parameter: IP-addressing parameters of the node. If modifications are required for "IP Parameter", they must be done also for CM579-PNIO and all other devices in this PROFINET line.
   Communication: Communication time set-up
   VLAN: virtual local area network ID
   RT Class: PROFINET IO RT (real time) type settings

3. Set station name to "ci502-pn-02" according to hardware settings.

   For numbers greater than 09 always make sure, that the last two decimal digits of the node’s "Station Name" in Automation Builder correspond to the position of module’s rotary switches (hexadecimal values): e.g., "ci502-pn-10" <-> "0A" or "ci502-pn-16" <-> "10".

4. Leave the default settings for "IP Parameter".

5. Adjust the communication time settings to get a Watchdog (ms) 24:
   - "Send clock (ms)": 4
   - "Reduction ratio": 2
   - "Phase": 1

6. Leave the default settings for "VLAN ID".

7. Leave the default settings for "RT Class".

   ![Information icon]

   If the node has the same device address (the last two digits of the device name) as set by means of the rotary switches on the module, all the node parameters will be loaded automatically upon initialization scan of the CI50x module. This allows, e.g., the module exchange without an engineering tool.
Create CI502-PNIO I/O mapping to symbols

1. Double-click “CI502_IO”.

2. Select “PNIO Module I/O Mapping”.

3. Fill in the variable names:

<table>
<thead>
<tr>
<th>Element</th>
<th>Hardware channel</th>
<th>Symbol</th>
</tr>
</thead>
<tbody>
<tr>
<td>Switch I5</td>
<td>CI502 DI8</td>
<td>xDI_08_CI502_I5</td>
</tr>
<tr>
<td>LED output DO8</td>
<td>CI502 DO 8</td>
<td>xDO_08_CI502</td>
</tr>
</tbody>
</table>
Add remote I/O expansion to project

Add a program POU to the project

1. Right-click “01 - POUs” in the device tree.
2. Select “Add object”.
3. Select “POU”.
4. Select “Add object”.
5. Fill in "_30_PNIO_test".
6. Select “Program”.
7. Select “Function Block Diagram”.
8. Select [Add] to add the POU.

Create a POU logic

1. Double-click “30_PNIO_test” in the device tree.

2. In the ToolBox, select “Assignment”.
3. Drag and drop “Assignment” into the "Start here" field in network "1".

4. Select “???” on the left side of the assignment, then select “...”.
5. In “IoConfig_Globals_Mapping” list, select “xDI_08_CI502_I5”.
6. Select [OK] to add this variable to the left side of the assignment connector.
7. Select “???” on the right side of the assignment, then select “...”.
8. In “IoConfig_Globals_Mapping” list, select “xDO_08_CI502”.
9. Select [OK].

Call the POU in PLC_PRG
1. Double-click “PLC_PRG”.
2. Select the next free line in “PLC_PRG” and press [F2].
   ⇒ “Input Assistent” opens.
3. Select “Module Calls”.
4. Open “Application”.
5. Open “10 POUs” and select “_30_PNIO test”.
6. Select [OK] to close the dialog.

Compile the project
Before logging-in to the CPU, you need to compile the complete code without any errors.
Select menu “Build ➔ Generate code”.

The result of the compiling is shown in the “Messages” field at the bottom of the screen.

If you skip the compiling and select “Login”, the Automation Builder will automatically trigger compiling in advance to logging-in.

Save the project

Select menu “File ➔ Save Project”. Alternatively, select the "save" icon in the tool bar. Alternatively, press [STRG] + [S].

Loading the project to the CPU

1. Download the project to the CPU as described in Chapter 1.1.17.2.7 , on page 61.
2. Check the notification window at the end of the download. In case of message "Boot parameters were changed. These changes will be applied after reboot", a reboot of the CPU is required after creation of the boot project.
The CPU is in "stop" mode.

Select menu "Debug ➔ Start [PLC_AC500_V3]". Alternatively, select the "start" icon in the tool bar. Alternatively, press [F5].

Test the function

Operate the switch I5 and observe:

- The LEDs of the relevant CI502 inputs and outputs.
- The online status of inputs and outputs within the POU.

1.1.17.3.6 Reset the CPU

Reset values and parameters

In some cases, it could be required to do a CPU reset, e.g., for resetting of counter values, parameters etc.
Fig. 12: Reset commands in “Online” menu

<table>
<thead>
<tr>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>warm</td>
<td>All variables are reset, except RETAIN PERSISTENT variables.</td>
</tr>
<tr>
<td>cold</td>
<td>Causes initialization of all variables, except PERSISTENT variables. By recommended creation of remanent variables always with both properties: PERSISTENT and RETAIN, this command resets all variables, except PERSISTENT RETAIN variables.</td>
</tr>
<tr>
<td>origin</td>
<td>All variables and the application project are reset.</td>
</tr>
</tbody>
</table>

Table 5: Behavior of variables of type VAR (local or global) and variables of type PERSISTENT RETAIN

<table>
<thead>
<tr>
<th>After online command</th>
<th>VAR</th>
<th>VAR PERSISTENT RETAIN</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Online change”</td>
<td>no change</td>
<td>no change</td>
</tr>
<tr>
<td>“Download”</td>
<td>initialization</td>
<td>no change</td>
</tr>
<tr>
<td>“Reset warm”</td>
<td>initialization</td>
<td>no change</td>
</tr>
<tr>
<td>“Reset cold”</td>
<td>initialization</td>
<td>no change</td>
</tr>
<tr>
<td>“Reset origin”</td>
<td>initialization</td>
<td>initialization</td>
</tr>
<tr>
<td>After power supply off</td>
<td>initialization</td>
<td>no change</td>
</tr>
</tbody>
</table>

Complete reset of the CPU

To do a complete reset of the CPU thereby erasing the application from the RAM and flash EEPROM do the following.
1. Right-click the station object “PLC_AC500_V3” in the device tree.
2. Select “Reset origin device [station name]”.
   ⇒ The application is completely erased from the CPU (complete project from all memory areas).

1.2 Automation Builder installation manager

Automation Builder installation manager allows you to install customer specific software packages, modify the existing installation, update installation information and to uninstall Automation Builder software packages in a comfortable and flexible way.

You can launch installation manager from the main menu of Automation Builder or from Windows start menu.
1. Open Automation Builder software.
   From the Tools menu, select Installation Manager.
2. As an alternative, launch installation manager from Windows start menu: “Start menu ➔ All Programs ➔ ABB ➔ Automation Builder ➔ ABB Automation Builder Installation Manager”. Installation manager starts.

Options:

- **Installed packages**: Shows all installed packages of Automation Builder.
- **Uninstall all**: Uninstalls the currently installed Automation Builder software.
- **Install Package**: Installs customer specific software packages.
- **Modify**: Adds or removes installed software packages.
- **Info Export**: Exports detailed information of installed packages in a notepad.
- **Check for Update**: Checks if your installed version of Automation Builder is up to date and checks for updates.

### 1.2.1 Installing customer specific package

Installation manager allows you to install customer specific software packages (CABPKG files). These packages are separately distributed to the customer based on the customer requirement.
1. In the installation manager, click **Install Package**.
2. Select the package to be installed (.cabpkg file) from the file system.
3. Select the components to be installed.
4. Click **Install**.
   > Data installation starts.
5. Successfully installed components are indicated with ✅.
   Errors during data download are indicated with ⚠️. Errors during download of any package component aborts the installation. In this case click **Show Log** and save the log data.
   Send the log file to ABB support team.
   Click **Finish** to end the wizard.

### 1.2.2 Adding or removing installed software packages

1. In the installation manager, click **Modify**.
   > The selection page opens.
   The selected software packages are installed already.
   The not selected software packages are not installed.
2. Select the software packages you want to install.
Unselect the software packages you want to uninstall.

You cannot unselect the main **ABB Automation Builder** software package.

If also an older Automation Builder version or Control Builder Plus version shall be installed for compatibility reasons, select the appropriate options under **Install also previous product versions**. This allows to open and edit a corresponding project in the original version without a previous project upgrade.

3. Click **Continue**.
The following three cases are possible:
- The selected software package starts downloading and installing.
- The unselected software package will uninstall.
- The unselected software package will uninstall first and then download and install the selected software package.

4. Successfully downloaded components are indicated with ☑.
Errors during data download are indicated with ☢. Errors during download of any package component aborts the installation. In this case click **Show Log** and save the log data. Send the log file to ABB support team.
Click **Finish** to end the wizard.

If you modify the type of installed edition, a warning message is displayed.

1.2.3 Checking for updates

In the installation manager, click **Check for Updates**.

If the installed Automation Builder version is up-to-date, the following message will appear.

If a newer Automation Builder version is available, you will get an option to download and install the new version.
1.2.4 Uninstalling Automation Builder

Installation manager offers a comfortable way to uninstall Automation Builder software. This will uninstall all related packages of Automation Builder platform as well, such as Mint Plugin, Automation Builder Extensions, Drive Manager etc.

1. In the installation manager, click **Uninstall all**.
   - A warning message is displayed to uninstall Automation Builder software.
     - Click **Yes** to continue.

2. If Automation Builder instances are running, a warning message is displayed. Close running instances of Automation Builder and click **Retry** to continue uninstallation.
   - With **Abort** uninstallation of the current package is stopped. Uninstallation is continued with the next package. With **Ignore**, uninstallation is forced. As this can lead to an erroneous uninstallation, we recommend you, *not* to use this option.

3. If installation manager was launched with “**Tools** ➤ **Installation Manager**”, the following message is displayed as Automation Builder is still running:
   - With **Yes** Automation Builder software is closed to continue uninstallation procedure.
   - With **No** uninstallation of the current package is stopped. Uninstallation is continued with the next package.

4. For each of the packages being uninstalled, system may prompt to continue uninstallation.
5. Successfully uninstalled components are indicated with 🔄. Errors during uninstallation are indicated with 🚨. Errors during uninstallation of any package component aborts the uninstallation. In this case click **Show Log** and save the log data. Send the log file to ABB support team.

Click **Finish** to end the wizard.

### 1.3 Programming with IEC 61131-3 editor

#### 1.3.1 CODESYS Development System

**Using CODESYS help**

CODESYS Help is intended to assist you in using the CODESYS Development System easily and successfully. You will find quick answers to questions and solutions to problems.

Each help component consists of a concept section and a reference section.

In the concept sections, we explain in detail all topics that are relevant for creating CODESYS projects. The concepts are supplemented with instructions that lead you step-by-step to the intended result.

In the reference sections, we provide complete reference works for the user interface and programming of CODESYS.

The following formats of CODESYS Help are provided:

- CODESYS Offline Help: CHM-based CODESYS Help
- CODESYS Online Help: Web-based CODESYS Help

In the CODESYS options, you determine whether to use CODESYS Offline Help or CODESYS Online Help.

You can call the context-sensitive help directly from the user interface of the CODESYS Development System. In CODESYS, when you position the cursor over an object, menu command, or programming element, and then press the [F1] key, the respective help page opens. As an alternative, you can use the commands in the "Help" menu. This is a full-text search. The index search is possible in CODESYS Offline Help only.

**Search operators for the offline help**

- **AND**
  - Used automatically, for example the input of the search terms Device Diagnosis has the same results as the input of *Device AND Diagnosis*
  - The * placeholder is used automatically. However, the * character must not be used as a wildcard because in this case the * character will be searched for specifically.

**Search operators for the online help**

- **AND**
- **OR**
- **NOT**
  - Example: abc NOT abcd: The search result includes all help pages that contain abc and excludes the pages with abcd.
- **ANDNOT**
  - ANDNOT is the combination of the search operators AND nd NOT.
- **ANDMAYBE**
  - Example: The search for abc ANDMAYBE xyz finds the help pages that contain abc and xyz, and all pages that contain only the string abc.
- **Placeholders**
  - *: Replaces any number of characters
  - ?: Replaces exactly one character

In the online help, you can use parentheses to group together multiple search operators for complex search queries. Example: ((profinet AND cycle) OR (Ethernet/IP AND cycle)) ANDNOT IRT
The CODESYS Development System IEC 61131-3 programming tool forms the core of the CODESYS software platform for tasks in industrial automation technology. With additional, integrated solutions for motion control, visualizations, and fieldbus connections, the usual practical requirements are covered in one system.

The free CODESYS Development System is a IEC 61131-3 programming platform for automation devices with control tasks. It provides diverse and comfortable engineering solutions to support you in your developing tasks:

- **Features**
  - Project configuration through wizards.  
    For this see in this Online Help:  
    - Chapter 1.3.1.3 “Creating and configuring a project” on page 124
  - Adaptability of the user interface.  
    For this see in this Online Help:  
    - cds_struct_application_programming,  
      -  
      - Chapter 1.3.1.1.2 “Customizing the user interface” on page 102
  - Creation of professional IEC 61131-3 controller applications with a host of standard features.  
    For this see in this Online Help:  
    - Chapter 1.3.1.9 “Programming of applications” on page 171
For this see in this Online Help:

<table>
<thead>
<tr>
<th>Feature</th>
<th>Page Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>User-friendly programming with mouse and keyboard in all IEC 61131-3 languages.</td>
<td>Chapter 1.3.1.24.1 “Programming languages and editors” on page 428</td>
</tr>
<tr>
<td>Appropriate editors for FBD, LD, IL, ST, SFC, additionally the variants CFC and Extended CFC.</td>
<td></td>
</tr>
<tr>
<td>Input assistance for the input and configuration of data.</td>
<td>Chapter 1.3.1.9.6 “Using input assistance” on page 208</td>
</tr>
<tr>
<td>Support of object-oriented programming.</td>
<td>Chapter 1.3.1.9 “Programming of applications” on page 171</td>
</tr>
<tr>
<td>Real object-oriented programming (OOP) fully compliant with the IEC 61131-3 standard in all IEC 61131-3 languages, without any additional tools.</td>
<td></td>
</tr>
<tr>
<td>Inheritance of POUS to similar application parts to reduce development time and errors.</td>
<td></td>
</tr>
<tr>
<td>Object-orientation is not a must: Functional and object-oriented programming can be used and mixed as required.</td>
<td></td>
</tr>
<tr>
<td>Comprehensive project comparison, also for graphic editors.</td>
<td>Chapter 1.3.1.5 “Comparing projects” on page 137</td>
</tr>
<tr>
<td>Library concept for an easy reutilization of application.</td>
<td>Chapter 1.3.1.19 “Using libraries” on page 371</td>
</tr>
<tr>
<td>Debugging and online features for the fast optimization of the application code and to speed up testing and commissioning.</td>
<td>Chapter 1.3.1.12 “Testing and debugging” on page 309</td>
</tr>
<tr>
<td>Integrated compilers for many different CPU platforms for optimizing the controller performance.</td>
<td>Chapter 1.3.1.25.3.12.2 “Dialog 'Project Environment' - 'Compiler Version’” on page 1070</td>
</tr>
<tr>
<td>Security features for the protection of the source code and the operation of the controller.</td>
<td>Chapter 1.3.1.25.3.11.3 “Dialog Box 'Project Settings' - 'Compileoptions’” on page 1062</td>
</tr>
<tr>
<td>Field bus support and programming of devices from different manufacturers.</td>
<td>Chapter 1.3.1.8 “Configuring I/O links” on page 155</td>
</tr>
<tr>
<td>Extensibility and adaptability without leaving the framework.</td>
<td>Chapter 1.3.1.21 “Managing packages and licenses” on page 375</td>
</tr>
<tr>
<td>Additionally:</td>
<td></td>
</tr>
<tr>
<td>Many seamlessly integrated tools for different kinds of automation tasks, for example CODESYS Visualization, CODESYS SoftMotion, CODESYS Application Composer.</td>
<td></td>
</tr>
<tr>
<td>Please always note the possibility to extend the functionalities by &quot;AddOn&quot;-Packages, provided in the CODESYS Store: CODESYS Store.</td>
<td></td>
</tr>
</tbody>
</table>

Customization of the user interface language

In the “Option ➤ International Settings” dialog you can customize the language of the user interface of the development system. This change will take effect the next time you start CODESYS. You can adjust the help language separately.
If you start CODESYS from the command line, you can add a parameter to adjust the user interface language.

See also

- Chapter 1.3.1.25.3.13.12 “Dialog ‘Options’ – ‘International Settings’” on page 1083
- Chapter 1.3.1.18 “Using the command-line interface” on page 366

1.3.1.1 Configuring CODESYS

CODESYS Development System allows to configure the behavior, the appearance, the content of the menus and the arrangement of the windows individually. In the “Tools” menu you find dialogs for customizing the user interface and to setup the CODESYS options.

See also

- Chapter 1.3.1.25.3.14 “Dialog ‘Customize’” on page 1089
- Chapter 1.3.1.25.3.13 “Dialog ‘Options’” on page 1073

1.3.1.1.1 Setting CODESYS options

You can configure the behavior and appearance of the CODESYS Development System in the different tabs of the “Options” dialog. The dialog opens by clicking “Tools ➔ Options”. Here you can configure the default settings for different editors and functionalities. These settings apply throughout CODESYS.

The settings are stored in your current user profile on your local system. For use on other systems, option settings, either user-specific or machine-specific (computer), can be exported to an XML file.

In V3.5 SP13 and later, CODESYS checks whether an older version is already installed when the development system is started for the first time. If this is the case, then the “Import Assistant” dialog opens for transferring the CODESYS options set with the older version.

See also

- Chapter 1.3.1.25.2.8.10 “Command ‘Options’” on page 959
- Chapter 1.3.1.25.2.8.11 “Command ‘Import and Export Options’” on page 959
- Chapter 1.3.1.25.3.1 “Dialog ‘Import Assistant’” on page 1039
- Chapter 1.3.1.1.2.1 “Customizing menus” on page 103
- Chapter 1.3.1.1.2.4 “Customizing keyboard shortcuts” on page 106
- Chapter 1.3.1.1.2.2 “Customizing toolbars” on page 104

1.3.1.1.2 Customizing the user interface

In CODESYS, you can customize the user interface by changing the window layout as well as the appearance of menus and commands according to your requirements.
**Customizing menus**

You can customize the menu commands of the CODESYS user interface. In a configuration dialog, you can add or remove menus.

**Removing menus and commands**

1. Choose the command “Tools ➔ Customize”.
   ⇒ The “Customize” dialog box opens. The “Menu” tab is visible.
2. Select a menu in the menu tree or a command in a menu.
3. Click “Delete”.
   ⇒ The menu or command is deleted from the menu tree.
4. Click “OK”.
   ⇒ The dialog box closes and the menu is customized.

**Adding menus**

1. Choose the command “Tools ➔ Customize”.
   ⇒ The “Customize” dialog box opens. The “Menu” tab is visible.
2. Scroll to the end of the menu tree.
3. Select the blank symbol ( ).
4. Click “Add Popup Menu”.
   ⇒ The “Add Popup Menu” dialog box opens.
5. Type a name for the new menu in the “Default text” field.
   If localization is unnecessary, then skip to step 9.
6. Click “Add Language”.
   ⇒ A drop-down list opens with available languages.
7. Choose the required language.
   ⇒ The language is added to the list of languages.
8. Click into the “Text” field and type the language-specific text.
9. Click “OK”.
   ⇒ The new menu is added at the bottom of the menu tree.
10. Change the menu order by clicking “Move up” and “Move down”. Click “OK” to close the “Customize” dialog box.

   *The new menu is displayed only when it contains a command.*

**Adding commands**

1. Choose the command “Tools ➔ Customize”.
   ⇒ The “Customize” dialog box opens. The “Menu” tab is visible.
2. Expand the branch of the menu where the new command should be added.
3. Select the blank symbol (□).
4. Click “Add Command”.
   ⇒ The “Add Command” opens dialog box.
      The dialog box lists all commands grouped by category.
5. Select the command to be added. Click “OK”.
   ⇒ The new command is added to the menu tree.
6. Change the menu order by clicking “Move up” and “Move down”. Click “Add separator” to add a border between commands. Click OK to close the “Customize” dialog box.
   ⇒ The new command is now available in the menu.

See also
● § Chapter 1.3.1.25.3.14.1 “Dialog ‘Customize’ - ‘Menu’” on page 1089
● § Chapter 1.3.1.1.2.2 “Customizing toolbars” on page 104

Customizing toolbars
You can customize the toolbars of the CODESYS user interface. In a configuration dialog, you can add or remove toolbars.

Removing toolbars and commands
1. Choose the command “Tools ➤ Customize”.
   ⇒ The “Customize” dialog box opens.
2. Choose the “Toolbars” tab.
3. Select a toolbar or a command from a toolbar tree.
4. Click “Delete”.
   ⇒ The toolbar or command is deleted.
5. Click “OK”.
   ⇒ The dialog box closes and the toolbar or command is removed.

Adding toolbars
1. Choose the command “Tools ➤ Customize”.
   ⇒ The “Customize” dialog box opens.
2. Choose the “Toolbars” tab.
3. Select the blank toolbar.
4. Click “Add Toolbar”.
   ⇒ The cursor blinks in the new toolbar.
5. Type a name.
6. Change the toolbar order by clicking “Move up” and “Move down”. Click “OK” to close the “Customize” dialog box.

CODESYS displays the new toolbar only when it contains a command.

Adding commands

1. Choose the command “Tools → Customize”.
   - The “Customize” dialog box opens.
2. Choose the “Toolbars” tab.
3. Expand the tree of the toolbar where the new command should be added.
4. Select the blank symbol (□).
5. Click “Add Command”.
   - The “Add Command” dialog box opens.
   - The dialog box lists all commands grouped by category.
6. Select the command to be added. Click “OK”.
   - The new command is added to the toolbar tree.
7. Change the toolbar order by clicking “Move up” and “Move down”. Click “Add separator” to add a border between commands. Click “OK” to close the “Customize” dialog box.
   - The new command is available in the toolbar.

See also
- § Chapter 1.3.1.25.3.14.3 “Dialog 'Customize' - 'Toolbars'” on page 1090
- § Chapter 1.3.1.1.2.1 “Customizing menus” on page 103

Customize command icon

CODESYS provides the capability of assigning customized icons to commands.

1. Select the command “Tools → Customize”.
   - The “Customize” dialog box opens.
2. Click the “Command icons” tab.
3. Select the category “Help” from the list on the left.
   - All commands in this category are listed on the right.
4. Select the command “Information”.
5. Click “Assign”.
   - A dialog box opens for selecting the icon file (*.ico).
7. Click the “Open” button.
   - The icon is assigned to the selected command.
8. Click “OK”.
Customizing keyboard shortcuts

CODESYS provides the capability of executing commands directly via keyboard shortcuts. You can customize or extend predefined keyboard shortcuts.

1. Choose the command “Tools ➔ Customize”.
   ⇒ The “Customize” dialog box opens.
2. Choose the “Keyboard” tab.
3. Select the category “Help” from the list on the left.
   ⇒ All commands in this category are listed on the right.
4. Select the command “Search”.
5. Click into the field “Press Shortcut Keys”.
6. Press `[Ctrl]+[Shift]+[S]`.
   ⇒ CODESYS adds the key combination to the field.
7. Click “Assign”.
   ⇒ The keyboard shortcut is assigned to the command.
8. Click “OK”.
   ⇒ You can call the “Search” command by pressing `[Ctrl]+[Shift]+[S]`.

See also
● Chapter 1.3.1.25.3.14.4 “Dialog Box ‘Customize’ - ‘Keyboard’ ” on page 1091

Changing the window layout

In CODESYS, you can easily customize the layout of different views to your individual needs.

1. Drag the view by the caption bar or by the tab.
   ⇒ Arrows are shown to mark possible destinations. Example: △
2. Drag the view to one of the arrows.
   ⇒ The destination is displayed as a blue-shaded area.
3. Release the left mouse button.
   ⇒ The window is inserted into the selected destination.

The window can also be placed outside of the CODESYS programming interface.
Resizing windows

1. Move the mouse pointer over the border between two windows or views.
   ⇒ The cursor becomes a left-right arrow.
2. Drag the border to another position.

   You can resize detached views by moving the frame lines.

See also

- Chapter 1.3.1.1.2.6 “Resizing windows” on page 107
- Chapter 1.3.1.1.2.7 “Auto-hiding windows” on page 107
- Chapter 1.3.1.1.2.8 “Switching between windows” on page 108

Auto-hiding windows

Hiding windows  When you hide a view, it is minimized to a tab in the frame of the user interface. When you move the pointer over the tab, the window is shown automatically.

1. Click into the window to be hidden.
2. Click “Window ➔ Auto Hide”.
   Or click the pin symbol (マーク) in the upper right corner of the view.
   ⇒ The window is hidden and only visible by a small tab on the edge of the main window.
3. Move the mouse pointer over the tab.
   ⇒ The window is shown as long as the mouse pointer hovers over the tab.

Showing windows

1. Click the tab of the hidden window.
2. Clear the check box “Window ➔ Auto Hide”.
   Or click the pin symbol (マーク) in the upper right corner of the view.
   ⇒ The window is permanently shown.

See also

- Chapter 1.3.1.1.2.5 “Changing the window layout” on page 106
- Chapter 1.3.1.1.2.7 “Auto-hiding windows” on page 107
- Chapter 1.3.1.1.2.8 “Switching between windows” on page 108
Switching between windows

It is possible to switch directly between the currently opened views and the editor windows.

1. Press the keystroke combination $\text{[Ctrl]}+\text{[Tab]}$. Continue pressing the $\text{[Ctrl]}$ key.
   - An overview opens with all active views and editors.
2. Continue pressing the $\text{[Ctrl]}$ key and select a window using the arrow keys.
3. Release the $\text{[Ctrl]}$ key.
   - The selected view or editor is activated.

See also
- Chapter 1.3.1.1.2.5 “Changing the window layout” on page 106
- Chapter 1.3.1.1.2.6 “Resizing windows” on page 107
- Chapter 1.3.1.1.2.7 “Auto-hiding windows” on page 107

1.3.1.2 Your first CODESYS program

Data security: In order to minimize the risk of data security breaches, we recommend the following organizational and technical measures:

- Avoid access to the PLC and control networks from open networks and the Internet.
- Use a VPN for remote access.
- Install a firewall.
- If you want to publish your visualization on the web, then set a password to prevent unauthorized access.
- Use the latest versions of gateway servers and web servers.

Install CODESYS

The installation includes the development system and the CODESYS Gateway and CODESYS Control Win V3, whose services are accessible in the Windows taskbar. These three programs are necessary in order to simulate a controller on your computer.

Contents of your first project

In this tutorial, you will program a simple refrigerator controller. The finished project RefrigeratorControl.project_archive can be found in the CODESYS installation directory in the Projects directory. In addition to the sample project, which you will create here step by step, the finished project contains a complete visualization with operation and diagnosis.

- As with a conventional refrigerator, the temperature is specified by the user via a rotary control.
- The refrigerator determines the actual temperature using a sensor. When it is too high, the refrigerator starts the compressor with an adjustable delay.
- The compressor cools until the desired temperature is reached, minus a hysteresis of 1 degree. The hysteresis is intended to prevent the actual temperature from fluctuating too much around the set temperature, which would result in the compressor constantly switching itself off and on.
- When the door is open, a lamp lights up inside the refrigerator. When the door is open too long, a beeping acoustic signal sounds.
- If the compressor does not reach the set temperature despite activity of the motor over a long period of time, then the buzzer emits a steady acoustic signal.

Engineering:
The cooling activity is controlled in the main program of the application. The signal management is controlled in another POU. The required standard function blocks are available in the Standard library. Because no physical temperature sensors and no physical actuators are connected in this sample project, you will also write a program to simulate the increase and decrease of the temperature. This will allow you to monitor the operation of the refrigerator controller afterwards in online mode.

Variables that are to be used by all POUs need to be defined in a global variable list.

**Preparation**

You have installed the CODESYS Development System and launched it with the default profile "CODESYS V<current version>". The frame window of the development system opens with the standard menu bar: No project is open yet.

**Creating the project and selecting the PLC device**

1. Click “File ➔ New Project”.
2. In the “Templates” view, select the “Standard project” template.
3. Specify a name and a storage location for the project and click “OK”.
   ⇨ The “Standard Project” dialog opens for entering the device type and the implementation language of the main program.
4. In the “Device” list box, select “CODESYS Control Win V3”.
5. In the “PLC_PRG in” list box, select “Continuous Function Chart (CFC)” and click “OK”.
   ⇨ The project opens in the CODESYS frame window.

On the left-hand side of the frame window, in the “Devices” view, you will see the device tree (“Devices” view). The previously selected PLC device is displayed with the default name “Device”.

The “Application” object for the application that you will program already exists below the “PLC Logic” object.

“Application” already contains an object for the main program PLC_PRG and the “Library Manager”.

The “Library Manager” already contains the libraries IoStandard and Standard. The library IoStandard is required for I/O configurations. The library Standard contains all functions and function blocks that are described by the IEC 61131-3 standard.

The “Task Configuration” object is located at the bottom of the device tree with the MainTask, which controls the processing of PLC_PRG. Visualization objects, for example, can be inserted below the “Task Configuration” at a later time.

In addition to the libraries IoStandard and Standard, you need the library Util for this sample project:

1. Double-click the “Library Manager” object in the device tree.
   ⇨ The Library Manager opens in its editor.
2. In the header bar of the editor, click “Add Library”. In the “Add Library” dialog, select the library Util.
   Click “OK” to confirm the selection.
   ⇨ The library Util is added to the Library Manager, and the POUs from the library are available for use in the project.
To program the control application in the project, see the following sections:

**Declaring the global variables**

First, declare the variables that you want to use across the entire application. To do this, create a global variable list below “Application”:

1. Select **Application** and click “Add Object ➔ Global Variable List” in the context menu. Change the default name GVL to Glob_Var and click “Add” to confirm.
   - The **Glob_Var** object appears below **Application**. The GVL editor opens to the right of the device tree.

2. When the textual view opens, it already contains the keywords **VAR_GLOBAL** and **END_VAR**. For our example, click **VAR_GLOBAL** in the right sidebar of the editor to activate the tabular view.
   - An empty row appears. The cursor is in the “Name” column.

3. In the context menu, click “Insert”.
   - An input field opens. At the same time, the “Scope” **VAR_GLOBAL** and the “Data type” **BOOL** are entered in the line automatically.

4. Specify rTempActual in the “Name” field.
5. Double-click in the field of the “Data Type” column.
   - Now the field can be edited and the button appears.

6. Click and then select “Input Assistant”.
   - The “Input Assistant” dialog opens.

7. Select the data type **REAL** and click “OK”.
8. Enter a numerical value in the “Initialization” column (example: 8.0).

Declare the following variables in the same way:

<table>
<thead>
<tr>
<th>Name</th>
<th>Data type</th>
<th>Initialization</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>rTempActual</td>
<td>REAL</td>
<td>1.0</td>
<td>Actual temperature</td>
</tr>
<tr>
<td>rTempSet</td>
<td>REAL</td>
<td>8.0</td>
<td>Set temperature</td>
</tr>
<tr>
<td>xDoorOpen</td>
<td>BOOL</td>
<td>FALSE</td>
<td>Status of door</td>
</tr>
<tr>
<td>timAlarmThreshold</td>
<td>TIME</td>
<td>T#30S</td>
<td>Time after compressor runs that a signal sounds</td>
</tr>
<tr>
<td>timDoorOpenThreshold</td>
<td>TIME</td>
<td>T#10S</td>
<td>Time after opening the door that a signal sounds</td>
</tr>
<tr>
<td>xCompressor</td>
<td>BOOL</td>
<td>FALSE</td>
<td>Control signal</td>
</tr>
<tr>
<td>xSignal</td>
<td>BOOL</td>
<td>FALSE</td>
<td>Control signal</td>
</tr>
<tr>
<td>xLamp</td>
<td>BOOL</td>
<td>FALSE</td>
<td>Status message</td>
</tr>
</tbody>
</table>

You will now describe the main function of the application program in the main POU **PLC_PRG**, which is created by default. The compressor is activated and cools when the actual temperature is higher than the set temperature plus a hysteresis. The compressor switches off when the actual temperature is less than the set temperature minus the hysteresis.

To describe this functionality in the CFC implementation language, follow the steps below:
1. Double-click **PLC_PRG** in the device tree.
   - The CFC editor opens in the **PLC_PRG** tab. The declaration editor appears above the graphical editor area in textual or tabular form. The “**ToolBox**” view is on the right side.

2. In the “**ToolBox**” view, select the “**Input**” element and drag it to a point in the CFC editor.
   - The nameless entry ??? was inserted.

3. In the CFC editor, click ??? for the input. Then click to open the input assistance. In the “**Variables**” category, select the variable rTempActual below “**Application**” -> **Glob_Var**. This is how to reference the global variable rTempActual here.
   - The input name is **Glob_Var.rTempActual**.

4. As in Step 3, create another input with the name of the global variable **Glob_Var.rTempSet**.

5. Create another input, and then click ??? and replace them with the name **rHysteresis**.
   - Because this is not the name of a known variable, the “**Declare Variable**” dialog opens. The name is already used in the dialog.

6. In the “**Declare Variable**” dialog, specify “**Data Type**” with **REAL** and “**Initialization value**” with **1**. Click “**OK**”.
   - The variable **rHysteresis** appears in the declaration editor.

7. In the “**ToolBox**” view, select “**Box**” element and drag it to a point in the CFC editor.
   - The POU opens in the CFC Editor.

8. Replace ??? with **ADD**.
   - The POU adds all inputs that are connected to it.

9. Connect the **Glob_Var.rTempSet** input to the **ADD** POU. To do this, click the output pin of the input and drag it to the upper input pin of the **ADD** POU.

10. In the same way, connect the input **rHysteresis** with the lower input of the **ADD** POU.
    - The two inputs **rHysteresis** and **Glob_Var.rTempSet** are now added by **ADD**.

11. To move an element in the editor, click an empty space in the element or click the the frame, so that the element is selected (red border, red highlight). Drag the element to the desired position.

12. Create another POU to the right of the **ADD** POU. Its purpose is to compare **Glob_Var.rTempActual** with the sum of **Glob_Var.rTempSet** and **rHysteresis**. Assign the **GT** function (greater than) to the POU. 
    - The **GT** POU works as follows: IF (upper input > lower input) THEN
    - output := **TRUE**;

13. Connect the input **Glob_Var.rTempActual** to the upper input of the **GT** POU.

14. Connect the output of the **ADD** POU to the lower input of the **GT** POU.

15. Now create a function block to the right of the **GT** POU that starts and stops the cooling compressor depending on the input conditions (Set – Reset). Specify the name **SR** in the ??? field. Press the enter key to close the open input field above the POU (**SR_0**).
    - The “**Declare Variable**” dialog opens.

16. Declare the variable with the name **SR_0** and the data type **SR**. Click “**OK**”.
    - The **SR** POU from the **Standard** library is instantiated. **SR** is used for the definition of **THEN** at the output of the **GT** POU. The inputs **SET1** and **RESET** appear.
17. Connect the output pin on the right of the GT POU to the SET1 input of the SR_0 POU.
   ✓ SR can set a Boolean variable from FALSE to TRUE and back again. When the condition at the input SET1 is satisfied, the Boolean variable is set to TRUE. The variable is reset again when the condition at RESET is satisfied. The Boolean (global) variable in our example is Glob_Var.xCompressor.

18. Create an “Output” element and assign it to the global variable Glob_Var.xCompressor. Drag a connecting line between Glob_Var.xCompressor and the output pin Q1 of SR.

Now specify the condition under which the compressor should switch off again (in this case, the RESET input of the SR POU gets a TRUE signal). To do this, formulate the opposite condition to that above. Use the SUB (subtraction) and LT (less than) POUs to do this.

The result is the following CFC chart:

Creating a POU for signal management in the ladder diagram editor

In another POU, you will now implement the signal management for the alarm buzzer and for switching the lamp on and off. The ladder diagram (LD) implementation language is suitable for this.

Handle each of the following signals in their own networks:

- When the compressor runs too long because the temperature is too high, a continuous acoustic signal sounds.
- When the door is open too long, an intermittent signal sounds.
- The light is on as long as the door is open.

1. Below the Application in the device tree, create a POU object of type “Program” using the “Ladder Diagram (LD)” implementation language. Specify the name Signals for the program.

   ✓ Signals is listed in the device tree next to PLC_PRG. The ladder diagram editor opens in the Signals tab. The declaration editor appears in the upper part, the “ToolBox” view to the right. The LD contains an empty network.

2. Create a program in the network so that an acoustic signal sounds when the compressor runs too long without reaching the temperature setpoint. To do this, insert a TON timer POU. It switches a Boolean signal to TRUE only after a specified time. Select a TON below “Function Blocks” in the “ToolBox” view and drag it to the empty network and to the “Start here” box which appears. Release the mouse button when the field turns green.

   ✓ The POU appears as a box with inputs and outputs and is automatically assigned the instance name TON_0. The line editor is open and the cursor is blinking there.
   ⊳ You have confirmed the instance name. The “Declare Variable” dialog opens.

4. Click “OK” to confirm the dialog.
   ⊳ Now the inserted POU `TON` is instantiated with the name `TON_0`. Moreover, the top input is displayed by default as a contact preceding the POU.

   To read the help for the function block `TON`, place the cursor in/on the POU in the "TON" string and press [F1].

5. Now you program so that the function block is activated as soon as the cooling compressor starts to run. For this purpose, name the contact at the upper input of the POU `Glob_Var.xCompressor`. You have already declared this Boolean variable in the GVL `Glob_Var`.

   When you begin to type a variable name at the input position, you automatically get a list of all variables with names that begin with the typed characters and can be used at this point. This assistance is a default setting in the CODESYS options for SmartCoding.

6. Insert the signal that is to be activated. To do this, drag a “Coil” from the “Ladder Diagram Elements” toolbox category to the output `Q` of the `TON` POU. Specify the name `Glob_Var.xSignal` for the coil.

7. Define the time period from the activation of the POU `TON_0` until the signal should sound. This definition takes place via the variable `Glob_Var.timAlarmThreshold`, which you insert for this purpose at the input `PT` of `TON_0`. To do this, click the fine-edged box to right of the input pin and enter the variable name.

8. Select the POU `TON` and click “Remove unused FB call parameters in the context menu”.
   ⊳ The unused output `ET` is removed.

9. In the second network of the LD, program so that the signal sounds intermittently when the door is open too long.
   In the editor window, click below the first network. In the context menu, click “Insert network”.
   ⊳ An empty network with the number “2” appears.

10. As in the first network, implement a POU `TON` for time-controlled activation of the signal. This time it is triggered by the global variable `Glob_Var.xDoorOpen` at the input `IN`. At the input `PT`, add the global variable `Glob_Var.timDoorOpenThreshold`.

11. In addition, from the library `Util`, add a POU `BLINK` at the output `Q` of the POU `TON` in this network and instantiate it as `Blink_0`.

12. The POU `BLINK_0` clocks the signal forwarding `Q` and therefore `Glob_Var.xSignal`.
    First, drag two “Contact” elements from the “ToolBox” view to the `OUT` output of the POU. Assign the variable `TON_1.Q` to the contact directly after the output `Q` and the global variable `Glob_Var.xDoorOpen` to the second contact.

13. Insert a “Coil” element after the two contacts and assign the global variable `Glob_Var.xSignal` to it.

14. To do this, declare the local variable `timSignalTime : TIME := T#1S;` and insert this variable at the inputs `TIMELOW` and `TIMEHIGH`. The cycle time is 1 second for `TRUE` and 1 second for `FALSE`.

15. Select the POU `TON` and click “Remove unused FB call parameters in the context menu”.
   ⊳ The unused output `ET` was removed.
16. In the third network of the LD, program so that the lamp lights up as long as the door is open. To do this, insert another network. In that network on the left, insert a contact `GlobVar.xDoorOpen`, which leads directly to an inserted coil `Glob_Var.xLamp`.

17. CODESYS processes the networks of an LD in succession. Now install a jump to Network 3 at the end of Network 1 in order to ensure that either only Network 1 or only Network 2 is executed:

   Select Network 3 by clicking in the network or in the field with the network number. In the context menu, click “Insert Label”. Replace the text Label: in the upper left part of the network with `DoorIsOpen`:

   Select Network 1. In the “ToolBox” view, in the “Common” category, drag the “Jump” element to the network. Place it on the “Add output or jump here” box which appears.

   ↩️ The jump element appears. The jump destination is still specified as `???`.

18. Select `???` and click ⬅️. Select `DoorIsOpen` from the possible label identifiers and click “OK” to confirm.

   ↩️ The label to Network 3 is implemented.

The LD program now looks like this:

![LD program image]

Calling the Signals program in the main program

In our program sample, the main program PLC_PRG should call the Signals program for signal processing.
1. In the device tree, double-click **PLC_PRG**.
   - The **PLC_PRG** program opens in the editor.
2. In the “ToolBox” view, drag a “Box” element to the editor of **PLC_PRG**.
3. Using the Input Assistant, add this POU from the “POU Calls” category to the call of the Signals program.

**Creating an ST POU for a simulation**

Because the application in this example project is not linked to physical sensors and actuators, you also need to write a program for the simulation of increase and decrease in temperature. This will allow you to monitor the operation of the refrigerator controller afterwards in online mode.

You create the simulation program in structured text.

The program increases the temperature until the main program **PLC_PRG** determines that the set temperature has been exceeded. Then the program activates the cooling compressor. Then the simulation program decreases the temperature until the main program deactivates the compressor again.
1. Below the application, insert a POU of type “Program” in the “ST” implementation language. Specify the name Simulation.

2. Implement the following code in the ST editor:

```plaintext
PROGRAM Simulation
VAR
    TON_1: TON;                            //The temperature is decreased on a time delay, when the compressor has been activated
    P_Cooling: TIME:=T#500MS;
    xReduceTemp: BOOL;                  //Signal for decreasing
    TON_2: TON;                            //The temperature is increased on a time delay, when the compressor has been activated
    P_Environment: TIME:=T#2S;          //Delay time when the door is closed
    P_EnvironmentDoorOpen: TIME:=T#1S;  //Delay time when the door is open
    xRaiseTemp: BOOL;                   //Signal for increasing
    timTemp: TIME;                      //Delay time
    iCounter: INT;
END_VAR

iCounter := iCounter + 1;       // No function, just for demonstration purposes.

// After the compressor has been activated due to TempActual being too high, the temperature decreases.
// The temperature is decremented by 0.1°C per cycle after a delay of P_Cooling
IF Glob_VAR.xCompressor THEN
    TON_1(IN:= Glob_Var.xCompressor, PT:= P_Cooling, Q=>xReduceTemp);
    IF xReduceTemp THEN
        Glob_Var.rTempActual := Glob_Var.rTempActual-0.1;
    TON_1(IN:=FALSE);
END_IF
END_IF

//If the door is open, the warming occurs faster; SEL selects P_EnvironmentDoorOpen
timTemp:=SEL(Glob_Var.xDoorOpen, P_Environment, P_EnvironmentDoorOpen);

//If the compressor is not in operation, then the cooling chamber becomes warmer.
//The temperature is incremented by 0.1°C per cycle after a delay of timTemp
TON_2(IN:= TRUE, PT:= timTemp, Q=>xRaiseTemp);
IF xRaiseTemp THEN
    Glob_Var.rTempActual := Glob_Var.rTempActual + 0.1;
    TON_2(IN:=FALSE);
END_IF
```

PLC Automation with V3 CPUs
Programming with IEC 61131-3 editor > CODESYS Development System

2020/12/10
3ADR010583, 1, en_US
116
We recommend that you use a visualization for convenient operation and monitoring of the entire control program. A visualization created with CODESYS is installed in the completed sample project for this tutorial, which is provided with the standard CODESYS Visualization installation (Projects directory). You can download this project to the controller and start it to see it working together with the visualization. When started, the Live_Visu begins with a representation of the refrigerator which reproduces the operation of the simulation program without you having to input any data. However, you can open and close the door by clicking the on/off switch. It is possible to adjust the default temperature by means of the needle of the rotary control. We will not cover the creation of the visualization in this tutorial. An corresponding tutorial is planned in the help for CODESYS Visualization.

Defining the programs to be executed in the task configuration

The default task configuration contains the call for the main program PLC_PRG. For our sample project, you also need to add the call for the Simulation program.

1. In the device tree, drag the Simulation entry to MainTask below the “Task Configuration”:
   - The Simulation program is inserted into the task configuration.

2. To view the task configuration, double-click the MainTask entry to open the editor.
   - In the table in the lower part of the editor, you see the POUs that are called by the task: PLC_PRG (entered by default) and Simulation. The call type of the task is “Cyclic” at intervals of 20 milliseconds. In online mode, the task will execute the two POUs one time per cycle.

Defining the "active application" for the communication with the PLC

The name of the Application is displayed in bold fonts in the “Devices” view. This means that this application is set as the "active application". The communication with the controller then refers to this application.

When there is only one application in a project, it automatically becomes the active application. If your application is not active yet, then activate it as follows:

▷ In the context menu of Application, click “Set Active Application”.
   - “Application” is now displayed in bold in the “Devices” view.

Debugging the application program

When you are inputting the code, CODESYS immediately alerts you to syntax errors by means of a wavy red underline below the text in question. Press [F11] to check the syntax of the entire application. The result of the check is shown in the message view. If necessary, click “View Messages” to open the message view. Then you can select a message and press the [F4] key to jump to the corresponding point in the code.

Afterwards, only an error-free application can be downloaded to the controller.
Starting the gateway server and PLC

CAUTION!
Check controller accessibility
For security reasons, controllers should not be accessible from the Internet or untrusted Networks under any circumstances! In particular, the TCP/IP programming ports (usually UDP-Ports 1740..1743 and TCP-Ports 1217 + 11740 or the controller specific ports) should not be accessible from the internet without protection. In case Internet access to the controller is needed, using a a safe mechanism is absolutely mandatory, such as VPN and password protection of the controller.

see also: Chapter 1.3.1.11.3 “Handling of device user management” on page 300

Starting the gateway server:
The program GatewaySysTray is installed as standard with CODESYS. You can communicate with the gateway server by means of this program. The gateway server is started automatically as a service when the system starts. Check that the Windows taskbar contains the program icon . When the gateway server is not running, the icon looks like this: . In this case, you can click the icon to open the gateway menu and click “Start Gateway”.

Starting the PLC:
The program CODESYSControlSysTray is installed by default with CODESYS. You can communicate with the CODESYS Control Service by means of this program. In CODESYS V3.5 SP2 and later, the control service is no longer automatically started on starting up the system. This is to protect against unauthorized access. Start the PLC as follows: In the Windows taskbar, click to open the PLC menu. Then click “Start PLC”. If the PLC is running, then the icon changes to . A dialog that appears at startup indicates that the started PLC allows programming access. Pay attention to the security notice above.

In the very first communication configuration: Adding the gateway

▷ In the “Devices” view, double-click Device (CODESYS Control Win V3).

◇ The “Communication Settings” tab opens in the Device editor.

If this is your first communication configuration with CODESYS V3, then now you need to define the local gateway server. If you have already defined the gateway server, then it will be shown on the “Communication Settings” tab. In this case, you can now continue to " Defining the communication channel" on page 120".

The gateway server is provided with the CODESYS installation.
1. Click “Gateway ➔ Add New Gateway”.
   ⇒ The “Gateway” dialog opens.

2. In the “Name” input field, specify a symbolic name for the gateway.
3. In the “Drivers” list box, select “TCP/IP”.
4. Double-click the right column of the “IP address” row and specify localhost in the input field.
5. Click “OK”.

⇒ The gateway is entered on the “Communication Settings” tab (1) of the device editor. When the gateway is running correctly, a filled green circle appears on the graphic of the gateway:

![Gateway graphic with green circle]

Defining the communication channel

Now define the communication channel to the device, which is then used via the gateway that you have set. To do this, double-click “Device” in the device tree to open the “Communication Settings” tab of the device editor.

1. Click “Scan Network” to search the local network for all available devices.

⇒ The “Select Device” dialog (1) appears with a list of all devices with which you can establish a connection.

![Select Device dialog]

2. Select the tree entry with the name of your computer.
3. Click “OK”.
   ⇒ The channel is now active and the associated information appears below the device graphic on the “Communication Settings” tab.

All communication actions now refer exactly to this channel. Remember this later if you have multiple communication channels in the project.

For the application on the PLC, see the following sections:

**Downloading the application to the PLC**

Requirement: The application has been compiled without errors. See "Debugging the application program" on page 117.

1. Click “Online ➔ Login”.
   ⇒ A dialog prompt opens to ask whether or not the application should be downloaded to the controller.
2. Click “Yes”.
   ⇒ The application is downloaded to the controller. The entries for the controller and the application in the “Devices” view are highlighted in green. Stop appears after the “Application” object. The current status of the controller appears in the taskbar:

   ![Stop](image)

   Run appears after the “Application” object.

Starting the application

If you have followed this tutorial precisely up until now, then you can use the “Application” on the PLC “Device (CODESYS Control Win V3”).

   ⇒ In the “Devices” view, in the context menu of the “Application” object, click “Start”.

   ⇒ The program starts running. The entries for the controller and the application in the “Devices” view are highlighted in green. Run appears after the “Application” object. The current status of the controller appears in the taskbar:

   ![Run](image)

**Monitoring and one-time writing of variable values at runtime**

In the following section, you can view the "monitoring" of the variable values in the various POUs, and you can set a specific variable value on the controller one time only from CODESYS.

You can see the actual values of the application variables in the online views of the POU editors or in monitoring lists. In the example here, we will focus on monitoring in the POU editor.

Requirement: The application is running on the controller.
1. In the device tree, double-click the objects PLC_PRG, Signals, Simulation, and Glob_Var to open the online views of the editors.

In the declaration part of each view, the actual value of the variable (1) on the controller appears in the table of expressions in the “Value” column (3) (see figure).

Monitoring in the implementation part depends on the implementation language. In the case of non-Boolean variables, the value is always located in a rectangular field to the right of the identifier. In the ST editor, this also applies to Boolean variables. We call this display "Inline monitoring". In the graphical editors, the value of a Boolean variable is shown by the color of the output connecting line (black for FALSE and blue for TRUE):
2. Observe how the variable values change in the various POUs. For example, you can see in the GVL Glob_Var how the values of rTempActual and xCompressor change due to the processing of the simulation program.

One-time setting of variable values on the controller:
1. Set the focus in the online view of the GVL Glob_Var.
2. To set a new specified value, double-click in the “Prepared Value” column (2) next to the expression rTempSet.
   ⇒ An input field opens.
3. Specify the value 9 and exit the input field.
4. To specify an open door, click once in the “Prepared Value” field next to the expression xDoorOpen. The value TRUE is specified. Click three more times to see that you can toggle the prepared value between TRUE, FALSE, and empty.
5. To write the prepared value TRUE one time only to the variable, press [Ctrl]+[F7].
   ⇒ The two values are each transferred to the “Value” column (3). Now the variable xDoorOpen no longer changes its value and the set temperature is 9 degrees. The variable timTemp changes to the value 1s because the refrigerator door is now “open” and the heating due to the Simulation should be faster than before (2s).

Setting of breakpoints and stepping at runtime

Debugging: For troubleshooting purposes, you need to check the variable values at certain points in the code. You can define breakpoints for the execution and initiate a step-by-step execution of the statements.

Requirement: The application has been downloaded to the controller and is running.
1. Double-click Simulation to open the program in the editor.
2. Place the cursor in the code line iCounter := iCoutner + 1; and press [F9].
   ⇒ The symbol ● is displayed before the line of code. It indicates that a breakpoint is set on this line. The symbol changes immediately to ●. The yellow arrow always points to the next statement to be processed. HALT ON BP. appears in the taskbar instead of RUN.
3. Observe the value of the variable iCounter in the inline monitoring or in the declaration part of the Simulation program.
   ⇒ The value of the variable no longer changes. The processing was stopped at the breakpoint.
4. Press [F5] to restart the processing.
   ⇒ After one cycle, the program stops again at the breakpoint. iCounter was incremented by 1.
5. Press [F8] to execute the next processing step.
   ⇒ The “RETURN” at the end of the line iCounter := iCounter + 1; statement is highlighted in yellow.
6. Press [F8] again to execute the next processing step.
   ⇒ The processing jumps to the editor of PLC_PRG. Repeatedly pressing [F8] shows how the program is executed step-by-step. The statement to execute is once again marked each time with a yellow arrow.
7. To disable the breakpoint and return to normal processing, set the cursor back in the code line and press [F9]. Then press [F5] to set the application back to RUN mode.
Requirement: The application has been downloaded to the controller and is running.

1. As above when stepping, monitor the line `iCounter := iCounter + 1;` in the Simulation program.

2. Press `[Ctrl]+[F5]` or click “Debug ➔ Single Cycle” to run a single cycle.
   ➔ The processing runs through one cycle and stops at the breakpoint again. `iCounter` was incremented by 1.

   ➔ The program runs again without stopping and without forced values. The `temp` variable has the value 1s again. RUN is displayed again in the status bar.

### Creating and configuring a project

#### What is a project?

- A project contains the objects that are necessary for the creation of a control program (`application`):
  - Pure POUs, for example programs, function blocks, functions, GVLs.
  - Objects that are additionally required in order to be able to execute the application on a PLC. For example, task configuration, library manager, symbol configuration, device configuration, visualizations, external files.
- You can program several applications and connect several control devices in a project.
- CODESYS manages device-specific and application-specific POUs in the “Devices” view (‘Device tree’), project-wide available POUs in the “POUs” view.
- For the creation of projects there are templates that already contain certain objects.
- Basic configurations and information for the project are defined in the “Project Settings” and the “Project Information”. For example:
  - Compiler settings
  - User management
  - Author
  - Data for the project file.

There are settings for the version compatibility of the project in the configuration dialog boxes for the “Project Environment”.

- You save a project as a file in the file system. Optionally you can pack it together with project-relevant files and information in a project archive. Saving in a source control system such as SVN is also possible
- Each project contains the information regarding the CODESYS version with which it was created. If you open it in another version, CODESYS will draw your attention to possible or necessary updates with regard to file format, library versions, etc.
- You can compare, export or import projects and create documentation for them.
- You can protect a project against change and completely, i.e. also against reading. By using a user management system you can purposefully control access to the project and even to individual objects in the project.

### Creating standard projects

1. Click “File ➔ New Project” and then the template “Projects ➔ Standard project”. Specify a name (example: `myProject`) and a location in the file system.
   ➔ The “Standard Project” dialog opens.
2. Select one of the standard devices CODESYS Control Win V3 from the "Device" list, and select "Structured Text (ST)" from the "PLC_PRG in" list. Then click "OK".

⇒ The project name myProject is listed in the title bar of the main window of the development system. The following structure is created in the device tree ("Devices" view).

![Device Tree Diagram]

Depending on the selected PLC, additional device nodes may also be shown.

The PLC is always displayed automatically with a "PLC Logic" subnode. The base objects below it are: an application with a "Library Manager" and a "Task Configuration". The task configuration defines the MainTask for processing the standard POU "PLC_PRG".

The library manager already contains Standard.library with base POUs (for example, counters, timers, and string functions) that can be used later when programming.

Now when you fill "PLC_PRG" with error-free code, you can download it to the controller and execute it without requiring any further programming objects.

In SP13 and later, the project setting "Integrity check" in "Project → Project Settings → Security" is activated automatically for a new created standard project. For detailed information on how to protect your project, please refer to the section "Protecting and Saving a Project".

See also
- Chapter 1.3.1.8.1 “Device tree and device editor” on page 155
- Chapter 1.3.1.3.2 “Adding objects” on page 125
- Chapter 1.3.1.9 “Programming of applications” on page 171
- Chapter 1.3.1.8 “Configuring I/O links” on page 155
- Chapter 1.3.1.9.8 “Using library POUs” on page 213
- Chapter 1.3.1.6 “Protecting and saving a project” on page 139
- Chapter 1.3.1.25.3.11.7 “Dialog 'Project Settings' - 'Security'” on page 1064

1.3.1.3.2 Adding objects

Requirement: A project is open. At least the highest node "<project name>" is shown in the "Devices" view ("Device tree") and in the "POUs" view.

The following instructions indicate a few possibilities when creating objects in the project.
1. Select an entry in the Device tree or in the POUs tree, for example the “Application” object.
2. Select the command “Project ➔ Add Object”.
   ⇒ Depending on the selected entry in the tree, CODESYS offers the suitable objects for selection.
3. For example, select the object “POU” and, in the dialog box “Add POU” which then appears, select the type “Program” with implementation language “Structured Text (ST)” and name “prog”.
   ⇒ After clicking on “Add”, CODESYS inserts a program object “prog” in the device tree below “Application”.
4. Select an object in the tree and select the command “Properties” in the context menu.
   ⇒ The dialog box “Properties” appears with the object-relevant categories. If you use a user management system you can, for example, restrict access to the object here.
5. Select an entry in the tree under which you would like to create a folder in order to collect certain objects in it. Please note: you cannot structure the arrangement of device nodes and device objects through folders that you have created yourself.
6. Select the command “Add Folder” in the context menu and, in the dialog box that appears, define a name for the new folder. Click on “OK”.
   ⇒ The folder appears in the tree.
7. Select an object in the tree and shift it by dragging it with the mouse within the tree to another position, for example into the folder.
8. Do not select an object in the Device tree; instead, set the focus on any desired free position in the CODESYS interface. Select the command “Project ➔ Add Object” and then, for example, the object “TextList”.
   ⇒ The object “Text List” is inserted in the “POUs” view!

See also
- § Chapter 1.3.1.3.1 “Creating standard projects” on page 124
- § Chapter 1.3.1.25.2.4.1 “Command ‘Add Object’” on page 897
- § Chapter 1.3.1.25.2.4.2 “Command ‘Add Folder’” on page 897
- § Chapter 1.3.1.25.2.3.22 “Command ‘Properties’” on page 896

### 1.3.1.3.3 Changing the compiler version

The version of the compiler with which code is generated in the current project for use on the target device is defined in the project settings.

The compiler version is independent of the CODESYS version. Hence, constant application code is created from the source code when the compiler version is set the same, even if this takes place from different CODESYS versions.

**NOTICE!**

Pay attention to the setting “Do not update” on the “Compiler Version” tab in the “Project Environment” dialog box: if this is activated and you open an older project, for which the “Newest” compiler version is still set, then CODESYS will continue to use the compiler version used last in the old project. This also applies if a different "current" version is defined in the new environment!
If you open a project in which the newest compiler version is not set, the “Project Environment” dialog box appears with a corresponding note and the possibility to update directly.

Requirement: A project is open.
1. Select the command “Project ➔ Project Settings” and then the “Compile Options” tab.
2. Select the desired “Fix Version” and click on “OK”.
   ⇒ The change is immediately effective.

See also
● § Chapter 1.3.1.11.4 “Generating application code” on page 303
● § Chapter 1.3.1.25.3.11.3 “Dialog Box ‘Project Settings’ - ‘Compile Options’” on page 1062
● § Chapter 1.3.1.25.3.12.2 “Dialog ‘Project Environment’ - ‘Compiler Version’” on page 1070

1.3.1.3.4 Opening a V3 project

Requirement: CODESYS is started (or a project is already open).
1. Click “File ➔ Open Project”.
2. In the “Open project” dialog, click any V3 project or project archive in the file system. A file filter is located in the lower right corner of the dialog. Note: If necessary, you can open the project as write protected in order to prevent any accidental changes to the existing project file.

If no other project is open, then the following cases are possible:

- You have selected a project that was saved with a later CODESYS version.
  This kind of project may include data that cannot be loaded. You can still open the project, but you have to understand that the project can behave unexpectedly because it cannot be interpreted completely. Objects that CODESYS could not load, either partially or completely, are highlighted in red in the device tree with the text “[unknown]” or “[incomplete]”. CODESYS cannot display objects in the editor that are unknown to the current version. CODESYS displays incomplete objects in the editor with a warning that the displayed contents may not correspond to the original. CODESYS cannot save incomplete projects as their original names. This is displayed by a write protection note in the upper right corner. You can save the file as another name.
- You have selected a project that CODESYS did not save properly after the last change, but the project option “Auto Save” was activated: The dialog “Auto Save Backup” opens for handling the backup.
- You have selected a project that is write-protected: You can either open the project in write-protected mode or remove the write protection.
- You have selected a project that is password-protected. You are prompted for the password.
- You have selected a project that is protected by a security key. If the security key is not plugged into the computer, then you are prompted to plug it in. Otherwise CODESYS opens the project without any information about the protection.
- You have selected a project that contains a package (add-on) or plug-in specific POU. In addition, the project was saved with a CODESYS installation that also had the relevant add-ons and plug-ins installed. However, the add-ons and plug-ins are not installed in the current CODESYS installation.
  A dialog opens with information that the project cannot be loaded with all POUs. The missing add-ons and plug-ins are listed with the version information. You can cancel opening the project and then perform an installation.
  ⇒ The “Project Environment” dialog opens.
3. Check the updates provided in the “Project Environment” dialog. For projects that were created with very old V3 versions, information may be displayed how to replace specific objects in order to continue using the project. Depending on whether you leave the project as it is or continue development under up-to-date conditions, you perform updates or you do not. After each update, a login without download or online change is no longer possible.

⇒ The project opens.

See also
- Chapter 1.3.1.3.5 “Opening a V2.3 project” on page 128
- Chapter 1.3.1.25.3.13.13 “Dialog ‘Options’ – ‘Load and Save’” on page 1083
- Chapter 1.3.1.6.1 “Setting up write protection” on page 142
- Chapter 1.3.1.6.2 “Assigning passwords” on page 143
- Chapter 1.3.1.25.2.1.2 “Command ‘Open Project’” on page 853

1.3.1.3.5 Opening a V2.3 project

A CoDeSys V2.3 project can be converted into a CODESYS V3 project only if the CODESYS V2.3 Converter package is installed in CODESYS V3. The package is available in the CODESYS Store.

Requirement: CODESYS is started (or a project is already open). You should be aware of the restrictions described below the following instructions.

1. Click “File ➔ Open Project”.

2. In the “Open Project” dialog, click any CoDeSys V2.3 project or project archive in the file system. For searching, you can set the file filter on the bottom right corner of the dialog.

⇒ If another project is still open, CODESYS instructs you to close it accordingly. After that the CoDeSys V2.3 converter automatically starts.

3. The V2.3 converter checks that the project can be compiled without errors. If so, then it processes the project automatically.

4. NOTE: If the project contains visualization objects with placeholder variables that the converter cannot resolve, the respective visualizations are shown as a group in place of the visualization references.

5. Device conversion: When a device (target system) is referenced in the project to be opened and no conversion rules are defined for the device, then the “Device Conversion” dialog opens. Specify here whether and how the converter should replace the previous device reference with a current one.

⇒ For replacement, the converter added the new device in the place of the old one in the device tree of the converted project.

6. Library conversion: if a library, for which no conversion rule has so far been defined, is referenced in the project to be opened, then the “Conversion of Library Reference” dialog opens. Specify here whether and how the converter should replace the existing library reference with a current one. If you select a library for which the project information is missing, then the “Enter Project Information” dialog opens in order to specify this information.

⇒ The converter loads the adapted project. Note: The redefined library references are to be found in the global Library Manager in the POUs view.
### Restrictions when reusing a CoDeSys V2.3 project in CODESYS

<table>
<thead>
<tr>
<th>Compilation:</th>
<th>The project has to be compilable without errors in CoDeSys V2.3. Note: CODESYS still issues warnings in V3 when compiling. These are caused by implicit conversions, which can lead to a loss of information (for example through sign changes). CODESYS checks &quot;case&quot; statements against the switch variable: CASE USINT OF INT is not checked in CoDeSys V2.3, but it issues an error message when imported into V3.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Controller configuration:</td>
<td>The &quot;Controller Configuration&quot; of a CoDeSys V2.3 project cannot be imported into V3. You have to recreate the device configuration and re-declare the variables used in the controller configuration.</td>
</tr>
<tr>
<td>Network variables:</td>
<td>For network variables, CODESYS creates V3 GVL objects and imports the variable declarations. However, the network properties are not imported. See the description of the network variable exchange for this.</td>
</tr>
<tr>
<td>Libraries:</td>
<td>All variables and constants that are used in a library also have to be declared in the library. It must be possible to compile the library in CoDeSys V2.3 without errors.</td>
</tr>
</tbody>
</table>
| Syntax and semantic restrictions since CoDeSys V2.3: | - FUNCTIONBLOCK is not a valid keyword instead of FUNCTION_BLOCK.  
- TYPE (declaration of a structure) must be followed by a ":".  
- ARRAY initialization** must have parentheses.  
- INI is no longer supported (you have to replace this in the code by the Init method).  
- In function calls it is no longer possible to mix explicit with implicit parameter assignments. Therefore the order of the parameter input assignments can be changed:  
  `fun(formal1 := actual1, actual2);`  // -> error message  
  `fun(formal2 := actual2, formal1 := actual1);`  // same semantics as the following line:  
  `fun(formal1 := actual1, formal2 := actual2);`  
- CoDeSys V2.3 pragmas are not converted. They produce an warning in V3.  
- The TRUNC operator now converts to the data type DINT instead of INT. CODESYS automatically adds a corresponding type conversion for a CoDeSys V2.3 import. |

### Visualization:

<table>
<thead>
<tr>
<th>Placeholders and their replacement</th>
<th>VAR_INPUT</th>
<th>Usage</th>
<th>Replacement</th>
</tr>
</thead>
<tbody>
<tr>
<td>Placeholders</td>
<td>VAR_INPUT</td>
<td>Usage</td>
<td>Replacement</td>
</tr>
<tr>
<td>PLC_PRG.$Local Var$.aArr[0] localVar: MyStruct; localVar.aArr[0] localVar := PLC_PRG.myStruc tVar</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$Var$.aArr[0] Var : MyStruct; Var.aArr[0] Var := PLC_PRG.myStruc tVar</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PLC_PRG.myStruc tVar.aArr[$Index$] Index : INT; PLC_PRG.myStruc tVar.aArr[Index] Index := 0</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Problematic placeholders

- **Placeholders within a text:**
  - Text: `$axle$-Axis`
  - Correction: `localVar : STRING;`
  - Text: `%s-Axis`
  - Text variable: `localVar`

- **Placeholder describes only one part of a variable name:**
  - Text: `axis$axis$spur$spur$.fActPosition`
  - Correction: Define only one placeholder for the `axis$axis$spur$spur$` placeholder.
  - axis_spur : MyFunctionBlock;
  - Then directly transfer the corresponding instance of the function block.
  - axis_spur := PLC_PRG.axis1spur2;

- **Placeholder is replaced by an expression:**
  - Text: `$Expression$ -> PLC_PRG.var1 + PLC_PRG.var2`
  - Correction: You must transfer the expression to an auxiliary variable and then transfer this auxiliary variable as an instance.

- **The placeholder describes a program name:** `$Program$.bToggle -> PLC_PRG.bToggle D`
  - The converter cannot transfer this form of setting placeholders in V3. However, you will rarely use it in practice.

- **Placeholder is replaced by different types:**
  - Text: `$Var$
  - Correction: Define two different placeholders in the interface for this.

- **The visualization is located in a library. You replace the placeholder later from any desired project when you use the visualization there.**
  - Correction: Here you have to replace the `TYPE.NONE` data types manually. However, there is also the possibility for you to integrate the library in a project and the placeholder is correctly replaced. If you now import this project, the data type is also determined correctly in the library.

### Non-importable elements:

- Trend, ActiveX – the import is not possible, because the implementation differs a great deal. In V3, a corresponding warning is issued and a corresponding manual reproduction is required.

### Programming languages

| ST, IL, FBD: | No restrictions |
CODESYS imports function blocks with parallel branches in such a way that the part before the branch is repeated for each branch. This corresponds to the generated code that CoDeSys V2.3 creates for parallel branches.

- **SFC:**
  - Step variables explicitly declared by the user must be declared locally in the SFC editor. You may not declare them as VAR_INPUT, VAR_OUTPUT or VAR_INOUT, because CODESYS cannot automatically adapt the calls. Explanation: Steps no longer use Boolean variables for the management of the internal states in V3, but also structures of the type SFCStepType.
  - Identifier: the following identifiers may not begin with an underscore character:
    - Names of IEC actions in the tree
    - Variables that are called in an IEC association list
    - Names of transitions that have been programmed out
  - Explanation: In V3 the implicit variables that CODESYS creates for actions are given an underscore character as prefix. An invalid identifier with a double underscore character would result.

- **CFC:**
  - Large boxes: The layout of large boxes can lose quality due to an import. The boxes may overlap one another too much. (Correction planned).
  - Macros: Macros cannot be imported. (Correction planned).

### See also
- Chapter 1.3.1.25.3.13.5 “Dialog ‘Options’ - 'CoDeSys V2.3 Converter’” on page 1077
- Chapter 1.3.1.25.1.31 “Object ‘Project Information’” on page 811
- Chapter 1.3.1.25.3.4 “Dialog ‘Device Conversion’” on page 1041
- Chapter 1.3.1.25.3.2 “Dialog ‘Library Reference Conversion’” on page 1040

### 1.3.1.3.6 Configuring a project

You can configure your CODESYS project using the following dialogs:
- "Project Settings": Basic settings on the behaviour of editors and of the compiler, on user management etc.
- "Project Information": Adding of individual and tagging information to the project
- "Project Environment": Defining which versions of the external and internal modules should be used, with the aim of achieving up-to-dateness and compatibility with each other.
Retrieving and editing project information

You can use the “Project Information” object to retrieve information about your project and the associated file, and edit certain information.

The object contains information about

- File attributes
- Meta-information, such as manufacturer, title, or author
- Properties with keys
- Statistics
- Licensing
- Signing: This way of signing translated libraries is deprecated, and for security reasons should only be used if compatibility with older versions is required. If this method is used, then later you can use a public key token to verify that the library was last signed by the library vendor. As a library vendor, it is therefore crucial that you make the public key used available to the customer, for example in the documentation.

CODESYS saves the project information as an object within the project. When you transfer a project to another system, the “Project Information” object is transferred with it. There is no need for a project archive.

You can use property keys to access the project information externally via function blocks. For a library project, you can also query information about the licensing.

Editing meta-information

1. Click “Project ➔ Project Information”.
2. The “Project Information” dialog opens.
3. Select the “Summary” tab.
4. Specify your data in the input fields (example: 0.0.0.1 in the “Version” input field).

   CODESYS creates a property with a key for each given value and manages them on the “Properties” tab. For a library project, CODESYS still uses the properties and sorts later in the library repository.

   If you select the option for CODESYS to create a functions block for these properties, then you can access the properties programmatically.

Creating functions for accessing properties

1. Click “Project ➔ Project Information”.
2. The “Project Information” dialog opens.
3. Select the “Automatically generate ’Project Information’ POUs” option.
Example

Requirement: The following property is defined.

```
Key = nProp1
Type= number
Value= 333
```

1. Select the “Automatically generate ’Project Information’ POUs” option.
2. Declare a property of the type DINT, for example `showprop : DINT;`.
3. Call the function `GetNumberProperty`: `showprop := GetNumberProperty("nProp1");`
   ⇒ You are granted access to the value in the application.

Note: The functions that are created with the “Automatically generate ’Project Information’ POUs” option can be used only if the runtime system supports the WSTRING data type. If this is not the case, then instead you can apply the “Automatically generate ’Library Information’ POUs” option. You can use the functions created in this way at least in the application to access properties. These functions are not registered in the runtime system.

Licensing library projects

If your project is a library project, then you can activate the library licensing in use here. The CODESYS Security Key is a dongle.

- Requirement: The project is a library project.
  1. Click “Project ➤ Project Information”.
     ⇒ The “Project Information” dialog opens.
  2. Click the “Licensing” tab.
  3. Select the “Activate dongle licensing” option.
  4. Specify the dongle data in “Firm code”, “Product code”, “Activation URL”, and “Activation mail”.
     ⇒ The library is licensed.

Deprecated: Signing a library project (only for version compatibility with CODESYS < SP15)

In CODESYS V3 SP15 and later, library signing is always certificate-based and can be executed by means of the “File ➤ Save Project as Compiled Library” command. In contrast to deprecated signing by using the settings in the “Project Information” - “Signing” dialog, the entire library is signed with the certificate.

In case you still have to sign a library project for a CODESYS version < V3 V3 SP15 for reasons of compatibility, then proceed as described below.

- Requirement: You have an available private key file with an associated token. In the project information of the library project, on the “Summary” tab, a library compatibility with a CODESYS version < V3 V3 SP15 is set
  1. Click “Project ➤ Project Information”.
     ⇒ The “Project Information” dialog opens.
  2. Click the “Signing” tab.
  3. Select the “Activate signing” option.
4. Specify the memory location of your private key file.
   ⇒ CODESYS enters the token automatically in “Public key token” (example: 427A5701DA3CF3CF).

5. Click “File ➔ Save Project as Compiled Library”.
   ⇒ The library project is saved as a files with the extension *.library. You will see the
token in the library repository or Library Manager in the library details and can com-
pare it with the library vendor's published token. In this way, you can check whether
the library actually originates from the intended vendor.

See also
● § Chapter 1.3.1.25.2.1.7 “Command ‘Save Project as Compiled Library’” on page 856

Creating private
key files

1. Click “Project ➔ Project Information”.
2. Click the “Signing” tab.
3. Click the “Create Private Key File” button.
   ⇒ The “Create Private Key File” dialog opens.
4. Select a safe location, e.g. D:\for lib developers only\mycomp_libkey.libpk
   and exit the dialog with “Save”.

See also
● § Chapter 1.3.1.25.1.31 “Object ‘Project Information’” on page 811

Making project settings

You can configure settings that affect the behavior of CODESYS and that of certain editors in
the “Project Settings” object. The settings are valid throughout the project and are applied
immediately for active editors. You can also access the dialog boxes of the object with the com-
mand “Project ➔ Project Settings”.

CODESYS saves the project settings as an object directly in the project. If you then transfer a
project to another system, the “Project Settings” object is also transferred with it, without a
project archive being required.

See also
● § Chapter 1.3.1.3.3 “Changing the compiler version” on page 126
● § Chapter 1.3.1.11.7 “Downloading source code to and from the PLC” on page 307
● § Chapter 1.3.1.9.13.2 “Analyzing code statically” on page 230
● § Chapter 1.3.1.25.2.4.12 “Command ‘Project Settings’” on page 902
● § Chapter 1.3.1.25.3.11.1 “Dialog ‘Project Settings’ - ‘SFC’” on page 1059
● § Chapter 1.3.1.25.3.11.2 “Dialog Box ‘Project Settings’ - ‘Users and Groups’” on page 1060
● § Chapter 1.3.1.25.3.11.3 “Dialog Box ‘Project Settings’ - ‘Compileoptions’” on page 1062
● § Chapter 1.3.1.25.3.11.4 “Dialog Box ‘Project Settings’ - ‘Compiler Warnings’” on page 1062
● § Chapter 1.3.1.25.3.11.5 “Dialog ‘Project Settings’ – ‘Source Download’” on page 1063
● § Chapter 1.3.1.25.3.11.6 “Dialog ‘Project Settings’ – ‘Page Setup’” on page 1064
● § Chapter 1.3.1.25.3.11.7 “Dialog ‘Project Settings’ – ‘Security’” on page 1064
● § Chapter 1.3.1.25.3.11.8 “Dialog ‘Project Settings’ – ‘Static Analysis Light’” on page 1066
● § Chapter 1.3.1.25.3.11.9 “Dialog ‘Project settings’ – ‘Visualization’” on page 1068
● § Chapter 1.3.1.25.3.11.10 “Dialog ‘Project Settings’ – ‘Visualization Profile’” on page 1069
1.3.1.4 Exporting and transferring a project

Export and import functions are available to you for the exchange of the data from CODESYS projects with other programs.

An exchange of CODESYS projects between CODESYS development systems takes place by way of a copy of the project file (*.project) or project archive (*.projectarchive).

See also
- Chapter 1.3.1.4.1 “Exporting and importing projects” on page 135
- Chapter 1.3.1.4.2 “Transferring projects” on page 136

1.3.1.4.1 Exporting and importing projects

CODESYS offers commands for the export and import of objects to and from a file. Two possibilities are available to you here:

- Export to or import from a CODESYS XML file (*.export)
  This format is completely compatible with the CODESYS project format. The objects are saved in a machine-readable XML format.
- Export to or import from an XML file in the PLCopen format (*.xml)
  You can use this format to exchange information with other programs (for example program editors or documentation tools). PLCopen XML defines a subset of the elements known in CODESYS. 100% compatibility is thus not guaranteed.

Exporting projects

Requirement: A project is open in CODESYS.

1. Select the command “Project ➔ Export…” or “Project ➔ Export PLCopenXML”.
2. Select the objects that you wish to export in the dialog box “Export” or “Export PLCopenXML”.
3. Click on “OK”.
4. Enter the file name and the location and click on “Save”.

Importing projects

Requirement: A project is open in CODESYS.

1. Select the command “Project ➔ Import…” or “Project ➔ Import PLCopenXML”.
2. In the dialog box “Import” or “Import PLCopenXML”, select the export file that you wish to import.
   ➔ A dialog box opens and displays the objects in a tree structure, which can be inserted at this point.
3. Select the object in the object tree, under which the objects to be imported are to be inserted.
4. Select the objects and click “OK”.
   ➔ The objects are added to the existing object tree.

See also
- Chapter 1.3.1.25.3.13.15 “Dialog ‘Options’ - ‘PLCopenXML’” on page 1084
- Chapter 1.3.1.25.2.4.22 “Command ‘Export PLCopenXML’” on page 911
- Chapter 1.3.1.25.2.4.23 “Command ‘Import PLCopenXML’” on page 911
- Chapter 1.3.1.25.2.4.21 “Command ‘Import’” on page 910
1.3.1.4.2 Transferring projects

If you wish to transfer a project to another computer and connect from there to the same PLC, without an online change or download being required, observe the following points.

- Make sure that the project requires only fixed versions of libraries (exception: interface libraries), visualization profile and compiler.
- Make sure that the boot application is up to date.

Then create a project archive, which you unpack on the other computer.

Transferring a project to another system

Requirement: A project is open on computer “PC1” that you transfer to another computer “PC2” and reconnect from there to the same controller.

1. Make sure that only libraries with fixed versions are integrated in the project, with the exception of pure interface libraries. To do this, open the “Library Manager” and check all entries that have a “*” instead of a fixed version specification.
2. Make sure that a fixed compiler version is set in the project settings. To check, select “Project ➔ Project Settings” and the “Compiler Options” category.
3. Make sure that a fixed visualization profile is defined in the project settings. To check, select “Project ➔ Project Settings” and the “Visualization Profile” category.
4. Make sure that the application that is presently open is the same as that which is presently in use on the PLC. This means that the “boot application” must be identical to the project in the programming system. To check, look at the project name in the title bar of the programming system window: If an asterisk is displayed behind the name, this means that the project has been modified, but not yet saved. It is then possible that the application and boot application do not correspond!

In this case, first create a (new) boot application. It depends on the PLC and the application properties, whether this takes place automatically during the download of the application. For explicit creation, select the command “Online ➔ Create boot application”. Then execute a download with the help of the commands “Online ➔ Login” and “Online ➔ Load”.

After that, start the application on the controller with the command “Debug ➔ Start”.

⇒ Now the desired application is running on the PLC, to which you wish to reconnect from the same project later on PC2.

5. Generate a project archive: Select “File ➔ Project Archive ➔ Save/Send Archive”. In the “Project Archive” dialog box, also select the following information:
   - “Download information files”
   - “Library profile”
   - “Referenced devices”
   - “Referenced libraries”
   - “Visualization profile”

Save the project archive in a place that is accessible by PC2.

6. Log out from the controller: To do this, select “Online ➔ Logout”. You can stop and restart the PLC without reservations, before you reconnect from PC2.

7. Extract the project archive to PC2: Select “File ➔ Project Archive ➔ Extract Archive” and open the archive saved above. In the “Extract Project Archive” dialog box, activate the same information as described above when generating the archive.

8. Open the project and log in to PLC “xy” again.

⇒ CODESYS does not demand an online change or download; the project runs.
1.3.1.5 Comparing projects

You can compare the currently open project with another project – a reference project. The differences in contents, properties, or access rights are detected and shown in a comparison view.

Clicking “Project ➤ Compare” opens the “Project Compare” dialog for you to configure and run the comparison. Then the result is shown in the comparison view “Project Compare - Differences” where the objects are aligned in a tree structure. Objects that indicate differences from the respective reference object are identified by colors and symbols. This is how you detect whether or not the contents, properties, or access rights are different.

For differences in the contents, you can also open the detailed compare view “Project Compare - <object name> Differences” in order to zoom into the object. In the detailed compare view, the contents of the object and reference object are displayed or their source code aligned. The detected differences are marked. Previously opened views are not closed. In this way, you can have any number of comparison views open and read them, in addition to the project compare view.

You can accept the detected differences from the reference project into the current project. This is possible only from the reference project into the open project. To do this, you activate differences (for example in the code) that should be accepted in the current project with the commands ✓, ✗, or ✕ in the active comparison view for accepting. These positions are highlighted in yellow. Make sure that any other open compare views are inactive (write-protected, read-only), therefore, you can activate differences to be accepted in exactly one comparison view only. When exiting the active compare view, if you confirm that the differences that are activated for acceptance are actually accepted into the current project, then the current project is modified.

In order to exit the project comparison completely, close the project compare view.

1.3.1.5.1 Creating a comparison view

Requirement: You have made changes in your current project and wish, for example, to compare it with the last-saved version. In the meantime, for example, you have added further POU's, removed a POU, changed single lines of code or the object properties in function blocks.

1. Select the command “Project ➤ Compare”.
   ➤ The “Project Comparison” dialog box opens.

2. Enter the path to the reference project, for example the path to the last-saved version of your current project.

3. Leave the activation of the comparison option “Ignore Spaces” as it is.

4. Click on “OK”.
   ➤ CODESYS opens the comparison view. Title: “Project Comparison – Differences”. The Device trees of the current project and the reference project are displayed alongside each other and the changed objects are marked in color.

5. Select an object marked in blue in the tree of the reference project (right). The current project no longer contains this object.

   Click on ✓ “Accept Single”
   ➤ CODESYS adds the object to the tree of the current project (left). The line has a yellow background. ✗ appears in the middle column.
6. Select an object marked in green in the tree of the current project (left). The reference project does not contain this object.

   Click on "Accept Single"
   ⇐ CODESYS removes the object again from the tree of the current project (left). The line has a yellow background. 🟢 appears in the middle column.

7. If CODESYS detects changes in the content of an object that is contained in both the current project and the reference project, this is indicated by red lettering. You can then switch to the detailed comparison view for the object by double-clicking on the object.

8. Close the comparison view and answer the query whether the changes made are to be saved with "Yes".
   ⇐ The changes become effective in the project.

1.3.1.5.2 Opening the Detailed Compare View

Requirement: For example, a user modified the code in a POU of the current project. You have performed the project comparison by clicking "Project ➔ Compare". The project compare view shows this POU highlighted in red in the aligned in the project tree.

1. Double-click the line of the aligned POU versions.
   ⇐ The compare view switches to the detailed compare view of the POU. The modified code lines are highlighted in gray and written in red.

2. Click ☑.
   ⇐ Code lines with changes (red) are extended by two lines: an line with insert (left, green) and a line with delete (right, blue).

3. Click ✗ again.
   ⇐ The code line is marked again as modified.

4. Move the mouse pointer to the code line marked as modified and click "Accept Single".
   ⇐ The code line from the reference project is activated for acceptance into the current project.

5. Click ☑.
   ⇐ The project compare view opens for the entire project. It is write-protected (read-only) to prevent you from activating differences for acceptance. The link highlighted in yellow above the tree view also indicates this.

6. Click the link: "Project compare view is read only because there are uncommitted changes in another view. Click here to switch to the modified view."
   ⇐ The detailed compare view opens again. The unconfirmed changes are highlighted in yellow.

7. Click ✗ in the tab of the view and confirm that the changes should be saved.
   ⇐ The detail project view is closed and the POU is overwritten. Now it corresponds to the POU of the reference project. The project view is active again so that you can continue working with project compare.

---

If you do not click the link, but click ✗ instead to close the editor of the project compare view, then you will also confirm the acceptance of changes into the current project. The detail changes are accepted and then the project compare is closed completely.
You can protect a project against unintentional changes by means of access and write protection. You can also provide it with read protection (knowledge protection).

Write protection:
The following options are available for providing the entire project with simple write protection:

- Select the "Open Read-Only" option when opening the project.
- You set the "Released" status in the "Project Information".
- You select the "read-only" option in the properties of the project file in the local file system.

In order to protect only certain objects in a project against changes, or to allow access only to certain users, you can use a user and access rights management (see below). Some target devices similarly support user and rights management. The access of CODESYS to objects and files of the target device can thus be restricted.

However, write protection and access protection do not serve as protection of expertise of the POU.s. Both CODESYS itself, automation platform plug-ins and persons with knowledge of the project file format can view or modify function blocks created with CODESYS.

Knowledge protection:
Knowledge protection of a project is done by encrypting the project file. Either with a project password, the CODESYS Security Key (dongle), or a certificate. We recommend protection by means of the key or the certificate because in this case no secret needs to be shared between authorized users. The desired type of project encryption is enabled in the project settings.

You can attain knowledge protection of a library by providing it as a target-system-independent "protected library" (*.compiled-library, *.compiled-library-v3). The library file no longer contains source code in this format, but only encrypted precompile context. The compiler is still able to interpret these data. Whether access by other CODESYS components or additional plug-ins is possible depends on their functionality and is to be observed in individual cases. Signing can increase protection even more.

Knowledge protection and copy protection of a boot application can be done by means of a run-time system dongle (simple or licensed) or encryption with a certificate. One of these options is enabled in the object properties of the application.

See also

- "User management and password manager" on page 140
- Chapter 1.3.1.6.3 “Protecting projects using a dongle” on page 144
- Chapter 1.3.1.6.2 “Assigning passwords” on page 143
- Chapter 1.3.1.6.5 “Protecting objects in the project by access rights” on page 146
- Chapter 1.3.1.19.1 “Information for library developers” on page 372
- Chapter 1.3.1.9.18 “Encrypting an application” on page 242

Encryption with certificates

In CODESYS, projects and applications can be encrypted with certificates and signed in order to protect them from unauthorized access.

To do this, you can configure specific security settings for each individual user profile. These settings are always used automatically when the user works with CODESYS projects. Therefore, they do not have to be redone for each project. The general configuration of the security features for a user profile is done in the "Security Screen" view of CODESYS. See the individual instructions below.

You can also encrypt a project file or an application for download or online change directly with a certificate:
● User-independent encryption for the current project is configured in the “Security” category of the “Project Settings”.
● User-independent encryption of the application is configured in the “Properties” dialog of the application object.

**NOTICE!**
When you encrypt a project, an application, or online code with a certificate, you will always require the certificate with a private key in order to open the object again.

If the CODESYS Security Agent add-on product is installed, then the “Security Screen” view provides an additional tab: “Devices”. This allows for the configuration of certificates for the encrypted communication with controllers.

**Certificates, Windows Certificate Store**
All available certificates are located in the Windows Certificate Store (“certmgr”) on your computer. There are two types of keys:

- Certificates with private keys
  - for file decryption
  - for digital signatures
- Certificates with public keys
  - for file encryption
  - for verifying digital signatures

The local Windows Certificate Store is usually filled with certificates by the IT administrator of the computer. Certificates are either created using special tools or the creation is requested by a trusted certification authority (CA).

If you receive a certificate file that you need to install yourself in the Windows Certificate Store, then double-click the file in the store directory. Depending on the type (certificate with private or public key only), the appropriate import wizard will appear.

See also
- § Chapter 1.3.1.23.1 “General information” on page 421
- § Chapter 1.3.1.6.7 “Encrypting projects with certificates” on page 149

**User management and password manager**
User accounts with different rights can be managed in CODESYS. For each account you can define the actions with which the user can access a project object.

The user management is configured in the “Project settings” in the category “Users and Groups”.

Before the creation of users and groups, please note the following:

- Rights can only be assigned to user groups. Therefore, you must assign each user to a group.
- There is automatically always a group ‘Everyone’ and by default every user and every other group is initially a member of this group. Thus each user account is automatically equipped with at least the defined standard rights.
- You cannot delete the group 'Everyone’, you can only rename it, and you cannot remove members from this group.
- Caution: by default "Everyone" does not have the right to change the current user, group and rights configuration!
● There is automatically always a group ‘Owner’ containing a user ‘Owner’. From V3.5 only the ‘Owner’ initially has the right to change the current user, group and rights configuration in a new project! Hence, only ‘Owner’ can assign this right to another group.

Initially the ‘Owner’ can log in with user name ‘Owner’ and an empty password. You can add further users to the group ‘Owner’ or remove users from it, but at least one member must be retained. Like ‘Everyone’, you cannot delete the group ‘Owner’ and it always possesses all access rights. This prevents a project from being rendered unusable by denying all access rights to all groups.

You can rename both the group ‘Owner’ and the user ‘Owner’.

● If the programming system or a project is restarted, no user is initially logged in to the project. However, the user can then log in via a certain user account with user name and password in order to obtain the access rights defined for the account.

● Each project has its own user management! Therefore, in order to obtain certain access rights to a library integrated into the project, for example, the user must explicitly log in to the library project.

Users and groups defined in different projects are not the same, even if they have the same names.

● A user management in a project only makes sense if it is connected with corresponding rights assignment for the access to project and objects. The project rights are generally managed in the dialog box “Rights” of the “User Management”. You can also change the access rights to an individual project object on the “Access control” tab of the “Properties” of the object.

● There are standard menu commands under “Project ➔ User Management” for logging into and out of a project as a defined user. A password manager permits the management of the login data on your computer.

From V3.5 only the ‘Owner’ initially has the right to change the current user, group and rights configuration in a new project! Hence, only ‘Owner’ can assign this right to another group.

NOTICE!
CODESYS stores the user passwords inaccessibly. If you forget a password, the user account becomes unusable. If you forget the ‘Owner’ password, the entire project may become unusable!

Password manager

The password manager enables you to save login data records that you enter during the login procedures for projects. It is accessible via a button in the login dialog box and offers fast access to the login data currently required. This can be helpful, for example, if you are working in parallel on several library projects that are protected by different passwords.

The password manager itself is protected by an individual master password. If you wish to use the password manager for the first time, CODESYS requests you to define this password in the password manager configuration dialog box. CODESYS notes the master password until you terminate the current CODESYS session. You must always input the password when you wish to log in to the password manager for the first time during a new session, or after you have changed it.

See also
● § Chapter 1.3.1.6.5 “Protecting objects in the project by access rights” on page 146
● § Chapter 1.3.1.6.6 “Logging in via user account and password manager” on page 147
● § Chapter 1.3.1.11.3 “Handling of device user management” on page 300

Rights management

Rights management for access to a project and objects in a project is necessary in order to make a user management meaningful.
The rights for a project are generally managed in the “Rights” editor of the “User Management”. You can also change the access rights to an individual project object on the “Access control” tab of the “Properties” dialog box of the object.

Before assigning rights, please observe the following:

- In a new project CODESYS always sets all rights for the execution of actions on objects with the default value ‘allowed’ (standard right). The only exception to this is the right to change the current user, group and rights configuration. Initially only the ‘Owner’ group has this right.
- If you are member of a group that is permitted to change rights, you can do this at any time for each right when working further on a project. You change a right by switching between ‘allowed’ and ‘forbidden’ or by resetting to the default.

See also

- Chapter 1.3.1.6.4 “Setting up a user management” on page 145
- Chapter 1.3.1.6.5 “Protecting objects in the project by access rights” on page 146

Filing, saving

Provide the project file with the desired protection before saving it in the file system; see above. For a read-only project file you are given various options so that you can still save the file, depending on the type of write protection.

If the project is to be opened later in an older CODESYS version, it makes sense to save the project for precisely this version (file type), since CODESYS will also inform you immediately about possible losses of data in the course of saving it.

If you wish to save library projects, please observe the rules for the creation of libraries. Also consider the possibility of installing a library directly in a library repository.

If you wish to continue to use a project on another computer, it makes sense not only to save the project file, but also to create a project archive from all relevant auxiliary files.

You can make a setting so that a backup copy of this project is created each time the project is saved. In addition you can configure CODESYS so that projects are generally automatically saved at certain time intervals.

If you wish to keep projects in a source control system, observe the corresponding add-ons for CODESYS. For example, the link to SVN is supported.

See also

- Chapter 1.3.1.25.3.13.13 “Dialog ‘Options’ – ‘Load and Save’” on page 1083
- Chapter 1.3.1.4 “Exporting and transferring a project” on page 135
- Chapter 1.3.1.6.8 “Save project” on page 150
- Chapter 1.3.1.6.9 “Saving/Sending the project archive” on page 152
- Chapter 1.3.1.19.1 “Information for library developers” on page 372
- Chapter 1.3.1.6.10 “Linking a project to the source control system” on page 152

1.3.1.6.1 Setting up write protection

A project can be protected against inadvertent changes by means of access and write protection. In addition, however, it can also be provided with read protection (know-how protection). You have the following options:

Open the project with write protection

Requirement: No project is opened.

1. Select “File ➔ Open Project”.
   ⇔ The dialog box “Open project” appears.

2. Select the project.
3. Click on the arrow button ☰ next to the “Open” button and select “Open read-only” from the menu.

 CODESYS opens the project. At the top right in the main window a line appears “Project file cannot be saved...”. You must now select one of the offered options if you wish to save the project file.

See also
● Chapter 1.3.1.3.4 “Opening a V3 project” on page 127

Providing projects with the attribute 'Released'

Requirement: project is opened.

1. Select “Project ➔ Project Information”, then the “Summary” tab.
2. Activate the option “Released”, confirm with “OK”.
3. Save the project, for example with [Ctrl]+[S].
4. Open the project again with the command “File ➔ Open Project”.

 CODESYS opens the project. At the top right in the main window a line appears “Project file cannot be saved...”. You can now directly remove the status “Released” again via the offered option if you wish to save the project file.

See also
● Chapter 1.3.1.25.1.31 “Object ‘Project Information’” on page 811

Providing a project in the file system with the property 'Read-only'

Provide the project file in its local file system with the property attribute ‘Read-only’.

 If you had already opened the project and you now attempt to save it under the same name, a dialog box appears informing you about the existent write protection. This dialog box provides you with the following options:

 You can save the project under another name or another path using the button “Save As...”.

 You can deliberately save the project under the same name and path and thus overwrite the existing version in the file system using the button “Overwrite”.

 You can abort the saving procedure using the “Cancel” button, for example to remove the write protection on the disk.

 If you re-open the project, a line appears at the top right in the main window 'The project cannot not be saved...'. You must now select one of the offered options if you wish to save the project file.

See also
● “General information about write and access protection” on page 139

1.3.1.6.2 Assigning passwords

Requirement: The project is open.
1. Click “Project ➤ Project Settings” and then select the “Security” category.
   ⇒ The dialog “Project Settings / Security” opens.
2. Select the “Encryption” option.
   ⇒ The option fields “Password”, “Dongle”, and “Certificates” are selectable.
3. Select the option “Password”.
   ⇒ The input fields for the encryption password appear.
4. Enter the encryption password in the input field “New password”.
5. Enter the encryption password for confirmation in the input field “Confirm new password”.
6. Click “OK”.
   ⇒ CODESYS saves the encryption password for the project. You must enter this password in order to be able to open the project again, even if it is to be loaded as a library reference.

CAUTION!
If you no longer know the encryption password, you can no longer open or restore the project!

See also
● § Chapter 1.3.1.25.3.11.7 “Dialog ‘Project Settings’ - ‘Security’” on page 1064
● § Chapter 1.3.1.6 “Protecting and saving a project” on page 139

1.3.1.6.3 Protecting projects using a dongle

Requirement: The project is opened and you have connected the CODESYS Security Key (dongle) to your computer.

1. Click “Project ➤ Project Settings” and then select the “Security” category.
   ⇒ The dialog “Project Settings / Security” opens.
2. Select the “Encryption” option.
   ⇒ The option fields “Password”, “Dongle”, and “Certificates” are selectable.
3. Select the option “Dongle”.
   ⇒ The dialog with the drop-down list “Registered Dongles” and the buttons “Add”, “Remove”, “Comment” and “Flash” opens.
4. Click “Add”.
   ⇒ The “Add registered dongle” dialog opens.
5. Select the CODESYS Security Key (dongle) from the “Dongle” drop-down list and optionally enter a comment.
6. Click “OK”.
   ⇒ The added dongle is listed in the list “Registered Dongles”.
7. Click “OK”.
   ⇒ The dongle is registered for the project. You must connect the dongle to your computer in order to be able to open the project again, even if it is to be loaded as a library reference.
NOTICE!
If the CODESYS Security Key registered for the project is lost, you can no longer open the project or restore it.

See also
- Chapter 1.3.1.25.3.11.7 “Dialog ‘Project Settings’ - ‘Security’” on page 1064
- Chapter 1.3.1.6 “Protecting and saving a project” on page 139

1.3.1.6.4 Setting up a user management

This concerns a user management for a CODESYS project file. Visualizations and devices can have their own user management.

The following guide describes how you can adapt the user management for the first time in a project. It deals with the definition of a user and a group to which he belongs.

Requirement: the project for which the user management is to be set up is opened. There is no adapted user configuration yet.

1. Select “Project Settings ➔ Users and Groups” and then the “Users” tab. The user Owner is already created by default.

2. Click on “Add”.
   ⇒ The dialog box “Add User” appears.

3. Enter a login name, for example 'Dev1', and a password. Leave the option “Activated” activated. Click on “OK”.
   ⇒ On creating a group for the first time, CODESYS now requests you to authenticate yourself to perform this action.

   In this case, enter 'Owner' as the “current user”. Do not enter a “password”, just click on “OK”.

   The user Dev1 appears in the list and is automatically a member of the group 'Everyone'.

4. Change to the tab “Groups”, in order to add the user to a new group.
   ⇒ The groups Everyone and Owner have already been created.

5. Click on “Add” in order to open the dialog box “Add Group”.

6. Specify at least one name for the new group, for example 'Developers'. Activate the checkbox next to the entry “User 'Dev1'” in the field “Members”. Click on “OK”.

   ⇒ The group “Developers” now appears with has user member 'Dev1'.

7. Switch to the “Users” tab.

   ⇒ The user “Dev1” now appears as a member of the groups 'Everyone' and 'Developers'.

You can take over the user management configuration from another project by using the “Export/Import” functions in the dialog box “Project Settings”, category “Users and Groups”.

PLC Automation with V3 CPUs
Programming with IEC 61131-3 editor > CODESYS Development System

2020/12/10  3ADR010583, 1, en_US
1.3.1.6.5 Protecting objects in the project by access rights

Protection of individual objects by setting access rights in the “Rights” editor

1. Select “Project ➤ User Management ➤ Rights”
   ⇒ The window of the “Rights” editor opens. On the left you can see the action categories, on the right the currently existing user groups.

2. Expand the relevant action category and below it the action for which you wish to change the right.

3. Select the goal of the action in the “Actions” window. In the “Rights” window, select the group for which you would like to change the right. Multiple selection is possible.
   ⇒ The buttons in the symbol bar are active.

4. Click on the appropriate button in order to change the right of the group for the action on the target object.
   ⇒ CODESYS updates the symbol in front of the group according to the new right. The right is immediately effective.

See also

● Chapter 1.3.1.25.3.6 “Dialog ‘Permissions’” on page 1042

Protection of individual objects by setting access rights in the object properties

Here you can configure whether the members of a group have the right to view, edit or remove the object and to add/remove child objects to/from the object.

1. Select the object in the navigator tree.

2. In the context menu, select the command “Properties” and in the dialog box select the category “Access Control”.

3. In the table under “Groups, actions and permissions”, double-click on the symbol of the right that you wish to change.
   ⇒ A selection list of the possible rights appears: “Grant”, “Deny”, “Clear”.

4. Select the desired right and click on “Accept” or “OK”.
   ⇒ The right is immediately effective for the action and group. The symbol changes accordingly.

See also

● Chapter 1.3.1.25.3.10.6 “Dialog ‘Properties’ - ‘Access Control’” on page 1050
1.3.1.6.6 Logging in via user account and password manager

Logging in to a project without using the password manager functions

Requirement: A project is open. You wish to log in as a defined user for this project or for a library integrated in it in order to edit one or the other with certain rights. You have the required login data for the respective project or the library.

1. Select “Project ➔ User Management ➔ User Logon”.
   ⇒ The dialog box “Logon” opens.
2. Select the project file from “Project/Library” and enter the required access data “User name” and “Password”.
3. Log in with “OK”.
   ⇒ If another user is already logged in, this user will automatically be logged out by the new login.

Setting a master password for the password manager

Requirement: A project is open. The dialog box “Login” is open for you to log in as a defined user for a project or for a library integrated in the project. You wish to use the password manager in order to save login data in it.

1. Select “Project ➔ User Management ➔ User Logon”.
2. In the dialog box “Logon”, click on the button.
   ⇒ If you are working for the first time with the password manager, the dialog box “Password Manager Configuration” opens.
3. Enter a character string as the future master password. Confirm it in the second line and click on “OK”.
   ⇒ CODESYS notes the master password until you terminate the current CODESYS session. You must always input this password when you wish to log in to the password manager for the first time during a new session, or after you have changed it.

NOTICE!
If you have forgotten your master password, you no longer have any possibility to access the login data already saved! In this case you can only reset the password manager. After that you must start again to save passwords in the manager!

Saving login data in the password manager

Requirement: A project is open. You wish to log in as a defined user for this project or for a library integrated in it in order to edit one or the other with certain rights. You have the required login data for the respective project or the library. These login data have not yet been saved in the password manager.

1. Select “Project ➔ User Management ➔ User Logon”, in order to open the “Logon” dialog box.
2. Select the project file from “Project/Library”.
3. Enter the user name and password for the project or the library.
4. Click on the button.
   ⇒ If you are working for the first time with the password manager, you will be requested to define a master password. Refer to the above guide ‘Setting a master password for the password manager’ for this.

When you call the password manager for the first time in this CODESYS session, you will be requested to enter the master password.
5. Enter the master password when requested to do so.
   ⇒ The password manager menu appears.

6. Select the option “Save the credentials locally on this computer”.
   ⇒ The login takes place. The data are saved in the password manager.

Getting the login data from the password manager

Requirement: A project is open. You wish to log in as a defined user for this project or for a library integrated in it in order to edit one or the other with certain rights. The login data required for this are already saved in the password manager.

1. Select “Project ➔ User Management ➔ User Logon” in order to open the “Logon” dialog box.
2. Click on the button.
   ⇒ If you are working for the first time with the password manager, you will be requested to define a master password. Refer to the above guide ‘Setting a master password for the password manager’ for this.
   When you call the password manager for the first time in this CODESYS session, you will be requested to enter the master password.
3. Enter the master password when requested to do so.
   ⇒ The password manager menu appears.
4. Select the appropriate entry “Use the stored credentials for <user name>”.
   ⇒ The login takes place automatically with the data read from the password manager.

Opening the password manager, changing the master password

Requirement: A project is open. You wish to open the password manager in order to view and/or edit the entries or to change the master password. You have already logged in once with the master password.

1. Select “Project ➔ User Management ➔ User Logon”, in order to open the “Logon” dialog box.
2. Click on the button.
   Select “Open the Password Manager”.
   ⇒ The password manager window opens.
3. Click on “Change master password” and make the change.

Logging out from the project

Requirement: A project is open. A user is logged in, which is recognizable by a name entry in the field “Current User” in the status bar.

⇒ Select “Project ➔ User Management ➔ User Logoff”. Alternatively, double-click on the field “Current User” in the status bar.
   ⇒ If the user is logged in to only one project, he will now be logged out without further interaction. “(nobody)” appears again in the field “Current User” in the status bar
   If the user is logged in to several projects, the dialog box “Logoff” opens. There, select the specific project or library project from which the user is to be logged out.

See also

- “User management and password manager” on page 140
- Chapter 1.3.1.25.2.4.24 “Command ‘User management’ – ‘Log in User’” on page 911
- Chapter 1.3.1.6.4 “Setting up a user management” on page 145
1.3.1.6.7 Encrypting projects with certificates

**Configuring a certificate for project file encryption in a user profile**

When a project is encrypted with a certificate, this certificate is needed for decryption to open the project. You can assign this certificate to specific user profiles. To do this, select the certificate from the Windows Certificate Store on the “User” tab of the “Security Screen”.

1. Double-click 🖼 in the status bar or click “View ➔ Security Screen”.
   ↩️ The “Security Screen” view opens.
2. In the “User” tab, select the user profile for which the communication will be encrypted. By default, the specified user profile is the one you have used on your computer to sign into Windows. You can also create a new user profile with 📦.
3. Click the button in the “Project file decryption” area.
   ↩️ The “Certificate Selection” dialog opens.
4. Select a certificate with a private key from the list “Available certificates in the local Windows Certificate Store”. Certificates with a private key are identified by the 🗝️ symbol.
5. Click 🖼.
6. The certificate is added to the upper part of the dialog.
7. Click “OK” to confirm your selection.
   ↩️ The selected certificate is displayed in the “Security Screen” in the “Project file decryption” area.

**Encrypting a project with a certificate**

A project encrypted with a certificate in connection with a user management allows you to restrict access to the project.

1. Click “Project ➔ Project Settings” and then select the “Security” category.
   ↩️ The “Project Settings / Security” dialog opens.
2. Select the “Encryption” option.
   ↩️ The option fields “Password”, “Dongle”, and “Certificates” are available.
3. Select the “Encryption” option.
   ↩️ The certificates available for project encryption are listed in the lower part of the dialog. If no certificate has been specified yet, then click 🖼 to select a relevant certificate in the “Certificate Selection” dialog. Then return to the “Project Settings” dialog. Now the certificate is specified for encryption. Now the project can only be edited on computers of users who also have the certificate for file decryption.

**Deleting a certificate in the user profile**

You delete the certificate in the “Security Screen” view, either directly on the “User” tab or in the “Certificate Selection” dialog. The deletion will follow in the other dialog.

- Dialog “Security Screen”, tab “User”, “Digital signature”, or “Project data decryption”: Select a certificate and click 🗝️
- Dialog “Certificate Selection”: in the “Security Screen” dialog, click 🖼 on the “User” tab. In the upper field of the “Certificate Selection” dialog, select the certificate to be deleted and click 🗝️

**Configuring a certificate for the digital signature in a user profile**

To ensure that the project is not only encrypted with a certificate, but also that its authorship and integrity can be verified, you can add a signature to the project:
1. Double-click 🖼 in the status bar or click “View ➔ Security Screen”.
   ➞ The “Security Screen” view opens.
2. In the “User” tab, select the user profile for which the digital signature will be created. By default, the specified user profile is the one you have used on your computer to sign into Windows. You can also create a new user profile with ⬆.
3. Click the button in the “Digital signature” area.
   ➞ The “Certificate Selection” dialog opens.
4. Select a certificate with a private key from the list “Available certificates in the local Windows Certificate Store”. Certificates with a private key are identified by the 📜 symbol.
5. Click 📜. The certificate is added to the upper part of the dialog.
6. Click “OK” to confirm your selection.
   ➞ The selected certificate is displayed in the “Security Screen” in the “Digital signature” area.

See also
● “Encryption with certificates” on page 139
● Chapter 1.3.1.25.2.3.18 “Command ‘Security Screen’” on page 891
● Chapter 1.3.1.25.3.18 “Dialog ‘Certificate Selection’” on page 1098
● Chapter 1.3.1.25.3.10.3 “Dialog ‘Properties’ - ‘Encryption’” on page 1047
● Chapter 1.3.1.25.3.11.7 “Dialog ‘Project Settings’ - ‘Security’” on page 1064
● Chapter 1.3.1.9.18 “Encrypting an application” on page 242

1.3.1.6.8 Save project
Saving a project under the same name
Requirement: The project is open. The project file is not write-protected.

▷ Select “File ➔ Save”.
   ➞ CODESYS saves the project file with the current project name, which appears in the title bar of the main window. If the project has been changed since it was last saved, then the project name is provided with an asterisk. If this is set in the CODESYS options in the category “Load and save”, then a backup copy will also be made.

Saving a project under a different name or format
Requirement: The project is open.

1. Select “File ➔ Save Project As”.
   ➞ The “Save Project” dialog box opens.
2. Select a storage location in the file system and the desired “File Type” (project file or library file) and the desired storage version. If you want to open the project later in an older version, then it makes sense to save for precisely this version, as you will then be informed immediately in the message window about possible data loss.
   ➞ If the project file is not write protected, then CODESYS saves it in the selected path. Otherwise you will be informed how to proceed.
3. If the current project contains add-ons that are not available in the selected memory format, then the “Extend Profile” dialog box opens.
4. Select the add-ons to extend the memory profile in order for the add-on data to be saved.
5. To save the memory profile permanently, click “Save Profile” and specify a name in the “Enter profile name” dialog box.

6. In the “Extend Profile” dialog box, select the “Use saved profile” option and click “Yes”.
   - CODESYS saves the project with the saved profile.

**Saving a read-only project**

Requirement: A read-only project is open.

- Select “File ➔ Save”.

If the write protection was assigned in CODESYS, then it will be displayed by a line in the top right corner of the main window. Depending on the current situation you will be offered one or more of the following actions so that you can still save the project:

- “Save project under a different file name on the disk”: Always appears and continues to the “Save File” dialog box, as for the “Save File As” command.
- “Exit read-only mode”: Appears if the “Open read-only” option is selected when opening the project.
- “Remove read-only attribute from the project on the disk”: Appears if the project file was provided with the 'Read-only' property in the local file system at the time of opening.
- “Remove identification ‘Released’ in the project information”: Appears only if this attribute is currently set.

If the write protection was assigned outside of CODESYS in the properties of the project file in the file system, you will be offered the following options when you attempt to save under the same name and path:

- “Save As”: You can save under a different name as with the “Save Project As” command.
- “Overwrite”: The write protection is removed from the project file and the file is saved under its existing name.

1. Click on the line in the top right corner of the main window that indicates the write protection.
   - The current options with which you can still save the project appear in a selection menu.

2. Select one of the options offered and perform any necessary actions.

3. Click “File ➔ Save” or “File ➔ Save As”.
   - The project can be saved.

**Saving of the project automatically; creating a backup copy**

Requirement: The project is open.

1. Click “Tools ➔ Options” (category “Load and Save”).
   - The “Load and Save” dialog box opens.

2. Activate the “Create backup files” option.

3. Activate the “Automatically save every … minute(s)” option and select a time interval.

4. Click “OK” to close the “Options” dialog box.
   - Each time the project is saved, CODESYS also creates a backup copy <project name>.backup.

CODESYS saves the project automatically at the specified time interval to a file <project name>.autosave in the project directory. If you open the project again after the development system was closed irregularly, then this file will be offered to you as an alternative to the file last saved by the user.
1.3.1.6.9 Saving/Sending the project archive

You can configure a project archive and then save it in the file system or send it directly in an e-mail.

To send, follow the guide below as far as point 9. There, click on the button “Send” instead of “Save” in order to directly open the standard e-mail program, in which a new mail will automatically be created with the project archive file as an attachment.

Requirement: A project is opened.

1. Select “File ➔ Project Archive ➔ Save/Send Archive”.
   ⇒ The dialog box “Project Archive” appears.

2. Activate the checkbox next to each object that is to be saved in the archive.

   In order to guarantee know-how protection, CODESYS will not automatically add unprotected libraries, not available as “compiled-library”, to a project archive. If you explicitly select such a library in the list of additional files, you will get an appropriate warning.

3. If you want to pack further files in the archive, click on “Additional Files”.
   ⇒ The dialog box “Additional Files” opens.

4. Click on “Add”.

5. Select the files and click “Open”.
   ⇒ The files are added to the list of additional files.

6. Click on “OK”.

7. Click on “Comment”.
   ⇒ The dialog box “Comment” opens.

8. Enter a comment and click on “OK”.

9. Click on the button “Save”.

10. Select a storage location and a file name and click on “Save”.
    ⇒ The project archive is saved in the file directory.

See also

-  Chapter 1.3.1.6 “Protecting and saving a project” on page 139
-  Chapter 1.3.1.25.3.13.13 “Dialog ‘Options’ – ‘Load and Save’” on page 1083

1.3.1.6.10 Linking a project to the source control system

To link your CODESYS projects to a source control system, check the following option:

The CODESYS SVN add-on provides the capability of directly linking to an SVN database. You can get the package at the CODESYS Store and install it with the help of the Package Manager. Refer to the corresponding help when using CODESYS SVN.
1.3.1.7 Localizing projects

You can display your project in different languages when you create and link localization files. The localization files correspond to those of the GNU gettext system. The format of the localization template files is *.pot (Portable Object Template), from which localization files *.po (Portable Object) are generated after translation.

You configure which categories of text information are localized in the project. Then you export these texts into a translation template. This template is a file in *.pot format (example: project_1.pot). You produce localization files in the format *.po (example: de.po, en.po, or es.po), either automatically with a corresponding external translation tool or manually with a neutral text editor. You can import the *.po files back into CODESYS and use them for localization.

The commands for using project localization are located in the menu “Project ➔ Project localization”.

See also
- § Chapter 1.3.1.25.2.4.14 “Command 'Project Localization' - 'Create Localization Template’” on page 903

Generating localization templates

Requirement: A project is open.

1. Click “Project ➔ Project Localization ➔ Create Localization Template”.
   ➔ The “Create Localization Template” dialog box opens.

2. Activate the categories of text information that should be included in the localization template.

3. “Position information” can also be included in the template. For each text to be translated, specify its location in the project. Select the positions to be displayed in the translation template: only the first position found, all positions found, or none.

4. Click the “Generate” button.
   ➔ The dialog box opens for saving a *.pot file to the file system. Save the localization template. Then you can process the file in a translation tool and generate localization files <language>.po in the required languages.

Format of the localization template: file *.pot

In the first line, the text categories are specified that were selected for the translation when generating the template:

Example: #: Content:Comments|Identifiers|Names|Strings: All four categories were selected.

Then each text to be translated is segmented in the form as in the following example:
Example

```plaintext
#: D:\Projects\pl1.project\Project_Settings:1
msgid "Project Settings"
msgstr ""
Line 1: Position information displayed as source code reference. Displayed only if this has
been configured when generating the translation file.

Line 2: Untranslated text as entry msgid (example: msgid "Project settings").

Line 3: Placeholder for the translation: msgstr "". Between the single straight quotation
marks, the translation in the *.po file must be inserted in the respective language.
```

Format of the localization file:  

*-  
<language>.po

You can generate a *.po file with a translation tool or create one using a neutral text editor
based on the *.pot file. For this purpose, you could change the file extension from *.pot to
*.po and edit the according to *.po standard format.

It is imperative to specify the language in the form of the usual culture abbreviation in the meta-
data of the file (example: "Language: de" for German. Then you insert the translations of the
individual texts between the straight quotation marks for the msgstr "" entries.

Example

```plaintext
"Language: de\\n"  
#: Content:Names  
#: D:\projects\pl1.project\Project_Settings:1  
msgid "Project Settings"  
msgstr "Projekteinstellungen"
```

Importing localization files / localizing projects

Requirement: For your project, localization files (<language>.po) were generated based on
the translation template *.pot. The project is open.

1. Click “Project ➔ Project Localization ➔ Manage Localizations”.
2. Click on the “Add” button.
   - The “Open Localization File” dialog box appears for selecting a *.po file from the file
     system.
3. Select one of the localization files (example: <project name>-de.po).
   - The dialog box closes and the affected texts appear in the project in the respective
     language. For example, if you specify the translation msgstr "Main program" for
     the POU name "PLC_PRG" in the English localization file, then the object name "Main
     program" appears in the device tree.
4. In the same way, you import the localization files for other language targets.

Switching localization, adding and removing localization files

Requirement: All required language are stored in the project by importing the corresponding
*.po file. The project is open.

1. Click “Project ➔ Project Localization ➔ Manage Localization”.
   - The “Manage Localization” dialog box opens. All stored localization files *-<language>.po appear in “Files”, as well as the entry “<original version>”.
2. Select the desired language and click the “Switch Localization” button.
   - The project appears in the selected language. When you select “<original version>”,
     the project is displayed in the original, unlocalized version and it cannot be edited.
Optional: Defining a default localization, toggling localizations

Select one of the available localizations and activate the “Default Localization” option.

Click “Project ➔ Project Localization ➔ Toggle Localization” to toggle the localization between the default localization and original version. By default, this command is also available with the button on the toolbar.

See also
- Chapter 1.3.1.25.2.4.15 “Command 'Project Localization' - 'Manage Localizations’” on page 904
- Chapter 1.3.1.25.2.4.16 “Command 'Project Localization' - 'Toggle Localization’” on page 905

1.3.1.8 Configuring I/O links

With the help of device objects, you can map hardware to be controlled in a tree structure in your CODESYS project. This makes for easy handling of the linking of hardware and application.

In the configuration editors of the device objects, you can configure the settings for the communication between CODESYS and the controller, and above all for I/O mapping. I/O mapping is the linking of the inputs and outputs of the controller with the variables of your application.

Depending on the device, access to controller objects at runtime can be regulated by means of an editable online user management in the CODESYS Development System. Moreover, communication with the controller depends on the current security settings.

See also
- Chapter 1.3.1.8.1 “Device tree and device editor” on page 155
- Chapter 1.3.1.8.2 “Mapping a hardware structure in the device tree” on page 159
- Chapter 1.3.1.8.3 “Configuring devices and I/O mapping” on page 162
- Chapter 1.3.1.11.2 “Encrypting communication, changing security settings” on page 296
- Chapter 1.3.1.11.3 “Handling of device user management” on page 300
- Chapter 1.3.1.16 “Using device applications” on page 358

1.3.1.8.1 Device tree and device editor

Device tree

In the “Devices” view (also called the device tree), you organize applications according to target device. In this view, you can see the PLC hardware and fieldbus systems, configure the hardware communication, and assign applications.

The root node of the device tree is a symbolic node entry: “<project name>”.

You insert device objects below this node for one or more PLCs, also referred to as target systems. Each device object represents a specific hardware component, for example a controller, fieldbus, bus coupler, drive, I/O module, or monitor. When you add objects, an add assistant helps you by offering all possible devices from your local device repository.

If you are already connected to a controller network, then you can scan the hardware for available devices and save them to the device tree in the current configuration.

For creating device objects in the device tree (mapping to the controlling hardware environment), specific rules apply (see below). The hierarchical layout of application objects and device objects defined the scope of other objects, such as libraries and GVLs.

There are programmable devices and parameterizable devices. The device type defines the possible insertion point in the device tree and the selection of objects that you can insert below the device. Programmable devices automatically get an additional “PLC Logic” node below the device object, which is merely for organizational purposes. Below this node, you insert the objects that are required for programming the device (this means the application(s) and GVLs or text lists, for example). An alternative to this concept is the use of a device application if the device supports this functionality. A device application combines the global data for all applications of a controller so that the applications can focus exclusively on control tasks.
Each device is defined by a device description and must be installed on the local system in order to be inserted into the device tree. The device description file defines the device properties for configurability, programmability, and possible connections to other devices.

The “POUs” view includes objects that can be used throughout the project. Programming objects that are intended for a specific application must be inserted below the application object in the “Devices” view (device tree).

Note the option of letting the active application run on a simulated device, which is provided by default within the development system. Currently, this simulation option is available for the CODESYS Control Win V3 target system. In simulation mode, you can also test the online functionality of the application without hardware. Click “Online ➔ Simulation” to activate simulation mode.

Note the option of establishing a connection to the device by means of the “Online Config Mode” command for an application, without having to load the application beforehand. This is useful for the initial commissioning of an I/O system because you can reference and test the I/Os with it in the PLC configuration, before you have programmed and downloaded the actual application.

Example of a device tree:
A device entry in the device tree consists of a symbol, the symbolic device name that is editable in the tree, and the device type (device name as defined in the device description).

You configure device communication, parameters, and IO mappings in the device editor dialogs. Open this editor by double-clicking the device object.

See also

- Chapter 1.3.1.16 “Using device applications” on page 358
- Chapter 1.3.1.25.2.4.35 “Command ‘Online Config Mode’” on page 915
- Chapter 1.3.1.25.2.4.31 “Command ‘Edit I/O Mapping’” on page 914
- Chapter 1.3.1.25.1.19 “Object ‘Device’ and generic device editor” on page 740
- Chapter 1.3.1.25.1.1 “Object ‘Application’” on page 710
Rules and procedures for arranging and configuring objects in the device tree

- **Insert objects**: To insert a device object, click “Add Device” or “Insert Device” by right-clicking in the device tree. For other objects, click “Add Object”. CODESYS always offers the appropriate objects at the currently selected position in the tree. Example: Modules for a PROFIBUS DP slave can be inserted only below a respective slave object, and applications inserted only below programmable devices. The selection of device objects also depends on which devices are installed in the device repository.

- You can only **insert device objects** on the level directly below the “<project name>” root node. If you select another object type, such as a text list, then CODESYS inserts this into the “POUs” view (project-global pool) automatically.

- **Insert applications**: A distinction is made here between the usual device tree structure with a “PLC Logic” node and the use of a “Device Application”. The following applies for the usual structure (for the device application, see the that chapter of the help).

You can insert an “Application” object only below the “PLC Logic” node (a programmable device). All applications must be named uniquely for each device. Below each application, you can insert more objects that are required for programming, such as POUs, DUTs, GVLs, or visualizations.

Below each application, you must insert a task configuration and configure the respective program calls (from application-specific POUs, or POU instances from the “POUs” view).

If multiple applications exist directly below a device, then you will need to define settings for the I/O processing of the device. This includes the variables of the application that CODESYS uses for communication with the target system. The settings are configured on the “PLC Settings” tab of the device editor.

Hierarchical layout of applications and scopes: You can add an application below another application. We refer to the resulting construct as the parent application and child application. In this case, the following applies: The child application can access objects in the parent application, but not the other direction. The reason for this is that a child application should always be removable or exchangeable without affecting the parent application.

**NOTICE**: After the parent application has been changed, the PLC removes a child application when an online change is performed.

- **Insert devices**: CODESYS inserts a device object in the tree as a node. If nodes are defined in the device description, then they are inserted automatically. A subnode can also represent a programmable device. Order of device objects in the tree (from top to bottom): For each level, the programmable devices (PLC Logic) are listed first and then the other types in alphabetical order.

- **Update devices**: A device that has been inserted into the device tree can be replaced using device applications, 5, en_USed by another version of the same device or by a device of another type (“Update devices”). A configuration tree below the device is available when possible.

- **Moving and deleting objects**: You can use the standard commands “Cut”, “Copy”, “Paste”, and “Delete” on objects, or drag an object to another position. When you copy objects, the new object gets the same name with an incremented number.

- **Network scan of the current hardware**: By default, creating the PLC configuration in the device tree is supported by the device editors with scan functionality. The current hardware environment is scanned and the detected modules are displayed in a dialog. From there, you can save the required device directly to the device tree. See the “Scan Devices” command.

See also

- % Chapter 1.3.1.8.2 “Mapping a hardware structure in the device tree” on page 159
- % Chapter 1.3.1.16 “Using device applications” on page 358
- % Chapter 1.3.1.25.2.4.6 “Command ‘Update Device’” on page 901

Device tree in online mode

In online mode, a symbol before a device entry indicates the device status:

- 📄: The PLC is connected, the application is running, the device is in operation, and data is being exchanged. The “Refresh I/Os in stop” check box on the “PLC Settings” tab can be selected or cleared.

- 📄: The PLC is connected and in “STOP”; and the “Refresh I/Os in stop” check box on the “PLC Settings” tab is cleared.

- 📄: The PLC is connected and the application is running. Diagnostic information is available.
The device is in preoperative mode and is not running yet. Diagnostic information is available.

⚠️: The device is not exchanging data; bus error, no configuration, or simulation mode.

💡: The device is running in demo mode for 30 minutes. When this time has elapsed, demo mode will expire and the fieldbus will end the data exchange.

⚠️: The device is configured, but not fully operational. No data is exchanged. Example case: CANopen devices when booting and in preoperative mode.

⚠️: Redundancy mode is active. The fieldbus master is not sending any data because another master is active.

❓: The device description could not be found in the device repository.

❓: The device itself is running, but a child device is not running. The child device is not visible due to a collapsed device tree.

The names of all connected devices and applications are highlighted in green.

The names of devices that are running in simulation mode are displayed in italics.

Additional diagnostic information is located on the “Status” tab of the respective device editor.

If you login while the device description on the target device is more recent than in the project, then a warning prompt opens with the possibility to cancel the process.

See also
- % Chapter 1.3.1.25.1.19 “Object 'Device' and generic device editor” on page 740
- % Chapter 1.3.1.8.3 “Configuring devices and I/O mapping” on page 162

Device editor

You can configure the settings for communication between CODESYS and the target device on the tabs of the device editor. Double-click the device object in the device tree to open the editor. The editor includes generic as well as specific tabs. Its title contains the device name.

See also
- % Chapter 1.3.1.25.1.19 “Object 'Device' and generic device editor” on page 740
- % Chapter 1.3.1.8.3 “Configuring devices and I/O mapping” on page 162

1.3.1.8.2 Mapping a hardware structure in the device tree

You map the hardware that you control with your application in the “Devices” view (device tree). To do this, insert device objects into this tree structure which represent physical devices in the network until the hierarchy reflects the controller network. Device objects are, for example, a PLC object, a fieldbus object, or a logical device.

Mapping a standard controller with fieldbus in the device tree

Below you will map hardware containing a standard PLC and a standard fieldbus.

1. Click “File ➔ New Project”.
2. In the “New Project” dialog, select the “Standard project” template.
3. In the “Standard Project” dialog, select the “CODESYS Control Win V3” device.

CODESYS inserts the Device “(CODESYS Control Win V3)” object in the device tree under the root node and below it the “PLC Logic” node, which always provides a programmable device object. An “Application” object exists below that, and the “Library Manager”, “PLC_PRG”, and “Task Configuration” objects exist below that.

4. Select the Device entry and click “Add Device” in the context menu.

The “Add Device” dialog provides a selection of the devices installed on your system. 'Installed on the system' means that CODESYS has received a device description for the respective hardware. You can thus use the device object and the associated configuration editor in the project.

5. Select one of the provided PLCs (example: “CAN-bus”) and click the “Add Device” button.

The “CAN-Bus” object appears in the device tree indented below Device “(CODESYS Control Win V3)”. The “Add Device” dialog remains open. If you now select the new “CAN-bus” entry, it automatically offers the objects that are insertable under it.

6. For example, select the “CANopen Device” from the “Local Device” category.

“CANopen_Device” is inserted indented under “CAN-bus” in the device tree.

7. Do you want to add a different device to the one you added before? Select the device entry and click “Update Device” in the context menu.

The “Update Device” dialog opens. It corresponds to the “Add Device” dialog used previously. You can select another device, which is then inserted in the tree in place of the previous one.

See also
- § Chapter 1.3.1.25.2.4.3 “Command 'Insert Device’” on page 898
- § Chapter 1.3.1.25.2.4.6 “Command 'Update Device’” on page 901

You can determine the devices in the network of the hardware environment (scan) and apply the devices found into the device tree of your project. If the scan function is permanently implemented in the PLC, you can perform the scan without further preparations. For this purpose, CODESYS establishes a brief connection to the controller. If the scan function is provided in a library, the device that includes the library must be inserted in the device tree and a login to the controller executed. Only then can the library be placed on the controller, making scanning possible.

Scanning is started by means of the “Scan for Devices” command. It refers to the controller that is currently selected in the device tree and linked to the project. For example, you can select an inserted PROFINET IO controller and use the command to determine the I/O devices and I/O modules assigned to it.

Requirement: Your project has a device configuration. The communication settings are correct. Gateways and hardware are running.
1. Select a PLC object in the device tree.
2. Click “Scan for Devices” in the context menu.
   ⇒ CODESYS establishes the connection to the hardware. The dialog “Scan for Devices” appears. Depending on the type of device it offers different functions. However, a table always appears, showing the devices found in the hardware: device name, type of device, station name, etc. Regarding this, see the help for the respective device editor.
3. So that the list displays only devices found that you do not yet have in the device configuration of your project, activate the “Show differences to project” option.
4. In order to apply a device into the device tree of the project, select the entry in the table and click the button “Copy to project”. All devices found are applied if you do not select an entry.
   ⇒ The corresponding entries are inserted in the device tree.

See also
● Ch. 1.3.1.25.2.4.5 “Command ‘Scan for Devices’” on page 899

Checking the controller configuration with the help of the ‘Online Config Mode’ command

With CODESYS you can test the correct wiring of the I/Os and the fieldbus of your target system, without having to develop a real application for the controller and loading it to the controller beforehand. This is interesting for the initial commissioning phase. Use the ‘Simple online configuration mode’ for this. If there is already an application on the controller, you can optionally also use the ‘Advanced online configuration mode’ if the device supports this. This allows you to read device parameters without having to login with the application concerned from CODESYS.

Simple online configuration mode

You create an implicit application by means of the “Online Config Mode” command in the context menu of the PLC object in the device tree. CODESYS automatically loads the application to the PLC and, via the application, automatically initializes all I/Os once. The application is called “HiddenOnlineConfigModeApp”. It is displayed in the device editor of the PLC on the “Applications” tab. You can use it to operate the I/Os with the following functions as in normal online mode:

- Read I/Os
- Write outputs
- Diagnosis (in the tree and on the status page)
- Scan (of the current hardware)
- Interactive online functions, if supported (for example, writing asynchronous messages)

Writing and forcing in the I/O mapping

In online configuration mode, the writing and forcing of values in the “I/O Mapping” dialog works differently to the way it works in real online mode. CODESYS writes the outputs into the I/O mapping table immediately after insertion. There is no “Prepared Value” column. Instead, you change the initial values directly after you double-click the “Current Value” column.

Testing the I/O access with the help of the simple online configuration mode:

Requirement: You have created a standard project with a device configuration in which you have configured an I/O mapping. In a program block you make read or write accesses to inputs or outputs of the hardware in order to check their wiring. The connection to the PLC of the hardware is configured in the communication settings. The PLC is running.

1. Select the PLC object in the device tree.
   ⇒ For example, the CODESYS Control Win V3 device is selected in the device tree.
2. Click “Online Config Mode” in the context menu.
   - CODESYS connects to the controller and the PLC object in the tree is given a green background.

3. Double-click the PLC object in the device tree in order to open the device editor. Select the tab “Applications”.

4. Click the “Update list” button.
   - The application HiddenOnlineConfigModeApp appears in the window “Applications on the controller”.

5. Start your program and check the behavior of the inputs and outputs.

Advanced online configuration mode (parameter mode)

If there is already a ‘real’ application on the controller and the device supports the ‘Advanced online configuration mode’, you can also optionally select the ‘Parameter mode’ after the command “Online configuration mode”. Then you can read the parameters of the real application on the controller without having to perform a standard login and – if compilation information is missing – having to download this application again. You can see the parameters on the “Configuration” tab of the device editor. The parameter mode prevents the inadvertent changing of the data on the controller; the applications remain untouched. Writing of the parameters is possible only if the driver supports it.

Setting up parameter mode for reading the device parameters:

Requirement: There are already one or more applications on the target device. The target device supports the advanced online configuration mode. You have opened the project containing these applications. You wish to look at the device parameters without having to login to the controller. The connection to the PLC of the hardware is configured in the communication settings. The PLC is running.

1. Select the PLC object in the device tree.
2. Click “Online Config Mode” in the context menu.
   - CODESYS connects to the controller and the PLC object in the tree is given a green background. The dialog “Select configuration mode” appears with a display of the application(s) on the PLC.

3. Select the desired application and click the “Parameter mode” button.
   - CODESYS checks whether the application in the project corresponds to the application on the PLC.

4. If CODESYS does not signal an error regarding the correspondence of the applications to each other, open the “Configuration” tab of the PLC device editor.
   - You can read the device parameters.

See also
   - "Chapter 1.3.1.25.2.4.35 “Command ‘Online Config Mode’" on page 915
   - "Chapter 1.3.1.25.1.19.3 “Tab ‘Parameters’” on page 746"

1.3.1.8.3 Configuring devices and I/O mapping

Configuring devices

You can configure the device objects inserted into the device tree in the associated device editor. The possibilities depend on the device description. The ‘generic device editor’ provides tabs that are supplemented as necessary by device-specific tabs.

Requirement: You have opened a standard project in whose device tree a standard PLC and below that a fieldbus device object are inserted.
1. Double-click the device object of the standard PLC in the device tree of your project.
   - The “<device name>” editor opens in the CODESYS main window. The “Communication Settings” tab is in the foreground. Change to the other tabs in order to make configuration settings for the controller. See the help pages for the generic device editor.

2. Double-click the fieldbus device object in the device tree of your project.
   - The “<fieldbus device name>” editor opens in the CODESYS main window. Specific tabs are available depending on the device. For the configuration options, see the help pages for the respective device editor. If the “Show generic device configuration views” option is selected in “Tools ➔ Options”, in the “Device Editor” category, then see also the tabs contributed by the generic device editor.

See also
- Chapter 1.3.1.25.1.19.1 “Generic device editor” on page 741

Read PLC parameter file to configuration

If the configuration parameters of the PLC have been changed by another device, for example via a visualization, a configuration file IoConfig.par is created on the controller. Depending on the device, a button may be in the parameter dialog of the device editor in online mode for writing the current parameters to such a file. In order to update the changed parameters inside your project as well, you can read in the parameter file in online mode.

Requirement: You have a standard project with a device tree in which the hardware environment is mapped with a PLC and a parameterizable device in the device tree. On the controller there is a file IoConfig.par, in which the device parameters were previously saved with the current values.

1. Add the “Read PLC parameter file to configuration” command from the “Devices” category in the “Tools ➔ Customize” dialog to an interface menu.
2. Select the PLC device object in the device tree and open the context menu.
3. Click “Read PLC parameter file to configuration”.
   - CODESYS automatically establishes a connection to the controller and reads in the parameters from the par-file. These are updated accordingly on the tab with the device parameters.

See also
- Chapter 1.3.1.25.2.4.34 “Command ‘Read PLC Parameter File to Configuration’” on page 915
- Chapter 1.3.1.25.3.14.1 “Dialog ‘Customize’ - ‘Menu’” on page 1089

General information about I/O mapping

Whether or not you can configure an I/O mapping to project variables or even to the entire function blocks depends on the type of device. Configuring an I/O map means linking input and output channels of the device with variables of the project. We also use the term ‘mapping’ for this.

Pay attention in general to the following for the mapping of inputs and outputs of a device to variables in CODESYS:
- You do not have write access to variables that are mapped to an input.
- You can map an existing variable to one input only.
- You can directly generate new global implicit variables in the I/O map and map them to a device channel.
- The memory layout of structures is specified by the device.
- You can change addresses and fix values in the I/O map.
● For each variable that is assigned to an I/O channel in the "I/O Mapping" dialog, you can cause 'force variables' to be generated during the compilation of the application (see further below). Using these variables you can, for example during the commissioning of a plant, force a value on the input or output via a visualization/HMI.

● Changes in the I/O map can be transferred to the controller with an online change.

● If a pointer to a device input is used, the access is considered to be a write access, for example `pTest := ADR(input);`. This leads to a compiler warning when code is generated: "...invalid assignment target". If you require a construct of this kind, you must first copy the input value `input` to a variable with write access.

● An I/O address can also be linked with a variable via the 'AT declaration' in the IEC code. Since a device configuration often changes again, however, we recommend that you make the assignments only in the device editor.

  If you use the AT declaration, note the following:
  - An AT declaration is permissible only with local or global variables, not with input or output variables of function blocks.
  - Implicit 'force variables' for I/Os (see below) cannot be generated for AT declarations.
  - If you use an AT declaration with structure variables or function block variables, all instances will access the same memory location. This then corresponds to the use of 'static variables' in classic programming languages such as 'C'.

<table>
<thead>
<tr>
<th>NOTICE!</th>
</tr>
</thead>
<tbody>
<tr>
<td>If a pointer to a device input is used, then the access (for example <code>pTest := ADR(input);</code>) is considered to be a write access. This leads to a compiler warning when code is generated: &quot;...invalid assignment target&quot;.</td>
</tr>
</tbody>
</table>

If you require a construct of this kind, you must first copy the input value (`input`) to a variable with write access.

As an alternative, you can assign a variable to an address in the programming code using the AT declaration. In view of possible changes of the device configuration, however, we recommend that you make the assignments only in the device editor.

You can export the I/O mapping configuration of a device to a csv file or import it from such a file.

See also

● § Chapter 1.3.1.25.1.19.11 “Tab <device name> I/O Mapping” on page 755
● § Chapter 1.3.1.25.2.4.33 “Command 'Export Mapping to CSV'” on page 915
● § “Generating implicit variables for the forcing of I/Os” on page 170

Linking a device input with an existing project variable ("mapping")

Requirement: A device that supports an I/O mapping configuration in CODESYS is inserted in the device tree of your project. On the "I/O Mapping" tab in the device editor you thus get a tabular display of the input and output channels of the device with specification of the addresses and data types.
NOTICE!

Mapping 'too large' data types

If a variable of a data type that is larger than a byte is mapped to a byte address, the value of the variable will be truncated to byte size there. For monitoring the variable value in the "I/O Mapping" dialog, this means the following: In the root element of the address, the value is displayed that the variable currently has in the project. The current individual bit values of the byte are displayed in succession in the bit elements below that, but this may not be sufficient for the entire variable value.

If a UNION is represented by I/O channels in the mapping dialog, it depends on the device whether mapping to the "root" element is also possible.

1. In a POU, declare, for example, a variable xBool4 of the type BOOL with which you wish to access an input of the target device from the application.
2. To open the device editor, double-click the device object in the device tree, and then the "<device name> I/O Mapping" tab.
3. Observe the “Variable” column with the display of the device input channels 🔄 and device output channels 🔄, which can still be sorted by organizational nodes 🗄, depending on the device. We assume that there is a device input of the type BYTE. It is displayed with its individual bit addresses (bit channels) below the BYTE node.
4. Note: When mapping structured variables, the editor prevents you from entering both the structure variable (example: %QB0) and individual structure elements (example: %QB0.1 and %QB0.2). Therefore, if there is a main entry with a subtree of bit channel entries in the mapping table, then the following applies: Then you can specify a variable either into the line of the main entry, or into the lines of the subelements (bit channels), but not into both. You can now occupy either the entire channel with a variable of a suitable type OR its individual bit-channel addresses with suitable variables of the type BOOL or BIT. First of all, double-click a bit input channel in the “Variables” column.

▷ An input field opens.
5. In order to place an existing variable on the channel, you must enter the desired project variable with the complete path. Press \( \text{ } \) to open the Input Assistant. Select, for example, the variable `Application.PLC_PRG.xBool4` declared in `PLC_PRG`.

The variable is inserted. The HMI symbol (\( \text{ } \)) is displayed in the “Mapping” column. The address is now struck through. That does not mean that the address is no longer available, because values of existing variables are managed at another memory space. But: in order to avoid ambiguities when writing the values, you should nevertheless not occupy the address with a further variable, especially in the case of outputs.

Note: For compiler version V3.5 SP11 and later, the initialization value of the variables is used automatically as the default value when mapping to an existing variable. You can edit the “Default value” field only if you map to a new created variable or if no mapping is specified. In older versions, users had to specify explicitly that the default value and initialization value were identical.

6. Delete the variable assignment again. Click the root of the channel, the `BYTE` node. Use the Input Assistant again to select the variable `Application.PLC_PRG.byte_gotodevice`.

The variable is inserted, all bit addresses of the main channel are struck through and you should not additionally occupy them.

See also
- Chapter 1.3.1.25.1.19.11 “Tab ‘<device name> I/O Mapping’” on page 755
In the following you will map a device output to a global implicit variable, which you recently create for this purpose directly in the “I/O Mapping” dialog.

The “I/O Mapping” dialog is thus a further place for declaring a global variable.

Requirement: A device that supports an I/O mapping configuration in CODESYS is inserted in the device tree of your project. On the “I/O Mapping” tab in the device editor you will thus see a tabular display of the input and output channels of the device with specification of the addresses and data types.

1. To open the device editor, double-click the device object in the device tree, and then the “<device name> I/O Mapping” tab.
2. Click in the mapping table on a channel entry in the “Variable” column in order to open an input field.
3. Enter a simple name (without ‘.’) for a new variable (for example, myBool).

   CODESYS creates the variable as an implicit global variable in the project and assigns it directly to the channel address. Therefore in this case the address does not appear struck through as in the case of mappings to existing variables.

Mapping a device input to a recently created project variable

Linking a device with a function block instance

If supported by the device, you can map entire function blocks to an input or output channel. This allows you to count the frequency of signal changes or scale a channel value for maintenance purposes, for example.

Here you will map a device output channel to a function block. In this example, the block scales the channel output value.

Requirement: A device with a digital output that supports FB mapping is linked in the project. There is a function block “Scale_Output_Int” with the following implementation. The attributes of the function block itself and before the output parameter with which the channel output is processed are important.

```plaintext
{attribute 'io_function_block'}
FUNCTION_BLOCK Scale_Output_Int
VAR_INPUT
  iInput : INT;
  iNumerator : INT;
  iDenominator : INT :=1;
  iOffset : INT := 0;
END_VAR
VAR_OUTPUT
{attribute 'io_function_block_mapping'}
iOutput : INT;
END_VAR
VAR
END_VAR
IF iDenominator <> 0 THEN
  iOutput := TO_INT(TO_DINT(iInput) * TO_DINT(iNumerator) / TO_DINT(iDenominator)) + iOffset;
END_IF
```

2020/12/10 3ADR010583, 1, en_US

167
1. Open the "<device name> I/O Mapping" tab of the device editor. Double-click the output that should be connected to the function block. Click the button "Add FB for IO channel".

   ⇒ The "Select Function Block" dialog opens. On the left side, you see at least the function block "Scale_Output_int" below the "Application" node. Libraries linked in the project that contain corresponding function blocks are also displayed for selection.

2. Select the POU myScaleOutputInt.

   ⇒ After clicking "OK", the path of the function block parameter iOutput in the "Variable" is entered in the mapping dialog. The path comprises the application name, the device channel name, and the selected FB output (example: Appl.Out_4_Int_myScale_Output_Int_1.iOutput).

3. Select the channel and click "Go to Instance".

   ⇒ The focus switches to the "<device name> IEC Objects" tab and the created entry for the new IEC object Out_4_Int myScale_Output_Int_1. In this view in online mode, you see the current value of the parameter iOutput for the channel Out_4_Int scaled by the FB. You can also write and force the value as in other monitoring views.

See also

- Chapter 1.3.1.25.3.3 "Dialog ‘Select Function Block’" on page 1040
- Chapter 1.3.1.25.1.19.11 "Tab ‘<device name> I/O Mapping’" on page 755
- Chapter 1.3.1.25.1.19.12 "Tab ‘<device name> IEC Objects’" on page 759
- Chapter 1.3.1.24.6.2.19 "Attribute ‘io_function_block’, ‘io_function_block_mapping’" on page 617

Changing and fixing an address value in the I/O map

You can change the address value of an entire channel (but not that of an individual subelement of the channel!) in the mapping table of the "<device name> I/O Mapping" tab. This allows you to adapt the addressing to a specified machine configuration and to retain the address value even if the layout of the modules changes. By default, a change of the layout leads to an automatic adaptation of the address values.

Requirement: Your project has I/O mapping. See the corresponding sections of the help page above.

1. To open the device editor, double-click the device object in the device tree, and then the "<device name> I/O Mapping" tab.

2. Click in the mapping table on a channel entry in the “Address” column in order to open an input field. This is only possible for the 'root' address of a channel, not for a particular one of its subelements.

   Therefore, change the top address entry of a channel in the table, for example from QB0 to QB1. Exit the input field.

   ⇒ The address value is changed. The symbol is displayed before the address. It indicates that the address is fixed. The addresses of the subelements of the channel are also changed accordingly. If you now change the position of the device object inside other device objects with input/output channels in the device tree, CODESYS does not adapt these addresses to the new order as would be the case without fixing.
3. In order to undo the manual change or fixing, open the input field of the address value again, delete the address entry and press the Enter key.

⇒ CODESYS resets the address and the subsequent addresses concerned to the values they had before the change and removes the symbol 🗑.

See also
- § Chapter 1.3.1.25.1.19.11 “Tab `<device name> I/O Mapping’” on page 755
- § Chapter 1.3.1.24.4.12 “Addresses” on page 566

Configuration of the I/O variable update

Depending on the device that you link in the project, CODESYS updates the variables applied to its inputs and outputs in different ways. You can explicitly change the settings for this in the “I/O Mapping” dialog.

See also
- § Chapter 1.3.1.25.1.19.11 “Tab `<device name> I/O Mapping’” on page 755

Monitoring of variables in the I/O map in online mode

Requirement: You have compiled an application with a device configuration containing I/O maps without error. The associated hardware and the bus system are running. You have connected to the controller by means of the “Online ➤ Login” command and have loaded and started the application.

1. Open the “I/O Mapping” tab of the PLC in the device editor. To open the editor, double-click the device object in the device tree.

⇒ The mapping table now additionally contains the “Current Value” and “New Value” columns.

   If a structure variable is mapped to the ‘root’ element of an address¹, CODESYS does not display a value in this line in online mode. If, for example, a DWORD variable is mapped to the address, however, the respective values are monitored both in the ‘root’ line and in the indented bit-channel lines below it.

   As a matter of principle, the field in the ‘root’ line always remains empty if the value would be composed of several subelements.

¹ ‘root’ = top element of this address in the Mapping dialog

2. Enter a certain variable value for an entry in the column “New value” and press [F7] to force or [Ctrl]+[F7] to write the value.

⇒ As in the case of monitoring in the declaration editor or in watch lists, the forced variable value is displayed in the column “Current Value” with a prefixed red F-symbol or the written value.

NOTICE!

Inputs and outputs that the PLC code does NOT use are not read by the PLC in online mode, as a result of which the displayed value could be incorrect. The “Current Value” of the variables concerned is displayed with a gray background.
Forced values on the controller

Values not used on the controller, value shown in gray

During the commissioning of a plant or machine it may be necessary to 'force' the values applied at the inputs and outputs. If a device supports this you can cause special 'force variables' to be generated for this purpose and use them, for example, in an HMI visualization.

Requirement: The device supports the functionality. You have a project in which an I/O map is configured for the device and which contains a program object PLC_PRG.

1. Open the device editor, "PLC Settings" tab, by double-clicking the device object in the device tree.
2. Activate the option “Generate force variables for IO mapping”.

Two variables are created for each I/O channel in accordance with the following syntax, in the process of which spaces in the channel name are replaced by underscores:

<device name>_<channel name>_<IEC address>_Force of type BOOL for the activation and deactivation of forces.

<device name>_<channel name>_<IEC address>_Value of the data type of the channel for defining the value that you wish to force on the channel.

These variables are available in the Input Assistant in the category "Variables" / "IoConfig_Globals_Force_Variables." You can use them in CODESYS in programming objects, in visualizations, in the symbol configuration, etc.

4. Open the function block “PLC_PRG”, set the focus in the implementation part and press F2.

The Input Assistant opens. The variables are available in the category "Variables" / "IoConfig_Globals_Force_Variables" as described above.

A rising edge at the 'Force variable' input activates the forcing of the respective input or output with the value given by the 'Value variable'. A falling edge deactivates the forcing. Deactivation by resetting the 'Force' variable to FALSE is the requirement for being able to force a new value.

Take note of the following restrictions.

- Forcing via the implicit force variables is only possible for channels that are mapped in the "I/O Mapping" of the device to an existing or recently created variable.
- Forcing via the implicit force variables is not possible for unused inputs and outputs or those that are mapped to a variable via an AT declaration in an application program.
- I/O channels that you wish to force via the mechanism have to be used by CODESYS in at least one task.
- CODESYS identifies forced inputs in the monitoring by the red Force symbol, but not forced input/outputs! The forced value is used only implicitly by the I/O driver for writing to the device.
See also
- Chapter 1.3.1.25.1.19.9 “Tab 'PLC Settings’’ on page 751
- Chapter 1.3.1.12.4 “Forcing and writing variables” on page 315

I/O mapping in one dialog for multiple devices

There is a table that displays the I/O map of a device plus the I/O maps of all subelements inserted below it in the device tree. There you can edit the I/O maps in exactly the same way as in the individual mapping tables of the respective device editors.

Requirement: In the device tree of your project there are several PLCs inserted that each enable an I/O mapping configuration.

1. Select the root node of the device tree and click “Edit I/O Mapping” in the context menu.
   - The “Edit I/O Mapping” dialog opens, in which the I/O mapping configurations of all devices inserted in the project are displayed in a table. You can edit the entries in the same way as in the “I/O Mapping” dialog of the associated device editor.

2. Now select one of the control objects in the device tree and select the “Edit I/O Mapping” command once again in the context menu.
   - The “Edit I/O Mapping” dialog now shows only the I/O table for the I/O mapping configurations found in and under the selected object.

3. Set a desired “Filter” in the bar above the table or enter a variable name in the “Search for variable” field in order to see the use of this variable in the mapping.
   - The method of working in this window is the same as that described for the “<device name> I/O Mapping” tab.

See also
- Chapter 1.3.1.25.2.4.31 “Command ‘Edit I/O Mapping’” on page 914

1.3.1.9 Programming of applications

For creating an application program which is executable on the PLC, you fill POUs with declarations and implementation code (source code), you define the mapping of the hardware I/Os on application variables, and you configure the task handling. After having checked and debugged your implementation, the CODESYS compiler will generate the application code which can be downloaded to the PLC.

The programming of the application POUs will be supported by the programming language editors and further features like for example textlists, image pools, alarm configuration, pragmas, refactoring and usage of ready modules provided by CODESYS Development System or by libraries.

1.3.1.9.1 Designating identifiers

Identifiers are names of variables and programming objects (for example programs, function blocks, and methods) and names of other objects of the application and project. There are rules that you must follow when assigning identifiers. Furthermore, there are also recommendations to help you designate uniform and expressive identifiers.

You designate variables identifiers in the variables declaration. These identifiers can e changed in the declaration section of the programming object. You designate identifiers for programming objects and other objects in the dialog box when adding the object. You can change the identifier of an existing object of the application or of the project in the properties dialog of the object. However, you cannot change the identifiers of objects that can be available only one time per application or project (for example, the “Library Manager” and “ImagePool” identifiers).

See also
- Chapter 1.3.1.24.7 “Identifiers” on page 644
1.3.1.9.2 Declaration of variables

Variable declaration: where and how

In CODESYS projects you can declare variables in the following places:

- Declaration part of a POU
- Dialog box “Auto Declare”, if a POU has the focus
- DUT, GVL or NVL editor
- I/O mapping configuration of an I/O device object

The variable declarations in global variable lists (GVL, NVL) and in the declaration part of POUs are carried out in the declaration editor. Special editors or dialog boxes are available for all other variants.

NOTICE!

By adding pragmas you can affect the behavior and the properties of one or more variables. More precise information can be found in the chapter "Use of pragmas".

Syntax for the variable declaration in POUs or global variable lists, if textual display is selected for the declaration editor:

```
<identifier> {AT <address>}:<data type> {:=<initialisation>};
```

- `<identifier>`: Name of the variable. The rules listed in the chapter "Identifiers" must be followed without exception when assigning an identifier. In addition, this chapter contains recommendations for standardizing the assignment of names.
- `{AT <address>}` (optional): You can directly bind a variable to a certain address with the help of the keyword AT.
- `<data type>`
- `{:=<initialisation>}` (optional)

In function blocks you can also declare variables with incomplete address specifications. In order to be able to use such a variable in a local instance, however, there must be a corresponding entry for this variable in the "variable configuration".

If you define a variable in the tabular declaration editor, the correct syntax is automatically produced.

See also

- Chapter 1.3.1.9.2.1 “Using the declaration editor” on page 173
- Chapter 1.3.1.24.5 “Data types” on page 569
- Chapter 1.3.1.24.2 “Variables” on page 489
- Chapter 1.3.1.25.2.2.31 “Command ‘Auto Declare’” on page 872
- Chapter 1.3.1.24.7 “Identifiers” on page 644
- Chapter 1.3.1.25.1.21 “Object ‘GVL’ - Global Variable List” on page 769
- Chapter 1.3.1.9.7 “Using pragmas” on page 211
- Chapter 1.3.1.9.2.2 “Using the ‘Declare variable’ dialog box” on page 174

Variable initialization

The standard initialization value for all declarations is 0. In the declaration part you can also specify user-defined initialization values for each variable and each data type.

The user-defined initialization starts with the assignment operator `:=` and consists of any valid expression of the programming language ST (structured text). You thus define the initialization value with the help of constants, other variables or functions. If you use a variable, you must also initialize it.
VAR

  var1: INT := 12;                 (* initialization value 12*)

  x : INT := 13 + 8;              (* initialzation value defined by an expression of constants*)

  y : INT := x + fun(4);          (* initialization value defined by an expression, that contains a function call; notice the order! *)

  z : POINTER TO INT := ADR(y);   (* not described in the standard IEC61131-3; initialization value defined by an adress function; Notice: the pointer will not be initialized during an Online Change *)

END_VAR

Examples

Notes on the order of initialization

From compiler version 3.5.3.40, variables in a function block are initialized in the following order: firstly, all constants in accordance with the order of their declarations, then all other variables in accordance with the order of their declarations.

NOTICE!

From compiler version 3.3.2.0, variables from global variable lists are always initialized before the local variables of a POU.

See also

● § Chapter 1.3.1.24.5.14 “Data type `ARRAY’” on page 581
● § “Initializing structures” on page 594
● § Chapter 1.3.1.24.5 “Data types” on page 569

Using the declaration editor

The declaration editor is used for declaring variables in the variable lists and POUs. The declaration editor offers two possible views: textual and tabular.

in the dialog in “Tools -> Options -> Declaration Editor”, you define whether only the textual view or only the tabular view is available, or whether you can switch between both views by means of the buttons on the right side of the editor view.

If the declaration editor is used in conjunction with a programming language editor, it appears as the declaration part at the top of the window of a POU.

Declaring in the textual declaration editor

The behavior and the appearance of the textual editor are configured with the settings in the dialog “Tools -> Options -> Text Editor”. The settings concern colors, line numbers, tab widths, indentations etc. The usual Windows functions are available, plus the IntelliMouse functions if necessary.

Requirement: You have opened a programming object (POU, GVL or NVL) of a project. The textual declaration editor has the focus.
Enter the variable declarations in correct syntax. With [F2] you can open the dialog "Input Assistant" for the selection of the data type or a keyword.

Declaring in the tabular declaration editor

In the tabular declaration editor, you add variable declarations to a table with the following columns: "Scope", "Name", "Address", "Data type", "Initialization", "Comment", and "Attributes" (pragmas).

Requirement: A programming object (POU or GVL) of a project is open. The tabular declaration editor has the focus.

1. Click the button in the declaration header or select the command "Insert" in the context menu.
   - CODESYS inserts a new row for a variable declaration and the input field for the variable name opens.
2. Specify a valid variable identifier.
3. Open the other fields of the declaration line as required with a double-click and select the desired specifications from the drop-down lists or with the help of the dialogs which appear.

See also

- Chapter 1.3.1.24.1.1 “Declaration editor” on page 428
- Chapter 1.3.1.25.2.2.31 “Command ‘Auto Declare’” on page 872
- Chapter 1.3.1.9.2.2 “Using the ‘Declare variable’ dialog box” on page 174
- Chapter 1.3.1.25.2.16.2 “Command ‘Edit Declaration Header’” on page 1009
- “Dialog ‘Input Assistant’ - Tab ‘Categories’” on page 874

Using the ‘Declare variable’ dialog box

Requirement: A programming object (POU or GVL) of a project is open.

1. Select the command “Edit ➔ Auto Declare”.
   - The dialog box “Auto Declare” opens.
2. Select the desired scope for the variable from the selection list “Scope”.
3. Enter a variable name in the input field “Name”.
4. Select the desired data type from the selection list “Type”.
5. If the initialization value deviates from the standard initialization value, enter an initialization value for the variable.
6. Complete your entries with a click on “OK”.
   - CODESYS lists the newly declared variable in the declaration part of your programming object.

With the help of pragmas in the declaration part you can affect the processing of the declaration by the compiler.

See also

- Chapter 1.3.1.25.2.2.31 “Command ‘Auto Declare’” on page 872
- Chapter 1.3.1.24.5 “Data types” on page 569
- Chapter 1.3.1.9.7 “Using pragmas” on page 211
Declaring arrays

Requirement: A programming object (POU or GVL) of a project is open.

1. Click “Edit ➔ Declare Variable”.
   ⇒ The “Declare Variable” dialog opens.
2. Select the desired scope for the array from the drop-down list “Scope”.
3. Enter an identifier for the array in the “Name” input field.
4. Click the arrow button (>>) next to the “Data type” input field and select the “Array Assistant” entry from the selection menu.
5. In the input fields “Dimension 1”, type in the lower and upper limit of the first dimension of the array (example: 1 and 3).
   ⇒ The field “Result” displays the 1st dimension of the array (example: ARRAY [1..3] OF ?).
6. In the input field “Basic type”, type in the data type of the array or use the “Input Assistant” ( ) or the “Array Assistant” (example: DINT).
   ⇒ The field “Result” displays the data type of the array now (example: ARRAY [1..3] OF DINT).
7. Define the second and third dimensions of the array according to steps 5 and 6 (example: Dimension 2: 1 and 4, Dimension 3: 1 and 2).
   ⇒ The “Result” field displays the array with the defined dimensions: ARRAY [1..3, 1..4, 1..2] OF DINT. The array consists of 3 * 4 * 2 = 24 elements.

In an array of variable length, declare the dimension limits with an asterisk placeholder (*). Arrays of variable length are permitted to be used only in VAR_IN_OUT declarations of function blocks, methods, or functions.

Example of a 2-dimensional array of variable length:
aiUnknownLengthData : ARRAY [*,*] OF INT;

8. Click “OK”.
   ⇒ In the dialog “Declare Variable” the field “Data type” displays the array.
9. To modify the initialization values of the array, click the arrow button (<<) next to the “Initialization value” input field.
   ⇒ The “Initialization Value” dialog opens.
10. Select the line of the array element whose initialization value you wish to modify. Example: Select array component [1, 1, 1].
11. Enter the desired initialization value in the input field below the list and click button “Use value on selected lines” (example: value 4).
   ⇒ CODESYS displays the changed initialization value of the selected line.
12. Click “OK”.
   ⇒ In the “Initialization value” field of the “Declare Variable” dialog, CODESYS displays the initialization values of the array (example: (4, 23(0))).
13. You can optionally enter a “Comment” in the input field.
14. Click “OK” in order to conclude the declaration of the array.
   ⇒ CODESYS adds the declaration of the array to the declaration part of the programming object.
Declaring global variables

Declaring global variables that are available within the application.

1. In the Device tree of your project, select the application in which the global variables are to be valid.
2. Select the context menu command “Add Object ➤ Global Variable List”.
   ⇒ CODESYS inserts the “GVL” in the Device tree under the application and opens it in the editor.
3. Select the menu command “Edit ➤ Auto Declare”.
   ⇒ The dialog box “Auto Declare” opens.
4. In the selection list “Scope”, select the entry “VAR_GLOBAL”.
5. In the field “Name”, enter a name for the global variable.
6. Select a data type from the selection list “Type”.
7. If your variable is to have an initialization value other than the standard initialization value, click on next to the field “Initialization”.
   ⇒ The dialog box “Initialization Value” opens.
8. Double-click on the cell “Init value” of your variable and enter the desired valid value.
9. Click on “OK”.
   ⇒ The initialization value is displayed in the dialog box “Auto Declare”.
10. Activate one of the “Flags” if necessary.
11. Confirm your entries by clicking on the button “OK”.
   ⇒ CODESYS inserts the declared variable in the GVL.

   The global variable is available in the total application of your project.

Declaring global variables that are available in the entire project.

1. Select the menu command “View ➤ POU’s”.
   ⇒ The “POUs” view opens.
2. In the “POUs” view, select the uppermost node with the project name and select the context menu command “Add Object ➤ Global Variable List”.
   ⇒ CODESYS inserts the “GVL” in the “POUs” view and opens it in the editor.
3. Select the menu command “Edit ➤ Auto Declare”.
   ⇒ The dialog box “Auto Declare” opens.
4. In the selection list “Scope”, select the entry “VAR_GLOBAL”.
5. In the field “Name”, enter a name for the global variable.
6. Select a data type from the selection list “Type”.
7. If your variable is to have an initialization value other than the standard initialization value, enter it in the column “Initialization”.
8. Activate one of the “Flags” if necessary.
9. Confirm your entries by clicking on the button “OK”.
   ⇒ CODESYS inserts the declared variable in the GVL.
   The global variable is now available in the entire project.

See also
● ☞ Chapter 1.3.1.25.2.2.31 “Command ‘Auto Declare™’ on page 872

Using task-local global variable lists

You can insert task-local global variable lists in the project in order to synchronize variables over multiple tasks. A task-local global variable list is cycle-consistent. The contents of the list are modified by only one task during a task cycle. Only this task writes to the variables in the list. All other tasks have read-only access to the list.

You declare task-local variables in the object ☞ “Global Variable List (task-local)” as a list, which syntactically is the same as a global variable list. There you also determine the task that has write access.

Implementing an application

☐ Requirement: CODESYS is open and a new standard project is created.
1. Select the “PLC Logic” node in the “Devices” view. Click “Add Object ➔ Application” in the context menu.
   ⇒ The “Add Application” dialog opens.
2. Specify the name appTasklocal.
3. Select the application and click “Add Object ➔ POU” in the context menu.
   ⇒ The “Add POU” dialog opens.
4. Specify the name ReadDate. Select “Program” as the type. Select “Structured Text” as the implementation language. Click “Add” to close the dialog.
   ⇒ The program ReadData is created.
5. Select the application again and click “Add Object ➔ POU” in the context menu.
6. Specify the name WriteData. Select the “Program” options as the type. Select “Structured Text” as the implementation language. Click “Add” to close the dialog.
   ⇒ The program WriteData is created.
7. Select the application and click “Add Object ➔ Task Configuration” in the context menu.
   ⇒ The task configuration is created with the standard task Task.
8. Select the standard task and click “Properties” in the context menu.
9. Specify the name Read in the input field.
   ⇒ The task is renamed. Moreover, the “Configuration” tab is open in the editor.
10. Click “Add Call” in the editor.
    ⇒ The “Input assistant” dialog opens.
11. Select the “ReadData” program in the configuration tree. Click “OK” to close the dialog.
   ✐ The Read task calls the ReadData program at runtime.

12. Select the task configuration and click “Add Object ➔ Task” in the context menu.
   ✐ The “Add Task” dialog opens.

13. Specify the name Write in the input field. Click “Add” to close the dialog.
   ✐ The Write task is created. The “Configuration” tab is opened in the editor.

14. Click “Add Call” in the editor.
   ✐ The “Input Assistant” dialog opens.

15. Select the “WriteData” program in the configuration tree. Click “OK” to close the dialog.
   ✐ The Write task calls the WriteData program at runtime.

16. Select the appTasklocal application and click “Add Object ➔ Global Variable List (Task-Local)” in the context menu.
   ✐ The “Add Global Variable List (Task-Local)” dialog opens.

17. Specify the name Tasklocals.

18. Select the Write task from the “Task with write access” drop-down list.

   ✐ The object structure for using task-local variables within an application is complete.

   Now you can code the objects.

   The application contains a task responsible for writing the data and a task responsible for reading the data.
The `diaData` array is populated with values in the `WriteData` program. The `ReadData` program tests whether or not the values of the array are as expected. If so, then the `bTest` variable yields the result `TRUE`.

The tested data is declared in the variable `g_diaData` as task-local GVL. This causes data access to be synchronized in the computer. In turn, this guarantees that the data is with cycle consistency even if the accessing programs are called from different tasks. It is taken into account that tasks can be interrupted by other tasks or can run simultaneously. The guarantee also applies above all if the application is running on a system with a multicore processor.

If the variable `g_diaData` is declared as a global variable list only and the `WriteData` and `ReadData` programs are called from different tasks, then the test will yield `FALSE` more often. Because then for example the values in `WriteData` can be changed while the values in `ReadData` are being read.

When using task-local variables, the compiler synchronizes the data of the variables, which takes a lot of time and memory. The synchronization is a complex task that is suitable above all for multicore operation. Select this data type only if sufficient resources are available. Then having this kind of data consistency is a convenient option for you without worrying about data synchronization. You can concentrate on the actual task of implementing the functionality.

Restrictions in the declaration

In the global task-local variable list, pay attention to the following:

- Do not assign direct addresses.
- Do not map direct addresses.
- Do not declare pointers.
- Do not declare references.
Do not instantiate function blocks.

Do not declare remanent variables as PERSISTENT and RETAIN at the same time.

Example:

```
VAR_GLOBAL RETAIN g_RemenantData END_VAR
```

When you edit the declaration in the list of task-local variables, you cannot login to the application afterwards with an online change.

Data access

Compiler errors:
The compiler reports write access in a task without write access as an error. However, not all write-access violations can be detected. The compiler can only assign static calls to a task. The call of a function block by means of a pointer or an interface is not assigned to a task, for example. Therefore, no write access is recorded there either. Moreover, pointers can point to task-local variables. Therefore, data can be manipulated in a read task. In this case, a runtime error is not issued. However, values that are modified by means of pointer access are not copied back in the shared reference of variables.

Read access:
Task-local variables are located at a different address in the list for each task. As a result, the ADR operator applied for a task-local variable yields another address in each task.

Controlling the task execution

A scheduler controls the time-based execution of multiple tasks and therefore also task switching. The strategy, which is tracked by the scheduler in order to control the allocation of the execution time, has the goal of preventing a task from being blocked. The synchronization mechanism is optimized to prevent blocking states (lock states) and at no time does a task wait for the action of another task.

Scheduling strategy:
- As long as the writing task saves a copy back to the shared reference, none of the reading tasks gets a copy.
- As long as a reading task gets a copy of the common reference, the writing task does not write back a copy.

Due to the scheduling method, no time can be determined when a reading task securely receives a copy of the writing task. In principle, the copies can run apart from one another. During the runtime of the sample application `appTasklocal`, you cannot expect each written copy to be edited once by the reader. For example, the reading task can edit the same array over multiple cycles, or the contents of the array can skip one or more values between two cycles. Pay attention to these effects of task switching.

For task-local variables on the other hand, it is guaranteed that task switching runs with cycle consistency.

A task with write access is halted for example between two accesses to the shared reference by a reading task for the duration of one cycle. Therefore, if n reading tasks exist, then the writing task can have n cycles of delay until the next update of the shared reference.

A writing task can also prevent a reading task from getting a reading copy in each cycle. Consequently, no upper limit can be specified for the number of cycles after which a reading task will definitely receive a copy. This can lead to problems especially when the reading task runs very slowly. For example, if a task runs only every hour and cannot access the task-local variables during this time, then the task works with a very old copy of the list.

You can remedy this by adding a time stamp for each task-local variable. In this way, you can at least determine whether the reading tasks access a current list. For example, add a variable of data type LTIME to the list of task-local variables and add the following code to the writing task:

```
tasklocal.g_ltimStamp := LTIME();
```
The compiler creates a copy for a list of task-local variables for each task. This creates a structure with the same variables as those in the list of task-local variables. Moreover, an array with this structure is created in which a dimension is created for each task. As a result, an array element is indexed for each task. If a variable in the list is accessed now in the code, then the task-local copy of the list is actually accessed. Furthermore, it is determined in which task the block is currently running and then the access is indexed accordingly.

For example, the line of code `diValue := TaskLocals.g_diaData[0];` from the above example is replaced by:

```c
diValue := __TaskLocalVarsArray[__CURRENTTASK.TaskIndex].__g_diarr[0];
```

`__CURRENTTASK` is an operator that is available in CODESYS V3.5 SP13 and later in order to determine the current task index quickly.

At runtime for the writing task at the end, the contents of the task-local list are written to the global list. For a reading task at the beginning, the contents of the global list are copied to the task-local copy. Therefore, for n tasks, there are n+1 copies of the list: One list serves as a shared reference and every task also has its own copy of the list. In addition, copying to or from the shared reference is also synchronized.

---

When deciding whether or not synchronization is the appropriate technique for your system, consider that the technical implementation requires more effort.

---

Task-local variables are designed for the use case "Single writer - multiple readers". When you implement a code that is called by different tasks, using task-local variables is a significant advantage. For example, this is the case for the sample application appTasklocal when it is extended by multiple reading tasks that all access the same array and use the same functions.

Task-local variables are especially useful on multicore systems. On these systems, you cannot synchronize tasks by priority. Then other synchronization mechanisms become necessary.

Do not use task-local variables when a reading task always works on the newest copy of the variable. Task-local variables are not suitable for this purpose.

A similar issue is the "Producer - Consumer" dilemma. This happens when a task produces data and another task processes the data. Choose another type of synchronization for this configuration. For example, the producer could use a flag to notify that a new date is present. Then the consumer uses a second flag to notify that it has processed its data and is waiting for new input. In this way, both can work on the same data. This removes the overhead for cyclic copying of data, and the consumer does not lose any data generated by the producer.

---

At runtime, there may exist multiple different copies of the task-local variable list in memory. When monitoring a position, not all values can be displayed. Therefore, the values from the shared reference are displayed for inline monitoring, in the monitoring list, and in the visualization for a task-local variable.

If you set a breakpoint, then the data of the task is displayed that ran to the breakpoint and was halted as a result. Meanwhile, the other tasks continue running. Under certain circumstances, the shared copy can be changed. In the context of the halted task, however, the values remain unchanged and are displayed as they are. You must be aware of this.

---

See also

- Chapter 1.3.1.25.2.2.31 “Command ‘Auto Declare’” on page 872
- Chapter 1.3.1.9.2.4 “Declaring global variables” on page 176
- Chapter 1.3.1.25.1.22 “Object ‘GVL’ - Global Variable List (task-local)” on page 769
1.3.1.9.3 Creating source code in IEC

Source code:

"Source code" is a term used for the implementation code, which you insert in the programming modules by using the appropriate programming language editors. The following programming module types are available for this purpose: POU (Program, Function, Function Block), Action, Method, Property, Interface.

Programming Language:

When creating a POU, you define, in which programming language the implementation should be inserted. Besides the IEC languages also CFC is available.

Programming Language:Editors:

You get a programming module editable in the corresponding programming language editor on a double-click on the programming module object. So, the module will appear either in the textual ST editor or in one of the graphical editors for FBD/LD/IL or CFC. Each editor consists of two windows: In the upper window you insert the declarations, in textual or tabular form, depending on the setting. In the lower window you insert the implementation code. The display and behaviour of each editor can be configured in the corresponding tab of the CODESYS "Options" dialog.

Regard the possibility to open a programming module for offline-editing even while the application is in online mode.

See also

● Chapter 1.3.1.24.1 “Programming languages and editors” on page 428

● Chapter 1.3.1.25.2.4.9 “Command ‘Edit Object (Offline)’” on page 902

FBD/LD/IL

A combined editor enables programming in the languages FBD (function block diagram), LD (ladder diagram) and IL (instruction list).

The basic unit of the FBD and LD programming is a network. Each network contains a structure that can represent the following: a logical or arithmetic expression, the call of a POU (function, function block, program etc.), a jump or a return instruction. IL actually requires no networks. In CODESYS, however, an IL program also consists of at least one network in order to support conversion to FBD or LD. In view of this you should also divide an IL program meaningfully into networks.

See also

● Chapter 1.3.1.24.1.5.1 “FBD/LD/IL editor” on page 461

Function block diagram (FBD)

The function block diagram is a graphically oriented IEC 61131 programming language. It works with a list of networks, where each network contains a structure that can contain logical and arithmetic expressions, calls of function blocks, a jump or a return instruction.

Boxes familiar from boolean algebra are used here. Boxes and variables are connected by connecting lines. The signal flow in the network runs from left to right. The signal flow in the editor runs from top to bottom, starting with network 1.

Example
CFC is also a programming language based on the same principle as FBD, but with the following differences:

– The CFC editor is not network-oriented.
– You can freely place the elements in the CFC editor.
– Direct insertion of feedbacks is possible.
– The order of execution is determined by a list of currently inserted elements, which you can change.

See also

● § Chapter 1.3.1.9.3.1.1 “Programming function block diagrams (FBD)” on page 184
● § Chapter 1.3.1.25.2.13 “Menu ‘FBD/LD/IL’” on page 992

Ladder diagram (LD)

The ladder diagram (LD) is a graphically oriented programming language that approximates an electrical circuit diagram. On the one hand the ladder diagram is suitable for designing logical switching units, but on the other you can also create networks just as in FBD. Therefore you can use LD very well for controlling calls of other program blocks.

The ladder diagram consists of a series of networks. A network is bounded on the left side by a vertical line (bus bar). A network contains a circuit diagram of contacts, coils, optional boxes (POUs) and connecting lines. On the left side of a network there is a contact or a series of contacts that relay the ON or OFF state, which corresponds to the boolean values TRUE and FALSE, from left to right. A boolean variable is associated with each contact. If this variable is TRUE, the status is relayed from left to right via the connection line. Otherwise OFF is relayed. Thus the coil(s) in the right part of the network receive(s) the value ON and OFF coming from the left and the value TRUE or FALSE is written accordingly into the boolean variable assigned to them.

If the elements are connected in series, this means an AND operation. If they are connected in parallel, this means an OR operation. A line through an element means a negation of the element. The negation of an input or an output is indicated by a circle symbol.

Example

IEC 61131-3 defines a complete LD command set, consisting of different types of contacts and coils. Contacts conduct the current (according to their type) from left to right. Coils store the incoming value. Contacts and coils are assigned to boolean variables. You can supplement an LD network by jumps, returns, labels and comments.

See also

● § Chapter 1.3.1.9.3.1.2 “Programming ladder diagrams (LD)” on page 186
● § Chapter 1.3.1.25.2.13 “Menu ‘FBD/LD/IL’” on page 992

Instruction list (IL)

The instruction list is an assembler-like IEC 61131-compliant programming language. It supports accumulator-based programming.

An instruction list (IL) consists of a series of instructions. Each instruction starts in a new line and contains an operator and, depending on the type of operation, one or more operands separated by commas. A label, followed by a colon, can be placed in front of an instruction. It serves the identification of the instruction and you can use the label as a jump destination. A comment must be the last element in a line. Empty lines can be inserted between instructions.
All IEC 61131-3 operators are supported, as are multiple inputs, multiple outputs, negations, comments, set/reset of outputs and conditional/unconditional jumps.

Each instruction is based primarily on the loading of values into the accumulator (LD instruction). After that the corresponding operation is executed with the parameter from the accumulator. The result of the operation is written again into the accumulator, from where you should store it purposefully with the help of an ST instruction.

The instruction list supports comparison operators (EQ, GT, LT, GE, LE, NE) and jumps for programming of conditional executions or loops. Jumps can be unconditional (JMP) or conditional (JMPC / JMPCN). In the case of conditional jumps, a check is performed as to whether the value in the accumulator is TRUE or FALSE.

Example

![Example Image]

See also

- Chapter 1.3.1.9.3.1.3 “Programming in instruction list (IL)” on page 187
- Chapter 1.3.1.24.1.5.3 “Modifiers and operators in IL” on page 466

Programming function block diagrams (FBD)

Creating a POU in the function block diagram (FBD) implementation language

1. Select an application in the device tree.
2. Select the command “Project ➔ Add Object ➔ POU”.
   ⇢ The dialog box “Add POU” opens.
3. Enter a name and select the implementation language “Function Block Diagram (FBD)”. Click on “Add”.
   ⇢ The POU is added to the device tree and opened in the editor. It consists of the declaration editor in the top part and the implementation part with an empty network in the lower part. The view “ToolsBox” is also automatically opened, in which the suitable elements, operators and function blocks for FBD programming are available.
Programming a network

1. Click inside the automatically inserted empty network in the implementation part.
   ⇒ The network is given a yellow background and the area at the left-hand side with the network number is given a red background.
2. Open the context menu with the right mouse button.
   ⇒ You obtain amongst other things the insert commands for the elements that can be inserted at this point.
3. Insert the elements required for your programming using the menu commands or by dragging in the elements from the toolbox.
4. For example, select the command “Insert Assignment”.
   ⇒ An assignment line is inserted. In each case three question marks stand for assignment source and assignment target.
5. Select the question marks and replace them with the desired variable. Input assistance is available for this purpose.
6. Move the cursor over the assignment line.
   ⇒ The possible insertion positions for further elements are displayed as grey diamonds. A click on a diamond selects that position and the suitable insert commands are once again available.
7. Alternatively, you can drag an element with the mouse from the toolbox into the network. For example, click in the tool box on the box element, keep the mouse button pressed and move the cursor over the network.
   ⇒ Each possible insertion position lights up green.
8. Release the mouse button in order to insert the box.
   ⇒ The box is displayed in the network. The type of box on the inside and the instance name above the box, which is required in the case of a function block, are still kept free with three question marks.
9. Select the string ??? inside the box and replace it with the name of the box. Input assistance is available for this purpose.
   ⇒ The inputs and outputs of the selected box are displayed. They are still kept free with question marks, as is the instance name in the case of a function block.

See also

- Chapter 1.3.1.25.1.28 “Object ‘POU’” on page 775
- Chapter 1.3.1.9.3.1 “FBD/LD/IL” on page 182

Programming line branches (subnetworks)

1. In the implementation part of your POU, insert a new network using the command “FBD/LD/IL ➔ Insert network” or drag it in from the tool box.
2. For example, drag an “ADD” operator into the empty network and replace the characters ??? with two variables of the type INT.
3. Drag the element “Branch” from the tool box into your implementation and release the mouse button at the green insertion position directly at the output of the operator.
   ⇒ The line branch splits the processing line at the output of the operator box into 2 sub-networks.

4. Further FBD elements and also further line branches can now be added to each of the two subnetworks.

See also
- § Chapter 1.3.1.24.1.5.4.9 “FBD/LD/IL element 'Branch'” on page 472
- § Chapter 1.3.1.25.2.13.33 “Command 'Insert Branch'” on page 1001

See also
- § Chapter 1.3.1.24.1.5.1 “FBD/LD/IL editor” on page 461

---

**Programming ladder diagrams (LD)**

**Creating a POU in the ladder diagram (LD) implementation language**

1. Select the application in the Device tree.
2. Select the command “Project ➔ Add Object ➔ POU”.
   ⇒ The dialog box “Add POU” opens.
3. Enter a name and select the implementation language “Ladder Diagram (LD)”.
   Click on “Add”.
   ⇒ CODESYS adds the POU to the Device tree and opens it in the editor. An empty network is inserted in the implementation part. The empty network is bounded on the left by a vertical line, which represents a bus bar. The view “ToolBox” is also automatically opened, in which the suitable elements, operators and function blocks for LD programming are available.

**Adding a contact and a function block (TON)**

☑ Requirement: a POU with the implementation language LD is opened in the editor and an empty network is inserted.

1. Click on the category “Ladder Elements” in the view “ToolBox”
2. Click on the “Contact” element, drag it into your network and release the mouse button at the insertion position “Start here”.
   ⇒ The contact is added on the left in the network directly against the vertical line.
3. Click on ??? and enter the identifier of a boolean variable. The input assistant is also available to you for this.
4. Click on “Function Blocks” in the view “ToolBox” and drag the function block “TON” onto an insertion position on the connecting line to the right of the inserted contact.
   ⇒ CODESYS inserts the box “TON” to the right of the contact. The contact is connected with the input IN of the TON box.
5. Enter a time constant at the input PT, for example T#3s.
   ⇔ If the variable of your contact goes TRUE, then the input IN of the TON box also goes TRUE. The TON box forwards the value TRUE to the output Q with a switch-on delay of T#3s, for example.

Inserting a closed line branch

Requirement: a POU with the implementation language LD is opened in the editor and an empty network is inserted.

1. Click inside the empty network and select the command “FBD/LD/IL ➔ Insert Contact”.
2. Select the connecting line to the left of the contact and select the command “FBD/LD/IL ➔ Set Branch Start Point”.
   ⇔ The starting point on the connecting line is marked by a red rectangle. CODESYS marks all possible end points of the branch with a blue rectangle.
3. Click on a blue rectangle in order to set the end point of your closed line branch.
   ⇔ CODESYS inserts the line branch between the starting and end points. The program flow will go through both branches up to the end point.

If you insert the line branch at a box instead of at a contact, the box will only be called if none of the other branches is TRUE.

See also
- Section 1.3.1.24.1.5.1 “FBD/LD/IL editor” on page 461
- Section 1.3.1.24.1.5.4.14 “Closed branch” on page 475
- Section 1.3.1.24.1.5.4.11 “LD element ‘Contact’” on page 473

Programming in instruction list (IL)

If necessary, IL can be activated in the CODESYS options.

Creating POUs in the instruction list (IL) implementation language

1. Select the application in the device tree.
2. Click “Project ➔ Add Object ➔ POU”.
   ⇔ The “Add POU” dialog opens.
3. Enter a name and select the implementation language “Instruction List (IL)”.
   Click “Add”.
   ⇔ CODESYS adds the POU to the device tree and opens it in the editor. A network is already inserted in the implementation part.

Programming networks (example: ADD operation)

Requirement: A POU (IL) is opened in the editor and possesses an empty network.
1. Click the line marked in color in the 1st column and enter the operator LD.
2. Press the [Tab] key.
   ⇒ The cursor jumps to the 2nd column
3. Enter the first summand of your ADD operation, for example 6.
4. Press [Ctrl]+[Enter] or select the command “FBD/LD/IL Insert IL Line After”.
   ⇒ CODESYS inserts a new instruction line. The first column of this line has the focus.
5. Enter ADD and press [Tab].
6. Enter the second summand of your ADD operation, for example 12.
7. Press [Ctrl]+[Enter]
8. Enter the operator ST and press [Tab].
9. Specify a variable of the data type INT, for example iVar.
   ⇒ The result – 16 in the example – is stored in the iVar.

**Calling function blocks**

Requirement: A POU (IL) is opened in the editor and possesses an empty network. A variable of the data type <function block> is declared in the declaration part (example: Cl:CTU;).

1. Click the line marked in color in the 1st column and select the command “FBD/LD/IL Insert Box”.
   ⇒ The input assistant opens.
2. Select the desired function block in the category “Function Blocks” or “Boxes”, for example the “CTU” counter from the “Standard” library, and click “OK”.
   ⇒ CODESYS inserts the selected function block “CTU” as follows:

   ![Function Block Example](image)

   3. Replace the strings ??? with the variable name and the values or variables for the inputs/ outputs of the function block.
4. As an alternative to inserting the function block via the input assistant, you can directly enter the call in the editor as shown in the picture at step 4.

See also

- © Chapter 1.3.1.24.1.5.1 “FBD/LD/IL editor” on page 461
- © Chapter 1.3.1.24.1.5.3 “Modifiers and operators in IL” on page 466

**Continuous function chart (CFC)**

The “Continuous Function Chart (CFC)” implementation language is a graphical programming language which extends the standard languages of IEC 61131-3.
You can graphically program a system by means of a POU in CFC. You insert elements and position them freely. You insert connections and wire the elements to a network so that a well-structured function block diagram is created. You can also insert feedback. You can read function block diagrams like an circuit diagram or a block diagram.

The execution order of a function block diagram is based on data flow. Moreover, a POU can process multiple data flows. Then the data flows do not have any common data. In the editor, multiple networks do not have any connections to each other.

On the other hand, POUs in FBD, LD, or IL have a network-based execution order.

The “Continuous Function Chart (CFC) - page-oriented” implementation language is also a graphical programming language which extends the standard languages of IEC 61131-3.

In this language, you can graphically program large, complex function block diagrams. The same elements and commands are available as for “Continuous Function Chart (CFC)”. In addition, you can arrange the code on as many pages as you like. This allows you to create extensive function block diagrams that are still easy to print. Furthermore, each page has border areas. You can arrange inputs and sink connection marks on the left, and outputs and source connection marks on the right. This helps you to insert connecting lines and provides a better overview.

Unfortunately, it is not possible to switch a POU between the “Continuous Function Chart (CFC) - page-oriented” and “Continuous Function Chart (CFC)” implementation languages.

Siehe auch
● ☞ Chapter 1.3.1.24.1.6.1 “CFC editor” on page 477
● ☞ Chapter 1.3.1.24.1.6.2 “CFC editor, page-oriented” on page 479
● ☞ Chapter 1.3.1.25.2.12 “Menu ‘CFC’” on page 977
● ☞ Chapter 1.3.1.25.3.10.12 “Dialog ‘Properties’ - ‘CFC Execution Order’” on page 1054

Automatic execution order by data flow

The execution order in POUs is uniquely determined in text-based and network-based editors. In the CFC editor, however, you can position the elements freely, so the execution order is initially not unique. For this reason, CODESYS determines the execution order by data flow and, in the case of multiple networks, by the topological position of the elements. The elements are sorted from top to bottom and left to right. Now the execution order is unique and makes sure that the POU is processed while optimized by time and by cycle.

You can get information about the chronological order of the elements in the chart and temporarily display the execution order. When you program networks with feedback you can define an element as the starting point in the feedback loop.

You can also explicitly edit the processing order in a CFC object if necessary. To do this, switch the “Auto Data Flow Mode” property of the CFC object to “Explicit Execution Order Mode”. In this mode, you have the option of editing the execution order by means of menu commands.

Before CODESYS Development System V3.5 SP15, you had to define the execution order explicitly for each POU. The was no mode switching.

Data flow

In general, data flow is described as the chronological order in which data is read or written when and how in which programming objects. A POU can process any number of data flows, which can also be executed independently of each other.
**Displaying the execution order**

By default, the execution order of a CFC object is determined automatically. The “Auto Data Flow Mode” property is selected for this. You can temporarily display the automatically determined execution order in the CFC editor.

1. Create a new project using the “Standard project” template and specify the name *Minimal* for example.

2. Extend the application with the function block `FB_DoIt` in the “ST” implementation language with inputs and outputs.

```plaintext
FUNCTION_BLOCK FB_DoIt
  VAR_INPUT
    iAlfa : INT;
    iBravo: INT;
    sCharlie : STRING := 'Charlie';
    xItem : BOOL;
  END_VAR
  VAR_OUTPUT
    iResult : INT;
    sResult : STRING;
    xResult : BOOL;
  END_VAR
  VAR
  END_VAR
  iResult := iAlfa + iBravo;
  IF xItem = TRUE THEN
    xResult := TRUE;
  END_IF
END_FUNCTION_BLOCK
```

3. Create the function block `ExecuteCFC` in the “CFC” implementation language.

```plaintext
PROGRAM ExecuteCFC
  VAR
    fb_DoIt_0: FB_DoIt;
    fb_DoIt_1: FB_DoIt;
    iFinal_1: INT;
    iFinal_0: INT;
    xFinal: BOOL;
  END_VAR
```

Recently created programming objects in CFC have the Auto Data Flow Mode selected. The execution order of the programming object is optimally defined internally.
4. Click “CFC ➔ Execution Order ➔ Display Execution Order”. 
   ⇒ The execution order of the object is shown. The boxes and inputs are numbered accordingly and reflect the chronological processing order. The numbering is hidden as soon as you click again in the CFC editor.

![Diagram showing execution order with boxes and numbers]

Determining the execution order in feedback networks

1. Create a CFC program with feedback.
   ⇒ The POU PrgPositiveFeedback counts.

   ```plaintext
   PROGRAM PrgPositiveFeedback
   VAR
   iResult: INT;
   END_VAR
   ```

2. Select an element within the feedback.
   ⇒ The selected element is highlighted in red.

3. Click “CFC ➔ Execution Order ➔ Set Start of Feedback”.
   ⇒ At runtime, this POU is processed first. The start POU of the feedback is defined and decorated with the ▶ symbol. The execution order is resorted and the selected element gets the number 0. (This is the lowest number of the feedback.)

4. Select the start POU again.
5. Click “CFC ➔ Execution Order ➔ Set Start of Feedback”.  
   ⇒ The POU is not selected as the start POU.

The execution order is defined internally.

6. Click “CFC ➔ Execution Order ➔ Display Execution Order”.  
   ⇒ The execution order by data flow is displayed.

The automatically defined execution order by data flow results in time- and cycle-optimized execution of the POU. You do not need any information about the internally managed execution order during the development process.  

In “Explicit Execution Order Mode”, it is your responsibility to adapt the execution order and to assess the consequences and impacts. This is another reason why the execution order is always displayed.

You can change the automatically defined execution order of a CFC object explicitly when you select the “Explicit Execution Order Mode” option for the object.

1. In the “Devices” or “POUs” view, select a CFC object.
2. In the context menu, click “Properties”.
3. Click the “CFC Execution Order” tab.  
   ⇒ The “Execution order” list box displays the currently selected mode.
4. In the “Execution order” list box, select “Explicit Execution Order Mode”.
5. Click “OK” to confirm the dialog.  
   ⇒ The Explicit Execution Order Mode property is selected. The networks are numbered in the CFC editor, and the following commands are provided in the “CFC ➔ Execution Order” menu for editing the execution order.
6. Open a CFC object.
7. Select a numbered element and click “CFC ➔ Execution Order ➔ Send to Front”.  
   ⇒ The execution order is resorted and the selected element has the number 0.

See also

- Chapter 1.3.1.24.1.6.1 “CFC editor” on page 477
- Chapter 1.3.1.24.1.6.2 “CFC editor, page-oriented” on page 479
- Chapter 1.3.1.25.2.12 “Menu ‘CFC’” on page 977
- Chapter 1.3.1.25.3.10.12 “Dialog ‘Properties’ - ‘CFC Execution Order’” on page 1054
Programming in the CFC editor

In the CFC editor, you can wire POUs to each other and create well-structured block diagrams.

The editor supports you in the following ways:

- Programming with elements and connecting lines
- Dragging instances and variables to the editing area
- Auto-routing the connecting lines
- Automatic linking
- Fixing of connecting lines by control points
- Collision detection
- Input assistance for connection marks
- Forcing and writing of values in online mode
- Movement of selection using arrow keys
- Reduced display of a POU without disconnected pins

Inserting elements and wiring with connecting lines

1. Drag a “Box” element and an “Output” element into the editor.
2. Click the output of the “Box” element.
   ⇨ The output is marked with a red box.
3. Drag a connecting line from the box output of the “Box” element to the box input of the “Output” element.
   ⇨ The cursor symbol changes when it reaches the box input.
4. Release the left mouse button.
   ⇨ The output pin of the box is wired to the input pin of the output.

You can also hold down the [Ctrl] key, select each pin, and then right-click “Connected Selected Pins”.

Calling of instances

1. Create a new project using the standard template and specify the name First for example.
   ⇨ The project First.project is created.
2. Extend the application with the function block `FB_DoIt` in the “ST” implementation language with inputs and outputs.

   ```
   FUNCTION_BLOCK FB_DoIt
   VAR_INPUT
     iAlfa : INT;
     iBravo: INT;
     sCharlie : STRING := 'Charlie';
     xItem : BOOL;
   END_VAR
   VAR_OUTPUT
     iResult : INT;
     sResult : STRING;
     xResult : BOOL;
   END_VAR
   VAR
     END_VAR
   iResult := iAlfa + iBravo;
   IF xItem = TRUE THEN
     xResult := TRUE;
   END_IF
   ```

3. Select the application and click “Add Object ➔ POU” in the context menu. Select the “Continuous Function Chart (CFC)” implementation language and the type Program. Specify the name `PrgFirst` for example.

   Click “OK” to confirm the dialog.

   The program `PrgFirst` is created and it opens in the editor. It is still empty.

4. Instantiate function blocks and declare variables.

   ```
   PROGRAM PrgFirst
   VAR
     iCounter: INT;
     fbDoIt_1 : FB_DoIt;
     fbDoIt_2 : FB_DoIt;
     iOut : INT;
     sOut: STRING;
     xOut: BOOL;
   END_VAR
   ```

5. Drag a “Box” element from the “ToolBox” view into the editor.

6. Click the ??? field and type in ADD.

   The box type is ADD. The box acts as an adder.

7. Click line 3 in the declaration editor.

   The declaration line of `iCounter` is selected.

8. Click in the selection and drag the selected variable into the implementation. Focus there on an input of the ADD box.

   An input has been created, declared, and connected to the box.

9. Click again in the selection and drag the variable to the output of the ADD box.

   An output has been created, declared, and connected to the box.

10. Drag an “Input” element from the “ToolBox” view to the implementation. Click its ??? field and type in 1.
11. Connect the 1 input to an input of the ADD box.
   ⇒ A network is programmed. At runtime, the network counts the bus cycles and stores the result in iCounter.

12. Click line 5 in the declaration editor.
   ⇒ The line is selected.
13. Click in the selection and drag the selected instance into the implementation.

⇒ The instance appears as a POU in the editor. The type, name, and POU pins are displayed accordingly.
14. Drag the `fbDoIt_2` instance to the editor. Interconnect the instances to each other and to inputs and outputs.

Example:

A program in ST with the same functionality might look like this:

```plaintext
PROGRAM PrgFirstInSt
VAR
  iCounter: INT;
  fbDoIt_1 : FB_DoIt;
  fbDoIt_2 : FB_DoIt;
  iOut : INT;
  sOut: STRING;
  xOut: BOOL;
END_VAR

iCounter := iCounter + 1;
fbDoIt_1(iAlfa := 16, iBravo := 32, sCharlie := 'First',
          xItem := TRUE, iDelta := 2, iResult => fbDoIt_2.iAlfa, xResult
          => fbDoIt_2.xItem);
fbDoIt_2(iBravo := fbDoIt_1.iResult, sCharlie := 'Second',
          iDelta := 2, iResult => iOut , sResult=> sOut, xResult =>
          xOut);
```

Creating connection marks

- Requirement: A CFC POU has connected elements.
- 1. Select a connecting line between two elements.
  - The connecting line is displayed as selected. The ends of the connecting line are marked with red boxes ( ).
- 2. Click “CFC ➜ Connection Mark”.
  - The connection is separated into a “Connection Mark - Source” and a “Connection Mark - Sink”. The name of the mark is generated automatically.
3. Click in the source connection marks.
   ➔ You can edit the name.

4. Specify a name SimpleMark for the source connection mark.
   ➔ The source connection mark and sink connection mark have the same name.

The following example shows how to use the “Route All Connections” command with control points.

1. Position the “Input” and “Output” elements. Connect the elements.

2. Position two “Box” elements on the line.
   ➔ The connecting line and the boxes are marked red because of the collision.

3. Click “CFC ➔ Routing ➔ Route All Connections”.
   ➔ The collision is resolved.
4. Change the connecting lines gradually.

5. Select the connecting line and click “CFC ➔ Routing ➔ Create Control Point”.

6. Change the connecting line as seen in the following example.

7. In the context menu, click “CFC ➔ Routing ➔ Remove Control Point” to remove the control point.

8. Unlock the connection by clicking “Unlock Connection” or by clicking the lock symbol.

9. Select the connecting line and click “Route All Connections”.

⇒ The connecting line has been changed manually and is now blocked for auto-routing. This is shown by a lock symbol at the end of the connection.

⇒ A control point is created on the connecting line. The connecting line is fixed to the control point.

⇒ You can also drag a control point from the “ToolBox” view to a line.

⇒ Use the control point for changing the connecting line according to your needs. You can set any number of control points.

⇒ The connecting line is routed automatically as seen in Step 3.
NOTICE!
Connections in a group are not auto-routed.

Reducing the display of a POU

- Requirement: A CFC POU is open. In the editor, its POUs with all declared pins are displayed.

1. Select a POU whose pins are partially disconnected.
   - Example: fb_DoIt_1

The POU needs space for all of the pins.

2. Click “CFC ➔ Pins ➔ Remove Unused Pins”.
   - Now the POU needs less space and is displayed only with the functionally relevant pins.

See also
- Chapter 1.3.1.24.1.2 “Common functions in graphical editors” on page 429
- Chapter 1.3.1.24.1.6.1 “CFC editor” on page 477
- Chapter 1.3.1.24.1.6.2 “CFC editor, page-oriented” on page 479
- Chapter 1.3.1.24.1.6.4 “Elements” on page 486

Structured text (ST), extended structured ext (ExST)

The ST editor is used for the programming of POUs in the IEC-61131-3 programming language Structured Text (ST) and Extended Structured Text. The Extended Structured Text offers some additional functions with regard to the IEC 61131-3 standard.
Structured Text is a programming language, comparable with other high-level languages such as C or PASCAL, which permits the development of complex algorithms. The program code consists of a combination of expressions and instructions, which can also be executed conditionally (IF... THEN... ELSE) or in loops (WHILE... DO).

An expression is a construct that returns a value following its evaluation. Expressions are also operators and operands together. You can also use assignments as expressions. An operand can be a constant, a variable, a function call or a further expression.

Instructions control how the expressions are to be processed.

For this text editor you can make various settings with regard to behavior, appearance and menus in the dialog boxes “Options” and “Adapt” in the “Tools” menu. The familiar Windows functions (for example IntelliMouse) are also available for this editor.

See also
- Chapter 1.3.1.9.3.3.1 “Programming structured text (ST)” on page 201
- Chapter 1.3.1.25.3.13.20 “Dialog ‘Options’ - ‘Text Editor’” on page 1086
- Chapter 1.3.1.25.3.14.1 “Dialog ‘Customize’ - ‘Menu’” on page 1089

ExST - Extended structured text

Extended Structured Text (ExST) is a CODESYS-specific extension of the IEC 61131-3 standard for Structured Text (ST).

See also
- Chapter 1.3.1.24.1.3.4.4 “ExST assignment ‘R’=” on page 434
- Chapter 1.3.1.24.1.3.4.3 “ExST assignment ‘S’=” on page 433
- Chapter 1.3.1.24.1.3.4.5 “ExST – Assignment as expression” on page 435

Programming structured text (ST)

Principle

The programming languages 'Structured Text' and 'Extended Structured Text' are programmed in the ST editor. The program code consists of a combination of expressions and instructions, which can also be executed conditionally or in loops. You must conclude each instruction with a semicolon ;.

The variables are declared in the declaration editor.

Creating a POU in the structured text (ST) implementation language

1. Select an application in the device tree.
2. Select the command “Project ➔ Add Object ➔ POU”.
   - The dialog box “Add POU” opens.
3. Enter a name and select the “Implementation language” “Structured Text (ST)”. Click on “Add”.
   - The POU is added to the device tree and opened in the editor.
   - Now insert the variable declarations in the upper part of the POU and enter the ST program code in the lower part of the POU.

See also
- Chapter 1.3.1.24.1.3.3 “ST expressions” on page 431
- Chapter 1.3.1.24.1.3 “Structured text and extended structured text (ExST)” on page 430
- Chapter 1.3.1.24.1.3.5.10 “ST function block call” on page 441
Sequential function chart (SFC)

Use the SFC editor for programming POUs in the IEC 61131-3 compliant SFC implementation language. SFC is a graphical programming language for describing the chronological sequence of individual actions in a program. For this purpose, actions (discrete programming objects) are assigned to step elements. Transition elements control the processing order of steps.

See also
- Chapter 1.3.1.24.1.4.1 “SFC editor” on page 442

Programming the sequential function chart (SFC)

Creating POUs in the SFC implementation language

1. Select an application in the device tree.
2. Click “Project ➔ Add Object ➔ POU”.
   ⇒ The dialog box “Add POU” opens.
3. Enter a name and select the implementation language “Sequential Function Chart (SFC)”. Click “Add”.
   ⇒ CODESYS adds the POU to the device tree and opens it in the editor.

Adding step-transitions

1. Select the transition after the initial step.
   ⇒ The transition is marked red.
2. Click “SFC ➔ Insert step-transition after”.
   ⇒ CODESYS inserts the “Step0” step and the “Trans0” transition.
3. Select the “Trans0” transition and click “SFC ➔ Insert Step-Transition”.
   ⇒ CODESYS inserts the “Trans1” transition and the “Step1” step before the “Trans0” transition.

You can also drag the “Step” and “Transition” elements into the diagram from the “Toolbox” view.

See also
- Chapter 1.3.1.24.1.4.7.1 “SFC elements 'Step' and 'Transition’” on page 452
- Chapter 1.3.1.25.2.11.6 “Command 'Insert Step-Transition’” on page 969
- Chapter 1.3.1.25.2.11.7 “Command 'Insert Step-Transition After’” on page 969
Adding entry actions

1. Select “Step0”.
2. Click “SFC ➯ Add entry action”.
   - By default, you are prompted to select the duplication mode for the step actions. You decide whether the reference to the existing step action objects is copied when the step is copied, or the objects are embedded. Embedding results in new step action objects being created when the step is copied. The duplication mode is defined in the “Duplicate when copying” step property. When it is deactivated, the copied steps call the same actions as the current step.
   - You can deactivate the prompt completely in the SFC properties.
   - The display of embedded objects in the “Devices” and “POUs” views can be deactivated by means of a menu command.
3. For this example, accept the default setting “Copy reference” and confirm by clicking “OK”.
   - The “Add Entry Action” dialog box opens.
4. Type the name "Step0_entry" and select “Structured Text (ST)” as the implementation language. Click “Add”.
   - CODESYS inserts the “Step0_entry” action below the POU in the device tree and opens the action in the editor.
   - In the "Step0_entry" entry action, you program statements to be executed one time when the “Step0” step becomes active.
5. Close the "Step0_entry" editor.
   - The “Step0” step is now marked with an “E” in the lower left corner. Double-clicking this marker opens the editor.
   ![Step0 Editor](image)
   - The entry action Step0_entry is now available in the properties of the step in “Entry action”. Other actions can also be selected there as needed.
6. Select Step0. Press [Ctrl]+[V] to copy the step.
   - The same entry actions inserted above are available in the inserted copy of the step. The new step then calls the same exact action.

See also
- Chapter 1.3.1.25.2.11.8 “Command ‘Add Entry Action’” on page 970
- Chapter 1.3.1.24.1.4.7.2 “SFC element ‘Action’” on page 454
- Chapter 1.3.1.25.3.13.1 “Dialog ‘Options’ - ‘SFC Editor’” on page 1074

Adding exit actions

1. Select “Step0”.
2. Click “SFC ➯ Add Exit Action”.
   - By default, you are prompted to select the duplication mode for the step actions of the step. See above for adding entry actions. Then the “Add Exit Action” dialog box opens.
3. Specify the name "Step0_exit" and select the implementation language "Structured Text (ST)". Click "Add".
   ⇒ CODESYS inserts the "Step0_exit" action below the POU in the device tree and opens the action in the editor.
   In the "Step0_exit" exit action, you program statements to be executed one time before the "Step0" step becomes inactive.
4. Close the "Step0_exit" editor.
   ⇒ The “Step0” step is now marked with an “X” in the lower right corner. Double-clicking this marker opens the editor.

![Step0]

You can define the exit action in the step properties of “Exit action”, as well as select other actions as needed.

See also
- Chapter 1.3.1.25.2.11.9 “Command 'Add Exit Action’” on page 970
- “2. Step actions” on page 455

Adding actions

1. Double-click the “Step0” step.
   ⇒ By default, you are prompted to select the duplication mode for the step actions of the step. See above for adding entry actions. The “Add Action” dialog box opens.
2. Type the name "Step0_active" and select the implementation language "Structured Text (ST)". Click "Add".
   ⇒ CODESYS inserts the “Step0_active” action below the POU in the device tree and opens the action in the editor.
   In the Step0_active step action, you program statements to be executed as long as the step is active.
3. Close the "Step0_active" editor.
   ⇒ The “Step0” step is now marked with a black triangle in the upper right corner.

![Step0]

You can define the action in the step properties of “Step action”, as well as select other actions as needed.

See also
- “2. Step actions” on page 455
Adding alternative branches

1. Select “Step1”.
2. Click “SFC ➔ Insert Branch Right”.
   ⇒ CODESYS inserts the “Step2” step to the right of “Step1”. The steps are connected as a parallel branch signified by two pairs of double lines.
3. Select one of the double lines.
   ⇒ The double line is marked red.
4. Click “SFC ➔ Alternative”
   ⇒ CODESYS converts the branch into an alternative branch. The double lines change into a single line.

You can convert an alternative branch into a parallel branch by clicking “SFC ➔ Parallel”.

See also
- Chapter 1.3.1.24.1.4.7.3 “SFC element 'Branch'” on page 457
- Chapter 1.3.1.25.2.11.10 “Command 'Parallel'” on page 970
- Chapter 1.3.1.25.2.11.12 “Command 'Insert Branch'” on page 971
- Chapter 1.3.1.25.2.11.13 “Command 'Insert Branch Right'” on page 971

Adding jumps

1. Select “Step2”.
2. Click “SFC ➔ Insert Jump After”.
   ⇒ CODESYS inserts the “Step” jump after the “Step2” step.
3. Select the “Step” jump destination.
   ⇒ You can type the jump destination manually or select it by using the input assistant. Select Step0.

See also
- Chapter 1.3.1.24.1.4.7.4 “SFC element 'Jump'” on page 458
- Chapter 1.3.1.25.2.11.16 “Command 'Insert Jump'” on page 973
- Chapter 1.3.1.25.2.11.17 “Command 'Insert Jump After'” on page 973

Adding macros

1. Select “Step1”.
2. Click “SFC ➔ Insert macro after”.
   ⇒ CODESYS inserts the “Macro0” macro after the “Step1” step.
3. Double-click the “Macro0” element.
   ⇒ The macro opens in the implementation section of the editor. The name "Macro0" is displayed in the caption.

4. Click “SFC ➔ Insert step-transition”.
   ⇒ CODESYS inserts a step-transition combination.

5. Click “SFC ➔ Zoom out of macro”.
   ⇒ The implementation section returns to the main diagram.

See also
-  “Chapter 1.3.1.24.1.4.7.5 “SFC element 'Macro’’” on page 458
-  “Chapter 1.3.1.25.2.11.18 “Command 'Insert Macro’’” on page 974
-  “Chapter 1.3.1.25.2.11.19 “Command 'Insert Macro After’’” on page 974

Adding associations

1. Select “Step2”.

2. Click “SFC ➔ Insert Action Association”.
   ⇒ CODESYS inserts an association to the right of the “Step2” step.

3. Click in the left field of the association to select the qualifier.
   ⇒ You can enter the qualifier manually or use the input assistant. Select "P".

4. Click in the right field of the association to select the action.
   ⇒ You can type the action or select it by using the input assistant.

See also
-  “1. IEC actions” on page 454
-  “Chapter 1.3.1.24.1.4.4 “Qualifiers for actions in SFC” on page 445
-  “Chapter 1.3.1.25.2.11.14 “Command 'Insert Action Association’’” on page 972
-  “Chapter 1.3.1.25.2.11.15 “Command 'Insert Action Association After’’” on page 973

See also
-  “Chapter 1.3.1.24.1.4.1 “SFC editor” on page 442

1.3.1.9.4 Configuring the memory reserve for an online change

You can configure a function block with a memory reserve to prevent its function block instances from having to be copied to new memory locations during an online change. This can happen when you make declaration changes for example adding variables to the function block. With memory reserves, this kind of online change runs faster and fewer problems occur. When the memory reserve is depleted, a message is displayed before the online change is performed.
NOTICE!
The memory reserve for a function block is best configured before downloading the application to the controller for the first time. If the memory reserve is configured when the application exists on the controller, then a more complicated online change is necessary.

In the future, bigger changes are made to a function block of a project, which could result in function block instances being copied to other memory locations in an online change.

- Requirement: The open project is not located on the controller yet.
  1. Click “View ➔ Online Change Memory Reserve Settings”.
     ⇒ The “Online Change Memory Reserve” view opens.
  2. Select the application of the project from the list box.
  3. Click “Build ➔ Build”.
  4. Click the “Scan Application” button.
  5. Select the “All” entry in the “Function blocks” area.
     ⇒ All function blocks of the application are displayed in the view.
  6. Select the corresponding function block for which you want to configure a memory reserve.
     ⇒ If the application is not on the controller yet, then the “Memory reserve” input field is editable.
  7. If the application is already on the controller, then click the “Enabled” button in the “Enable Editing” area.
     Note that if you modify the memory reserve of an application that is already on the controller, then the instances of all affected function blocks have to be copied to the memory.
  8. Specify the size of the memory reserve (in bytes) and click “Apply for Selection”.
     ⇒ The given number of bytes is shown in the “Memory reserve” field in the table.
  9. Click “Build ➔ Build” and then click “Scan Application”.
     ⇒ The following information is updated in the function block list for the configured function block: “Size”, “Instance count”, “Additional memory for all instances”, and “Remaining size of the memory reserve”.

Now when you download the application to the controller, the function block instances occupy the required memory and also the memory reserve. In the future, bigger changes of the function block can be downloaded to the controller with the online change without having to copy all function block instances to memory.

See also
- § Chapter 1.3.1.25.2.3.19 “Command ‘Settings of Memory Reserve for Online Change’” on page 894
- § Chapter 1.3.1.25.2.6.6 “Command ‘Online Change’” on page 928
1.3.1.9.5 Function block — Calling functions or methods with external implementation

A runtime system can include the implementation of a function block, function, or method (for example, from a library). If you create a POU in your application with the same name by using the “External implementation” without an implementation, then you can execute the existing implementation. Please make sure that you declare local variables only in an external function block. External functions and methods must not contain local variables.

When the application is downloading, CODESYS searches for and links the associated implementation in the runtime system for each external POU.

Objects with the property “External implementation” are postfixed with (EXT) after the object name in the “Devices” or “POUs” view.

See also

● Chapter 1.3.1.25.3.10.4 “Dialog ’Properties’ - ’Build’” on page 1048

Creating POUs with external implementation

1. Click “Project ➔ Add Object ➔ POU”.
2. Activate “Function block”, Method, or “Function” and specify the name of the associated implementation of the runtime system. Close the dialog box by clicking “Add”.
   ➔ The runtime system POU is created in the “POUs” view. The name is postfixed with (EXT).
3. Right-click the POU and select “Properties”.
   ➔ The dialog box opens.
4. Click the “Build” tab.
5. Select the “External implementation (Late link in the runtime system)” check box.
   ➔ The POU is declared and you can implement a POU call.

Creating methods with external implementation

1. Select a function block in the device tree or in the POUs view.
2. Select “Add Object ➔ Method” and type the name of the associated implementation of the runtime system. Click “Add” to close the dialog box.
   ➔ The method is created.
3. Right-click the method and select “Properties”.
   ➔ The dialog box opens.
4. Click the “Build” tab.
5. Select the “External implementation (Late link in the runtime system)” check box.
   ➔ The method is declared and you can implement a method call. The method name is postfixed with (EXT) in the “Devices” or “POUs” view.

1.3.1.9.6 Using input assistance

CODESYS provides tools and features to help you code when creating programs.
**Input assistant**
The input assistant provides all program elements that you can insert at the current cursor position. Open the “Input Assistant” dialog by clicking “Edit ➔ Input Assistant” or by pressing [F2].

See also
- “Dialog 'Input Assistant' - Tab 'Categories’” on page 874

**Dialog 'Auto Declare’**
This dialog supports the declaration of variables.

See also
- “Chapter 1.3.1.25.2.2.31 “Command 'Auto Declare’” on page 872

**"List components"**
The "List components" function is an input tool in textual editors to help you input valid identifiers. Activate this function by clicking “Tools ➔ Options” and then the “SmartCoding” category. Select the “List components after typing a dot (.)” check box.

- If you type a dot (.) instead of a global variable, then a drop-down list opens with all available global variables. You insert the selected variable after the dot by double-clicking a variable in the drop-down list or by pressing [Enter].
- If you type a dot (.) instead of a global variable after a function block instance variable or a structure variable, then CODESYS opens a drop-down list with all global variables, all input and output variables for the function block, or all structure members. You insert the selected variable after the dot by double-clicking a variable in the drop-down list or by pressing [Enter].
  
  Note: When you also want to choose from the local variables of function block instances, select the “Show all instance variables in input assistant” option in the CODESYS options (SmartCoding category).
- If a component access (with a dot) for a drop-down list has already happened, then the last selected entry is preselected at the next component access.
- When you type any sequence of characters and then press [Ctrl]+[Space], a drop-down list opens with all available POU’s and global variables. The first element in this list that starts with the sequence of characters is selected by default and you can insert it by double-clicking it or by pressing [Enter].
  
  Matches with the entered character string are highlighted in yellow in the drop-down list.
  
  If the entered character string is changed, then the displayed drop-down list is refreshed.
- In the ST editor, you can filter the displayed drop-down list by scopes:
  
  Depending on the displayed drop-down list, you can use the [Arrow right] and [Arrow left] keys to toggle between the following drop-down lists:
  - “All items”
  - “Keywords”
  - “Global declarations”
  - “Local declarations”
- CODESYS displays a tooltip if you type an opening parenthesis for a POU parameter when calling a function block, a method, or a function. This tooltip includes information about the parameters as they are declared in the POU. The tooltip remains open until you click to close it or you change the focus away from the current view. If you accidentally close the tooltip, then you can reopen it by pressing [Ctrl]+[Shift]+[Space].

You can use the pragma attribute ‘hide’ for excluding variables from the "List components" feature.
Examples

Typing structure variables:

Calling a function block:

See also

- Chapter 1.3.1.25.3.13.11 “Dialog ‘Options’ - ‘SmartCoding’” on page 1082
- Chapter 1.3.1.24.6.2.13 “Attribute ‘hide’” on page 613

Short form feature

The short form feature allows you to type abbreviated forms for variable declarations in the declaration editor and in textual editors where variables declarations are possible. Use this feature by pressing [Ctrl]+[Enter] to end a declaration line.

CODESYS supports the following short forms:

- All identifiers become variable identifiers except the last identifier of a line.
- The data type of the declaration is determined by the last identifier of the line. The following applies:
  - B or BOOL yields BOOL
  - I or INT yields INT
  - R or BOOL yields BOOL
  - S or STRING yields STRING
- If a data type is not defined using this rule, then the data type is automatically BOOL, and the last identifier is not used as the data type (see Example 1).
- Depending on the type of declaration, every defined constant becomes an initialization or string length definition (see Example 2 and 3).
- An address, such as %MD12, is automatically extended with the AT attribute (see Example 4).
- Any text after a semicolon (;) is converted into a comment (see Example 3).
- All other characters in the line are ignored (see exclamation mark in Example 5).

Examples

<table>
<thead>
<tr>
<th>Example</th>
<th>Short Form</th>
<th>Resulting declaration</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>A</td>
<td>A: BOOL</td>
</tr>
<tr>
<td>2</td>
<td>A B I 2</td>
<td>A, B: INT := 2;</td>
</tr>
<tr>
<td>3</td>
<td>ST S 2; A string</td>
<td>ST:STRING(2); (* A string *)</td>
</tr>
<tr>
<td>4</td>
<td>X %MD12 R 5 Real Number</td>
<td>X AT %MD12: REAL := 5.0;(* Real Number *)</td>
</tr>
<tr>
<td>5</td>
<td>B !</td>
<td>B: BOOL</td>
</tr>
</tbody>
</table>
See also
- Chapter 1.3.1.9.2 “Declaration of variables ” on page 172

Smart tag functions
Smart tags make it easier to write program code by suggest appropriate commands directly at the programming element. When you place the cursor over a programming element that has a smart tag function, the symbol appears. When you click the symbol, the commands that you can choose from are shown. Available smart tags:
- The smart tag function provides the “Declare Variable” command for undeclared variables in the implementation part of the ST editor.

See also
- Chapter 1.3.1.25.2.2.31 “Command ‘Auto Declare’” on page 872

1.3.1.9.7 Using pragmas

What is a pragma in CODESYS?
A pragma is a special statement in the source code of an application that influences the properties of variables when precompiling or compiling (code generation).

There are different pragmas for different purposes (example: initialization of a variable, monitoring of a variable, adding a variable to the symbol configuration, forcing the display of messages during the compilation process, and behavior of a variable under certain conditions).

NOTICE!
Uppercase and lowercase characters must be maintained

How and where to insert a pragma?
A pragma statement is specified in braces. Attributes and texts used within the statement are enclosed in single straight quotation marks. The opening brace may follow immediately after a variable name. Opening and closing braces must be located on the same line.

Examples

{warning 'This is not allowed'}
{attribute 'obsolete' := 'datatype fb1 not valid!'}

Possible insertion points:
- Declaration part of a POU
  In the textual declaration editor, you specify pragmas directly as a line. In the tabular editor, you specify pragmas that must be located before the first declaration line in the dialog box “Edit Declaration Part” / “Attributes”.
- Global variable list
- Implementation part of a POU (on a separate line or on a line with code text)
  FBD/LD/IL: Pragmas in networks of the FBD/LD/IL editor are entered as a label. Click the command “FBD/LD/IL ➔ Add Label” and replace the default text “Label:” in the text field with the appropriate pragma statement. To use a pragma in addition to a label, you specify the pragma first and then the label.

NOTICE!
Pragmas in CODESYS are not one-to-one implementations of C preprocessor directives. You must position a pragma like an ordinary statement. It must not be used within an expression.
Incorrect and correct positions for a conditional pragma

**INCORRECT:**

{IF defined(abc)}
IF x = abc THEN
(ELSE)
IF x = 12 THEN
(END_IF)
y := {IF defined(cde)} 12; {ELSE} 13; {END_IF}
END_IF

**CORRECT:**

{IF defined(abc)}
IF x = abc THEN
(IF defined(cde))
y := 12;
(ELSE)
y := 13;
(END_IF)
END_IF
ELSE
IF x = 12 THEN
(IF defined(cde))
y := 12;
(ELSE)
y := 13;
(END_IF)
END_IF

In the “Properties” dialog box (“Compile” category), you can specify "defines" that can be queried in pragmas.

Scope:
Depending on the type and contents of a pragma, it may influence the following:

- subsequent declarations
- the next statement only
- all subsequent statements until it is canceled by a corresponding pragma
- all subsequent statements until the same pragma is executed with other parameters or the end of the code is reached. In this context, "code" means the declaration part, implementation part, global variable list, and type declaration. Therefore, a pragma influences the entire object when the pragma is alone on the first line of the declaration part and is not superseded or canceled by another pragma.

What kinds of pragmas are there?
The CODESYS pragmas are divided into the following categories:

- Attribute pragmas (influence compiling and precompiling)
- Message pragmas (print user-defined messages when compiling)
- Conditional pragmas (influence code generation)
- User-defined pragmas

See also

- % Chapter 1.3.1.9.2.2 “Using the 'Declare variable' dialog box” on page 174
- % Chapter 1.3.1.24.6.2 “Attribute pragmas” on page 601
1.3.1.8 Using library POUs

Libraries are collections of objects that you can link to your application. You can use the objects contained in libraries in exactly the same way as objects that you have defined in the project.

Libraries can contain:

- POUs, such as function blocks or functions
- Interfaces and their methods and attributes
- Data types, such as enumerations, structures, aliases, and unions
- Global variables, constants, parameter lists
- Text lists, image collections, visualizations, and visual elements
- External files, such as documentation
- Cam plate tables

The library manager is used for managing libraries in a project. You use the library repository dialog box for installing the library in advance onto the system.

See also

- Chapter 1.3.1.19 “Using libraries” on page 371

Using library POUs

The following instructions describe how to insert the counter module CTUD from the Standard library into your program.

1. Open a POU in the editor and place the cursor in the declaration part.
2. Specify the name for the function block instance, followed by a colon (example: iCounter1:).
4. In the category “Structured Types”, select the CTUD function block from the Standard library (subfolder “Counter”).
   
   Activate the option “Insert with namespace prefix”.
5. Click “OK” to close the dialog box.

   The function block is inserted with a namespace prefix into the declaration part: iCounter1:Standard.CTUD.

See also

- Chapter 1.3.1.19 “Using libraries” on page 371

1.3.1.9 Managing text in text lists

Text lists are used for preparing visualization texts in multiple languages. You can specify the texts in Unicode format so that all languages and characters are possible. You can export text lists and then translate the texts outside of the current project.

CODESYS differentiates between static text (managed in the “GlobalTextList” object) and dynamic text (managed in objects of type “TextList”). Static texts exist in the visualization and can change only the displayed language while in runtime mode. The text ID stays the same. Dynamic texts can be controlled by means of an IEC variable that contains the text ID. In this way, you can display varying text in a visualization element in runtime mode. For example, you can configure a text field so that is shows an error text for an error number.
Both text list types include a table with text entries. An entry consists of an ID for identification, the output text, and its translation. In a text list or global text list, you can translate an output text in any number of languages. The translations are the basis for the language selection and the language switch in visualizations.

**Adding a language and translating text**

Requirement: A project is open with a text list or global text.

1. Double-click an object of type “TextList” or “GlobalTextList” in the device tree or POUs view.  
   ⇒ The “Textlist” menu is shown in the menu bar and the text list opens in the editor.
2. Click “Textlist ➔ Add Language”.
3. Specify a name for the language (example: en-US). Click “OK” to close the dialog.
   ⇒ A column is displayed with the heading en-US.
4. Type in the translation of the source text into the column.

You can correct the name of a language in the table by means of the command “Rename Language” in the context menu of the text list.

**Exporting a text list**

.Requirement: A project is open with a text list or global text.

1. Double-click the object “GlobalTextList” or an object of type “TextList”.  
   ⇒ The object opens.
2. Click “Textlist ➔ Import/Export Text Lists”.
   ⇒ The “Import/Export” dialog opens.
3. At “Choose export file”, click for more (….) and select the directory and file name (example: Text_lists_exported).
4. Select the “Export” option.
5. Click “OK” to close the “Import/Export” dialog.
   ⇒ CODESYS exports to a file the text list entries of all text lists of the project. The table contains a column with the text list names.
Example

Contents of the file

<table>
<thead>
<tr>
<th>TextList</th>
<th>Id</th>
<th>Default</th>
<th>en_US</th>
</tr>
</thead>
<tbody>
<tr>
<td>Text_list_A</td>
<td>A</td>
<td>Information A</td>
<td>Information A_en</td>
</tr>
<tr>
<td>Text_list_A</td>
<td>B</td>
<td>Information B: OK</td>
<td>Information B_en: OK</td>
</tr>
<tr>
<td>Text_list_A</td>
<td>C</td>
<td>Information C</td>
<td>Information C_en</td>
</tr>
<tr>
<td>Text_list_A</td>
<td>D</td>
<td>Information D</td>
<td>Information D_en</td>
</tr>
<tr>
<td>Text_list_A</td>
<td>E</td>
<td>Information E</td>
<td>Information E_en</td>
</tr>
<tr>
<td>Text_list_A</td>
<td>F</td>
<td>Information F</td>
<td>Information F_en</td>
</tr>
<tr>
<td>AlarmGroup</td>
<td>2</td>
<td>Warning 2</td>
<td></td>
</tr>
<tr>
<td>AlarmGroup</td>
<td>1</td>
<td>Warning 1</td>
<td></td>
</tr>
<tr>
<td>GlobalTextList</td>
<td></td>
<td>Information B</td>
<td>Information B_en</td>
</tr>
<tr>
<td>GlobalTextList</td>
<td></td>
<td>Information A</td>
<td>Information A_en</td>
</tr>
<tr>
<td>GlobalTextList</td>
<td></td>
<td>Switch</td>
<td>Switch</td>
</tr>
<tr>
<td>GlobalTextList</td>
<td></td>
<td>Counter: %i</td>
<td>Counter: %i</td>
</tr>
</tbody>
</table>

Preparing the exported file for input assistance

- Requirement: A file is created (example: Text_lists_exported) by means of the command “Import/Export Text Lists”. It contains the texts of the text lists of the project.
  2. Click in “Text file for textual "List components" ” and select a file (example: Text_lists_exported). Click “OK” to close the dialog.

(1): “Texts”, “Text”

See also
- “List components” on page 209

Importing files with text list entries

A file to be imported has the .csv format. The first line is a header (example: TextList Id Default en_US). The other lines contain text list entries. You get this kind of file by exporting the text lists of the project to a file. There you can edit the text list entries and then import the file outside of CODESYS. When importing, CODESYS handles the text list entries differently for the GlobalTextList and for dynamic text lists.

- CODESYS does not create new text list entries for an unknown ID.
- CODESYS ignores changes that affect the ID or the source text.
- CODESYS accepts changes in the translations.
TextList

- For a new ID, CODESYS supplements the corresponding text list with a text list entry.
- For an existing ID that does not agree in the source text, the source text of the text list is overwritten with the source text of the file.
- CODESYS accepts changes in the translations.

Importing a file

Requirement: A project is open with a text list or global text.
1. Double-click the object “GlobalTextList” or an object of type “TextList”.
   ⇒ The object opens.
2. Click “Textlist ➔ Import/Export Text Lists”.
   ⇒ The “Import/Export” dialog opens.
3. In the “Choose file to compare or to import” input field, click for more (…) and select the directory and file (example: Text_lists_corrected.csv).
4. Select the “Import” option.
5. Click “OK” to close the dialog.
   ⇒ CODESYS imports the text list entries of the file into the respective text lists.

Example Contents of the file

<table>
<thead>
<tr>
<th>TextList</th>
<th>Id</th>
<th>Default</th>
<th>en_US</th>
</tr>
</thead>
<tbody>
<tr>
<td>Text_list_A</td>
<td>A</td>
<td>Information A</td>
<td>Information A_en</td>
</tr>
<tr>
<td>Text_list_A</td>
<td>B</td>
<td>Information B: OK</td>
<td>Information B_en: OK</td>
</tr>
<tr>
<td>Text_list_A</td>
<td>C</td>
<td>Information C</td>
<td>Information C_en</td>
</tr>
<tr>
<td>Text_list_A</td>
<td>D</td>
<td>Information D</td>
<td>Information D_en</td>
</tr>
<tr>
<td>Text_list_A</td>
<td>E</td>
<td>Information E</td>
<td>Information E_en</td>
</tr>
<tr>
<td>Text_list_A</td>
<td>F</td>
<td>Information F</td>
<td>Information F_en</td>
</tr>
<tr>
<td>Text_list_A</td>
<td>G</td>
<td>Information G</td>
<td>Information G_en</td>
</tr>
<tr>
<td>AlarmGroup</td>
<td>2</td>
<td>Warning 2</td>
<td></td>
</tr>
<tr>
<td>AlarmGroup</td>
<td>1</td>
<td>Warning 1</td>
<td></td>
</tr>
<tr>
<td>GlobalTextList</td>
<td></td>
<td>Information B</td>
<td>Information B_en</td>
</tr>
<tr>
<td>GlobalTextList</td>
<td></td>
<td>Information A</td>
<td>Information A_en</td>
</tr>
<tr>
<td>GlobalTextList</td>
<td></td>
<td>Switch</td>
<td>Switch</td>
</tr>
<tr>
<td>GlobalTextList</td>
<td></td>
<td>Counter: %i</td>
<td>Counter: %i</td>
</tr>
</tbody>
</table>

These contents are applied to the text lists with the same name in the project.

See also

- ¶ Chapter 1.3.1.25.2.20.6 “Command ‘Import/Export Text Lists’” on page 1021

Comparing text lists with a file and exporting differences

Requirement: A project is open with a text list or global text.
1. Double-click the object “GlobalTextList” or an object of type “TextList”.
   ⇒ The object opens.
2. Click “Textlist ➔ Import/Export Text Lists” in the context menu.
   ⇒ The “Import/Export” dialog opens.
3. In the “Choose file to compare or to import” input field, click for more (…) and select the directory and file name of the comparison file (example: Text_lists_corrected.csv).
4. For “Choose export file”, click […] and select the directory and file that contains the comparison result.
5. Select the “Export only text differences” option.
6. Click “OK” to close the dialog.

   CODESYS reads the import file and compares the text list entries that have the same ID. If they do not agree, then CODESYS writes the text list entries of the text list to the export file. 

   For the global text list, CODESYS compares the translations of the same source texts. If they do not agree, then CODESYS writes the text list entries to the export file.

See also
- Chapter 1.3.1.25.2.20.1 “Command ‘Add Language’” on page 1020
- Chapter 1.3.1.25.2.20.6 “Command ‘Import/Export Text Lists’” on page 1021
- Chapter 1.3.1.25.2.20.7 “Command ‘Remove Language’” on page 1022
- Chapter 1.3.1.25.1.35 “Object ‘TextList’” on page 823

Managing static text in global text lists

The global text list is the central location for texts that are displayed in the visualization.

When you configure a text for the first time in visualization element, CODESYS creates the global text list. CODESYS fills in the table as you create more texts. Therefore, the table includes all texts automatically that you create in the project visualizations. CODESYS assigns incremental IDs as integers, beginning at 0.

You can check, update, and compare the global text list with the static texts of the visualization. You cannot edit the source text or the ID directly in the table. However, you can replace a source text with another source text by creating and importing a replacement file. Menu commands are provided for this purpose.

Configuring visualization elements with static text

A text in a “GlobalTextList” can contain a format definition.

☐ Requirement: A project is open with a visualization. The “GlobalTextList” object contains the texts that are defined in the project visualizations.

1. Double-click the visualization.

   ⇒ The editor opens.

2. Select an element with the “Text” property (example: “Text field”).

3. Type in some text in the “Text” property (example: Static Information A).

   ⇒ CODESYS adds the text to the global text list in the POU view.

Checking the global text list

☐ Requirement: A project is open with a visualization. The “GlobalTextList” object contains the texts that are defined in the project visualizations.

1. Double-click the “GlobalTextList” object in the POUs tree.

   ⇒ The table opens with the static texts.

2. Click “Text List ➤ Check Visualization Text IDs”.

   ⇒ CODESYS reports when a source text of the text list does not match the static text that is identified by the ID. The source text in the global text list and the text in the visualization with the same ID do not match.
Updating IDs of the global text list

☐ Requirement: A project is open with a visualization. The “GlobalTextList” object contains the texts that are defined in the project visualizations.
1. Double-click the “GlobalTextList” object in the POUs tree.
   ⇒ The list opens with the text list entries.
2. Click “Text List ➤ Update Visualization Text IDs”.
   ⇒ CODESYS adds text to the global text list when a text in the “Static Text” property does not match the source text in the project visualizations.

Removing the global text list and creating current IDs again

☐ Requirement: A project is open with a visualization. The “GlobalTextList” object contains the texts that are defined in the project visualizations.
1. Right-click the “GlobalTextList” object in the POUs tree and select the “Delete” command.
   ⇒ The object is removed.
2. Open a visualization.
3. Click “Visualization ➤ Create Global Text List”.
   ⇒ In the POU view, a new “GlobalTextList” object is created. The global text list contains the static text from the existing project visualizations.

Removing IDs from the global text list

☐ Requirement: A project is open with a visualization. The “GlobalTextList” object contains the texts that were defined in the project visualizations.
1. Double-click the “GlobalTextList” object in the POUs tree.
   ⇒ The table opens with the texts.
2. Click “Text List ➤ Remove Unused Text List Entries”.
   ⇒ CODESYS removes the text list entries with IDs not referenced in the project visualizations.

Updating the global text list with a replacement file

A replacement file has the CSV format. The first row is a header: defaultold defaultnew REPLACE. The following rows contain the old source texts, the new source texts, and then the REPLACE command. Tabs, commas, and semicolons are permitted separators. A combination of separator characters in a file is not permitted.

defaultold  defaultnew  REPLACE
Information A  Information A1  REPLACE

When you import a replacement file, CODESYS processes the replacement file row by row and performs the specified replacements in the “GlobalTextList”. In addition, CODESYS replaces the previous text with the replacement text in the visualizations. If the replacement text already exists as static text, then CODESYS recognizes this and harmonizes the static text and leaves only one text list entry.
Requirement: A project is open with a text list or global text.

1. Double-click the “GlobalTextList” object.
   ⇒ The object opens.

2. Click “Text List ➔ Import/Export Text Lists”.
   ⇒ The “Import/Export” dialog opens.

3. At the “Choose file to compare or to import” input field, click for more (…) and select the directory and file (example: ReplaceGlobalTextList.csv).

4. Select the “Import replacement file” check box.

5. Click “OK” to close the dialog.
   ⇒ The texts in the text lists and the visualizations are replaced.

**Example**

The global text list contains the following source texts:

- GlobalTextList Counter: %i
- GlobalTextList Counter: %i
- GlobalTextList Information A
- GlobalTextList Information a
- GlobalTextList Information Aa
- GlobalTextList Switch

The replacement file contains the following replacements:

- defaultold defaultnew REPLACE
- Counter: %i Counter2: %i REPLACE
- Counter: %i Counter2: %i REPLACE
- Information A Information A2 REPLACE
- Information a Information A2 REPLACE
- Information Aa Information A2 REPLACE
- Switch Switch2 REPLACE

CODESYS detects duplicate text list entries and removes them. Afterwards, the global text list contains the following source texts:

```
<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>Switch2</td>
</tr>
<tr>
<td>4</td>
<td>Counter2: %i</td>
</tr>
<tr>
<td>3</td>
<td>Information A2</td>
</tr>
</tbody>
</table>
```

The texts in the visualization have been replaced.
Managing dynamic text in text lists

You can create and translate texts in a text list for dynamic texts in order to show them dynamically in a visualization or in the alarm management. The object of type “Text list” can be located globally in the POUs view or below an application in the device tree. It contains a table with text list entries that you can edit and extend. A text list entry consists of an ID for identification, the output text, and its translation. You can add new text list entries to a text list. Menu commands are provided for this purpose.

Creating text lists for dynamic text output

Requirement: A project is open with a visualization.

1. Select an application in the POUs view or device tree and click “Project ➔ Add Object”.
2. Select “Text list”.
3. Type a name (example: Textliste_A). Click “Add” to close the dialog.
   ➔ An object of type “Text list” is created.
4. Click below the “Default” column and open the input field. Type a text (example: Information).
   ➔ The source text is created. It is used as a key in the table and as a source text for translations.
5. Type any string in the “ID” column (example: A).
   ➔ A text list entry is defined with source text and ID. If you configure the “Dynamic texts” property of an element in a visualization, then you can select the text list Textliste_A and assign the ID A.
6. Double-click in the blank line at the end of the table below “Default” and type in more text list entries.

Displaying text dynamically

In a visualization, you can configure the dynamic output of texts that were created in a text list by configuring the “Dynamic texts” property of an element. You can directly assign a text list and an ID, as well as IEC variables, where you set the values programmatically.
Requirement: A project with visualization is open and a text list is in the device tree.

1. Open the text list (example: `Text_list_A`).
2. Double-click the visualization.
   - The editor opens.
3. Drag an element to the visualization (example: a “Text field”).
4. Configure its “Dynamic texts” property by selecting one in the “Text list” property (example: `Text_list_A`) and add an ID from the text list into the “Text index” (example: ‘A’).
   - Pay attention to the single straight quotation marks. You can also assign an IEC variable of type `STRING` for the text list name and ID.
   - The IEC variables allow for programmatic access to the texts of the text lists.
5. Build the application, download it to the controller, and start it.
   - The visualization shows the text from the text list in the text field: Information A.

See also
- © Chapter 1.3.1.25.1.35 “Object 'TextList’” on page 823

1.3.1.9.10 Using image pools

An image pool is a table of image files. CODESYS references image files for use in the project (for example, in a visualization) uniquely by the ID and name of the image pool. A project can include several image pools. You can create Image pools in the device tree below the application or in the POU pool. In a library project, you can use the object properties of an image pool to turn it into a symbol library for the visualization.

We recommend that you reduce the size of image files as much as possible integrating them. This will optimize the loading time of the visualization in every visualization type: TargetVisu, WebVisu and development system.

If you insert an image element into a visualization and enter an ID (“Static ID”) in the element properties, then CODESYS automatically creates a global image pool. CODESYS uses the default name “GlobalImagePool” for this.

Please note the following when the ID of an image file appears in several image pools.
- Search order: If you selected an image managed in “GlobalImagePool”, then you do not have to enter the name of the image pool. The search order for image files is as follows:
  - 1. GlobalImagePool
  - 2. Image pools assigned to the currently active application
  - 3. Image pools next to the GlobalImagePool in the POU window
  - 4. Image pools in libraries
- Unique access: You can reference a selected image directly and uniquely by appending the image ID to the name of the image pool in the following syntax “<pool name>.<image ID>”.

See also
- © Chapter 1.3.1.25.1.24 “Object ‘Image Pool’” on page 770
- © Chapter 1.3.1.25.3.10.16 “Dialog ‘Properties’ - ‘Image Pool’” on page 1057
Creating image pools

1. Select the “Application” object in the device tree.
   Click “Project ➔ Add Object ➔ Image Pool”.
   > The “Add Image Pool” dialog box opens.
2. Type a name for the image pool (for example, "Images1") and click “Add”.
   > The image pool is added to the device tree.
3. Select the image pool object and open by choosing the command “Project ➔ Edit Object”.
4. Double-click the field in the “ID” column and assign an appropriate ID (for example, "Icon1").
   You can also add new images to the list by clicking “Imagepool ➔ Add Image File”.
5. Double-click the field in the “File name” column. Click for more settings (…).
   > The “Select Image” dialog box opens.
6. Click for more settings (…) and select the image file.
   > A thumbnail of the image file is displayed in the field of the column “Image”. The name of the file is displayed in the field of the column “File name”.

The image file can be references only by the name Images1.Icon1.

See also
- Chapter 1.3.1.25.2.15.1 “Command 'Insert Image’” on page 1009

Using image files in the 'Image' visualization element

When you insert an image element into a visualization, you can define the image type.
- Static image: Enter the image ID of the image file or the name of the image pool plus the image ID into the element configuration (property “Static ID”). Please note the comments for the search order and access.
- Dynamic image: Type the variable for defining the image file ID (for example, PLC_PRG.imagevar) in the element configuration (“Bitmap ID variable” property). You can exchange a dynamic element in online mode depending on a variable.

See also
- Visualization element 'Image' Chapter 1.3.5.18.1.5 “Visualization element 'Image’” on page 1272

Using image files for the visualization background

You can set an image in the background definition of a visualization. You can define the image by the name of the image pool plus the filename, as described above for a visualization element.

See also
- Command ‘Background’ Chapter 1.3.5.19.2.10 “Command ‘Background’” on page 1565

1.3.1.9.11 Integrating C Modules

With the C code integration plugin, externally implemented C code files can be included in CODESYS projects and C stubs can be generated from IEC objects.

In CODESYS, the “C Code Module” object type is available for this purpose. The C code files and the used IEC objects are located below a “C Code Module”. A file directory on the hard disk with C code files is assigned to each C code module.

In the project, you can generate IEC objects from a C code file in the format *.h or *.hpp (header file) in order to use them in other POUs.
The generation of C-stubs is intended for the following use cases:

- A C code file accesses an IEC object: A C code file cannot access an IEC object directly. It can access only the C stub that was generated from the IEC object.
- Generation of precompiled modules that you can merge into a library project.

After being imported, the imported source code files are part of the CODESYS project and they are therefore decoupled from the original files on the disk.

During compilation, a dynamic module is generated from a C code module and saved as part of the project. Information, warnings, and errors are displayed in the message view in the "C Code Module" category.

All dynamic modules of an application are transferred and loaded to the runtime system during the download. The runtime system must support dynamic linking for this.

---

**License for the runtime system**

The runtime system requires a license that permits C modules to be loaded. Without this license, dynamic modules cannot be linked during the download, and therefore the download will be aborted.

---

The dynamic modules are part of the boot application and they are reloaded and activated when the controller is restarted. The "Reset Origin" command unloads all C code modules in the application. The "Reset Cold" and "Reset Warm" commands do not lead to a repeated initialization of the C code modules.

---

**NOTICE!**

- **No C code for simulation mode**
  
  In simulation mode, C code is not generated and loaded to the runtime system. To simulate the code contained in the C modules anyway, you can implement it for this purpose in the respective IEC objects of the C code module.

---

CODESYS does not support the monitoring of variables in C code files or the setting of breakpoints in C source code.

**Precompiled module in a library:**

C code integration provides the capability of assigning a precompiled runtime module (example: ".dll") in the library to a device and then to save it in the library. Then, these modules can be loaded dynamically.

See also:

- Chapter 1.3.1.25.1.16 “Object ‘C Code Module’” on page 737
- Chapter 1.3.1.25.1.17 “Object ‘C Code File’” on page 739
- Chapter 1.3.1.25.1.18 “Object ‘C Implemented Library’” on page 739
- Chapter 1.3.1.11.3 "Handling of device user management" on page 300

---

**Configuring C code modules**

Requirement: A project is open that already includes a C code module.

1. Click the object "C Code Module" in the device tree.
2. Select the command “Properties” in the context menu.
3. Open the “Build” tab in the “Properties” dialog.
4. Specify the file path of the Visual Studio installation on your computer. The input assistant and the search tool (magnifying glass) are also available.
5. Specify the file path of the MS Windows SDK installation on your computer. The input assistant and the search tool (magnifying glass) are also available.
6. Specify a file path for CODESYS to store the temporary compile files.
Importing folders with C source files from the file directory

Requirement: A project is open. The project controller supports the integration of C code.

1. Select “Application” in the device tree and click “Project ➔ Add Object ➔ C Code Module”.
2. If necessary, specify a new name for your C code module in the “Add C Code Modules” dialog. If you do not, then your object will be given the standard name “C Code Module”.
3. Click the symbol (…) next to the “Source directory” input field.
4. The “Find Folder” dialog opens.
5. In the “Find Folder” dialog, select the folder containing the C source files (*.c, *.cpp, *.h, or *.hpp).
6. When you select the “Monitor folder for source code changes” option, CODESYS displays a message when changes have been made to the C source files in the selected folder of the file system.
7. Click “Add”.
   ⇒ CODESYS inserts the C code module into the device tree with the folders “Extensions”, “IEC interface”, and “Source Files”.
8. In the device tree, click the plus symbol (+) of the “Source Files” folder.
   ⇒ The imported C source files are listed in the open folder.
9. If you double-click one of the C source files (éd), then the C code file opens in your editor.

Importing individual C code files

Requirement: A project is open that already includes a C code module.

1. Click the object “C Code Module” in the device tree.
2. Click “Project ➔ Add Object ➔ C Code File”.
3. In the “Add C Code-File” dialog, use the input assistant (…) to select a file in *.c, *.cpp, *.h, or *.hpp format, and then click “Add”.
   ⇒ CODESYS inserts the selected C code file into the device tree below the “C Code Module”.
4. If you double-click the new C code file (éd) in the device tree, then it opens in the editor for modification.

Generating empty C code files

Requirement: A project is open that already includes a C code module.

1. Click the object “C Code Module” in the device tree.
2. Click “Project ➔ Add Object ➔ C Code File”.
3. In the “Add C Code File” dialog, specify the name for the new C code file with the appropriate file extension and click “Add”.
   ⇒ CODESYS inserts the selected C code file into the device tree below the “C Code Module”.
4. If you double-click the new C code file (éd) in the device tree, then it opens in the editor for modification.
Converting C code files into IEC objects for use as programming objects in applications

Requirement: A project is open that includes a C code module and C code files. For example, the C code file contains the following C code:

```c
int adder(int a, int b);
```

1. Click a C code file with the file extension *.h. In this example, it is test.h.
2. Click “Build ➤ C-Integration ➤ Create IEC Interface”.
   - The dialog “Create C Interface” opens and lists the file test.h and its function adder (int, int). Both are activated for the import.
3. Click “Import”.
4. CODESYS generates the “adder (FUN)” function and inserts it as an object in the “IEC Interface” folder in the device tree.
5. When you double-click the “adder (FUN)” object, it opens in the editor.
   - It contains the following declaration part:

```c
FUNCTION adder::DINT
VAR_INPUT
   → a::DINT;
   → b::DINT;
END_VAR
VAR
END_VAR
```
6. You can now call the adder function in the implementation part of a POU (example: adder (diVar1, diVar2);).

Creating C stubs

Requirement: A project is open that includes a C code module. A POU is added to the C code module and this POU has implemented code.

- In the device tree, select the POU below the C code module and click “C-Integration ➤ Create Stub Implementation in C”.
- CODESYS creates the objects “iec_external.c” and “iec_external.h” and adds them to the “Extensions” folder in the device tree.
- In the message view (“C Code module” category), you will find a message that an m4 file has been successfully created.

When you click “Create Stub Implementation in C”, the application is compiled automatically. If errors occur in the process, then these are indicated in the message view. In addition, please monitor the messages in the “C Code Module” category.

Assigning pre-compiled runtime modules to devices and saving them in libraries

Requirement: A library (*.library) is open in CODESYS.
1. Click “View ➔ POUs”.
   ⇒ The “POUs” view opens and displays the library project and its objects.
2. Select the library project and click “Project ➔ Add Object ➔ C-Implemented Library”.
3. Click “Add” in the “Add C-Implemented Library” dialog.
   ⇒ CODESYS adds the object “C Implemented Library” to the “POUs” view.
4. Double-click the “C Implemented Library” object.
   ⇒ The object opens in its editor
5. Click “Add” in this editor.
   ⇒ The “Select Device” dialog opens.
6. In the “Object file” input field, specify the name of a dynamically loadable module in the format *.dll or *.so.

   ![NOTICE!]
   The *.dll file must contain the title of the library project in its name. For example, if the library project is named XYlib, then the “Object file” must be called: <Name>_XYlib.dll.

7. In the “Device” window, select a device for assignment of the “Object file”.
8. Click “Select Device”.
   ⇒ CODESYS displays the created device file assignment in the editor on the tab “Compiled Components”.
9. Save the library project.

See also
- § Chapter 1.3.1.25.1.18 “Object 'C Implemented Library'” on page 739

### 1.3.1.9.12 Programmatic access to I/Os

CODESYS provides the following features for mapping project variables to input, output and memory addresses:

- Assignment of project variables to input, output and memory addresses in the “I/O Mapping” tab of the device editor
- Programmed access to I/Os
  - Variables configuration
  - AT declaration

   ![NOTICE!]
   We recommend that you define the mapping of project variables to input, output and memory addresses in the “I/O Mapping” of the editor of the respective device.

See also
- § Chapter 1.3.1.8.3 “Configuring devices and I/O mapping” on page 162
Variables configuration - VAR_CONFIG

Use the variables configuration for mapping variables of functions blocks to the process map. For declarations in the function block, assign the variables to the device inputs/outputs without providing the full address. Later, the exact address is provided centrally for all function block instances of the application in a global variable list including VAR_CONFIG declarations. This global variables list with the VAR_CONFIG declarations is termed the "variables configuration".

NOTICE!
For changes to variables that are assigned to I/O addresses, CODESYS displays them immediately in the process map. For changes to variables that are mapped by a variables configuration, CODESYS displays them not until the end of the responsible task.

Declaration of variables in functions blocks

When declaring variables in a function block, declare the variables between the keywords VAR and END_VAR and assign incomplete addresses to the variables. Mark these incomplete addresses with an asterisk (*).

Syntax:
<identifier> AT %<I|Q>*: <data type>;

Example
Define two local I/O variables: the input variable xLocIn and the output variable xLocOut.

FUNCTION_BLOCK locio
VAR
  xLocIn AT %I*: BOOL := TRUE;
  xLocOut AT %Q*: BOOL;
END_VAR

Final definition of addresses in the variables configuration of the global variables list

In the global variables list that you use as the variables configuration, define the variable declarations with the absolute addresses between the keywords VAR_CONFIG and END_VAR.

You must declare the VAR_CONFIG variables with the complete instance path, separating the individual POU and instance name by a dot (.). The declaration must include an address whose class (input/output) agrees with the class of the incomplete address (%I*, %Q*) in the function block. The data type must also agree.

Syntax:
<instance variable path> AT %<I|Q><location>: <data type>;

If the path instance does not exist, then an error is reported. CODESYS prints an error also if there is not an address configuration available for a variable that you declared with an incomplete address.

Example
The locio function block in the example above is used in a program as follows:

PROGRAM PLC_PRG
VAR
  locioVar1: locio;
  locioVar2: locio;
END_VAR

A correct variables configuration in a global variable list could then look like this:

VAR_CONFIG
  PLC_PRG.locioVar1.xLocIn AT %IX1.0 : BOOL;
  PLC_PRG.locioVar1.xLocOut AT %QX0.0 : BOOL;
  PLC_PRG.locioVar2.xLocIn AT %IX1.0 : BOOL;
  PLC_PRG.locioVar2.xLocOut AT %QX0.3 : BOOL;
END_VAR
Creating a variables configuration

Requirement: You have a project open that includes a controller configuration with a field device. The project contains a program (e.g. PLC_PRG) and a function block (e.g. func1). The field device has inputs and outputs. The textual view is selected in the options for the declaration editor.

1. Double-click a function block in the device tree (e.g. func1).
   ⇒ The function block editor opens.
2. Type the following between the keywords VAR and END_VAR:
   xLocIn AT %I*: BOOL := TRUE;
   and
   XLocOut AT %Q*: BOOL; in the next line.
   ⇒ You have declared an input variable xLocIn and assigned it to the incomplete input address %I* of a field device. You have assigned the declared output variables have to the incomplete output address %Q*.
3. Click the PLC_PRG object in the device tree and add the following to the declaration section of the program between VAR and END_VAR:

   locioVar1: func;
   locioVar2: func;
4. Right-click the “Application” object in the device tree and click “Add Object,” “Global Variable List” and then click “Add” in the “Add Global Variable List” dialog box.
   ⇒ The global variables list is added to the device tree and opens in the editor.
5. Change the keyword VAR_GLOBAL to VAR_CONFIG.
6. Click “Declarations,” “Add All Instance Paths”.
   ⇒ The following instance paths are added:

   PLC_PRG.logioVar1.xLocIn AT %I*;
   PLC_PRG.logioVar2.xLocIn AT %I*;
   PLC_PRG.logioVar1.xLocOut AT %Q*;
   PLC_PRG.logioVar2.xLocOut AT %Q*;
7. Now, replace the incomplete addresses %I* and %Q* with the absolute, complete addresses.

See also
● Chapter 1.3.1.24.2.10 “Configuration variables - VAR_CONFIG” on page 497
● Chapter 1.3.1.9.12.2 “AT declaration” on page 228
● Chapter 1.3.1.24.4.12 “Addresses” on page 566

AT declaration

In the variables declaration, the code AT assigns a project variable to a specific input address, output address, or memory address of the PLC that is configured in the device tree. You can also define the assignment of variables to an address in the “I/O Mapping” dialog of the device in the PLC configuration.

Syntax

<variable name> AT <address> : <data type>;

<address> : %<memory area prefix> ( <size prefix> )? <memory position>
The AT declaration allows you to give the address a meaningful name. You can make any necessary changes for the input or output signals at just one location, for example in the declaration.

### Examples

<table>
<thead>
<tr>
<th>Declaration</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>VAR wInput AT %IW0 : WORD; END_VAR</td>
<td>Variable declaration with address information of an input word</td>
</tr>
<tr>
<td>VAR xActuator AT %QW0 : BOOL; END_VAR</td>
<td>Boolean variable declaration</td>
</tr>
<tr>
<td>Note: For Boolean variables, one byte is allocated internally if a single bit address is not specified. A change in the value of <code>xActuator</code> affects the range from <code>QX0.0</code> to <code>QX0.7</code>.</td>
<td></td>
</tr>
<tr>
<td>VAR xSensor AT IX7.5 : BOOL; END_VAR</td>
<td>Boolean variable declaration with explicit specification of a single bit address. On access, only the input bit 7.5 is read.</td>
</tr>
<tr>
<td>VAR xSensor AT IX* : BOOL; END_VAR</td>
<td>For the address specification, the placeholder <code>*</code> is given instead of the memory position. The final address specification is done in the variables configuration.</td>
</tr>
<tr>
<td>Note: This is possible in function blocks.</td>
<td></td>
</tr>
</tbody>
</table>

If you assign a variable to an address, please note the following:

- You cannot write to variables that are placed at inputs. This will cause a compiler error.
- You can perform AT declarations only for local and global variables, not for input/output variables of POUs.
- Furthermore, AT declarations cannot be used in persistent variable lists.
- If you use AT declarations for structure components or function block variables, then all instances use the same memory. This is just like using static variables in classic programming languages, such as C.
- The memory layout of structures also depends on the target system.

---

**NOTICE!**

If you do not specify a single bit address explicitly, then Boolean variables are allocated byte-by-byte.

### Example

```plaintext
PROGRAM PLC_PRG
VAR
  xVar AT %QW0 : BOOL;
END_VAR

xVar := TRUE;
```

When the variable `xVar` is written, the output memory range from `QX0.0` to `QX0.7` is affected.

See also

- § Chapter 1.3.1.9.12.1 “Variables configuration - VAR_CONFIG” on page 227
- § Chapter 1.3.1.24.2.10 “Configuration variables - VAR_CONFIG” on page 497
- § Chapter 1.3.1.24.4.12 “Addresses” on page 566
1.3.1.9.13 Checking syntax and analyzing code

CODESYS provides useful functions for detecting errors and assisting you while you create programs. The syntax check flags errors and prints them to the message view as early as the programming phase.

The static code analysis in CODESYS also assists you in complying with defined coding guidelines and detecting weak constructs.

See also

- Chapter 1.3.1.9.13.1 “Checking syntax” on page 230
- Chapter 1.3.1.9.13.2 “Analyzing code statically” on page 230

Checking syntax

When you input code, the precompile in CODESYS already runs some basic checks. Then, wavy underlines appear under buggy code in the editor and an error message is printed to the messages view.

After programming, you must check the syntax of the application by clicking “Build ➔ Build” or by pressing [F11]. In this step, the compiler checks the program and prints any other errors.

CODESYS automatically generates the application code from the source code that was written in the development system. This is done automatically before downloading the application to the controller. Before the application code is generated, a test is performed for checking the assignments, the data types, and library availability. In addition, the memory addresses are allocated when the application code is generated. You can execute this command explicitly by clicking “Build ➔ Generate Code”. This is useful for detecting any errors in your source code, even when the controller is not connected yet.

CODESYS prints all errors and warnings to the "Build" category of the messages view. Double-clicking the error message opens the respective POU in the editor with the buggy code marked. As an alternative, you can also jump to the buggy code by right-clicking the error message.

Please note the settings for this in the CODESYS options.

See also

- Chapter 1.3.1.25.3.13.11 “Dialog 'Options' - 'SmartCoding’” on page 1082

Analyzing code statically

You can subject your source code also to static analysis (lint) during the code generation. This determines whether or not your source code complies with the coding guidelines that you defined - according to the idea behind the lint analysis tool.

- You activate the rules to the checked in the “Project Settings” dialog in the “Static Analysis Light” category. The check itself is performed automatically each time code is generated, for example when you click “Build ➔ Generate Code”. If divergence from the rules is determined, then it is reported as an error message in the “Build” category of the message view. The reported errors have the prefix SA<number>.

![NOTICE!]

For static code analysis with “Static Analysis Light”, only the application code of the project is checked. Libraries are excluded from the check.

GVL variables in the “POUs” view are not necessarily checked: If you have a project with several applications, then only the objects in the active application are checked. If you have only one application, then the objects in the common POU pool are also checked.
"Static Analysis Light" includes only a reduced set of rules in the default development system. A larger set of rules, additional naming conventions, and metrics are available when you install the CODESYS Static Analysis add-on.

**Deactivating lines of code in the implementations with pragmas from the static analysis**

By means of the pragma `{analysis ...}`, you can mark code so that the specified rules are not checked. As a result, the marked lines of code are not subjected to static analysis. The marked code is ignored during the check.

**Syntax:**

```
{analysis <sign><rule number>|,<other combinations of signs and rules, comma-separated>}
```

- `<rule number>`: Deactivate the rule SA<rule number>.
- `<rule number>`: Activate the rule SA<rule number>.

**Requirement:** Rules are activated in the “Project Settings” dialog.

1. Add the pragma `{analysis -<number>}` above the line of code that contains code not to be checked first of all. For example, for the rule SA0024:
   - The line of code is the first line of the code snippet that is not checked with rule 24.
2. Add the pragma `{analysis -<number>}` below the line of code that contains code not to be checked first of all. For example, for the rule SA0024:
   - The line of code above is the last line of the code snippet that is not checked with rule 24.

**Example:**

**Ignore untyped literal**

```plaintext
{analysis -24)
nTest := 99;
iVar := INT#2;
{analysis +24)
```

The rule “SA0024: Untyped literals only” is deactivated for two lines. An error is not issued although the code does not correct to: `nTest := DINT#99;`

**Example:**

**Ignore several rules**

```plaintext
{analysis -10, -24, -18)
...
{analysis +10, +24, +18)
```

“SA0010: Arrays with only one component”
“SA0018: Unusual bit access”
“SA0024: Untyped literals only”

**However, you cannot deactivate the rule SA0004: “Multiple Write Access on Output” with a pragma.**
Excluding programming objects with pragmas from the static analysis

Syntax:
{attribute 'analysis' := '-<rule number>[,<other negative rule numbers, comma-separated>]'}

When you insert the attribute pragma in the declaration part of a programming object, the specified rules are excluded for the entire programming object. If multiple rules are excluded, then the rules are each comma-separated with a dash and a number. A pragma statement for activation is not required.

Example

{attribute 'analysis' := '-33, -31'}
TYPE LocalData :
STRUCT
  iLocal : INT;
  uiLocal : UINT;
  udiLocal : UDINT;
END_STRUCT
END_TYPE

The rules SA0033 and SA0031 are ignored for the structure LocalData.

{attribute 'analysis' := '-100'}
big: ARRAY[1..10000] OF DWORD;

The rule SA0100 is ignored for the array big.

See also
- Chapter 1.3.1.25.3.11.8 “Dialog ‘Project Settings’ - ‘Static Analysis Light’” on page 1066

1.3.1.9.14 Orientation and navigation

Using the cross-reference list to find occurrences

The occurrences of symbols of a variable, a POU (program, function block, function), or a DUT can be displayed in a cross reference list. Then you can jump from the list directly to the corresponding locations in the project.

There are two ways to search for occurrence locations of a symbol:

- Plain text search: You manually specify a text (symbol name, placeholder) in the “Cross Reference List” view.
- Search for a specific declaration:
  - In the “Cross-Reference List” view, you select the declaration from the input assistant.
  - The focus is on a symbol name in the POU editor and you start the cross-reference search from the context menu.
  - The focus is on a symbol name in the POU editor, the “Cross-Reference List” view is open, and the cross-reference search executes automatically.
  - In the “Cross-Reference List” view which already lists occurrence locations for several declarations, you limit these results to a specific declaration.

Cross references with text search by symbol name

Requirement: The “Cross-Reference List” view is open.
1. Specify a string in the field next to the name, for example the identifier of the variable for which you want to find the occurrence location in the project. Example: "iCounter".
   
   For the text search, you can use the asterisk "*" (for any number of characters) or the question mark "?" (for an exact number of characters) combined with a substring of a variable identifier.
   
   Use the percent sign "%" to search for IEC addresses. Examples: "%MW8", "%M?".

2. Click the button to start a text search in the project.
   
   The view "Cross-Reference List" opens and displays the occurrence locations for the iCounter variable. The declaration parts are always displayed in the project with the occurrence location indented.

3. Double-click an occurrence location in the cross-reference list.
   
   The respective object opens in the editor with the marked occurrence location.

Requirement: A POU is open in the editor.

Cross-references for a specific symbol declaration

From the POU editor, with a menu command

1. Set the cursor at the identifier of the symbol (variable, POU) in the declaration part or implementation part.
2. Click “Browse for Symbol ➔ Browse Cross-References” in the context menu or “Edit” menu.
   
   The “Cross-Reference List” view opens and shows the occurrence locations of the variables or POU.

If the “Cross-Reference List” view is already open, then you can also search the occurrence locations for a specific result as follows:

From the POU editor, automatic

Select the “Automatically list selection in cross reference view” check box in “Tools ➔ Options” (“SmartCoding” category). Select the name of the symbol in the POU, or set the cursor in the name.

Depending on the position of the selection or cursor, the cross-reference list automatically shows the occurrence locations for the respective symbol.

In the cross-reference list view, with input assistance

In the “Cross-Reference List” view, use the input assistant 📅 to specify a symbol name in the field next to “By declaration”.

The cross reference list displays the occurrence locations for the symbol.

In the cross-reference list view, limited to a specific declaration

If multiple declarations for a symbol are listed in the “Cross-Reference List” view, for example after a text search, then you can reduce the display to one result: Select the line with the desired declaration and click the button or click “Limit Results to Current Declaration” in the context menu.

The cross-reference list includes only the occurrence locations for the selected declaration.

See also

- Chapter 1.3.1.25.2.3.13 “Command ‘Cross Reference List’” on page 886
- Chapter 1.3.1.25.2.2.29 “Command ‘Browse Cross-References’” on page 871
- Chapter 1.3.1.25.3.13.11 “Dialog ‘Options’ - ‘SmartCoding’” on page 1082
Finding declarations

CODESYS provides the capability of searching the entire project for the definition location of a variable or function. The block that includes the definition opens in the editor with the marked declaration.

Finding the declaration of a variable

Requirement: You have opened a POU in the editor.

1. Set the cursor at an identifier in the implementation section.
2. Click “Edit ➔ Browse ➔ Go to Definition”.
   ⇒ The POU with the declaration opens in the editor with the variable definition marked. If the definition is located in a compiled library, then the respective block opens in the library manager.

> You can execute this command in both online and offline mode.

Examples

The following block includes a function block definition (fbinst), a program call (prog_y()), and a function block call (fbinst.out):

```
VAR fbinst:fb1; ivar:INT; END_VAR prog_y(); ivar:=prog_y.y;
res1:=fbinst.out;
```

If the cursor is located at prog_y, then the command opens the program prog_y in the editor.
If the cursor is located at fbinst, then this command focuses in the declaration section at line fbinst:fb1;
If you set the cursor at out, then this command opens the function block fb1 in its editor.

See also
● Chapter 1.3.1.25.2.2.36 “Command ‘Go to Definition’” on page 875

Setting and using bookmarks

Bookmarks are used for easy navigation through long programs. You can use bookmarks in all implementation language editors, except SFC (sequential function chart). Commands help to navigate directly to the marked position in the program.

Setting and deleting bookmarks

Requirement: The POU is open in the editor.

1. Set the cursor at any program line.
2. Click “Edit ➔ Bookmarks ➔ Toggle Bookmark”.
   ⇒ A bookmark is set at this position in the program. This is marked by the bookmark symbol ☐.
3. Set several bookmarks at different places in the program.
4. Set the cursor at a bookmarked program line.
5. Click “Edit ➔ Bookmarks ➔ Toggle Bookmark”.  
⇒ The bookmark is removed. The bookmark symbol □ is deleted.

As an alternative to this, you can delete one or more bookmarks in the “Bookmarks” view by clicking the ✗ button. For this purpose, the corresponding bookmarks have to be selected in the “Bookmarks” view.

```
Click “Edit ➔ Bookmarks ➔ Clear All Bookmarks (Active Editor)” to remove all bookmarks from the active POU.
```

```
In order to delete all bookmarks in a project, click “Clear All Bookmarks”. However, for this command to be available, you first have to add it to a menu by means of the command “Tools ➔ Customize”.
```

See also
- % Chapter 1.3.1.25.2.2.22 “Command ‘Toggle Bookmark’” on page 869
- % Chapter 1.3.1.25.2.2.27 “Command ‘Clear All Bookmarks (Active Editor)’” on page 871
- % Chapter 1.3.1.25.2.2.28 “Command ‘Clear All Bookmarks’” on page 871

Jumping to bookmarks within a POU

Requirement: The POU is open in the editor. Multiple bookmarks are set.

1. Click “Edit ➔ Bookmarks ➔ Next Bookmark (Active Editor)”.  
⇒ Depending on the current cursor position, the cursor jumps to the next bookmark (see below).

2. Click “Edit ➔ Bookmarks ➔ Previous Bookmark (Active Editor)”.  
⇒ Depending on the current cursor position, the cursor jumps to the previous bookmark (see above).

See also
- % Chapter 1.3.1.25.2.2.23 “Command ‘Next Bookmark (Active Editor)’” on page 870
- % Chapter 1.3.1.25.2.2.25 “Command ‘Previous Bookmark (Active Editor)’” on page 870

Jumping to bookmarks of different POUs in a project

A project is open with multiple POUs. Multiple bookmarks are set in different POUs.

1. Click “View ➔ Bookmarks”.  
⇒ The “Bookmarks” view opens.

All bookmarks in the project are listed in a table in the view.

2. Click the ✳ “Next Bookmark” button.  
⇒ In the “Bookmarks” view, the bookmark in the row below the selected bookmark is selected.

The POU with the recently selected bookmark in the table opens in the editor and the row with the bookmark is selected in the POU.
3. As in step 2, you can click the ‘Previous Bookmark’ button to jump to the bookmark in the project that is displayed in the row above it in the “Bookmarks” view.

See also
- Chapter 1.3.1.25.2.3.11 “Command 'Bookmarks’” on page 884
- Chapter 1.3.1.25.2.2.26 “Command 'Previous Bookmark’” on page 870
- Chapter 1.3.1.25.2.2.24 “Command 'Next Bookmark’” on page 870

1.3.1.9.15 Searching and replacing in the entire project

In CODESYS you can search for strings in single objects or project-wide. If required, you can replace the string found.

1. Choose the command “Search” in the main menu “Edit ➔ Search Replace”.
   - The dialog “Find” opens.
2. Enter the string to be found in the field “Find what”.
3. Activate the search options
4. Define the objects to be searched by choosing an entry from the combobox “Search”.
5. Click on the button “Find Next”.
   - The first hit is displayed.
6. Click on the button “Replace” to replace the string found by a different one.
7. Click on the button “Find All” to get a list of all hits.

See also
- Chapter 1.3.1.25.2.2.2 “Command 'Find', 'Find in Project’” on page 863
- Chapter 1.3.1.25.2.2.3 “Command 'Replace', 'Replace in Project’” on page 864

1.3.1.9.16 Refactoring

In general, refactoring is a technique for improving the design of existing software code without changing the way it functions.

In CODESYS, refactoring provides functions for renaming objects and variables and updating referenced pins. You can display all occurrences of renamed objects and variables and then rename them all at once or individually. In “Tools ➔ Options”, you can also configure where CODESYS will prompt you for refactoring.

Renaming global variables

Requirement: A project is open that includes at least a function block “FB” and a global variable list. The global variable list “GVL” is open in the editor and contains a variable declaration (example: iGlobal). “FB” uses iGlobal.

1. Select the global variable name iGlobal.
2. Right-click the variable and click “Refactoring ➔ Rename iGlobal”.
3. In the “Rename” dialog, type a name in the “New name” input field, for example iGlobalOK, and click “OK”.
   - The “Refactoring” dialog opens. In the device tree view on the left, the “GVL” and “FB” objects are highlighted in red and yellow. In the view on the right, “FB” in is open in its editor and iGlobal has already been renamed as iGlobalOK.
4. Click “OK”.
   - No global variable iGlobal is in your project. Now iGlobalOK is everywhere.
1. Select the global variable name iGlobal.

2. Right-click the variable and click “Refactoring ➔ Rename iGlobal”.

3. In the “Rename” dialog, type a name in the “New name” input field, for example iGlobalTest, and click “OK”.
   ⇒ The “Refactoring” dialog opens. In the device tree view on the left, the “GVL” and “FB” objects are highlighted in red and yellow. In the window on the right, the function block “FB” is open in its editor. iGlobalTest is listed instead of iGlobal.

4. Right-click in the view on the right.

5. Click “Reject this Object” and click “OK”.
   ⇒ The global variable iGlobal is available in “FB” in your project. The variable iGlobalTest is now specified in the objects where the previous variable occurred.

In the declaration part of blocks, you can add and delete input and output variables by using the refactoring commands. CODESYS performs updates at the occurrence locations and calling locations of the blocks. You can accept or reject these updates individually. The “Refactoring” dialog also opens for this purpose.

Requirement: The FCT (function type) POU is open in the editor. The function already contains the input variables input1, input2, and inputx. They are called in the PLC_PRG and POU programs.

1. Set the focus in the declaration part of the FCT function.

2. Click “Refactoring ➔ Add Variable”.
   ⇒ The default dialog opens for declaring variables.

3. Declare the variable input_3 with the scope of VAR_INPUT and data type INT. Click “OK” to close the dialog.
   ⇒ The “Refactoring” dialog opens (see figure below). The affected locations are marked in yellow. (1)+(2)

4. In the upper right corner, select “Add inputs with placeholder text” from the drop-down list. (3).

5. In the left side of the window, click one of the highlighted objects (for example, PLC_PRG). Right-click and choose the “Accept Whole Project” command to add the new variable at the new location of use in FCT for the entire project.
   ⇒ You see the change in the implementation part of PLC_PRG in the view on the right: The placeholder _REFACTOR_ appears at the location where the new variable was added.

6. Click “OK” to close the “Refactoring” dialog.

7. Click “Edit ➔ Find”. Search the project for “_REFACTOR_” to check and edit the affected locations.

8. Note: As an alternative, you can insert the new variable with another initialization value without working with a placeholder first. In this case, in Step 4 you select “Add inputs with the following value” and type the value in the field on the right side of the drop-down list.
Example of a new variable with placeholder text in a CFC block:

Please note that you can also remove variables with refactoring.

**Reordering variables in the declaration**

In the declaration part of function blocks, you can change the order of declarations by refactoring. This is possible for declarations with scope `VAR_INPUT`, `VAR_OUTPUT`, or `VAR_IN_OUT`.

**Requirement:** The declaration part of a POU is open and includes declarations, for example:

```plaintext
VAR_INPUT
  invar2 : INT;
  invar1 : INT;
  in : DUT;
  bvar : BOOL;
  invar3 : INT;
END_VAR
```

1. Right-click in this declaration block to access the context menu.
2. Click “Refactoring ➔ Reorder Variables”.
   - The “Reorder” dialog opens with a list of `VAR_INPUT` variables.
3. Drag the "invar1 : INT;" entry to the position before the "invar2." entry.
   ⇒ The invar1 declaration is now at the top position.
4. Click “OK” to close the dialog.
   ⇒ The “Refactoring” dialog opens. The affected locations are marked in yellow (see figure above).
5. Click “OK” to accept the new order for the function block.

Changing a variable declaration and applying refactoring automatically

Refactoring helps you in the declaration when renaming variables (by means of "Auto declare").

- Requirement: Function block fb_A.
  1. Click “Tools ➔ Options”.
     ⇒ The “Options” dialog opens.
  2. Select the “Refactoring” category.
  3. In “Auto-Declare”, activate the options “On renaming variables” and “On adding or removing variables, or for changing the namespace”.
  4. Double-click the function block fb_A.
  5. Select a variable in the declaration of fb_A, for example iA. As an alternative, you can set the cursor before or in the variable.
  6. Specify “Edit ➔ Declare variable” ([Shift]+[F2]).
     ⇒ The “Declare Variable” dialog opens. The dialog includes the settings of iA.
  7. Change the name of iA to iCounter_A.
  8. The option “Changes by means of refactoring” appears and is activated.
  9. Click “OK”.
     ⇒ The dialog “Refactoring” “Renaming from iA to iCounter_A” opens. All locations affected by the variable renaming are marked there.
10. Click “OK” to close the dialog.
     ⇒ The changes are applied.

See also
- Chapter 1.3.1.25.2.2.39 “Command ‘Refactoring’ - ‘Rename <...>’” on page 876
- Chapter 1.3.1.25.2.2.40 “Command ‘Refactoring’ - ‘Update Referenced Pins’” on page 877
- Chapter 1.3.1.25.2.2.41 “Command ‘Refactoring’ - ‘Add Variable’” on page 878
- Chapter 1.3.1.25.2.2.42 “Command ‘Refactoring’ - ‘Remove <variable>’” on page 879

1.3.1.9.17 Task configuration

In the task configuration, you define one or more tasks for controlling and executing the application program in the controller. Each application must include a “Task Configuration” object.

A task is a time-based flow unit of an IEC program. You define a task with a name, a priority, and a type, which determines which condition triggers the start of the task. You can define this condition either by time (cyclic-interval, freewheeling) or by the occurrence of an internal or external event to process the task. Examples of an event are the rising edge of a global project variable or an interrupt event of the controller.
A task calls one or more program blocks (POUs). These programs can be application-specific (objects below the application in the device tree) or project-specific (objects available in the POU window). In the case of a project-specific program, the application instances the project-global program. If CODESYS processes the task in the current cycle, then the programs are executed for the duration of a cycle.

With the combination of priority and condition, you define the order in which the tasks are processed. You can configure a watchdog for each task, and you can link a start, stop, and reset directly to the execution of the project block.

Rules for the processing order of the defined tasks:

- If the task condition is satisfied, then CODESYS processes the task.
- If several tasks satisfy the condition for processing at the same time, then CODESYS processes the tasks with the highest priority first.
- If several tasks with the same priority level satisfy the condition for processing at the same time, then CODESYS processes the longest waiting task first.
- The program calls are processed in the order they appear in the configuration dialog of the task.
- If a called program has the same name in the device tree of the application and in a library or project-global in the POU window, then the application program is used.

Attention

All tasks share one process map. The reason is as follows: When each task has its own individual process map, performance is compromised. However, the process map can be consistent only with one task. When you create a project, you must ensure that the application copies the input and output data to a safe location in case of conflicts. Modules, such as the library SysSem, provide the capability of solving consistency and synchronization problems.

Consistency problems can also occur when accessing other global objects, such as global variables or blocks. Consistency problems always occur if several tasks read and write to one variable. Modules, such as the library SysSem, are available as a solution.

Creating a task configuration

Requirement: The open project includes a program-type POU and a “Task Configuration” with a “Task” object has been inserted below “Application” in the device tree.

1. Double-click the task object below “Task Configuration” in the device tree.
   - The “Configuration” tab of the task object opens.
2. In the “Type” dropdown list., click “Cyclic”.
   - The “Interval (e.g. t#200ms)” input field appears.
3. Enter t#300ms in the “Interval (e.g. t#200ms)” input field.
4. Click “Add Call”.
   - The Input Assistant opens.
5. In “Input Assistant ➔ Categories” ➔ “Programs”, click the desired POU and then click “OK”.
   - CODESYS inserts the selected POU into the POU list of the “Configuration” tab and below the task object in the device tree.
   - When the application is executed from the controller, CODESYS executes the selected POU in cyclical intervals of 300 ms.

- Chapter 1.3.1.25.1.38.1 “Tab 'Configuration’” on page 839
Multicore

Object: “Task Configuration”

More and more of today's controllers have processors with multiple CPU cores. In order to take advantage of this for the IEC program and improve performance, CODESYS provides options for distributing the IEC tasks over these CPU cores. In order to benefit from a multicore processor as well, the IEC program has to be distributed into multiple tasks.

There are two basic different strategies in CODESYS for distributing IEC tasks over CPU cores:

- Fixed linking of an IEC task to a CPU core:
  The task is always executed on this specific CPU core. In this case, multiple tasks can also be bundled to one and the same CPU core if the IEC program is not yet able to be executed on multiple CPU cores at the same time.

- The IEC task is executed on all CPU cores:
  The operating system takes control of distributing the task on the CPU cores.

**NOTICE!**

When the IEC tasks are distributed over CPU cores, some changes result in the behavior in the IEC program, which have to be considered:

- The processing of IEC tasks by priority is no longer a given. They are processed by priority only if the tasks are bundled together to one CPU core.
- The cycle consistency of the data in the IEC task with the highest priority is no longer a given. Therefore, the data has to be copied locally at the beginning of the IEC task cycle if the values should not change during the cycle.
- For consistent counters (incrementer, decrementer), the atomic external library function `SysCpuAtomicAdd()` should always be used (for more details, see `SysCpuHandling.library`).

**NOTICE!**

Data consistency

- Bit access (data type `BIT`) is not processed consistently on marker CPUs in the IEC program. For this we recommend that you use the external library function `SysCpuTestAndSetBit()` (for more details, see `SysCpuHandling.library`).
- Simple data types up to a width of 32 bits (`BIT, BYTE, WORD/INT, DWORD/DINT, etc.`) are processed consistently in the IEC program on marker CPUs as well.
- Data types with 64 bits (`LINT, LWORD, LREAL`) are processed consistently in the IEC program on marker CPUs only. You do not have to do anything explicit in this case.
- To access complex data types (`STRINGs, FBs, STRUCTs, ARRAYs`), you have to make arrangements yourself for the synchronization/consistency.
- In the task configuration in the “Variable Usage” tab, you can define whether a date in IEC has read or write access.
- On multicore systems, a memory reordering effect can occur. For more information, see `__MemoryBarrier()` IEC-Operator.

See also

- ¶ Chapter 1.3.1.25.1.19.10 “Tab 'PLC Shell'” on page 753
- ¶ Chapter 1.3.1.25.1.37.4 “Tab 'Variable Usage'” on page 838
- ¶ “Multicore operators” on page 509
Distributing tasks over multiple processor cores

Requirement: At least two tasks are defined in your application, for example “MainTask (IEC task)” and “LowTask (IEC task)”.

1. Open the “Task Configuration” object in the editor.
2. Switch to the “Task Groups” tab.
   - The overview shows a “IEC-Tasks” task group.
3. Add a new task group by clicking the “Add Group” button.
   - The group “NewGroup” is added.
4. Double-click the group name “NewGroup” and change it to “LowGroup”.
5. Select an assignment in “Core”, for example “1”.
6. Open the task “LowTask (IEC task)” in the editor.
7. Select the new group “LowGroup” in “Task Group”.
   - The task “LowTask” is processed now by processor core 1 due to its group membership in “LowGroup”.

See also
- Chapter 1.3.1.25.1.37.5 “Tab ‘Task Groups’” on page 838
- Chapter 1.3.1.25.1.38.1 “Tab ‘Configuration’” on page 839

Display the processor load in the trace for each processor core

In CODESYS, you can visualize the processor load in a “DeviceTrace” object. For more information, refer to the following help section about displaying device traces in the project:

See also
- Chapter 1.3.1.13.3.4 “Accessing all traces on the controller” on page 342

1.3.1.9.18 Encrypting an application

You achieve the know-how protection and copy protection of a boot application with the help of PLC-specific license management and its settings in the object properties of the application. In this case, the download code and boot application are encrypted.

Encryption with a dongle

Requirements: You have a project with an application that you want to download to the controller as an encrypted boot application. A security key for license management is connected to your computer.

1. Select the application in the device tree.
2. Select the “Properties” command in the context menu.
   - The “Properties - <application name>” dialog opens.
3. Click the “Encryption” tab.
4. For “Encryption Technology”, select the “Simple Encryption” option and type the “Product Code” that you received from the hardware manufacturer for the controller. Depending on the controller, it is protected either by a security key (firmcode is shown automatically) or by an integrated Wibu SD card for example.
5. Click “Online ➔ Login” and download the application.
   - If the matching security key and/or valid license is available, then you can download the application to the controller. By default, a boot application is automatically created at this time in the controller directory. The default setting is defined in the application “Properties”, in the “Boot Application” category.
6. Logout, change the application, and login again.
   ⇒ You are prompted to perform an online change. The dialog provides the option of
   updating the boot application on the PLC. If the security key and license match, then
   you can log in. If not, then you receive a corresponding message.

**Encrypting with certificates**

Requirements: You have a project with an application that you want to download to the con-
troller as an encrypted boot application. In the Windows Certificate Store of your computer, you
have a certificate of this controller for encrypting the application. Note: In case you want to
download the application to different controllers, you will need the appropriate certificate for
each controller.

1. Select the application in the device tree.
2. Select the “Properties” command in the context menu.
   ⇒ The “Properties - <application name>” dialog opens.
3. Click the “Encryption” tab.
   ⇒ The “Certificates” group is enabled.
5. If there are not any certificates listed in the table, then click the button.
   ⇒ The “Certificate Selection” dialog opens for selecting a certificate from the local Win-
dows Certificate Store.
6. In the lower area, select a certificate and add it to the upper area by clicking the button,
   Click “OK” to confirm.
   ⇒ The certificate is shown in the “Certificates” group of the “Encryption” dialog.
7. Select the certificate and click “Apply” or “OK”.
   ⇒ The certificate is now used to encrypt the application. It can only be transferred to the
   controller on computers that have an corresponding key installed in the Windows Cer-
tificate Store.

See also
- ☞ Chapter 1.3.1.23.3 “Security for the runtime system / PLC” on page 423
- ☞ Chapter 1.3.1.6 "Protecting and saving a project" on page 139
- ☞ Chapter 1.3.1.6.7 “Encrypting projects with certificates” on page 149
- ☞ Chapter 1.3.1.25.3.10.3 “Dialog ‘Properties’ - ‘Encryption’” on page 1047

**Signing a boot application**

1. Click in the status bar of CODESYS to open the “Security Screen” view. Then select a
   certificate with a private key for a user profile for the “Digital signature”. The procedure is
described in the instructions "Configuring a certificate for the digital signature in a user
profile".
2. Double-click the certificate for the “Digital signature” in the “User” tab.
   ⇒ The “Certificate” dialog opens.
3. On the “Details” tab, click “Copy to file”.
   ⇒ The “Certificate Export Wizard” starts.
4. In the “Export Private Key” prompt, select the “No, do not export the private key” option.
5. For “Export File Format”, select the “DER encoded binary X.509 (.CER)” option.
6. In the next step, select a file name and the location for the certificate.
7. After the last step “Finish”, a message appears that the export was successful.
8. After successful export to CODESYS, open the device editor by double-clicking the controller in the device tree and selecting the “Files” tab for the file transfer.

9. Select the “Path” cert/import in the right side of the “Runtime” dialog.

10. On the left side of the dialog for “Host”, select the path in the file system where you saved the exported certificate and selected the certificate.

11. Click ➤. The certificate is copied to the cert/import folder.

12. Click the “PLC Shell” tab.

13. Type the command cert-import trusted <file name of the certificate.cer> in the input line of the tab and press the [Enter] key. Note that the file name is specified with the extension .cer; otherwise the certificate is not imported successfully.
   ➤ The certificate is created on the controller under trusted. With this certificate, the controller can test the integrity of the boot application.

14. Open the “Security Screen” by double-clicking ✨ in the status bar.

15. If you want that downloads, online changes, and boot applications of your project are always encrypted, then select the “Enforce signing of downloads, online changes and boot applications” option in the “Security level” group on the “User” tab. To do this, the “Enforce encryption of downloads, online changes and boot applications” option also has to be selected.

See also :

- Chapter 1.3.1.6.7 “Encrypting projects with certificates” on page 149
- “Encryption, signature” on page 421
- Chapter 1.3.1.25.2.3.18 “Command ‘Security Screen’” on page 891
- Chapter 1.3.1.25.3.10.3 “Dialog ‘Properties’ - ‘Encryption’” on page 1047

Encryption the download, online change, and boot application

Requirement: The CODESYS Security Agent add-on product is installed.

The “Security Screen” view provides an additional tab: “Devices”. This allows for the configuration of certificates for the encrypted communication with controllers. In this case, see the help for CODESYS Security Agent.

Alternatives:

If you the CODESYS Security Agent is not available to you, then you can proceed as follows by means of the PLC shell of the device editor:

In order to use certificates on the controller for the encryption of downloads, online changes, and boot applications, these certificates first have to be generated on the controller and loaded from the controller and installed in the Windows Certificate Store.

- Requirement: You are connected to the controller.

1. Open the device editor by double-clicking the controller in the device tree, and select the “PLC Shell” tab.
   ➤ The tab appears with a blank display window. Below that is a command line.

2. Type ? in the command line and press the [Enter] key.
   ➤ All commands are listed in the display window.

3. Type the following command in the command line: cert-getapplist.
   ➤ All used certificates are listed with information about components and availability with certificates.

4. If no certificate is available for the CmpApp component, then type the command cert-genselfsigned <Number of the Component in the applist>.
5. Click the “Log” tab and then the refresh button ( הראשי).
   ➤ The display shows whether or not the certificate was generated successfully.

6. Type in `cert-getcertlist` and press the [Enter] key.
   ➤ Your own certificates are listed that can be used for encryption. The information
     Number and Key usage(s) are useful in the next step.
     Number: The number is specified as a parameter in the next step.
     Key usage(s): Data encryption means that this is a certificate of the controller
     for a download, online change, and boot application.

7. Export the required certificate by typing in the command `cert-export own 0` and press
   the [Enter] key.
   ➤ The display shows that the certificate has been exported to a cert directory.

8. Click the “Files” tab of the device editor.

9. Click the refresh button ( הראשי) in the right part of the dialog in “Runtime”.
   ➤ The list of files and directories is refreshed.

10. Open the “cert” folder in the list and then the “export” subfolder.

11. In the left part of the dialog in “Host”, open the directory where the certificate of the con-
    troller will be loaded.

12. In the right part of the dialog, select the certificate that you have exported and click
    ➤ The certificate is copied to the selected directory.

13. In the file explorer, go to the directory where the certificate was copied and double-click
    the certificate.
   ➤ The “Certificate” dialog opens and shows the information about this certificate.

14. On the “General” tab, click “Install Certificate”.
   ➤ The “Certificate Import Wizard” starts.

15. In the “Certificate Storage” dialog, for “Certificate Import Wizard”, select the “Store all
certificates in the following store” option and then select the “Controller Certificates” folder.
   ➤ The controller certificate is imported into the Windows Certificate Store in the
   “Controller Certificates” folder. Now the certificate is available for the encryption of
   boot applications, downloads, and online changes.

16. Open the “Security Screen” by double-clicking in the status bar.

17. If you want that downloads, online changes, and boot applications of your project are
    always encrypted, then select the “Enforce encryption of downloads, online changes and
    boot applications” option in the “Security level” group on the “User” tab.

18. Open the “Project” tab and double-click the application in the “Encryption of boot
application, download and online change” area.
   ➤ The properties dialog of the application opens.

19. Click the “Encryption” tab, select “Encryption with certificates” in the “Encryption
technology” list box, and click
   If the “Enforce encryption of downloads, online changes and boot applications” option
   is selected in the “Security Screen”, then “Encryption with certificates” is already selected.

20. In the “Certificate Selection” dialog, select the respective certificate from the “Controller
    Certificates” folder and click

21. Click “OK” to confirm the dialog.
   ➤ The certificate is displayed in the properties dialog.
22. Confirm the properties dialog of the application.
   ⇒ The certificate is shown on the “Project” tab of the “Security Screen” in the “Encryption of boot application, download and online change” group.
   The boot application, download, and online change are encrypted.

See also
- Help for the CODESYS Security Agent add-on product
- Chapter 1.3.1.25.1.19.10 “Tab ‘PLC Shell’” on page 753
- Chapter 1.3.1.25.2.3.18 “Command ‘Security Screen’” on page 891

Deleting a certificate for the encryption of boot application, download and, online change

Requirement: The CODESYS Security Agent add-on product is installed. A certificate with the information “Encrypted Application” is already installed on your computer.

1. In the “Security Screen” view, on the “Project” tab, in the bottom view, click the entry for the application.
   ⇒ The “Properties” dialog for the application opens with the “Encryption” tab.
2. For “Encryption Technology”, select “Encryption with certificates”. In the “Certificates” group, click .
3. In the “Certificate Selection” dialog, delete the certificate as described above.
4. Click “OK” to close the “Certificate Selection” dialog.
   ⇒ The certificate is no longer displayed in the “Properties” dialog.

See also
- Help for the CODESYS Security Agent add-on product
- Chapter 1.3.1.25.2.3.18 “Command ‘Security Screen’” on page 891

1.3.1.9.19 Unit conversion

You define a conversion rule when you want to convert data for another system of units. This data is executed for a specific order of magnitude and unit of measure.

Conversion rules are defined in a “Unit Conversion” object. CODESYS automatically implements each conversion rule as a function block <name>_Impl and instances it as <name>.
Each conversion rule includes Convert and Reverse methods for use as function blocks.
Locations where you access a variable, you can link the variable to a conversion rule. The input assistant provides conversion rules in the “Function Blocks” and “Instance Calls” categories.
After execution, the result is a converted value according to the conversion rule.
In a visualization, an IEC variable that is configured in an element property can also be linked to a conversion rules.

Defining unit conversions

1. Double-click a “Unit conversion” object in the device tree.
   ⇒ The respective editor opens with a table of the defined conversion rules. You edit a rule in “Type setting” and a respective condition in “Condition setting”.

246
3ADR010583, 1, en_US
2020/12/10
2. Double-click the “Add new entry” field and type a name.
   ⇒ CODESYS implements the <name>_Impl function block and instances it as <name>.
3. Double-click the “Type” field and click a type from the drop-down list.
   ⇒ Input fields are displayed below the table for editing the conversion rule. The input fields vary according to selected type.
4. Change the conversion rule in the input fields.
   ⇒ The changes are displayed in the “Setting” category of the table.
5. Double-click the “Condition” field and click a condition type from the drop-down list.
   ⇒ Input fields are displayed below the “Condition Setting” category of the table to edit the condition. The input fields vary according to selected type.
6. Edit the condition.
   ⇒ The changes are displayed in the “Condition Setting” category of the table.

**Defining switchable unit conversions**

You can define which conversion rule is applied to a specific language or condition.

1. Double-click a “Unit Conversion” object in the device tree.
   ⇒ The respective editor opens with a table of the predefined conversion rules.
2. Click the “Add new entry” field and type a name.
   ⇒ Example: Conv_A_LanguageDependent
3. Double-click the “Type” field and click “Switchable conversion”. Double-click the “Condition” field and click “Language”.
   ⇒ Below the main table, the “Switchable Conversion” table is displayed with “Condition setting”.
4. In the “Switchable Conversion” table, double-click a predefined conversion rule from the drop-down list in the “Switchable conversion name” column, for example Conv_AInInch. In “Condition Setting”, type a value in the “For condition 'Language'” input field, for example en.
   ⇒ CODESYS executes the Conv_AInInch conversion rule only if the language set in the visualization manager is “en”.
5. In the “Switchable Conversion” table, double-click a predefined conversion rule from the drop-down list in the “Switchable conversion name” column, for example Conv_AInMM. In “Condition Setting”, type a value in the “For condition 'Language'” input field, for example de.
   ⇒ CODESYS executes the Conv_AInMM conversion rule only if the language set in the visualization manager is “de”.
6. Apply the Conv_A_LanguageDependent conversion rule in the application or visualization.
   ⇒ If the set language in the visualization is English, then the application visualization applies the Conv_AInInch conversion rule. If the set language in the visualization is German, then the application visualization applies the Conv_AInMM conversion rule. The current visualization language is located in the VisuElems.CurrentLanguage variable.

**Applying conversion rules**

Add a conversion rule to objects that access IEC variables.
1. In the device tree, double-click an object that accesses IEC variables in order to link an IEC variable to a conversion rule at that location.

2. Declare a variable for the conversion result of the IEC variable.

   ST sample code: rConvertedA : REAL;

3. Use the input assistant to apply the conversion rule with the `Convert` method and then assign the result to the variable.

   ST sample code to link the IEC variable to the conversion rule: rConvertedA := ConvRule_A.Convert(rA);

Applying reverse conversion rules

1. In the device tree, double-click an object that accesses an IEC variable.

2. Declare a variable for the result of the conversion rule.

   ST sample code: rReverseA: REAL;

3. Apply the reverse conversion rule with the `Reverse` method and then assign the result to the variable.

   rReverseA := ConvRule_A.Reverse(rConvertedA);

Example

Requirement: The conversion rule is Conv_XtoY.

ST call

```st
PROGRAM A_PRG
VAR
  rA : REAL;
  rConvertedA : REAL;
  rReverseToA : REAL;
END_VAR
rConvertedA := Conv_XtoY.Convert(rA);
rReverseToA := Conv_XtoY.Reverse(rConvertedA);
```

CFC call

In the CFC editor, define the instance name of the conversion rule via the block. Select the method in the block.

See also

- Chapter 1.3.1.25.1.44 “Object ‘Unit Conversion’” on page 849
- Chapter 1.3.1.9.6 “Using input assistance” on page 208
1.3.1.20 Data persistence

The lifespan of a variable and its data begins at the time when the variable is created and ends
at the time when the variable is deleted and its memory is freed. The time when the variable is
created, initialized, or instantiated depends on the declared scope. The time when the memory
is freed usually depends on the scope as well. For example, the memory of global variables is
freed by exiting the application.

They can retain data longer than usual. The following mechanisms are provided for this pur-
pose.

Mechanisms for data retention

- (A): Persistent global variable list with the keyword PERSISTENT RETAIN
  Persistent variables retain their values when the application is reloaded. Moreover, the
  values are restored after a download, warm start, or cold start.
- (B): Retain variables with the keyword RETAIN
  Retain variables retain their values after a warm start, but not after reloading the application,
  a download, or a cold start.
- (C): Variables of the Persistence Manager of the CODESYS Application Composer
  Variables of the Persistence Manager are stored in an external file.
- (D): Recipe variables
  Recipe variables and their values are stored in a recipe file.

See also

- Chapter 1.3.1.9.20.3 “Retaining data with variables of the persistence manager” on page 255
- Chapter 1.3.1.9.20.2 “Preserving data with retain variables” on page 254
- Chapter 1.3.1.9.20.4 “Preserving data with recipes” on page 255
- Chapter 1.3.1.24.2.12 “Persistent variable - PERSISTENT” on page 498
- Chapter 1.3.1.24.2.13 “Retain variable - RETAIN” on page 500
- Persistence Manager
- Chapter 1.3.1.25.1.23 “Object ‘Persistent variable list’” on page 770

Mechanisms in comparison

Which mechanism is suitable for which application? Some common use cases are considered
in the table. The specific examples refer to a building control system.
<table>
<thead>
<tr>
<th>Uses case</th>
<th>(A) Persistent variables</th>
<th>(B) Retain variables</th>
<th>(C) Variables of the Persistence Manager</th>
<th>(D) Recipe variables</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>The application must maintain device settings. Example: After a power failure, the building control has to have information available about how long a window blind needs to be raised.</td>
<td>Suitable(^1) Preferred use case In this case, you can also use retain variables instead of persistent variables. This is advantageous for variables whose declaration is often changed.</td>
<td>Suitable Preferred use case Retain variables are an advantage when their declarations are changed often.</td>
<td>Suitable(^2) This is advantageous for controllers that do not have any hardware support. Special functionalities make this possible, such as double file buffering.</td>
</tr>
<tr>
<td>2</td>
<td>The application must maintain values also after program changes or extensions.</td>
<td>Suitable(^1) Preferred use case</td>
<td>Suitable</td>
<td>Suitable(^2) Possible, but complicated.</td>
</tr>
<tr>
<td>2a: Rare extensions Example: An application programmer extends the program with a new switch and installs a new light. The building control must still have saved values available until then.</td>
<td>Suitable</td>
<td>Suitable</td>
<td>Suitable(^2) Possible, but complicated.</td>
<td></td>
</tr>
<tr>
<td>2b: Unrestricted changes, including deleting or changing the data type of variables The building control is running and is persistent. When an application programmer adds a new functionality to the controller and therefore adds another persistent variable to a function block, the values saved up to that point must be retained. For example, the program in an FB is extended with a variable that controls the automatic switching off of a previously uncontrolled lamp after a certain time. The building control must have the times of all controlled lamps available after the extension.</td>
<td>Not suitable</td>
<td>Suitable Data from retain variables are preserved as far as possible after an online change.</td>
<td>Suitable as far as possible(^2) Preferred use case</td>
<td></td>
</tr>
<tr>
<td>2c: The application must maintain values after a download.</td>
<td>Suitable</td>
<td>Not suitable</td>
<td>Suitable</td>
<td>Suitable</td>
</tr>
</tbody>
</table>
### Uses case

<table>
<thead>
<tr>
<th>Uses case</th>
<th>(A) Persistent variables</th>
<th>(B) Retain variables</th>
<th>(C) Variables of the Persistence Manager</th>
<th>(D) Recipe variables</th>
</tr>
</thead>
<tbody>
<tr>
<td>3  The application must be able to use different value sets.</td>
<td>Not suitable</td>
<td>Not suitable</td>
<td>Not suitable</td>
<td>Suitable Preferred use case</td>
</tr>
<tr>
<td>Example: The operating settings for summer, winter, and holidays must be saved and imported when needed.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4  The application must be able to use settings from another system.</td>
<td>Not suitable</td>
<td>Not suitable</td>
<td>Suitable</td>
<td>Suitable^2</td>
</tr>
<tr>
<td>It must be possible to transfer settings to another plant using similar variables.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5  The application must provide human readable data.</td>
<td>Not suitable</td>
<td>Not suitable</td>
<td>Suitable</td>
<td>Suitable^3</td>
</tr>
<tr>
<td>The user must be able to read, compare, and edit the data.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

1 Disadvantage: Only possible if the runtime system supports this mechanism and an NVRAM memory or UPS is available. Advantage: Speed; recommended application: 1 and 2a

2 Disadvantage: In the case of large variable sets (> 10000), long delays during initialization and shutdown are to be expected. Advantage: No special memory is required; value retention exists even in case of changes, extensions, or deletions.

3 Advantage: Editable remotely, transferable. Disadvantage: Complicated

### Lifespan of variables when calling online commands

<table>
<thead>
<tr>
<th>User input in the “Online” menu</th>
<th>Variable with usual lifespan</th>
<th>RETAIN</th>
<th>PERSISTENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Command “Online Change”</td>
<td>Neither RETAIN nor PERSISTENT</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>Command “Reset warm”</td>
<td>-</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>Command “Reset cold”</td>
<td>-</td>
<td>-</td>
<td>x</td>
</tr>
<tr>
<td>Command “Download”</td>
<td>-</td>
<td>-</td>
<td>x</td>
</tr>
<tr>
<td>Command “Reset origin”</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

x : Variable retains its value
- : Variable is initialized

^1 Note: For the structure of persistent data, refer to the information in "Mechanism for downloading".

See also

- ≈ "Mechanism for downloading" on page 252
- ≈ Chapter 1.3.1.25.2.6.6 “Command ‘Online Change’” on page 928
- ≈ Chapter 1.3.1.25.2.6.11 “Command ‘Reset Origin’” on page 932
Lifespan of variables when downloading a boot project

The values of ordinary variables lose their value and are reinitialized.

The values of persistent variables are protected when:

- The structure of the persistent variable in memory matches the structure in the persistent data list.

The values of retain variables are protected when:

- The structure of the persistent variable in memory matches the structure in the persistent data list.
- The persistent variables match the application (GUID has to agree).

A "Retain mismatch" occurs when the requirements for restoring the values of retain variables and persistent variables are not met when the application is booted. The response to this discrepancy is described in the documentation of the hardware manufacturer.

Note: For the structure of persistent data, refer to the information in "Mechanism for downloading".

See also

- “Mechanism for downloading” on page 252

Preserving data with persistent variables

Persistent variables retain their values after reloading the application, and after a download, warm start, or cold start.

A special non-volatile memory area on the controller, for example as NVRAM or UPS, is required to extend the lifespan. Securing the data in such a memory does not require any additional time, which is an advantage over data retention with the Persistence Manager. If the controller does not provide hardware support, then the data is usually stored in a file. Then the data will be retained if you shut down the controller correctly. In the event of a power failure or a pulled plug, however, data will be lost.

Behavior

Value retained for

- Uncontrolled exit
- Warm start by calling the "Reset Warm" command
- Cold start by calling the "Reset Cold" command
- Repeated download of the application

Reinitialization for

- Call of the "Reset Origin" command

Therefore, persistent variables are reinitialized only if you reset the controller to the factory settings (for example, when you click “Online ➔ Reset Origin”).

If, on the other hand, you download the application again, the persisted data is retained if possible. That depends on how profound the changes that led to the download were. Changing the application name always leads to a full reinitialization. Changes to the implementations never lead to a reinitialization: the data persistence is completely preserved. Changes to the declarations lead to an initialization of the new variables only if the existing variables are persistent, when you change the declarations so that the persistent variable list remains consistent. This is the case when you add a new variable or delete an existing one. Inconsistencies can occur if you edit and change the identifiers or data types of previously declared persistent variables.

Mechanism for downloading

Editing the variable list in the persistence editor causes the variable list to be edited automatically before it is saved, not to be saved as it is shown in the editor.
During post-processing, a variable that you have removed is replaced by a placeholder variable with the same memory requirement. As a result, the subsequent variables retain their addresses in the process image. Moreover, a variable you add is moved to the end of the list. Post-processing can neutralize changes that would lead to a loss of persistence. But you create gaps that use additional memory.

When downloading, the CRC value of the variable list and the length of the list (number of variables) are stored on the controller. When downloading again, the new test value is compared with the test value currently on the controller. Then the variable list is compared successively up to the specified length. If you have edited a declaration (for example, the name or data type), then the variable is reinitialized. Otherwise its value is retained. When the download is repeated, CODESYS checks whether the variable list declared in the persistence editor is still consistent with the variable list already on the controller.

The mechanism works well when the variables themselves are not modified significantly. Too extensive changes of the identifiers and the data types continue to lead to a reinitialization and the loss of persistence. If you anticipate frequent changes due to your application requirements, then this kind of a list is not recommended. Moreover, in an online change after a data type change, a persistent variable is less robust than a variable with a normal lifespan.

It is good practice to clear any gaps in the variable list after a while (command “Reorder List and Clear Gaps”). After cleaning, however, the list no longer matches the list on the controller and you have triggered an initialization of all persistent variables. The persistence of all variables is lost.

For versions before V3.5 SP1, changes in the persistence editor always lead to reinitialization.

Recovering data with the recipe manager

To clean up the global persistent variable list without losing persistence, you can save the data in a recipe using the Recipe Manager. This creates a list for all variables of the persistent variable list in the recipe manager, and at the same time its current values are stored by the controller as a recipe. Then execute the command “Reorder List and Clear Gaps” and perform a download again. Now when you execute the command “Restore Values from Recipe”, the values saved in the recipe are restored.

Changing an existing declaration in the persistent variable list

If you change the name or data type of a variable, this is interpreted as a new declaration and causes a re-initialization of the variables at the next online change or download. For complex data types, a change occurs when a new component is added, or when you change the type of a variable from INT to UINT in the depth of a used structure used, for example.

Basically, complex user-defined data types are not suitable for administration in a persistent variable list, because even small changes cause the variable to be initialized with all components.

Double allocation of memory in the case of instance paths

You can persist global variables or variables declared locally in a function block or program. To do this, add the keyword PERSISTENT to the declaration. In addition, you insert the instance path to this variable in the persistent global variable list. To do this, execute the “Add All Instance Paths” command in the persistence editor.

Persistence is guaranteed by the following mechanism:

- The cyclic tasks in which the variable is accessed are determined.
- At the end of the first cyclic task (in each cycle), the variable is copied to the persistent global variable list.
- After restarting the controller, the value of the persistent variable is copied to the ordinary variable.
The disadvantage of this mechanism is that memory is allocated both at the place of declaration and at the place of the instance path. This persistent variable has a double memory allocation. Moreover, the data is copied to both places in each cycle. This can be time consuming, especially when large structured values are involved.

**Memory location in the case of persistent function block instances**

A function block instance is always stored completely in memory. This is necessary so that the same code can work on different instances. If only one variable in a function block is marked with PERSISTENT, then the function block instance is stored completely with all variables in remanent memory, although only the one variable is treated as persistent. However, non-volatile memory is not available to the same extent as main memory.

A function block with a pointer to an instance in SRAM as a variable is not stored in the protected memory.

**Importing from CoDeSys V2.3 projects**

When you open a CoDeSys V2.3 project to import it into CODESYS V3, the declarations of persistent variables are not preserved. You have to revise the declarations and create them again in a separate persistent global variable list.

See also

- § Chapter 1.3.1.24.2.12 “Persistent variable - PERSISTENT” on page 498
- § Chapter 1.3.1.25.2.17.4 “Command ‘Add all instance paths”’ on page 1012
- § Chapter 1.3.1.3.5 “Opening a V2.3 project” on page 128

**Preserving data with retain variables**

Retain variables preserve their values after a warm start. However, the degree of value retention for persistent variables is higher.

A special non-volatile memory area on the controller, for example as NVRAM or UPS, is required to extend the lifespan. Securing the retain variables in such a memory does not require any additional time, which is an advantage over data retention with the Persistence Manager. If the controller does not provide hardware support, then the data is usually stored in a file. Then the data will be retained if you shut down the controller correctly. In the event of a power failure or a pulled plug, however, data will be lost.

**Declaration**

To declare a retain variable, add the RETAIN keyword to a variable declaration.

**Behavior**

Value retained for

- Uncontrolled exit
- Call of the "Reset Warm" command

Reinitialization for

- Repeated download of the application
- Call of the "Reset Cold" command (in contrast to persistent variables)
- Call of the "Reset Origin" command

When you restart an application, its variables are usually initialized with an explicitly preset initial value or with a default value. Variables marked with the RETAIN keyword are managed in a separate memory area depending on the target system and retain their value. Then the variables are protected from power failure, for example. This means that you can apply retain variables to a parts counter in a production line so that you can continue counting even after a power failure.
Function block instances are stored as one block in memory. This is necessary so that the same code can work on different instances. If a variable is marked with `RETAIN` in a function block, then each instance of the function block is protected with all variables. This is also true for the variables of the function block that are not marked this way. However, non-volatile memory is not available to the same extent as main memory.

A function block with a pointer to an instance in SRAM as a variable is not stored in the protected memory.

When you open a CoDeSys V2.3 project to import it into CODESYS V3, the declarations of retain variables are preserved and remain effective as before.

See also
- Chapter 1.3.1.24.2.13 “Retain variable - RETAIN” on page 500
- Chapter 1.3.1.3.5 “Opening a V2.3 project” on page 128

Persistent variables are managed in the Persistence Manager of the CODESYS Application Composer. The functionality of the “Persistence Manager” does not need any special memory on the controller in order to preserve values and data.

In the declarations, the variables managed in the Persistence Manager are marked with the pragma `{attribute 'ac_persist'}`.

The pragma makes sure that the variable with this attribute is managed in the Persistence Manager of the Application Composer. The variable value is retained even if you change the declaration of the variable, delete a variable from the application, or add a new one. The value is retained even if you change the data type and use the appropriate conversions.

The variables of the Persistence Manager are stored with their values in an external archive file in TXT format.

The application code is extended with the code of the Persistence Manager, which leads to a greater memory requirement. This is at the expense of performance. Moreover, reading and especially writing a large number of persistent variables can take a long time. As a result, the executing task also blocks the execution for a long time.

- You can load and edit the TXT file in an external editor such as Notepad++.
- You can use the persistent variables of the file in another application.
- You can configure the behavior of persistent variables by defining persistence groups, assigning variables to them, and configuring the groups with their own save and read behavior.

See also
- Persistence Manager

Variables are managed persistently in the Recipe Manager. The Recipe Manager does not need any special memory on the controller in order to preserve values and data.
Declaration

A recipe definition consists of a set of variables with values and is created and edited in the "Recipe Manager" object and saved to a file.

Functionality

- You can include a variable in multiple recipes, each with different values.
- In online mode, you can read in the actual values of the variables from the controller and save them as recipe values (specified value).
- You can use the Recipe Management library to programmatically implement the creation and editing of a recipe.
- You can save and backup a recipe as a recipe file.

See also

- § Chapter 1.3.1.13.2 “Changing values with recipes” on page 331
- § Chapter 1.3.1.25.2.17.2 “Command ‘Save Current Values to Recipe’” on page 1011
- § Chapter 1.3.1.25.2.17.1 “Command ‘Reorder List and Clean Gaps’” on page 1011
- § Chapter 1.3.1.25.2.17.3 “Command ‘Restore Values from Recipe’” on page 1011
- § Chapter 1.3.1.25.1.33 “Object ‘Recipe Manager’” on page 820
- § Chapter 1.3.1.25.1.34 “Object ‘Recipe Definition’” on page 823

Declaring VAR PERSISTENT variables

Below you will declare persistent variables in a persistent variable list and in a POU.

Requirement: A project is opened and contains a program POU. You have enabled the option for the textual view in the "Declaration Editor" category of the options (menu command “Tools ➔ Options”).

1. Add the “Persistent Variables” object to the application object with the menu command “Project ➔ Add Object”.

   ⇒ CODESYS adds the persistent variable list “PersistentVars” below the application object in the device tree and the editor opens.

2. In the editor, enter a variable declaration, for example

   ivarpersist1 : INT; between VAR_GLOBAL PERSISTENT RETAIN and END_VAR.

3. Double-click the POU in the device tree.

   ⇒ The editor of the POU opens.

4. Specify the following declaration in the declaration part:

   VAR PERSISTENT
   ivarpersist2 : INT;
   END_VAR

5. Click “Build ➔ Build”.

   ⇒ The message view opens. If CODESYS has compiled the application without errors, then close the message window and continue with the next step. Otherwise, correct the error(s) and select the menu command “Build ➔ Build” again.

6. Click the tab “PersistentVars” to select the persistent variable list as the active editor and select the menu command “Declarations ➔ Add All Instance Paths”.

   ⇒ CODESYS adds the persistent variable from the persistent variable list “PersistentVars” to the POU:

   // instance path of the persistent variables created
   POU.IVARPERSIST2 : INT
Saving the values of a persistent variable list in a recipe

Requirement: a project is opened and a persistent variable list with declarations of persistent variables exists under an application object.

1. Double-click on the controller in the device tree and select the tab “Communication Settings”.
2. Select your gateway and click on the button “Scan Network”.
   ⇒ Your device is shown in bold in the tree view of the gateway.
3. Select your device and click on the button “Set Active Path”.
4. Select your application object in the device tree and select the context menu command “Set Active Application”.
   ⇒ The application object is displayed in bold.
5. Select the menu command “Online ➔ Login”.
   ⇒ Your application is logged in to the controller and the controller and the application object in the device tree have a green background.
6. Double-click on the persistent variable list and select the command “Declarations ➔ Save Current Values to Recipe”.
   ⇒ CODESYS creates the objects “Recipe Manager” and “PersistentVariables” under the application object.
7. Select the menu command “Online ➔ Logout”.
   ⇒ The application is logged out from the controller.

See also
- Chapter 1.3.1.24.2.12 “Persistent variable - PERSISTENT” on page 498
- Chapter 1.3.1.24.2.13 “Retain variable - RETAIN” on page 500
- Chapter 1.3.1.25.1.23 “Object ‘Persistent variable list’” on page 770
- Chapter 1.3.1.25.2.17.4 “Command ‘Add all instance paths’” on page 1012

1.3.1.9.21 Alarm management

You can use the alarm management as part of an application in order to determine and record critical process states and to display them with the help of visualization elements. You can take notice of alarms via the application or user inputs in the visualization and react accordingly with acknowledgment and subsequent actions. CODESYS provides special visualization elements and a library for this.

An alarm management consists of the following elements

- **Alarm class**: an alarm class serves the typifying of alarms, i.e. it defines a basic parameter set for an alarm. You must specify the class when creating an alarm. For example, it defines how an alarm must be confirmed, whether it is to be recorded and displayed, or whether it is to be written to a file.

- **Alarm group**: an alarm group is for the concrete configuration of one or more alarms. In the alarm group you define the conditions for the individual alarms and assign the alarms to the alarm classes. Furthermore you define there whether the alarms of this group are archived.

- **Alarm storage**: an alarm storage is for defining the parameters for the archiving of the alarms that occur.
The visualization of CODESYS offers the elements “Alarm Table” and “Alarm Banner” for displaying the alarm messages.

So that you have access to the alarms from the application, the library AlarmManager.library offers certain functions and interfaces.

See also
- Chapter 1.3.1.25.1.6 “Object ‘Alarm Class’” on page 715
- Chapter 1.3.1.25.1.7 “Object ‘Alarm Group’” on page 718
- Chapter 1.3.1.25.1.8 “Object ‘Alarm Storage’” on page 721
- Chapter 1.3.1.9.21.2 “Configuring the alarm management” on page 258
- Visualization Element ‘Alarm table’ Chapter 1.3.5.18.1.22 “Visualization element ‘Alarm table’” on page 1395
- Visualization Element ‘Alarm banner’ Chapter 1.3.5.18.1.23 “Visualization element ‘Alarm banner’” on page 1402

Defining alarms

For alarm management in CODESYS the following definitions apply to an alarm. An alarm definition consists in principle of
- a general description (ID, message text, etc.)
- a description of the cause of the alarm (expression to be monitored, limits, min. pending time, etc.)
- a description of the effects of the alarm (actions, display properties, confirmation method etc.).

Acknowledgement of alarms: the main reason for the use of alarms is to inform the user about an alarm situation. In order to ensure that the user has taken notice of the information, CODESYS provides you with various actions that you define in the alarm class. The user must acknowledge the alarm in order to delete it from the alarm list. The special types of alarm acknowledgment are described in the status diagrams as status transitions.

Alarm event: an alarm event is linked to an alarm condition. While an alarm condition can exist over a lengthy period, the alarm event describes the momentary occurrence of a change, for example the change from the normal state to the alarm state. In the CODESYS alarm configuration the same names (Active, ACK, Inactive) are used for the three types of events and the associated alarm status.

The following features are supported in CODESYS:
- Deactivation of the alarm generation for simple alarms as well as for alarm groups
- Selection of the alarms that are to be displayed, by definition of alarm groups and priorities
- Display of all alarm events in an alarm table
- Visualization elements “Alarm Table” and “Alarm Banner” in the CODESYS visualization

Configuring the alarm management

Creating alarm configurations

1. Select an application in the device tree.
2. Click “Project ➔ Add Object ➔ Alarm configuration”.
   ➔ The “Add Alarm Configuration” dialog box opens.
3. Click “Add”.
   ⇒ CODESYS adds the “Alarm Configuration” to the device tree below the application.
   In addition, CODESYS inserts the predefined alarm classes (“Error”, “Info”, and “Warning”) and the alarm storage “AlarmStorage” below the alarm configuration. The predefined objects are available to you if you need them.

### Configuring alarm classes

1. Select the “Alarm Configuration” object in the device tree.
2. Click “Project ➔ Add Object ➔ Alarm Class”.
   Type in a name (example: "PartsDeficit") and click “Add”
   ⇒ The “PartsDeficit” alarm class is added to the device tree. The object opens in the editor.
3. Set the following parameters for the class. These apply to all alarms that belong to this class group:
   - “Priority”: 10
   - “Archiving”
   - “Acknowledgment method”: REP_ACK
   - “acknowledge separately”:

4. In the following, you define that the bPartsDeficit variable has the value TRUE when alarm messages occur: Double-click the “Action” column in the “Notification actions” group.
   Select the “Variable” action.
   Press the [Enter] key.
   ⇒ The action is added and all options are activated.
5. Select the “Activate” check box, and clear the “Deactivate” and “Confirm” check boxes.
6. Click inside the input field “Variable” and enter bPartsDeficit or select the variable with the help of the input assistant
   Enter the value TRUE into the right field.
   ⇒ The ...bPartsDeficit := TRUE statement is displayed in the “Details” column.
7. Select the “Background color” "Red" in the “Active” row of the “Display Options” group.
   Select the “Background color” "Yellow" in the “Wait for confirmation” row in the “Presentation Options” group.

All messages in the “PartsDeficit” class set the variable bPartsDeficit to TRUE. Active messages are highlighted in red in the "Alarm table" visualization element. Deactivated messages that are not acknowledged are highlighted in yellow.

See also
- Chapter 1.3.1.25.1.6 “Object ‘Alarm Class’” on page 715
- Visualization Element Alarm table ➔ Chapter 1.3.5.18.1.22 “Visualization element ‘Alarm table’” on page 1395
- Visualization Element Alarm banner ➔ Chapter 1.3.5.18.1.23 “Visualization element ‘Alarm banner’” on page 1402

### Configuring alarm groups

In the following, you define an alarm group with an alarm for monitoring PartsDeficit. You have already created the applicable alarm class above.
1. Select the “Alarm Configuration” object in the device tree.

2. Click “Project ➔ Add Object ➔ Alarm Group”.
   Type in the name “PartsDeficit” and click “Add”
   ⇒ The “PartsDeficit” alarm group is added to the device tree. The object opens in the editor.
   CODESYS creates the text list “PartsDeficit”.

3. Set the following parameters:
   ● “Archiving”: (none)

4. Double-click in the “Observation type” field.
   Select the “Digital” monitoring type.

5. Configure the alarm as follows:
   ● “Class”: PartsDeficit
   ● “Message”: PartsDeficit at Station 1 - level: <LATCH1> ("<LATCH1>" is a placeholder for latch variable 1.)
   ● “Min. pend. time”: t#5s
   ● “Latch var 1”: ...iFuellstand

6. In the “Digital” group, click the “Expression” field and type in bPart1Empty or select the variable by using the input assistant.
   Type in the value TRUE in the right field.
   ⇒ The ...bTeil1Leer = TRUE statement is displayed in the “Details” column.

If the bTeil1Leer variable switches to TRUE, then CODESYS triggers a message. The value of the iFuellstand variable is printed with the message. The acknowledgment response and the display are executed according to the “PartsDeficit” alarm class. The alarm is not archived.

See also
●  Chapter 1.3.1.25.1.7 “Object ‘Alarm Group’” on page 718

Configuring alarm storage

1. Select the “Alarm Storage” object in the device tree.

2. To change the name, select the “Properties” command in the context menu.
   Change the name from "AlarmStorage" (example: "DatabaseFileAlarmStorage") and click “OK”.

3. Double-click the “DatabaseFileAlarmStorage” object in the device tree to open the editor.

4. Configure as follows:
   ● “Subdirectory”: Leave this field blank. Specifying a directory is optional only on the PLC where the alarm storage file will be saved.
   ● “Limit” group: Select the “No limit” check box.
   ⇒ The alarm storage is configured. In this case, CODESYS saves the alarms internally in an unlimited buffer.

See also
●  Chapter 1.3.1.25.1.8 “Object ‘Alarm Storage’” on page 721
Calling event alarms in the program

You define the alarms in the object Alarm Group. The monitoring type “Event” is thereby available. Contrary to the other monitoring types, which trigger the alarm depending on a condition, you can trigger the alarm with the type “Event” via a function block call.

Defining the event alarm in the alarm group

1. Create an alarm group.
2. Define an alarm of the monitoring type “Event” in the alarm group.
3. In the column “Class”, select an alarm class with the acknowledgement method “ACK” or create a new alarm class.

See also
● “Configuring alarm classes” on page 259

Program call for triggering the event alarm

1. Set the cursor at the program position at which the function-block call is to take place.
2. Use the RaiseEvent function from the AlarmManager library:
   Enter 'AlarmManager', followed by a period.
   ⇒ A window opens containing a list of all insertable elements (function 'List components').
3. Select AlarmGlobals and enter a period.
4. In the same way, insert the elements g_AlarmHandler and RaiseEvent.
   ⇒ You get the following line:
   Alarmmanager.AlarmGlobals.g_AlarmHandler.RaiseEvent.
5. Now define the transfer variables AlarmGroup_ID and “Alarm_ID”. To do this, enter an opening parenthesis.
   ⇒ A tooltip appears, showing information about the transfer variables
6. Press [F2]
   ⇒ The input assistant opens
7. On the “Categories” tab (3), select the category “Variables” (4).

8. Select the desired group ID variable (1).
   ➞ The variable is applied into the program line.

9. Enter a comma and insert the desired alarm ID variable (2) in the same way as in step 7.
   Input a closing parenthesis and a semicolon.
   ➞ You get the following program line:
     
     ```
     Alarmmanager.AlarmGlobals.g_AlarmHandler.RaiseEvent(Alm_AlarmConfiguration.Alarmgroup_IDs.ID_AlarmGroup1,
     Alm_AlarmGroup1_Alarm_IDs.ID_0);
     ```

See also

● “List components” on page 209

1.3.1.9.22 Using POUs for implicit checks

CODESYS provides special POUs that implement implicit monitoring functions. At runtime, these functions check the array limits or subrange types, the validity of pointer addresses, or division by zero.

1. Select the “Application” object in the device tree.
   Click “Project ➔ Add Object ➔ POU for Implicit Checks”
   ➞ The “Add POU for Implicit Checks” dialog box opens.

2. Select the desired functions.

3. Click “Add”.
   ➞ The selected POUs are inserted below the “Application” in the device tree.

4. Open the POUs in the editor.
5. Adapt the implementation suggestion to your requirements.

CAUTION!
To obtain the feature for monitoring functions, do not edit the declaration section. However, you are permitted to add local variables.

See also
● Chapter 1.3.1.25.1.28 “Object ‘POU’” on page 775
● Chapter 1.3.1.25.1.29 “Object ‘POUs for Implicit Checks’” on page 796

1.3.1.9.23 Object-oriented programming
CODESYS supports object oriented programming with function blocks and for this purpose provides the following features and objects:

● Methods
● Interfaces
● Properties
● Inheritance
● Method call, virtual function call
● Definition of function blocks as extensions of other function blocks

See also
● Chapter 1.3.1.25.1.28.4 “Object ‘Interface’” on page 783

Extension of function blocks
The extension of a function block is based on the concept of inheritance in object-oriented programming. A derived function block thereby extends a basic function block and in doing so is given the properties of the basic function block in addition to its own properties.

The extension of a function block means:

● The inherited function block contains all data and methods that are defined by the basic function block. You can use an instance of the basic function block in every context in which CODESYS expects a function block of the type of the basic function block.

● The derived function block can overwrite the methods that you have defined in the base function block. This means that the inherited function block can define a method with the same name, the same inputs and the same output as is defined by the basic function block. Tip: You have the following support when overwriting methods, actions, attributes, and transitions that are inherited by the base block: When you insert a method, action, etc. below an inherited block, the “Add Object” dialog includes a combo box with a list of methods, actions, etc. used in the base block. You can accept these and adapt them accordingly.

● The derived function block may not contain function block variables with the same names as used by the basic function block. The compiler reports this as an error. The only exception: If you have declared a variable in the basic function block as VAR_TEMP, then the inherited function block may define a variable with the same name. In this case, the inherited function block can no longer access the variable of the basic function block.

● You can directly address the variables and methods of the basic function block within the scope of the inherited function block by using the SUPER pointer.
NOTICE!
Multiple inheritance is not permitted.
Exception: A function block can implement multiple interfaces, and an interface can extend other interfaces.

Extension of a basic function block by a new function block

Requirement: the currently opened project possesses a basic function block, for example “POU_1(FB)”, which is to be extended by a new function block.

1. Right-click the “Application” object in the device tree and select “Project ➔ Add Object ➔ POU”.
   ⇒ The “Add POU” dialog opens.
2. Type the name for the new POU in the “Name” input field, for example “POU_Ex”.
3. Select “Function block”.
4. Click “Advanced” and then the more button (≡).
5. In the category “Function blocks” under “Application” in the input assistant, select the POU(FB) that is to serve as the basic function block, for example POU_1, and click “OK”.
6. As an option, you can select an “Access modifier” for the new function block from the drop-down list.
7. Select from the “Implementation language” combo box (example: “Structured text (ST)”).
8. Click “Add”.
   ⇒ CODESYS adds the POU_Ex function block to the device tree and opens the editor.
   The first line contains the text:
   ```
   FUNCTION_BLOCK POU_Ex EXTENDS POU_1
   ```
   The function block POU_Ex extends the basic function block POU_1.

Extension of a basic function block by an existing function block

Requirement: The open project possesses a base function block (example: POU_1(FB)) and another function block (example: POU_Ex(FB)). The function block POU_Ex(FB) is also to be given the properties of the basic function block. This means that POU_Ex(FB) should extend POU_1(FB).

1. Double-click the function block POU_Ex(FB) in the device tree.
   ⇒ The function block editor opens.
2. Extend the existing entry in the top line FUNCTION_BLOCK POU_Ex with EXTENDS POU_1.
   ⇒ The function block POU_Ex extends the basic function block POU_1.

See also
- § Chapter 1.3.1.9.23.2 “Implementing interfaces” on page 265
- § Chapter 1.3.1.9.23.3 “Extending interfaces” on page 267
- § Chapter 1.3.1.24.2.14 “SUPER” on page 501
- § Chapter 1.3.1.24.2.15 “THIS” on page 502
- § Chapter 1.3.1.25.1.28.2 “Object ‘Function Block’” on page 778
- § Chapter 1.3.1.25.1.28.8 “Object ‘Property’” on page 791
- § Chapter 1.3.1.25.1.28.9 “Object ‘Action’” on page 794
- § Chapter 1.3.1.25.1.28.10 “Object ‘Transition’” on page 796
Implementing interfaces

Implementing interfaces is based on the concept of object-oriented programming. With common interfaces, you can use different but similar function blocks the same way.

A function block that implements an interface has to include all methods and attributes that are defined in that interface (interface methods and interface attributes). This means that the name and the inputs and outputs of the methods or attributes must be exactly the same. When you create a new function block that implements an interface, CODESYS adds all methods and attributes of the interface automatically to the tree below the new function block.

**NOTICE!**
If you add more interface methods afterwards, then CODESYS does not add these methods automatically to the affected function block. To perform this update, you must execute the "Implement Interfaces" command explicitly.

For inherited function blocks, you have to make sure that any methods or attributes that were derived through the inheritance of an interface also receive the appropriate implementation. Otherwise they should be deleted in case the implementation that was provided in the basis should be used. Respective compile error messages or warnings are displayed, prompted automatically by added pragma attributes. For more information, refer to the help page for the "Implementing Interfaces" command.

**NOTICE!**

- You must assign the interface of a function block to a variable of the interface type before a method can be called via the variable.
- A variable of the interface type is always a reference of the assigned function block instance.

A variable of the interface type is a reference to instances of function blocks. This kind of variable can refer to every function block that implements the interface. If there is no assignment to a variable, then the variable in online mode contains the value 0.
The interface contains the `GetName` method.

```plaintext
METHOD GetName : STRING
```

The functions blocks A and B implements the interface I1:

```plaintext
FUNCTION_BLOCK A IMPLEMENTS I1
FUNCTION_BLOCK B IMPLEMENTS I1
```

For this reason, both function blocks must include a method named `GetName` and the return type `STRING`. Otherwise the compiler reports an error.

A function includes the declaration of a variable of interface I1 type.

```plaintext
FUNCTION DeliverName : STRING
VAR_INPUT
  l_i : I1;
END_VAR
```

Function blocks that implement the I1 interface can be assigned to these input variables.

Examples of function calls:

```plaintext
DeliverName(l_i := A_instance); // call with instance of type A
DeliverName(l_i := B_instance); // call with instance of type B
```

Calling of interface methods:

In this case, it depends on the actual type of `l_i` whether the application calls `A.GetName` or `B.GetName`.

```plaintext
DeliverName := l_i.GetName();
```

---

### Implementing an interface in a new function block

- Requirement: The open project has at least one interface object.

1. Right-click “Application” in the device tree and select “Project ➔ Add Object ➔ POU”.
   - The “Add POU” dialog box opens.
2. Type the name for the new POU in the “Name” input field, for example “POU_Im”.
3. Select “Function block”.
4. Click “Implemented” and then the more button ().
5. In the input assistant, select the interface from the category “Interfaces”, for example ITF1, and click on “OK”.
6. To insert more interfaces, click ( ) and select a another interface.
7. As an option, you can select an “Access modifier” for the new function block from the selection list.
8. Select from the “Implementation language” combo box (example: “Structured text (ST)”).
9. Click “Add”.
   - CODESYS adds the “POU_Ex” function block to the device tree and opens the editor.
   - The first line contains the text:
     ```plaintext
     FUNCTION_BLOCK POU_Im IMPLEMENTS ITF1
     ```
   - The interface and its methods and properties are now inserted below the function block in the device tree. Now you can type program code into the implementation part of the interface and its methods.

---

Examples

- Chapter 1.3.1.25.2.23.2 “Command ‘Implement Interfaces’” on page 1038
Implementing an interface in an existing function block

- Requirement: The currently open project has a function block (example: “POU_Im”) and at least one interface object (example: “ITF1”).

1. Double-click the “POU_Ex(FB)” POU in the device tree.
   - The POU editor opens.
2. Extend the existing entry in the uppermost line `FUNCTION_BLOCK POU_Im with IMPLEMENTS ITF1`.
   - The “POU_Im” function block implements the “ITF1” interface.

See also
- “Chapter 1.3.1.25.1.28.2 “Object ‘Function Block’” on page 778

Extending interfaces

You can extend interfaces just like function blocks. The interface is then also given the interface methods and interface properties of the basic interface in addition to its own.

1. Select the object “Application” in the device tree.
2. Select the command “Project ➔ Add Object ➔ Interface”.
   - The dialog box “Add Interface” opens.
3. Enter a name for the new interface.
4. Activate the option “Extended” and click on the button 
5. The input assistant opens.
6. From the category “Interfaces”, select the interface that is to be extended by the new interface.

- “Chapter 1.3.1.25.1.28.4 “Object ‘Interface’” on page 783

Calling methods

To implement a method call, the actual parameters (arguments) are passed to the interface variables. As an alternative, the parameter names can be omitted.

Depending on the declared access modifier, a method can be called only within its own namespace (INTERNAL), only within its own programming module and its derivatives (PROTECTED), or only within its own programming module (PRIVATE). For PUBLIC, the method can be called from anywhere.

Within the implementation, a method can call itself recursively, either directly by means of the THIS pointer, or by means of a local variable for the assigned function block.

Method call as a virtual function call

Virtual function calls can occur due to inheritance.

Virtual function calls enable one and the same call to call various methods in a program source code during the runtime.
In the following cases the method call is dynamically bound:

- You call a method via a pointer to a function block (for example `pfub^.method`).
  In this situation the pointer can point to instances of the type of the function block and to instances of all derived function blocks.
- You call the method of an interface variable (for example `interface1.method`).
  The interface can refer to all instances of function blocks that implement this interface.
- A method calls another method of the same function block. In this case the method can also call the method of a derived function block with the same name.
- The call of a method takes place by means of a reference to a function block. In this situation the reference can point to instances of the type of the function block and to instances of all derived function blocks.
- You assign `VAR_IN_OUT` variables of a basic function block type to an instance of a derived FB type.
  In this situation the variable can point to instances of the type of the function block and to instances of all derived function blocks.

**Example**

<table>
<thead>
<tr>
<th>Overloading methods</th>
<th>The function blocks <code>fub1</code> and <code>fub2</code> extend the function block <code>fubbase</code> and implement the interface <code>interface1</code>. The methods <code>method1</code> and <code>method2</code> exist.</th>
</tr>
</thead>
</table>
| PROGRAM PLC_PRG     | **VAR_INPUT**  
|                     | `b : BOOL;`  
| END_VAR             | **VAR**  
|                     | `pInst : POINTER TO fubbase;`  
|                     | `instBase : fubbase;`  
|                     | `inst1 : fub1;`  
|                     | `inst2 : fub2;`  
|                     | `instRef : REFERENCE to fubbase;`  
| END_VAR             |
| IF `b` THEN         | `instRef REF= inst1;`  
|                     | (* reference to fub1 *)  
|                     | `pInst := ADR(instBase);`  
| ELSE                | `instRef REF= inst2;`  
|                     | (* reference to fub2 *)  
|                     | `pInst := ADR(inst1);`  
| END IF              |
| `pInst^.method1();` | (* If `b` is TRUE, `fubbase.method1` will be called, otherwise `fub1.method1` is called *)  
| `instRef.method1();` | (* If `b` is TRUE, `fub1.method1` will be called, otherwise `fub2.method1` is called*)  

On the assumption that `fubbase` in the above example contains two methods `method1` and `method2`, it overwrites `fub1.method2`, but not `method1`. The call of `method1` takes place as follows:

- `pInst^.method1();`
- If `b` is TRUE, then CODESYS calls `fubbase.method1`. If not, then `fub1.method1` is called.

**Additional outputs**

In accordance with the IEC 61131-3 standard, methods can have additional outputs declared, like normal functions. With the method call, you assign variables to the additional outputs.

Detailed information about this can be found in the topic “Function”.

---

3ADR010583, 1, en_US
Syntax for the call:

\[
\text{<function block name>.<method name>(<first input name> := <value> (, <further input assignments>)+, <first output name> => <first output variable name> (,<further output assignments>)+ )};
\]

Example

Declaration

\[
\text{METHOD PUBLIC DoIt : BOOL}
\text{VAR_INPUT}
\quad \text{iInput_1 : DWORD;}
\quad \text{iInput_2 : DWORD;}
\text{END_VAR}
\text{VAR_OUTPUT}
\quad \text{iOutput_1 : INT;}
\quad \text{sOutput_2 : STRING;}
\text{ENDVAR}
\]

Call

\[
\text{fbInstance.DoIt(iInput_1 := 1, iInput_2 := 2, iOutput_1 => iLocal_1, sOutput_2 => sLocal_2);}
\]

When the method is called, the values of the method outputs are written to the locally declared output variables.

Calling a method even if the application is in the STOP state

In the device description it is possible to define that a certain function block instance (of a library function block) always calls a certain method in each task cycle. If the method contains the input parameters of the following example, CODESYS processes the method even if the active application is presently in the STOP state:

Example

\[
\text{VAR_INPUT}
\quad \text{pTaskInfo : POINTER TO DWORD;}
\quad \text{pApplicationInfo: POINTER TO _IMPLICIT_APPLICATION_INFO;}
\text{END_VAR}
\]

\[
\text{(*Now the status of the application can be queried via}
\text{pApplicationInfo and the instructions can be implemented: *)}
\text{IF pApplicationInfo^.state = RUNNING THEN <instructions> END_IF;}
\]

Calling methods recursively

Use recursions mainly for processing recursive data types such as linked lists. Generally, we recommend that you be careful when using recursion. An unexpectedly deep recursion can lead to stack overflow and therefore to machine downtime.

Within their implementation, a method can call itself:

- Directly by means of the THIS pointer
- Indirectly by means of a local function block instance of the basic function block

Usually, a compiler warning is issued for such a recursive call. If the method is provided with the pragma \{attribute 'estimated-stack-usage' := '<estimated_stack_size_in_bytes>'\}, then the compiler warning is suppressed. For an implementation example, refer to the section "Attribute 'estimated-stack-usage". 
1.3.1.10 Working with control networks

With the following functionalities, CODESYS supports communication between controllers (PLC) and the insertion of a safety device below a PLC:

- **Symbol configuration**: CODESYS creates symbols with certain access rights for the variables in an application. With these symbols, you can access the variables from outside, for example from an OPC server.

- **Data source manager**: Manages the connection settings and the data transmission to remote devices (data sources). The transmitted data is mapped in data source variables that are accessed in the visualization or local application. An example of this is a control panel that controls remote devices and visualizes the state of the device as an HMI application.

- **Network variables**: Network variables are variables whose values are accessible to different controllers in the network. The variables must be defined in rigid, identical lists in both the transmitter device and the receiver device. These lists are assigned to applications, but can be located in different projects.

- **A safety device** can be inserted below a PLC in the device tree. The communications links of the safety devices to the field devices, controller networks, and development system are routed via this controller.

1.3.1.10.1 Network and addressing

Constructing a control network hierarchically, so that extensive self-configuration is possible. In CODESYS the network topology is mapped to clear addresses and the routing algorithm is kept simple by structured addresses. There is direct and relative addressing and automatic address determination during the bootup of the system.

Network topology

It is recommended to set up a network system so that the following are possible:

- Extensive self-configuration (address assignment)
- Transparent support for every communication medium
- Transport of data packets between different networks
The routing mechanism should be so simple that each network node can reroute data packets, even if it has a low memory capacity. Therefore, avoid extensive routing tables, complex calculations or queries at runtime.

Construct the control network hierarchically. Each node may possess a parent node and any number of child nodes. A node without a parent is a "top level" node. Cycles are not permitted, i.e. each control network has the structure of a tree.

Parent-child relationships result from the specification of certain network areas. A network area can be, for example, a local Ethernet or a serial point-to-point connection. We differentiate between the main network (mainnet) and the subnetworks (subnet). Each node belongs at the most to one main network, to which its parent node, if one exists, also belongs. For each node any desired number of subnets can be configured, for which the node acts in each case as a parent.

A network area may have only one parent node. Therefore, a configuration in which a network area is defined at the same time as a subnet of several nodes is invalid.

See also
- Chapter 1.3.1.10.1.2 “Addressing and routing” on page 271
- Chapter 1.3.1.10.1.3 “Address structures” on page 272

### Addressing and routing

Addressing means: the topology of the control network is mapped to unique addresses.

A node address is composed hierarchically: for each network connection the associated block driver determines a local address, which uniquely identifies the node within the local network. The complete node address is formed as follows: The local address is placed in front of the subnet index of the local network assigned by the parent. In turn, the subnet index is placed in front of the node address of the parent. The length of the subnet index (in bits) is thereby determined by the device. The length of the local address, conversely, is determined by the type of network. A node without a main network is a top level node with address 0. A node with a main network that contains no parent is likewise a top level node. It is given the local address of the main network.

See an example of a control network here:
In the example the addresses of the nodes are represented in hexadecimal notation. The first 4 digits represent the address of the respective parent in the main network, for example 0x007A=122 for PLC1. The next byte (in blue lettering) is reserved for the subnet index and is followed by the local address, for example C=12 for node ID 12. The structuring of the addresses makes a lean routing algorithm possible. Routing tables, for example, are thus unnecessary. Information is queried only locally: via its own address and via the address of the parent node. On this basis a node can correctly process the data packets:

- If the destination address corresponds to the address of the current node, then this is meant to be the receiver.
- If the destination address starts with the address of the current node, then the data packet is either meant directly for a child or for a descendant of the node and must be forwarded.
- In all other cases the receiver is not a descendant of the current node and the data packet must be forwarded to its own parent.

Relative addressing is a special case: relative addresses do not contain the node number of the receiver, but directly describe the path from the sender to the receiver. The principle is similar to the relative path in the file system: the address consists of the number of steps via which the packet must be transported upwards. These are the steps to the corresponding parent and from the subsequent path downwards to the destination node.

The advantage of relative addressing is that two nodes in the same subtree can continue to communicate if the complete subtree is shifted to another place in the entire network. Whereas the absolute node addressing has to be modified due to this shift, the relative addressing is still valid.

Address determination

For a node to know its own address it must either know the address of its parent node or know that it is a top level node. For this purpose the node dispatches a message during the bootup to all network devices for address determination. As long as it receives no response to this message, the node considers itself to be a top level node, but continues to search for a possible parent. A parent node responds by announcing its address. The node will thus independently complete its address and will announce it to the subnets. An address determination can be accomplished during the bootup or at the request of the PC used for programming.

See also

- Chapter 1.3.1.10.1.1 “Network topology” on page 270
- Chapter 1.3.1.10.1.3 “Address structures” on page 272
**Example of network address coding**

Length: 11 bit

Address: 111 1000 1100

<table>
<thead>
<tr>
<th>Byte</th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit</td>
<td>7</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Reserved (0)

**Node addresses**

The node address indicates the absolute address of a node within a control network and is therefore unique within the whole "network tree". The address is composed of up to 15 address components, each of which occupies 2 bytes. The lower a node is located within the network hierarchy, the longer its address.

The complete node address consists of the partial addresses of all preceding nodes and the partial address of the node itself. Each partial address consists of one or more address components. The length is therefore always a multiple of 2 bytes. The partial address of a node is formed from the network address of the node in its main network and the subnet index of the main network in the case of the parent node. The bits required for the subindex are determined by the router of the parent node. Filler bits can be inserted between the subnet index and the network address in order to ensure that the length of the partial address is a multiple of 2 bytes.

**Special cases:**

- A node without a main network: this means that there is neither a subnet index nor a network address in the main network. In this case the address is set to 0x0000.
- A node in the main network without a parent: In this case a subnet index with the length 0 is assumed. The partial address corresponds to the network address, if necessary extended by filler bits.

---

**Example of node addresses**

**Main network**

- Network address: 0x274 (12 bits)
- Length of subnet index: 4 bits

**Partial address of B:**

- 0x1000010C:
  - 0\_0\_0\_1\_0\_0\_0\_0\_0\_0\_1\_0\_0\_0\_1\_1\_0

- Node address of B: 0274 : 1000 : 010C
The node address is always specified in hexadecimal. The individual address components (two bytes in each case) are separated by a colon ":". Since this represents an array of bytes and not a 16-bit value, the components are not displayed in the Intel format. For manually input addresses, missing parts in an address component are supplemented by leading zeros: "274" = "0274". In order to improve the legibility, the display should also always contain the leading zeros.

**Absolute and relative addresses**

Communication between two nodes can be based on relative or absolute addresses. Absolute addresses are identical to node addresses. Relative addresses specify a path from the sender to the receiver. They consist of an address offset and the descending path to the receiver.

The (negative) address offset describes the number of address components by which a packet must be passed upwards in the tree before it can be passed back down by the common parent node. Since nodes can use partial addresses that consist of more than one component, the number of parent components to be passed is always equal to the address offset. This means that the demarcation between the parent nodes is no longer clear. For that reason the common start of the address of the communication partners is used as the parent address. Each address component is counted as an upward step, independent of the current parent node. Each error resulting from this assumption can be detected by the corresponding parent node and must be handled by it accordingly.

After achieving the common parent node the relative path, as an array of address components, is followed downwards as usual. Formal: the node address of the receiver is formed by removing the last address offset components from the node address of the sender and by appending the relative path to the remaining address.

**Example of the formation of node addresses**

In the following example each address component is represented by a letter; in each case a dot separates nodes from each other. Since a node can carry several address components, there are some in the example that are represented with several letters.

Node A: a.bc.d.ef.g

Node B: a.bc.i.j.kl.m

- Address of the lowest common parent: a.bc
- Relative address from A to B: -4/i.j.kl.m (The number 4 results from the 4 components, d, e, f and g, which must pass on the data packet in the upward direction)

In order to guarantee correct operation of the routing, the relative address must be adapted each time it passes an intermediate node. It is sufficient to adapt the address offset. This is always done by the parent node: If a node receives a data packet from one of its subnets, the address offset is increased by the length of the address component of this subnet.

- If the new address offset is < 0, then the data packet must be passed further upward.
- If the address offset is >= 0, then the data packet must be passed on to the child node whose local address corresponds to the relative path, starting from the address offset. First of all, however, the address offset must be increased by the length of the local address of the child node, so that the child node sees the correct address.

A special situation results if the error mentioned above occurs during the determination of the common parent node. In this case the address offset of the actual parent node is negative, but this value is larger than the length of the partial address of the subnet from which the packet originated. So that the next node sees a correct relative address in this case, the node concerned must do the following: it must discover the error, calculate the local address of the child node on the basis of the address of the predecessor node and the length difference, and adapt the address offset accordingly. In this case, too, the address components as such remain unchanged; only the offset is changed.

**Broadcast addresses**

There are two types of broadcast - global and local. A global broadcast is sent to all the nodes in a network. The empty node address with a length of 0 is reserved for this purpose.

Local broadcasts are sent to all the devices in a network area. For this purpose, all the bits of the network address are set to 1. This is permissible both in relative and in absolute addresses.
A block driver must be able to process both kinds of broadcast addresses. This means: empty network addresses as well as network addresses whose bits were all set to 1 must be interpreted and sent to all devices concerned.

### 1.3.1.10.2 Symbol configuration

Use the symbol configuration for preparing symbols with specific access rights for project variables. With these symbols, you can access the variables from outside, for example from an OPC server. When generating code, CODESYS also generates a symbol file (*.xml) that includes the description of the symbols.

The symbol file is stored in the project directory. The name of the symbol file is composed as follows: `<project name>..<device name>..<application name>.xml`

**Example**

`proj_xy.PLC1.application.xml`

You can also generate the symbol file with the “Generate Code” command. This is very useful when downloading to the PLC is not possible.

The variables that you export as symbols can be bundled in the symbol configuration editor or defined in the variables declaration using the `attribute 'symbol'` pragma. Another option is using the element in the SFC editor, where you can define the implicitly generated element variables that should be exported to the symbol configuration.

The name of the symbol is generated in the symbol configuration in the following syntax: `<application name>..<POU name>..<variable name>`. When accessing the variable, you must always provide the complete symbol name in this syntax.

**Example**

`MyApplication.PLC_PRG.a` or `MyApplication.SymFB.a`

As a rule, read-only access applies to symbols for input addresses and for variables that are mapped to input channels. Write access is possible for testing purposes in simulation mode only.

The symbol file is loaded with the application to the PLC. Depending on the device description, this file can be generated as an additional (child) application. This application is then listed in the “Application” tab of the device editor. Syntax: `<application name>..symbols`. The symbol application is regarded as a "normal" application with respect to the maximum number of applications on the PLC.

If your controller has a user management, then you can assign different access rights to a symbol to the individual user groups (clients). To do this, place the same symbol in different symbol sets and allow the individual user groups (clients) either to access a symbol set or not. An on-site operator or an operating data record, for example, receives more information and access to the same symbols as telemaintenance.

See also

- § Chapter 1.3.1.25.1.36 “Object ‘Symbol Configuration’” on page 824
- § Chapter 1.3.1.24.6.2.37 “Attribute ‘symbol’” on page 635
- § Chapter 1.3.1.24.1.4.7.6 “SFC element properties” on page 459
- § Chapter 1.3.1.25.1.19.4 “Tab ‘Applications’” on page 747
- § Chapter 1.3.1.25.2.6.21 “Command ‘Simulation’” on page 937
Creating a symbol configuration

Requirement: The project can be compiled without any errors.

1. Select the “Application” object in the device tree.
2. Click “Project ➔ Add Object ➔ Symbol Configuration”.
   ⇒ The “Symbol Configuration” object is added to the device tree and the objects editor opens.
3. Open the “View” menu of the editor and activate the categories of variables that should be provided in the configuration editor. Click “Build” in the symbol configuration editor.
   ⇒ All variables (according to the currently defined filter in the “View” menu) are displayed in a tree structure.
4. Select the check boxes of individual variables.
   Note: Pay attention to the current settings (see the “Settings” button in the menu bar of the editor).
   ⇒ In the field below the menu bar of the editor, information is provided about the current situation with accompanying instructions, as well as controls for corrective actions.
5. Follow the prompt in the field below the menu bar. In the following case, this should be only the information that the modified symbol configuration is transferred with the next download or online change.
   Click “Build ➔ Generate Code” in the CODESYS main menu.
   ⇒ The <project name>.<device name>.<application name>.xml file is generated in the project directory.

CODESYS transmits the symbol configuration to the PLC for an application download or online change.

See also
● § Chapter 1.3.1.25.1.36 “Object 'Symbol Configuration'” on page 824

Creating symbol sets with different access rights for different control clients

A symbol set is a defined set of symbols. If supported by the target device, you can combine different symbol sets from the symbols of the application in the symbol configuration editor. The information about the symbol sets is downloaded to the controller. Then you can define the user group that has access to each symbol set. Rights are assigned in the “Symbol Rights” tab of the device editor.

As a result, symbol sets allow different client-specific access rights to a symbol in the controller.

You can download changes to a symbol set definition to the controller in an online change. When the application is deleted on the controller, the symbol sets are also deleted. When building the application, you can create and save a symbol file in XML format for each symbol set.

In the following section, you will see an example of steps for creating symbol sets and the assignment of rights on the controller:

Requirements: The application has a defined symbol configuration in the project. The “Enable symbol sets” option is enabled in the settings of the symbol configuration. The controller has a user management. For the example here, there should be a user group that has the necessary rights for the servicing of the plant. By default, this type of user group, named “Service”, is already created.

1. Define the connection to the controller in the “Communication Settings” of the device editor.
2. Click the button in the editor of the “Symbol Configuration” in order to create a new symbol set. Specify a name of the group (“startup”) in the dialog “Add a New Symbol Set”.
3. Click the button (“Build”) in the toolbar of the dialog in order to display all symbols available in the project. Select the users who should belong to the group. Save the project.
4. Click the button “Configure Symbol Rights”.
   ⇒ The “Symbol Rights” tab of the device editor opens.

5. Click the button (“Synchronization”) to synchronize the display of the symbol sets with
   the device.
   ⇒ If you have not enabled user management on the controller yet, then you will see a
dialog box in the “Users and Groups” tab prompting you to do it.

6. Click “OK” in the dialog box and click the “Users and groups” tab.

7. Click the button (“Synchronization”). Click “Yes” to confirm that user management
   should be enabled.
   ⇒ The “Device User Login” dialog opens.

8. Click “OK” in the dialog box and click the “Users and groups” tab.

9. Click “Synchronization” button.
   ⇒ In “Symbol sets”, you see all sets that have currently been downloaded for the appli-
cation (for this example, at least "startup"). In “Rights”, a table shows the user groups
that are created in the user management of the controller. In the example, we assume
that the default groups “Administrator” and “Service” have been created. When a
symbol set is selected on the left, you see on the right the access rights of the indi-
vidual user groups to this symbol set (access granted; access not granted). The
possible type of access is already defined for each symbol in the symbol configuration
(read, write, execute).

10. On the left, select the “Startup” symbol set and double-click the preset minus sign for
   “Administrator” as well as for “Service”.
   ⇒ The symbol changes into a plus sign. The “Administrator” and “Service” now have
   access to the symbols in the “Startup” symbol set.

See also
● Chapter 1.3.1.11.3 “Handling of device user management” on page 300

1.3.1.10.3 Network variables

The values of network variables can be exchanged between different PLCs in a network. The
variables must be defined in strict, identical lists on both the sender device and receiver device,
and only one device application defines the network variables. The lists can be in one or more
projects.

The network variable list in the sender is a global variable list where specific log and transfer
parameters are defined in their object properties. By adding these properties, you create a "net-
work variable list (sender)" from an ordinary “GVL”. You can also insert a “Network Variable
List (Sender)” object directly into the device tree when this object already has these parameters
set.

The network variable list in the receiver is of type “Network Variable List (Receiver)”. When
creating one, select the respective network variable list of the server. As an alternative, you can
read this variable list from an export file that was generated from the sender list. An export file is
required anyway for defining the sender list in another project.

The network variables are transmitted as broadcasting in one direction only: sender to receiver.
However, it is also possible for a device to contain both sender and receiver lists.
For the NetVarUdp library version 3.5.7.0 and later, a receiver channel is no longer assigned when confirmed transfer is not selected. In this way, network variable exchange is also possible between two controllers on one hardware device.

**NOTICE!**
- If the exchanging devices should be senders and receivers, then the variable list identifiers must be unique in order to prevent abnormal operation. The variable list identifiers are defined in the “Properties” dialog of an object GVL.
- Data exchange via network variables is not possible when:
  - The device (target system) does not support it.
  - A firewall blocks the communication.
  - Another client or application is using the UDP port that is set in the properties of the network variable list.
  - More than one application per sender device and receiver device use network variable lists.
- Only arrays that have limits defined with a literal or constant are transmitted to the receiver application. Constant expressions are not permitted for this purpose.
  - Example: "arrVar : ARRAY[0..g_iArraySize-1] OF INT ;" is not transmitted, but "arrVar : ARRAY[0..10] OF INT ;" is transmitted.
- The maximum size of a network variable is 255 bytes. The possible number of network variables is unlimited.
- If the size of the GVL exceeds the maximum length of the network telegram, then the data is split into multiple telegrams. Depending on the configuration, this can result in data inconsistencies.

Communication by means of network variables is also possible when the PLCs operated with applications from different versions of the development system (V2.3, V3). However, in this case, you cannot use the export/import mechanism for matching the variable lists exactly in the sender and receiver projects. The reason is that an variable export file (*.exp) that is generated from V2.3 does not include the required amount of information necessary for creating a receiving NVL in V3. There is no respective network parameter configuration as a GVL file, which you exported from the sender previously. To get this file, you must recreate the V2.3 NVL in V3 first. Then you can generate an export file and create a receiving NVL in V3 based on this.

An alternate to data exchange between PLCs is the use of data sources. As opposed to the broadcasting method for exchanging network variables, defined point-to-point connections are created between one application and a remote data source.

See also

- § Chapter 1.3.1.25.3.10.10 “Dialog ‘Properties’ - ‘Network Variables’” on page 1052
- § Chapter 1.3.1.25.1.21 “Object ‘GVL’ - Global Variable List” on page 769

Configuring a network variable exchange

The following steps are necessary for exchanging network variables between the sender device and receiver device.
1. **Creating a network variable list in the sender device and generating an export file**

Requirements: An application is inserted in the device tree of the PLC that has been employed as the sender device.

1. Select the application and insert a “Network Variable List (Sender)” object. Make the following settings in the “Add Network Variable List (Sender)” dialog: network type: UDP, example: "NVL_Sender".
2. Double-click the NVL object to open the respective editor and type the declarations of the network variables. Example:
   ```
   VAR_GLOBAL
   iglobvar:INT;
   bglobvar:BOOL;
   strglobvar:STRING;
   END_VAR
   ```
3. Right-click the NVL object in the device tree to open the “Properties”. Open the “Network Variables” tab in the “Properties” dialog. This shows the following settings: Network type: UDP; List identifier: 1; Pack variables; Cyclic transmission: every 50 ms.
4. Note: You can also convert an existing GVL into a network variable list by configuring its network variable properties.
5. Click the “Link To File” tab in the “Properties” dialog of the “NVL_Sender”. Define a file name `<export>.gvl` and a location in the file system for the export file of the GVL. Select the “Export before compile” check box.
6. Click “Build ➔ Build” to compile the application.

The export file for the network variable list is now located in the defined folder.

See also

- Chapter 1.3.1.25.3.10.10 “Dialog ‘Properties’ - ‘Network Variables’” on page 1052

2. **Creating an associated network variable list in the receiver device**

Requirements: A sender device and a receiver device exist in the device tree. An application with a task configuration is inserted below the device. An NVL or a GVL is created below the sender device as network variable list to be sent.

1. Select the application of the receiver in the device tree and click “Add Object ➔ Network Variable List (Receiver)”.
   - The “Add Network Variable List (Receiver)” dialog opens.
2. In the dialog, select the previously created NVL of the sender device and type a name (for example, "NVL_Receiver"). CODESYS populates this recipient list automatically with the variable declarations from the sender list.
   - Note: As an alternative, you can select the “Import from file” option and load the export file that was generated previously from the sender list.

3. **Testing the network variable exchange**

Requirements: A network variable list (sender) exists in the sender device, a network variable list (receiver) exists in the receiver device, and both lists have identical variable declarations.

1. Below the application in the sender device, create a program that increments a network variable. Example: `iglobvar := iglobvar + 1;`.
2. Configure the application task so that this program calls it.
3. Below the application in the receiver device, create a program that writes the value of this network variable to a local variable. Example: ilocalvar:=iglobvar;.

4. Configure the application task so that this program calls it.

5. Download both applications to the controllers and start them. (Set the application as active, login, download, and start.)

6. In the online views of the editors of both programs, check whether the values of iglobvar match in the receiver and the sender.

Configuration of network variable communication between V2.3 and V3 PLCs

Requirements: A global variable list exists in a V2.3 project with network variables that you want to read into your V3 project. In the V3 project in the device tree below the PLC, there is an application with a program that uses the variables.

1. Open the “Properties” dialog of the GVL in the V2.3 project (sender).
2. Insert another PLC into the project as a dummy with an application and a task. Then create the V2.3 GVL below it as follows:
3. Insert a “Network Variable List (Sender)” object named "dummy23" below the dummy application.
4. Right-click “dummy23” to open the “Properties” dialog. Click the “Network Variables” tab. Type the same network properties as the GVL in the V2.3 project.
5. Fill “dummy23” with the same variable declarations as in the V2.3 GVL.
6. Define a location for an export file "dummy23.gvl" in the “Link To File” tab of the “Properties” for “dummy23”.
7. Set the application of the dummy PLC as active and press [F11] to compile. CODESYS creates the export file.
8. Set the application of the physical PLC as active and right-click to open the “Add Object” dialog.
9. Select “Network Variable List (Receiver)” and type a name (for example, "NVL_from_23").
10. Select the “Import from file” check box and select dummy23.gvl, which was generated previously.
11. Check “NVL_from_23”, which has just been created. You must receive the same variable declarations as those in the sender GVL in the V2.3 project.

See also
- ¶ Chapter 1.3.1.25.1.26 “Object 'Network Variable List (Sender)’” on page 774
- ¶ Chapter 1.3.1.25.1.27 “Object 'Network Variable List (Receiver)’” on page 775

1.3.1.10.4 Data source manager

In order to have read/write access to remote devices and their running applications, you can add a data source manager to your application with one or more data sources.

The functionality of the data source manager allows for establishing connections and communication to remote devices, and it makes its data available through data source variables. At this time, the partners communicate by means of a point-to-point connection. Depending on the network where the controllers are located, a connection is established via different data source types and different connection types.

During the development phase, you can create a variable list offline by means of local symbol configuration files. In this way, you can develop a local application offline based on the symbol information without a connection to a data source.
The communication between remote and local devices communicate by means of data source types. CODESYS provides the following two types.

**CODESYS Symbolic:**
Data is accessed by means of symbolic monitoring. A symbol file on the remote device is read and the stored variable information is used for the data source variables and the data transfer. The advantage is that the application does not have to be updated in the local device when someone edits modifies the remote application without updating its symbol configuration. If the symbol file is also located on your development system (either a file or a symbol configuration object as part of your project), then the local symbol file can also be read. Then, you can work offline during the development phase.

You will usually use CODESYS Symbolic. Do not use CODESYS Symbolic only if no resources are available for a symbol configuration in the remote PLC.

**CODESYS Application V3:**
Data is accessed by means of address monitoring. This requires that the address information between the remote PLC and the local device match. Disadvantage: For changes to the remote application, the local application (for example the HMI application) is updated. The advantage is that a symbol configuration is not required in the remote application.

Use this communication for embedded or mini PLCs when there are no available resources for the symbol configuration.

See also
- ![Chapter 1.3.1.25.1.9 “Object 'Data Source Manager’” on page 723](#)

**Connection types**
The functionality of the data source manager allows for establishing a point-to-point connection from a local device to one or more remote devices (even through networks). Select the appropriate connection type.

When the communication is based on the data source type **CODESYS Symbolic**, the following connection types are possible:

- **“CODESYS V2”**: The devices are located in the same network. The V2 runtime system on the remote PLC provides a communication interface.
- **“CODESYS V2 (via gateway)”**: The devices are not located in the same network. They are connected via a V2 gateway. Note: For this connection, a “CoDeSys V2.3 Gateway Server” (V2 gateway) must be installed on the development computer where CODESYS V3 is running.
- **“CODESYS V3”**: The devices are located in the same network. The V3 runtime system on the remote PLC provides a communication interface.
- **“CODESYS V3 (via gateway)”**: The devices are not located in the same network. They are connected via a V3 gateway.

When the communication is based on the data source type **CODESYS Application V3**, the runtime system of the local application needs valid communication parameters in order to establish the connection. The network scan function can support you when configuring the data source.
Data transmission

When the local application is in runtime mode, the data source variables that appear in the data source editor of the “Variables” tab are updated in configurable time intervals. The remote application is also executed at this time. Variables that are configured in the visualization, in the trend, as alarms, or for recipes are transferred and stored automatically. When a variable is accessed in IEC code only, the variable is not updated automatically. In this case, you must activate the “Update always” option in the data source editor of the “Variables” tab.

The communication types support the (read or write) data access to variables of the source PLC for the following data types:

- Scalar value at highest level
  Example: PLC_PRG.hugo

- Property to a program or GVL by means of a call when it is marked with `{attribute monitoring := 'call'}`.
  Example: PLC_PRG.PropertyCall

- Variable that is mapped to bit addresses.
  Example in PLC_PRG:
  ```plaintext
  AT %MX0.5 : BOOL;
  ```

- Variable (type `BIT`) in a function block
  Declaration in PLC_PRG:
  ```plaintext
  x, y : BIT, access: PLC_PRG.dutInst.y
  ```

- Structured obtainable variable
  Example in PLC_PRG:
  ```plaintext
  outerInst.innerInst.dwVar
  ```

- Property to POU instance when it is marked with `{attribute monitoring := 'call'}`.
  Example: PLC_PRG.instance.PropertyCall

- Property at highest level and to an instance when it is marked with `{attribute monitoring := 'variable'}`.
  Note: This cannot be written by monitoring or by the data sources.

- Array access with literal index
  Example:
  ```plaintext
  PLC_PRG.arrOfInts[3], PLC_PRG.inst.arrOfBool[1]
  ```

- Nested access (for example, array of structures or structure of arrays)

CAUTION!

Bit access used in visualizations that are transferred via a data source connection function only if they contain literal offset specifications. A visualization cannot process an offset specification by defined constants.

Adding initial data sources

For data exchange between your local device and a remote device, add a “Data Source Manager” object and then a “Data Source” below your application in the device tree. An assistant guides you through the configuration of the data source. Then you can modify the settings in the editor of the object at any time. However, it is not possible to modify the data source type later.
Use the data source type “CODESYS Symbolic”, even if there are not any resources available in the remote PLC for a symbol configuration. As long as the symbol configuration is not affected after an application change in the remote device, you have the advantage that the application does not have to be updated in the local device.

When adding a data source, select a data source type. Then specify the connection settings of the point-to-point connection to the remote device. Ideally, the remote device is being executed in the meantime and CODESYS can establish the connection immediately. Then all available data source variables are displayed from the remote PLC. Select the variables that should be transferred. You can also select all variables. Then the data source is initialized automatically, the data source variables are created below the “DataSources_Objects” folder, and an additional data source is added below the data source manager.

When the data is transferred by means of symbolic monitoring and the symbol file is stored on your development system, then you can read the variable information from the symbol file and work offline. The symbol file is stored as either a file on your development system or a symbol configuration object as part of your project (in CODESYS).

The initial settings can be modified at any time in the editor of the data source.

See also
- Chapter 1.3.1.25.1.9 “Object ‘Data Source Manager’” on page 723

A “CODESYS Control Win V3” is running on the remote device. In addition, an application is running with a symbol configuration.

1. Insert a “Data Source Manager” object below your application in the device tree.
2. Select the “Data Source Manager” object and click “Add Object ➔ Data Source”.
   - The “Add Data Source” dialog opens.
3. Specify the data source name in “Name”.
   - Example: ds_Remote_Device
4. Select the “CODESYS Symbolic” data source type.
   - Data The data is transmitted via symbolic monitoring. The “Initialize Data Source Wizard - Provider Settings” dialog opens.
5. Select the connection type “CODESYS V3”.
6. Select the “Node name” option for “Type of name or address”.
7. In “Name or address of device”, specify the connection parameters for configuring the remote device. Example: “[03A7]”
   - The connection to the remote device is established and the application is read. In addition, the “Initialize Data Source Wizard - Browse Data Items” dialog opens. Below the “Variables” item, the read remote control variables are displayed in a tree view. The top node is the application that is displayed with its remote application name.
8. In the tree view, select the control variables to be transmitted, and click “Exit”.
   - The data source is configured initially. The object ds_Remote_Device is added below the node “Data Source Manager”. The object is open and the data source variables that were created are listed in the tree view on the “Variables” tab. The GVL ds_Remote_Device, where the data source variables are declared, is displayed below the folder “DataSource_Objects”.
Initial connection of devices symbolically with 'CODESYS V3 (via gateway)'

A “CODESYS Control Win V3” is running on the remote device. In addition, an application is running with a symbol configuration. The remote device is located in another network so that the communication must be routed via a gateway.

1. Insert a “Data Source Manager” object below your application in the device tree.
2. Select the “Data Source Manager” object and click “Add Object ➔ Data Source”.
   The “Add Data Source” dialog opens.
3. Specify the data source name in “Name”.
   Example: ds_Remote_Device
4. Select the “CODESYS Symbolic” data source type.
   The data is transmitted via symbolic monitoring. The “Initialize Data Source Wizard - Provider Settings” dialog opens.
5. Select the connection type “CODESYS V3 (via gateway)”. You can also specify the communication parameters for the gateway.
6. Select the “Node name” option for “Type of name or address”.
7. In “Name or address of device”, specify the connection parameters for configuring the remote device. Example: “[03A7]”
   The connection to the remote device is established and the application is read. In addition, the “Initialize Data Source Wizard - Browse Data Items” dialog opens. Below the “Variables” item, the read remote control variables are displayed in the tree view. The top node is the application that is displayed with its remote application name.
8. In the tree view, select the control variables to be transmitted, and click “Exit”.
   The data source is configured initially. The object ds_Remote_Device is added below the node “Data Source Manager”. The object is open and the data source variables that were created are listed in the tree view on the “Variables” tab. The GVL ds_Remote_Device, where the data source variables are declared, is displayed below the folder “DataSource_Objects”.

See also
● Table 43 “Connection settings for CODESYS V3 (via gateway)” on page 731

Initial connection of devices symbolically with 'CODESYS V2'

A “CoDeSys V2.3 SP PLCWinNT V2.4” is running on the remote device. In addition, an application is running with a symbol configuration.

1. Insert a “Data Source Manager” object below your application in the device tree.
2. Select the “Data Source Manager” object and click “Add Object ➔ Data Source”.
   The “Add Data Source” dialog opens.
3. Specify the data source name in “Name”.
   Example: ds_Remote_Device
4. Select the “CODESYS Symbolic” data source type.
   - Data The data is transmitted via symbolic monitoring. The “Initialize Data Source Wizard - Provider Settings” dialog opens.

5. Select the connection type “CODESYS V2”.

6. In “Connection Settings”, specify the connection parameters for configuring the remote device.
   
   Example: Driver type TCP/IP (Level 2 Route), address localhost, port 1200

   - The connection to the remote device is established and the application is read. In addition, the “Initialize Data Source Wizard - Browse Data Items” dialog opens. Below the “Variables” item, the read remote control variables are displayed in a tree view.

7. In the tree view, select the control variables to be transmitted, and click “Exit”.
   
   - The data source is configured initially. The object ds_Remote_Device is added below the node “Data Source Manager”. The object is open and the data source variables that were created are listed in the tree view on the “Variables” tab. The GVL ds_Remote_Device, where the data source variables are declared, is displayed below the folder “DataSource_Objects”.

See also

- Table 40 “Connection settings for CODESYS V2” on page 729

---

A “CoDeSys V2.3 SP PLCWinNT V2.4” is running on the remote device. In addition, an application is running with a symbol configuration. The remote device is located in another network so that the communication must be routed via a gateway.

1. Insert a “Data Source Manager” object below your application in the device tree.

2. Select the “Data Source Manager” object and click “Add Object ➔ Data Source”.
   
   - The “Add Data Source” dialog opens.

3. Specify the data source name in “Name”.
   
   - Example: ds_Remote_Device

4. Select the “CODESYS Symbolic” data source type.
   
   - Data The data is transmitted via symbolic monitoring. The “Initialize Data Source Wizard - Provider Settings” dialog opens.

5. Select the connection type “CODESYS V2 (via gateway)”.
   
   - You can also specify the communication parameters for the gateway.

6. In “Connection Settings”, specify the connection parameters for both the gateway and the device configuring the remote device.
   
   Example: Driver type TCP/IP (Level 2 Route), address localhost, port 1200

   - The connection to the remote device is established and the application is read. The “Initialize Data Source Wizard - Browse Data Items” dialog opens. Below the “Variables” item, the remote control variables are displayed in the tree view.

7. In “Name or address of device”, specify the connection parameters for configuring the remote device. Example: “[03A7]”
   
   - The connection to the remote device is established and the application is read. The “Initialize Data Source Wizard - Browse data items” dialog opens. Below the “Variables” item, the read remote control variables are displayed in a tree view. The top node is the application that is displayed with its remote application name.
8. In the tree view, select the control variables to be transmitted, and click “Exit”.
   - The data source is configured initially. The object `ds_Remote_Device` is added below the node “Data Source Manager”. The object is open and the data source variables that were created are listed in the tree view on the “Variables” tab. The GVL `ds_Remote_Device`, where the data source variables are declared, is displayed below the folder “DataSource_Objects”.

See also

- Table 41 “Connection settings for CODESYS V2 (via gateway)” on page 730

**Initial addition of data source variables from a symbol file**

Ideally, the same symbol file on the remote device is saved on your development system.

1. Insert a “Data Source Manager” object below your application in the device tree.
2. Select the “Data Source Manager” object and click “Add Object ➔ Data Source” in the context menu.
   - The “Add Data Source” dialog opens.
3. Specify the data source name in “Name”.
   - Example: `ds_Symbols`
4. Select the “CODESYS Symbolic” data source type.
   - Data The data is transmitted via symbolic monitoring. The “Initialize Data Source Wizard - Provider Settings” dialog opens.
5. Click the item “From symbol file” in “Variable information”.
6. In “Select symbol file”, specify the storage location and the file name of the symbol file. When code is generated, an XML symbol file is created in the project directory by default.
   - Example: `D:\Projects\V3.5 SP11\Project_A.Device.Application.xml`
   - Tip: When a symbol file is specified, no additional connection settings have to be configured. A connection is not established. You work offline. You have to configure the connection settings only when you need current data from the controller that is transferred online. Activate the option “From connection settings” in the “Variable information” settings.
7. Click “Next”.
   - The “Initialize Data Source Wizard - Search Data Source Variables” dialog opens. Below the “Variables” item, the read symbols are displayed in the tree view.
8. Select the symbols to be transferred in the tree view. Then select “Exit”.
   - The data source is configured initially. The object `ds_Symbols` is added below the node “Data Source Manager”. The object is open and the data source variables that were created based on the symbol file are listed in the tree view on the “Variables” tab. The GVL `ds_Symbols`, where the data source variables are declared, is displayed below the folder “DataSource_Objects”.

See also

- Table 41 “Connection settings for CODESYS V2 (via gateway)” on page 730
Your active project contains the control application for the remove device. The control application includes a symbol configuration with symbols that are added to your local application as data source variables.

1. Insert the “Data Source Manager” object below your local application in the device tree.
2. Select the “Data Source Manager” object and click “Add Object ➔ Data Source”.
   - The “Add Data Source” dialog opens.
3. Specify the data source name in “Name”.
   - Example: ds_Symbols
4. Select the “CODESYS Symbolic” data source type.
   - Data The data is transmitted via symbolic monitoring. The “Initialize Data Source Wizard - Provider Settings” dialog opens.
5. Select the item “<remote device>.<application>.symbol configuration” in “Variable information”.
   - Example: Device.Application.Symbol Configuration
   - Tip: When a symbol file is specified, no additional connection settings have to be configured. A connection is not established. You work offline.
6. Click “Next”.
   - The “Initialize Data Source Wizard - Search Data Source Variables” dialog opens. Below the “Variables” item, the read symbols are displayed in the tree view.
7. Select the symbols to be transferred in the tree view. Click “Finish”.
   - The data source is configured initially. The object ds_Symbols is added below the node “Data Source Manager”. The object is open and the data source variables that were created based on the symbol configuration are listed in the tree view on the “Variables” tab. The GVL ds_Symbols, where the data source variables are declared, is displayed below the folder “DataSource_Objects”.

See also
- Table 41 “Connection settings for CODESYS V2 (via gateway)” on page 730

A “CODESYS Control Win V3” is running on the remote device. The project of the remote device is located on your development computer. The developed application there does not contain a symbol configuration.

Use this communication connection only if no resources are available for a symbol configuration in the remote PLC.

1. Insert a “Data Source Manager” object below your application in the device tree.
2. Select the “Data Source Manager” object and click “Add Object ➔ Data Source”.
   - The “Add Data Source” dialog opens.
3. Specify the data source name in “Name”.
   - Example: ds_Remote_Device
4. Select the “CODESYS ApplicationV3” data source type.
   - Data The data is transmitted via address monitoring. The “Initialize Data Source wizard - Provider settings” dialog opens.
5. Select the “Other project” option for “Select project type”.

Initial addition of data source variables from a symbol configuration

Initial connection of devices with address monitoring
6. For “Choose file”, specify the file and storage location of the project on the remote device. Example: C:\Data\Projects\PLC_A.project.
   - The remote device is displayed in the tree view of the window below, and as a result the connection was established.
7. Select the link “From device”.
   - The connection parameters to the remote device are read and displayed in the dialog. The connection is configured.
8. Click “Next>”.
   - The “Initialize Data Source Wizard - Browse Data Items” dialog opens. Below the “Variables” item, the remote control variables are displayed in the tree view.
9. In the tree view, select the control variables to be transmitted, and click “Exit”.
   - The data source is configured. A connection is established. The settings are stored in the object and can be modified in the editor of the object.

The data source is configured initially. The object ds_Remote_Device is added below the node “Data Source Manager”. The object is open and the data source variables that were created are listed in the tree view on the “Variables” tab. The GVL ds_Remote_Device, where the data source variables are declared, is displayed below the folder “DataSource_Objects”.

See also
- “Tab 'Communication' for CODESYS ApplicationV3” on page 732

### Editing data source variables

In runtime mode, the remote data is saved to the data source variables. The data source variables and their mapping to the remote variables are displayed in the data source editor below of the “Variables” tab. If the local and remote variables have the same names and the same data types, then the data is mapped 1:1. The variables and the data types are created automatically. That is the regular procedure.

You can also map to existing variables. This is necessary, for example, if a visualization includes a data type in an interface. Then the same data must be passed to this visualization. In this case, the declared local variable and the remote variable have the same data type, for example from one library. Moreover, you can map a local variable with a conforming data type to a remote variable. The data type can be created in the “Type Mappings” tab.

The specifically created variables and data types are declared in the “DataSources_Objects” folder. For each data source, a global variable list of the same name as the data source is declared there. Moreover, the data source variables usually have the identical or conforming data type as the remote control variable and they are declared as user-defined data types (DUT objects). Considering all data sources, multiple declaration of the same data types is avoided.

Do not edit the data interface in the “DataSources_Objects” folder manually. It is created initially when adding a data source. Changes can be made later in the editor of the data source.

See also
- Chapter 1.3.1.10 “Working with control networks” on page 270
- Chapter 1.3.1.10.4.4 “Updating data interfaces” on page 291
- Chapter 1.3.1.25.1.10 “Object 'Data Source' - Tab 'Variables'” on page 725

### Selecting variables for data transfer

You can edit the selection of the data source variables.
Requirement: The remote device and its application are running. A data source manager is already inserted below the local application with a data source.

1. Open the editor of the data source.
2. Select the “Variables” tab.
3. Click “Update Variables”.
   ⇒ The “Browse Variables” dialog opens.
4. Activate the variables that should be transferred and click “OK” to close the dialog.
   ⇒ The data source variables are modified according to the selection. The declaration of variables and data types is also modified.
   The “Variables” tab shows the modified selection. Moreover, the mapped remote variable is listed in the “Remote variable” column.

See also

- Chapter 1.3.1.10 “Working with control networks” on page 270
- Chapter 1.3.1.10.4.4 “Updating data interfaces” on page 291
- Chapter 1.3.1.25.1.10 “Object ‘Data Source’ - Tab ‘Variables’” on page 725

Mapping remote variables to a new variables

You need to map a remote variable to a global implicit variable that is created new. That is the regular procedure for transposing data source to 1:1.

Requirement: A project is open. A data source manager and a data source below it are located in the device tree of the local application.

1. Open the editor of the data source.
2. Select the “Variables” tab.
   ⇒ The data source variables are listed.
3. Select a variable and click the symbol in the “Create or map” column.
4. Specify a name in “Local variable”.
   ⇒ A variable is declared automatically and it contains the same value as the mapped remote variable.

Mapping remote variables to an existing variables

You need to map a remote variable to an existing variable.

Requirement: A data source manager and a data source below it are located in the device tree of the local application. The remote data that should be transferred is displayed in the editor of the data source in the “Variable” tab

1. Open the editor of the data source.
2. Select the “Variables” tab.
3. Select a variable and click the symbol in the “Create or map” column.
   ⇒ A variable contains the same value as the mapped remote variable.
First, create a conforming data type and then use it for a variable.

1. Open the editor of the data source.
2. Select the “Type Mappings” tab.
3. Select the data type in the list that you want to edit.
   - The elements of the data type are listed in the window below the data type list
4. Specify a name for the data type. Example: DataType_A. Select the name for the remote data types to which the local type should conform. Example: Library1.DataType_A.
5. Modify it in the window below the data type list and remove the elements that are not necessary for the data transfer.
6. Select the symbol for this data type in the “Create or map” column.
   - The data type DataType_A is declared in the “DataSources_Objects” folder.
7. Select the “Variables” tab.
8. Specify a name in the “Local variable” column. Example: Var_A
9. Select the symbol in the “Create or map” column.
10. Specify the data type DataType_A in the “Mapping type” column.
11. Select the remote variable with the data that should be transferred. Example: appPLC_A.Data_A. Use the input assistance for this.
   - A variable Var_A is declared automatically with the user-defined data type DataType_A. During data transfer, it receives the data of the mapped remote variables.
Example

Library SnakeUtil

The example demonstrates how variables of the data source are created. At this time, new variables are created, data is mapped to existing data types and their variables, and new data types are created with type-conforming mapping.

The remote PLC uses POU instances from the SnakeUtil library and the HMI device visualizes these POU instances. This is why the HMI application requires a variable in the operating interface that has a data type appropriate for a visualization template. As a result, the SnakeUtil library is linked integrated into the HMI application and the HMI variables instantiate the SnakeUtil.SnakeVisu visualization function block.

The following library function blocks from the SnakeUtil library are used in the remote PLC.

- Function block SnakeUtil.Snake: Equipped with much logic and calling from external functions.
- DUT SnakeUtil.PositionInfo: Two values (of the variables x and y)
- DUT: SnakeUtil.DrawingInfo: Image ID
- The SnakeUtil.SnakeVisu visualization function block with transfer parameter SnakeUtil.Snake visualizes the Snake function block.

The following settings are entered in the editor of the “Type Mappings” tab:

In the visualization, a frame is inserted with a reference to SnakeUtil.SnakeVisu. This expects to have the type SnakeUtil.Snake.

The data types SnakeUtil.PositionInfo and SnakeUtil.DrawingInfo are mapped to existing data types (symbol in the “Create or map” column). The data types are small and contain data only.

The SnakeUtil.Snake function block is very complex and calls external functions that are not available in the HMI visualization. The function block with code is not required in the visualization. You need a less extensive but compatible and conforming type in the HMI visualization. Therefore, do not create the original data type directly. Instead, first modify the original data type and remove the unnecessary elements. Then create the new data type Snake by selecting the symbol in the “Create or map” column.

Editing communication

You have added a “Data Source Manager” object and below it a “Data Source” object below your application in the device tree. The connection parameters are displayed in the data source editor of the “Communication” tab. You can modify it there.

The data source type and the current connection type are listed in the status bar. It is not possible to modify the data source type later.

See also

- Chapter 1.3.1.10.4.1 “Adding initial data sources” on page 282
- Chapter 1.3.1.25.1.12 “Object ‘Data Source’ - Tab ‘Communication’” on page 728

Updating data interfaces

The data source variables are updated cyclically in runtime mode. Only the data is updated that either is used in the current visualization or has the property “Update always”.

2020/12/10
You can define the time interval. Moreover, you can define variables whose data is transferred in each update interval, and therefore they are always update. To update variables that are not used in the application code, you can implement an update programmatically with the help of interface functions from the data source manager.

<table>
<thead>
<tr>
<th>NOTICE!</th>
</tr>
</thead>
<tbody>
<tr>
<td>If data traffic between the remote and local device is too high, then the update rate is reduced automatically. This can lead to an incomplete transfer.</td>
</tr>
</tbody>
</table>

See also

- [Chapter 1.3.1.25.1.10 “Object ’Data Source’ - Tab ’Variables’” on page 725](#)

### Setting the update rate

1. Open the editor of the data source.
2. Click the “General and Diagnosis” tab.
3. Specify a value in the “Update rate” field.

   **Example:** 100

   - The data from the remote device to the local device is transferred every 100 ms.

See also

- [Chapter 1.3.1.25.1.13 “Object ’Data Source’ - Tab ’General and Diagnosis’” on page 734](#)

### Selecting the variable for ’Update always’

<table>
<thead>
<tr>
<th>NOTICE!</th>
</tr>
</thead>
<tbody>
<tr>
<td>Avoid updating too many variables always. Each update produces additional data traffic at the connection between the remote and local devices. When data traffic is too high, the update rate is reduced automatically. This can lead to an incomplete transfer.</td>
</tr>
</tbody>
</table>

1. Open the editor of the data source.
2. Activate the option “Update always” for a variable.

   - The data of the variables is transferred at each update cycle, even when the data has not changed.

See also

- [“Tab ’Variables’” on page 725](#)

### Updating data programmatically

The data source manager provides interface functions in the Datasources library. If a data source manager is integrated in the application code, then the global variable g_Datasources is instantiated automatically. This provides access to the interface functions.

Then you can update individual variables that are not called in the active visualization.
The variable `ivar` is activated and deactivated by means of methods from the `Datasources` library so that its value is transferred. Furthermore, you can configure that the variable is updated only over a defined duration in order to save transfer capacity.

```cpp
//Synchronize with DatasourcesTask and block until access is possible
//Regard the feedback in ERR_OK or in ERR_DE_MULTITASKING_LOCKED
if (Datasources.BeginDataConfiguration(TRUE)) {  
    g_DataServer.UseData(ADR('RemoteDevice.Application.PLC_PRG.iVar'));  
    // Deactivate variable
    g_DataServer.ReleaseData(ADR('RemoteDevice.Application.PLC_PRG.iVar'));  
    g_DataServer.EndDataConfiguration();
}
```

The data configuration is started with `BeginDataConfiguration(TRUE)`, thus initializing the synchronization of the task `DatasourceTask` with the application task. The value `TRUE` blocks the processing until the access to the variable is possible; `FALSE` repeats access attempts without blocking. The return values `ERR_OK` and `ERR_DS_MULTITASKING_LOCKED` provide feedback about the access attempts.

When synchronization is successful, the variable is activated by means of the `UseData` method. Then the data configuration is completed with the `EndDataConfiguration` method and the synchronization triggered again with the task `DatasourceTask`.

The `ReleaseData` method is used in the same way for deactivating the variable again at the desired processing time.

### Using remote data

The variables that are listed in the data source editor of the “Variables” tab (and declared in the “DataSources_Objects” folder) can be used in your application like IEC variables. For example, you can visualize the variables. If multiple data sources are available and therefore conflicts occur regarding unique variable names, then you must specify the data source name as the prefix. If no conflicts occur, then this is not necessary and you can map the variables without a data source prefix.

`<data source name>.<function block name>.<variable name>`

### Displaying variable values from the remote device

You need to show the variable value `iTemp` of a remote device in a visualization element of a visualization in the local application (with the data source manager).

Initial situation: A data source `dsRemotePLC` is below the local data source manager where the connection to the remote device is configured. In addition, the variable `iTemp` is selected in the data source editor of the “Variables” tab.

1. Select the visualization element in the editor view. Select the properties “Text variables”-“Text variable” in the “Properties” view.
2. Select the `iTemp` variable.
   - The variable mapping is qualified. Example: `dsRemotePLC.PLC_PRG.iTemp`.
3. Select the “Text” property of the visualization element and type in the following:
   ```text
   Temperature: %s
   ```
   - The value of the `iTemp` variable from the remote device `RemoteDevice` is displayed.
4. Download and start the remote application.
5. Download and start the local application.
   - The visualization starts and displays the actual value of `iTemp`. 
1.3.10.5 Subordinate safety controller

If a safety controller is below the standard controller, then the communication with the development system and the data exchange run via the standard controller. The communication links of the safety controller can interrupt the execution of commands that affect the standard controller. You find a notice about this for each these command.

Possible interruptions

- Temporary interruption: During the execution of the command (for example: download), the connections with the safety controller are interrupted first and then are automatically available again afterwards. If the interruption time is too long, then safety-oriented reactions can occur in the output devices and connected network variable receiver safety controllers. Then in the safety controller, the corresponding communication errors must be acknowledged (if not done automatically) in order to end the safety-oriented reactions. This affects the connection to their field devices and network variable receiver connections to other sender safety controllers. In the case of a connected safety controller with network variable senders, the communication errors must be acknowledged in the other safety controllers.

- Permanent interruption: The execution of commands (for example: delete) leads to an interruption that is ended again by another action (for example: download). As a result of the interruption, safety-oriented reactions can occur in the output devices and connected network variable receiver safety controllers. After ending the interruption, the corresponding communication errors must be acknowledged in the safety controller (if not done automatically) in order to end the safety-oriented reactions.

For a subordinate safety controller, the routing runs via "<Name of SafetyApp>_Mapping". In some cases, it can happen that the user can see this application in the device tree.

CAUTION!

No commands may be executed in the application "<Name of SafetyApp>_Mapping".

- Chapter 1.3.1.10 “Working with control networks” on page 270
1.3.1.11 Transferring an application to the PLC

1.3.1.11.1 Configuring the connection to the PLC .................................................. 295
1.3.1.11.2 Encrypting communication, changing security settings .......................... 296
1.3.1.11.3 Handling of device user management ............................................. 300
1.3.1.11.4 Generating application code ............................................................. 303
1.3.1.11.5 Downloading the application code, logging in, and starting the PLC .... 304
1.3.1.11.6 Generating boot applications ............................................................ 305
1.3.1.11.7 Downloading source code to and from the PLC ............................. 307
1.3.1.11.8 Distributing application code to different memory areas ................. 308

In order to transfer your application to the PLC, the program must be compiled without any errors and the connection settings for the PLC must be set.

If the communication with the controller is encrypted and/or restricted to specific users, then you need the respective certificates and permissions. See here:

- Chapter 1.3.1.11.3 “Handling of device user management” on page 300
- Chapter 1.3.1.9.18 “Encrypting an application” on page 242

You can edit the basic security policy for communication with the device in a dialog on the “Communication Settings” tab of the device editor. See here:

- Chapter 1.3.1.11.2 “Encrypting communication, changing security settings” on page 296

If these requirements are fulfilled, then the application is downloaded to the PLC at login.

1.3.1.11.1 Configuring the connection to the PLC

The connection to the controller is established via a gateway. This gateway can be your development computer or another network computer connected to the controller. The “Communication Settings” dialog is available for configuring the connection path. This dialog opens automatically when you attempt to log in, but the communication settings have not been configured yet.

If the communication with the controller is encrypted and secured by means of user management, then you need a corresponding certificate and credentials to establish the connection to the controller. In this case, see the relevant instructions on the “Encrypting communication and Changing Security Settings” help page.

Requirement: The project can be compiled without any errors. A programmable logic controller (PLC) is inserted in the device tree.

1. In the device tree, select the PLC and click “Project ➔ Edit Object”.
   ☐ The PLC opens in the editor.
2. Select the “Communication Settings” tab.
3. Click “Scan Network” in the menu bar.
   ☐ The “Select Device” dialog opens. All available devices in the network are shown on the left side.
4. Select the required device and click “OK”.
   ☐ The connection path for the PLC is set.
You can reset the communication settings view to the original view in the CODESYS options of the device editor.

See also
- Chapter 1.3.1.11.2 “Encrypting communication, changing security settings” on page 296
- Chapter 1.3.1.25.1.19.2 “Tab ‘Communication Settings’” on page 741
- Chapter 1.3.1.25.3.13.9 “Dialog ‘Options’ – ‘Device Editor’” on page 1081

1.3.11.2 Encrypting communication, changing security settings

NOTICE!
Recommendations for data security

In order to minimize the risk of data security violations, we recommend the following organizational and technical actions for the system where your applications are running. As far as possible, avoid exposing the PLC and control networks to open networks and the Internet. Use additional data link layers for protection, such as a VPN for teleaccess. Install firewall mechanisms. Restrict access to authorized people. Change any default passwords regularly before and after commissioning.

Use the security features supported by CODESYS and the respective controller, such as encryption of communication with the controller and intentionally restricted user access.

Communication with the device can be protected by means of encryption and user management on the device. You can change the current security preset on the “Communication Settings” tab of the device editor.

Establishing a connection to the controller, logging in, installing a trusted certificate for encrypted communication

- Requirement: Encrypted communication with the controller and user management are enforced on the controller. However, there is no individual password yet, no certificate installed on your computer, and the connection to the controller is not configured yet.

1. In the device tree, double-click the controller.
   - The device editor opens.
2. Click the “Communication Settings” tab.
3. Click “Scan Network”.

- Requirement: Encrypted communication with the controller and user management are enforced on the controller. However, there is no individual password yet, no certificate installed on your computer, and the connection to the controller is not configured yet.
4. Select a controller.
   ⇒ A dialog opens, informing you that the certificate of the device does not have a trusted signature for communication. You are prompted whether or not to install this certificate as trusted in the local "Controller Certificates" store on your computer.

   NOTICE!
   A controller certificate installed in this way is valid for only 30 days.
   This gives you time for the following long-term solutions:
   – Creation of an additional self-signed certificate with a longer term (for example, 365 days). You can do this on the security screen if you have installed the CODESYS Security Agent, even if a certificate already exists. Using the PLC shell of the device editor is not a convenient workaround.
     See below: "Configuring encrypted communication with a controller certificate with a more long-term validity period"
   – Importing a CA-signed certificate. This is currently only possible via the PLC shell commands of the runtime system. Therefore we recommend to use self-signed certificates first.

5. Click “OK” to confirm the dialog prompt.
   ⇒ The certificate is listed as trusted.
   After accepting the self-signed certificate for the first time, you can establish an encrypted connection with the controller again and again without further prompts.

6. If user management is currently enforced, then you are prompted to login. When you login for the first time, enter “Administrator” as your user name and password. Then the “Password expired, please enter a new one!” dialog opens to define an individual password.
   ⇒ You can now login to the controller as usual.

7. All saved controller certificates (from step 5) are kept in the local Windows Certificate Store on your computer. You can access this by means of “Execute”, certmgr.msc command.
   ⇒ All registered certificates for encrypted communication with controllers are listed here in “Controller Certificates”.

Requirement: The CODESYS Security Agent add-on product is installed. You want to replace the temporary certificate (as described above) acquired the first time you connected to the protected controller with a certificate with a longer validity period.

In this case, the “Security Screen” view provides an additional tab: “Devices”. This allows for the simple configuration of certificates for the encrypted communication with controllers. For operation, see the help for CODESYS Security Agent: "Encrypted Communication with Devices via Controller Certificates".

Choose this less convenient method when the CODESYS Security Agent is unavailable to you. In this case, you can set up a certificate with a more long-term validity period for communication encryption on the “PLC Shell” tab of the device editor.
☑ Requirement: You are connected to the controller.

1. At first, you check if a qualified certificate is already on the controller. If no certificate is available, then you create a new certificate.
   
   Open the device editor by double-clicking the controller in the device tree, and select the **PLC Shell** tab.
   
   ⇒ The tab appears with a blank display window. Below that is a command line.

2. Type the following command in the command line: `cert-getapplist`.
   
   ⇒ All used certificates are listed. The list includes information about the runtime component and whether or not the certificate is available.

3. If a certificate still does not exist for the component **CmpSecureChannel**, then type the following command in the input line:

   `cert-genselfsigned <number of the component in the apllist>`

4. Click the **Log** tab and then the refresh button (⇒).
   
   ⇒ The display shows whether or not the certificate was generated successfully.

5. Change back again to the **PLC Shell** tab and type the command `cert-getapplist`.
   
   ⇒ The new certificate for the component **CmpSecureChannel** is displayed.

6. In the next two steps, activate encrypted communication in the security screen of CODESYS.

7. Open the **“Security Screen”** by double-clicking in the status bar.

8. On the **“User”** tab, select the **“Enforce encrypted communication”** option in the **“Security Level”** group.
   
   ⇒ The communication to all controllers is encrypted. If there is not a certificate on a controller, then you cannot login to it.

   The connecting line between the development system, the gateway, and the controller is displayed in yellow on the **“Communication Settings”** tab of the device editor of the controller.

9. As an alternative to the **“Enforce encrypted communication”** option which applies to all controllers, you can also define encrypted communication for specific controllers only. To do this, select the **“Communication Settings”** tab in the editor of the respective controller. Then click **“Encrypted Communication”** in the **“Device”** list box.
   
   ⇒ The communication to this controller is encrypted. If there is not a certificate on the controller, then you cannot login to it.

   The connecting line between the development system, the gateway, and the controller is displayed in yellow on the **“Communication Settings”** tab of the device editor of the controller.

10. When you login to the controller for the first time, a dialog opens with information that the certificate of the controller is not signed by a trustworthy authority. In addition, the dialog displays information about the certificate and prompts for you to install it as a trustworthy certificate in the local store in the **“Controller Certificates”** folder.

    When you confirm the dialog, the certificate is installed in the local store and you are logged in to the controller.

    In the future, communication with the controller will be encrypted automatically with this control certificate.
11. To increase security for key exchange for controllers < V3.5 13.0, you can generate Diffie–
    Hellman parameters on the controller. To do this, type the command cert-gendhparams
    in the input line.

    This is no longer required for controllers >= V3.5.13.0.

    !
    **NOTICE!**
    Caution: Generating the Diffie-Hellman parameters can last for several
    minutes or even several hours. However, this process must be executed
    only one time for each controller. The Diffie-Hellman parameters increase
    security for key exchange and for future attacks against encrypted data
    recording.

Changing the communication policy (encryption, user management)

- Requirement: The connection to the device is established.
- 1. In the device tree, double-click the controller.
   ➜ The device editor opens.
- 2. Click the “Communication Settings” tab.
- 3. Open the “Device” menu in the header of the editor. Click “Change Communication
   Policy”.
   ➜ The “Change Communication Policy” dialog opens.
- 4. In the upper part of the dialog, you can toggle between the “Optional encryption”,
   “Enforced encryption”, and “No encryption” settings.
- 5. In the lower part of the dialog, you can toggle between the “Optional user management”
   and “Enforced user management” settings.

Enabling and disabling enforced encrypted communication

- Requirement: The device supports encrypted communication.
- 1. In the device tree, double-click the controller.
   ➜ The device editor opens.
- 2. Click the “Communication Settings” tab.
- 3. Open the "Device" menu in the header of the editor. Click “Encrypted Communication”.
   The status toggles between enabled and disabled.
   ➜ If the “Encrypted communication” option is selected, then the connection line between
   the development system, the gateway, and the device is highlighted in the editor in
   bold and in color in the graphical representation.

See also
- § Chapter 1.3.1.11.3 “Handling of device user management” on page 300
- § Chapter 1.3.1.9.18 “Encrypting an application” on page 242
- § “Encryption with certificates” on page 139
1.3.1.11.3 Handling of device user management

NOTICE!
Recommendations for data security

In order to minimize the risk of data security violations, we recommend the following organizational and technical actions for the system where your applications are running. As far as possible, avoid exposing the PLC and control networks to open networks and the Internet. Use additional data link layers for protection, such as a VPN for teleaccess. Install firewall mechanisms. Restrict access to authorized people. Change any default passwords regularly before and after commissioning.

Use the security features supported by CODESYS and the respective controller, such as encryption of communication with the controller and intentionally restricted user access.

For devices that support a device user management, the device editor includes the “Users and Groups” tab and the “Access Rights”. When offered by the device, you can view and also edit the user management for the device. Here, the defined user groups can be granted or denied specific rights on the controller.

The device user management can already be set up in the device description.

Note the commands in the “Online ➔ Security” menu. You can easily add, edit, or remove a user account on the controller where you are currently logged in.

In order for the “Access Rights” tab to be available in the device editor, the corresponding CODESYS option must be selected in the device editor and unlocked in the device description. If the device editor is not available, then contact the manufacturer of the controller.

In order to grant access rights to user groups, users and user groups first have to be configured on the “Users and Groups” tab of the device editor. User management first has to be set up on the controller before access rights can be configured on it.

General information about device user management

Note that access rights can only be granted to groups, not to individual users. Therefore every user has to be a member of a group.

Access rights can be granted for the following actions which are performed on the individual objects of the controller:

- View
- Modify
- Add/Remove
- Execute

Each object on the controller is usually assigned to just one component. And all of the above access rights can be used by the object. However, in most cases only the following two rights are required for an object:

- “View”
- “Modify”

The objects are organized in a tree structure. There exist two root objects for both of the separate types of objects: “Runtime objects ➔ Device” and “File system objects ➔ /”.

In the file system objects, the rights can be granted to folders of the current execution directory of the controller.
In the runtime objects, all objects are managed that have online access in the controller and therefore have to control the access rights.

In general, the access rights are inherited from the root object (also “Device” or “/”) to the subobjects. This means that if you deny or explicit grant a right to a user group on a parent object, then this first affects all child objects.

For the rights, a distinction also has to be made as to whether a right is explicitly granted, denied, or neutral. Neutral means that it is neither explicitly granted nor denied. In this case, the right of the parent object applies. However, if no right has been explicitly granted or denied in the entire hierarchy of the object, then it is by definition initially denied. This means that all rights are denied at first.

The only exception is the access right for the “View” action. This is explicitly granted initially for every user group both on the “Device” runtime object as well as on the “/” file system object. This allows read access to all objects at first, unless it is explicitly denied in child objects.

The “Everyone” user group has a special position here: This right only applies at the very end of the rights check if no other user group on a level has explicitly granted or denied access.

For an overview table for the objects, see the “Tab ‘Access Rights’” chapter.

See the following instructions for handling the editor for the device user management:

**First-time login on the controller in order to edit or view its user management**

Requirement: the controller has a device user management.

1. Double-click the controller device object in the device tree.
   ➞ The device editor opens.
2. Click the “Users and Groups” tab.
3. Click [Synchronization].
   ➞ A dialog opens prompting whether the device user management should be activated.
4. Click “Yes” to confirm the prompt.
   ➞ The “Device User Logon” dialog opens.
5. Specify “Administrator” as the “User Name” and “Password”.
   ➞ The “Password expired, please enter a new one!” dialog opens.
6. Specify a new “Password”. The password strength is displayed.
   ➞ After you click “OK” to confirm, the device user management is shown in the editor view.

**Setting up a new user in the user management of the controller**

Requirement: the controller has a device user management. You have the corresponding access data.

1. Double-click the controller device object in the device tree.
   ➞ The device editor opens.
2. Click the “Users and Groups” tab.
3. Click [Synchronization] to load the user management configuration from the controller to the editor. If you are not logged in to the device yet, then the “Device User Logon” dialog opens for entering the user name and password.
   ➞ The user management configuration of the device is shown in the editor.
4. In the “Users” view, click “Add”.
   ⇒ The “Add User” dialog opens.

5. Specify the name of the new user and assign the user to a group. This counts as the
   user’s minimum required default group. The user can be assigned to other groups later.
   Define and confirm a “Password” for the user. Define whether the user can change the
   password and whether the user has to change the password at the first login. Click “OK”
   to confirm.
   ⇒ The new user appears in the “Users” view as a new node and in the “Groups” view as
   a new subentry of the selected default group.

Changing of access rights to controller objects in the user management of the controller

Requirement: the controller has a device user management. You have the corresponding
access data.

1. Double-click the controller device object in the device tree.
   ⇒ The device editor opens.

2. Click the “Access Rights” tab.

3. Click (Synchronization) to load the rights management configuration from the controller
to the editor. If you are not logged in to the device yet, then the “Device User Logon”
dialog opens for entering the access data.
   ⇒ The access rights management configuration of the device is shown in the editor.

4. Select the object whose access right you want to change to the left in the object tree.
   ⇒ In the “Rights” view, a table shows the access rights to this object for all configured
   user groups.

5. Double-click the right in the table that you want to change.
   ⇒ If the object has child objects, then a dialog prompts whether you want to modify the
   rights for the child objects.

6. Click “Yes” or “No” to close the prompt.
   ⇒ The rights are switched from “allowed” + to “not allowed” −, or the other way around.
The symbol in the table cell changes accordingly. Explicitly set rights appear in the
  table as green or red symbols. Rights that are inherited from a parent object appear as
  gray symbols.
1. Double-click the controller device object in the device tree.
   ⇒ The device editor opens.
2. Click the “Users and Groups” tab.
3. Click . Select the file (.dum) from the local file system with the saved user management and click “Open” to confirm.
   ⇒ The users and groups described in the file are shown in the editor.
4. Edit the configuration however you like. For example, change the user password or add a new user.
5. Click the “Synchronization” button to transfer the configuration to the device.
   ⇒ A dialog asks you to select the desired operation.
6. Select the “Download the editor content to the device and overwrite the user management there.” option.
   ⇒ The “Device User Logon” dialog opens.
7. Provide valid credentials in order to login to the controller.
   ⇒ After you login successfully, the changes are transferred from the editor to the device. As long as the synchronization is not switched off, CODESYS automatically transfers more changes to the controller that are made in the editor.

See also
- § Chapter 1.3.1.25.1.19.14 “Tab ‘Access Rights’” on page 762
- § Chapter 1.3.1.25.3.13.9 “Dialog ‘Options’ - ‘Device Editor’” on page 1081
- § Chapter 1.3.1.25.1.19.13 “Tab ‘Users and Groups’” on page 760
- § Chapter 1.3.1.8.3 “Configuring devices and I/O mapping” on page 162
- § Chapter 1.3.1.11.2 “Encrypting communication, changing security settings” on page 296

### 1.3.1.11.4 Generating application code

The application code is the machine code that a PLC executes when you start an application. CODESYS automatically generates the application code from the source code that was written in the development system. This is done automatically before downloading the application to the PLC. Before the application code is generated, a test is performed for checking the assignments, the data types, and library availability. In addition, the memory addresses are allocated when the application code is generated.

You can execute this command explicitly by clicking “Build ➔ Generate Code”. This is useful for detecting any errors in your source code, even when the PLC is not connected yet. The errors are printed to the message view (“Build” category).

**NOTICE!**

If you have encrypted the application, then consider the following information: If a (new) boot application generated on request after an online change, then the boot application is formed in the RAM with the current code that is NOT encrypted.
Explicit generation of application code

Requirement: The application can be compiled without any errors.

▷ Click “Build ➔ Generate code”.

⇒ The application code is generated. Detailed information about memory allocation is printed in the message view.

See also

● % Chapter 1.3.1.11.6 “Generating boot applications” on page 305
● % Chapter 1.3.1.25.2.5.3 “Command ‘Generate Code’” on page 917

Messages when generating application code

When you generate application code, CODESYS prints information about memory allocation in the message view. Gaps arise in the memory because reallocation is only for new and changed blocks and variables due to the incremental memory compile. Online changes have the same effect. This fragmentation reduces the amount of available memory. However, you can completely reallocate the memory by clicking “Clean” and therefore increase the amount of free memory.

More information about messages at code generation: Syntax errors and errors that CODESYS detects during code generation and memory allocation appear in the message view in the “Build” category. At each code generation, information is also provided about the code size, the data size (in bytes), the contents of the allocated memory, and the highest address used (byte). It depends on the PLC which data and code is stored in which memory areas. For example, code and data is located in the same area on the CODESYS Control Win V3. For the addresses %I, %M, and %Q, memory is always reserved, even when a variable is not assigned to an address. After cleaning the application, the memory is reassigned completely. In this case, small gaps may result from the predefined alignment (normally 8). Larger gaps result from changing a date without cleaning, for example increasing an array range. In this case, only the affected POUs are rebuilt. Online changes have the same effect. Resulting gaps are reused when possible for other changes. As small gaps can no longer be used at all in some cases, the size of the "largest correlating memory gap" of the affected memory area (in bytes) is issued as meaningful information for the current code generation, as well as its percentage of the total memory.

Please note the options for generating applications.

See also

● % Chapter 1.3.1.25.3.10.9 “Dialog ‘Properties’ - ‘Application Build Options’” on page 1051

Encrypting the application code

See also

● % Chapter 1.3.1.9.18 “Encrypting an application” on page 242

1.3.1.11.5 Downloading the application code, logging in, and starting the PLC

In order to download the source code of your application to the PLC, you must log in to the PLC with application. If there are several applications in the project, then you must switch explicitly to the correct application first.

When you download an application to the PLC, CODESYS performs the following checks:

The list of applications on the PLC is compared with the applications available in the project. If they do not match, then you are prompted to download the application that is not on the PLC yet or delete existing applications.
For "externally implemented" blocks in the application to be downloaded, CODESYS checks whether these are available on the PLC. If they are not available, then the message "unresolved reference(s)" is printed to a dialog prompt and to the message view. Then CODESYS compares the parameters (variables) of the blocks in the application to be downloaded and the parameters of the same-named blocks in the application that exists on the PLC (signature check). If there are any discrepancies, then the message "invalid signature(s)" is printed to a dialog prompt and to the message view.

If the "Download Application Info" check box is selected in the application properties, then additional information about the application contents are downloaded to the PLC.

If multiple applications exist for the same device, then notice that the "I/O Mapping" dialog contains the definition for which of the applications is used for the I/O mapping of the device.

See also
● "Chapter 1.3.1.25.1.1 "Object ‘Application’" on page 710"

**Transferring an application and starting the program**

Requirement: The application contains no errors and the communication settings of the PLC are correct. The application does not exist yet on the PLC: The application and the communication with the controller are not encrypted.

1. Select the required application in the device tree. Skip to Step 3 if you have only one application.
2. Click “Set Active Application”.
   ⇒ The application name appears in bold typeface.
3. Click “Online ➔ Login”.
   ⇒ A dialog prompts you whether the application should be created on the PLC.
4. Click “Yes” to confirm.
   ⇒ The application is downloaded to the PLC.
5. Click “Debug ➔ Start” or press the [F5] key.
   ⇒ The application is running on the controller.

### 1.3.11.6 Generating boot applications

A boot application is the application that is started automatically when the controller is switched on or started. For this to happen, the application on the controller must exist as a file `<application name>.app`.

For each application that is running on the controller, a boot application can also be saved there.

By default, CODESYS generates the boot application automatically when an application is downloaded and transfers them to the PLC. The defaults for generating automatically are located in the “Boot application” category of the application “Properties”. When logging in with a changed application, you are still prompted whether or not to generate a new boot application.

In addition, you can create a (new) boot application at any time in online mode by clicking “Online ➔ Create boot application”.

You can create and save a local copy of a boot application in offline mode as well. Then, you can copy this application to the controller with external tools. In this way, you transfer an application to the controller, even when there is no connection to CODESYS.
Generating boot applications on the controller automatically and explicitly

Requirement: Offline mode; the application is compiled without errors. The connection to the controller is configured and the controller is running. The application is active. The following steps demonstrate the options:

1. Click “Online ➔ Login”.
   ⇒ The boot application file `<application name>.app` is created on the controller with the checksum of the boot application `<application name>.crc`.

2. Click “Online ➔ Create Boot Application” explicitly.
   ⇒ The files on the controller are replaced by new files.

3. Log out.

4. Change the application. Log back in to the controller.
   ⇒ You are prompted whether an online change should be performed. You see the “Update boot application” check box in the same dialog box. This is cleared by default, but this can be changed in the “Boot Application” category of the application “Properties”.

5. Keep the check box cleared and continue login.
   ⇒ A new boot application file is not created.

   ⇒ The boot application that was created above is running on the controller.

You can save the encrypted boot application on the controller. These settings are defined in the “Application Build Options” category of the application “Properties”.

See also
- § Chapter 1.3.1.25.3.10.9 “Dialog ‘Properties’ - ‘Application Build Options’” on page 1051
- § Chapter 1.3.1.25.3.10.2 “Dialog ‘Properties’ - ‘Boot Application’” on page 1047
- § Chapter 1.3.1.14.1 “Executing the online change” on page 355
- § Chapter 1.3.1.25.2.6.4 “Command ‘Create Boot Application’” on page 926

Creating boot applications in offline mode

Requirement: Offline mode; the application is compiled without errors. You want to generate a boot application for an application and save it in the file directory for copying it later to the controller by using external tools (without CODESYS).

1. Click “Online ➔ Create Boot Application”.
   ⇒ A dialog box opens for specifying a save location in the local file system.

2. Click a save path and then click “Save”.

3. If the application has changed since the last boot application was generated, then you are prompted to use a new code for the boot application. In this case, click “Yes”.
   ⇒ The “Save as” dialog box opens.

4. Select a directory and click “Save”.
   ⇒ The boot application file `<application name>.app` is created in the given path.

You are prompted whether or not the build information for the boot application is saved.
5. Click “Yes”.
⇒ The build information is saved to the project directory as a file named `<application
name>.compileinfo`. It is a requirement for a possible online change the next time
the application is updated. Please note: Clicking “Build ➔ Clean” deletes this file.

See also
● § Chapter 1.3.1.25.2.6.4 “Command ‘Create Boot Application’” on page 926

When using CODESYS Control Win V3

When using a CODESYS Control Win V3, the application name must also be included in the
configuration file (*.cfg).

[CmpApp]
Application.1=MyApplication

1.3.11.7 Downloading source code to and from the PLC

CODESYS provides the capability of loading project source code to a PLC as a project archive.
You can then transfer this project archive back to the development system from the PLC as
needed.

Requirement: The connection settings are configured for the affected controllers.

**Downloading source code to the PLC**

1. Choose the command “File ➔ Source Download”.
⇒ The dialog box “Select Device” opens.

2. Select the controller to receive the source code. Click “OK”.
⇒ CODESYS writes the archive file `Archiv.prj` to the controller.

![By choosing the command “Online ➔ Source Download to Connected Device”,
you can load the source code directly to the connected device.]

See also
● § Chapter 1.3.1.25.2.1.11 “Command ‘Source Download’” on page 860
● § Chapter 1.3.1.25.2.6.7 “Command ‘Source Download to Connected Device’” on page 929
● § Chapter 1.3.1.25.3.11.5 “Dialog ‘Project Settings’ – ‘Source Download’” on page 1063

**Loading source code from the PLC**

1. Choose the command “File ➔ Source Upload”.
⇒ The dialog box “Select Device” opens.

2. Select the controller to send the source code. Click “OK”.
⇒ The “Extract Project” dialog box opens.
3. Select the destination directory where you want to extract the project archive. Click “Extract”.
   ⇒ CODESYS extracts the project archive to the directory.

4. Then you are prompted to open the project archive. Click “Yes”.
   ⇒ The project opens.

See also
●  Chapter 1.3.1.25.2.1.10 “Command ‘Source Upload’” on page 859

1.3.1.11.8 Distributing application code to different memory areas

CODESYS provides the capability of storing application code in different memory areas on the controller. The requirement is that you are using a controller that supports this functionality. For these specific controllers, your application automatically gets “POU Locations” object for you to assign different memory areas to your individual POUs.

NOTICE!
Also note manufacturer-specific or controller-specific specifications or information about this functionality.

☐ Requirement: The application does not contain any compile errors.

1. Double-click the “POU Locations” object in the “Devices” view.

2. In the open object, click the Generate Code button.
   ⇒ All program blocks of the application are displayed with the respective object type, current location in the memory, and code size.

3. Double-click the entry in the “Configured Location” column for the POUs that you want to store in another memory area.
   ⇒ A list box opens.

4. Select a memory area from the drop-down list.

5. Click “Clean” and then “Generate Code”.
   ⇒ The POUs are stored at the reconfigured memory locations.

See also
●  Chapter 1.3.1.25.1.3 “Object ‘POU Locations’” on page 713
1.3.12 Testing and debugging

CODESYS provides various options for testing your application and detecting errors. You can start your application in simulation mode, even without connecting any hardware. Using breakpoints and stepping commands, you can examine specific parts of a program. By writing values to variables, you can influence the running program.

Commands are provided that reset your application in various different ways, from resetting only non-persistent variables to completely resetting the controller to factory settings.

See also
- Chapter 1.3.1.12.2 “Using breakpoints” on page 310
- Chapter 1.3.1.12.3 “Stepping through a program” on page 313
- Chapter 1.3.1.12.5 “Resetting applications” on page 318

1.3.12.1 Testing in simulation mode

Use simulation mode for testing and debugging your program when you do not have a physical target device. In this mode, the application is started on a simulated device.

The command is available only when you are logged out.

Requirement: You program contains no errors (compiler error messages or compile errors) and you are not logged in.

1. Activate simulation mode as follows:
   - Click “Online ➔ Simulation”, or
   - Right-click the controller in the device tree and click “Simulation”.
     - The name of the controller in the device tree is displayed in italics. In the status line, “Simulation” appears highlighted in red. The “Simulation” command is selected in the main menu.

2. Click “Online ➔ Login”.

3. When logging in with the active application, you will be prompted whether the application “Sim.<device name>.<application name>” should be created and loaded. Click “Yes” to confirm.
   - The application is logged onto the PLC.

4. Now you can check and correct the program flow with the commands provided in the main menu in “Debug”.

5. Log out from the controller and end the simulation mode.

See also
- Chapter 1.3.1.25.2.6.21 “Command ‘Simulation’ on page 937

Limitations
- The focus of the simulation mode is testing and debugging your program of the PLC. That means the functionality of the simulated PLC is limited. Keep in mind that some POUs have no function. They are not creating any compile or download errors, they will simple not work.
- Without an extra available “Virtual Commissioning” license the “Online mode” of the simulated PLC is limited to 2 hours. After 2 hours starting from the “Login”, the “Online mode” is automatically terminated and the PLC is logged out.
- It is not possible to create a “Boot Application” in the simulated PLC. Every “Login” starts with an empty simulated PLC and a download of the application is required.
- When logging in to a simulated PLC the first time a “Windows Security Alert” is displayed. Depending on the application, e.g. if any network communication is implemented, it might be necessary to allow the “Virtual AC500” to communicate on one or multiple network types.
1.3.1.12.2 Using breakpoints

Breakpoints are commonly used for debugging programs. CODESYS supports breakpoints in all IEC editors.

You can set breakpoints at specific positions in the program to force an execution stop and to monitor variable values. You can set special data breakpoints to halt program execution when the value of a specific variable changes.

The halt at a breakpoint or data breakpoint can be linked to additional conditions. You can also redefine breakpoints and data breakpoints as execution points where specific code is executed instead of stopping the program.

The “Breakpoints” view provides an overview of all defined breakpoints. It also includes additional commands for processing batch changes to multiple breakpoints.

In the editor, the following symbols identify the status of a breakpoint or execution point:

- Breakpoint enabled
- Breakpoint disabled
- Breakpoint is set in another instance of the POU open in the editor.
- Halted at breakpoint
- Conditional breakpoint enabled
- Conditional breakpoint disabled
- Execution point enabled
- Execution point disabled
- Conditional execution point enabled
- Conditional execution point disabled
- Data breakpoint enabled
- Data breakpoint disabled
- Halted at data breakpoint
- Data execution point enabled
- Data execution point disabled
- Halted at data execution point
- Conditional data execution point enabled
- Conditional data breakpoint enabled

See also

- Chapter 1.3.1.25.2.3.12 “Command ‘Breakpoints’” on page 885

Whether or not data breakpoints can be set and how many depends on the target system.

Program execution stops at a data breakpoint when the value of a particular variable or memory address changes. As with ordinary breakpoints, the halt can be linked to an additional condition, or specific code can be processed instead of the halt (converted to a data execution point).

You set a data breakpoint either by means of the “New Data Breakpoint” command in the “Debug” menu or by means of the “New” button in the “Breakpoints” view. You specify a qualified variable name or a memory address directly which is to be monitored for changes in its value.
Example

In the following sample code, the memory of the variable `iNumber` is overwritten unintentionally. However, a data breakpoint at the variable `iNumber` will detect when its value changes. The processing then stops with a corresponding message at the array access, which overwrites the variable value: `Idx = 7`. See also below: "Setting a data breakpoint".

```plaintext
PROGRAM PLC_PRG
VAR
  Idx : INT;
  Ary : ARRAY[0..3] OF BYTE;
  iNumber : INT := 55;
END_VAR
FOR idx := 0 TO 6 DO
  Ary[idx] := 0;
END_FOR
```

Breakpoints in applications with multiple tasks

Basically, debugging is not possible for multiple tasks at the same time. While you are working on a task with breakpoints or stepping, breakpoints are ignored in other tasks.

If a block containing a breakpoint is used by multiple tasks, then only the debug task is halted because it arrives at the breakpoint first. All other tasks continue. The “Call Stack” dialog shows which task is currently halted.

If you need a breakpoint to affect only one specific task, then you can define this in the breakpoint properties.

Breakpoints operate separately for each application so that a "HALT ON BP" does not affect any other applications. This applies also to parent/child applications, even if the breakpoint is set in a block that is used by several applications and whose code is located only once on the PLC.

NOTICE!

The I/Os that are called by the debug task are not updated at a halt in the breakpoint, even if you select the “Refresh I/Os in Stop” check box in the PLC settings.

If the application stops at a breakpoint on the PLC, then an online change or download causes all tasks to halt which means the PLC will stop. In this case, CODESYS prompts you whether or not to continue with the login.

See also

- Chapter 1.3.1.25.2.3.15 “Command ‘Call Stack’” on page 888

Setting a single breakpoint (example in ST editor)

Requirement: The application is in online mode and running. The operating mode is “Debug”.

1. In the editor, open a POU programmed in structured text (ST).
2. Place the cursor in the line where a breakpoint will be set.
3. Click “Debug ➔ Toggle Breakpoint” or press [F9].
   - The line is marked in red and identified by the "breakpoint enabled" symbol ( ● ). If the program is halted at the breakpoint, then the line is marked by the "stop at breakpoint" symbol ( ● ). Program processing is stopped and identified in the status line by the red-highlighted HALT ON BP status.
4. Click “Debug ➔ Start” or press [F5].
   - The program continues.
5. Set more breakpoints and check the variable values at the break position.
6. Place the cursor in the line where a breakpoint should be removed.
7. Click “Debug ➔ Toggle Breakpoint” or press [F9].
   ⇒ The marking disappears. The breakpoint is deleted.

See also
● § Chapter 1.3.1.25.2.7.9 “Command ‘Toggle Breakpoint’” on page 940

Defining a breakpoint condition (example in ST editor)

1. In the editor, open a POU programmed in structured text (ST).
2. Place the cursor in the line where a breakpoint will be set.
3. Click “Debug ➔ New Breakpoint”.
   ⇒ The “Breakpoint Properties” dialog opens.
4. Click the “Condition” tab.
5. Click “Break when the hit count is a multiple of” in the “Hit Count” list box.
   Type the value "5" in the field to the right.
6. In addition, you can define a Boolean condition for when the breakpoint should be active.
   Select the “Break, if true” check box. Type a Boolean variable in the text field to the right.
7. Select the “Enable breakpoint immediately” check box.
8. Close the dialog.
   ⇒ The line is marked red and identified by the "conditional breakpoint enabled" symbol (○)

Monitor the running program. As long as the Boolean variable for the condition is FALSE, the breakpoint condition is not fulfilled and the program continues to run. If you set the variable to TRUE, then the condition is fulfilled and the program halts at the breakpoint every 5th pass.

See also
● § Chapter 1.3.1.25.3.5 “Dialog ‘Breakpoint Properties’” on page 1041

Defining an execution point (example in ST editor)

1. In the editor, open a POU programmed in structured text (ST).
2. Place the cursor at the position for an execution point.
3. Click “Debug ➔ New Breakpoint”.
   ⇒ The “Breakpoint Properties” dialog opens.
4. Click the “Execution Point Settings” tab.
5. Select the “Execution point” check box.
   In the “Execute the following code” field, type the following statement: iCounter := iCounter + 1;
   In the “Print a message in the device log” field, type the following text: Execution point reached {iCounter}
6. Close the dialog.

When the program reaches the execution point, it does not halt, but executes the code defined above. In addition, a message is issued to the device log.

See also

● Chapter 1.3.1.25.3.5 “Dialog ‘Breakpoint Properties’” on page 1041

### Setting a data breakpoint

**Requirement:** The application is in online mode and running.

1. Click “View ➔ Breakpoints”.
2. Click “Debug ➔ New Data Breakpoint”.
3. Click the button in the “New breakpoint” dialog (“Data” tab).
4. In the “Input assistant” dialog (“Watch Variables” tab), select the variables for which the program should halt when changed.
   As an alternative, specify the qualified name of the variable on the “Data” tab directly in the input line. Example: PLC_PRG.iNumber. The exact number of bytes to be monitored is specified as the “Size”. A value that corresponds to the data type is set here automatically by default. You can also specify fewer bytes to be monitored.
5. In the “Breakpoints” view, select the line with the data breakpoint and click the button.
   The line is marked and identified by the “Data breakpoint enabled” symbol (●). When the program reaches the data breakpoint (meaning when the value of the selected variables changes), the program processing halts. In the implementation part of the POU, the next line is identified by an arrow (●). This is identified in the status line by the **HALT ON BP** status highlighted in red.
6. Click “Debug ➔ Start” or press [F5].
   The program continues running and halts again when the value of the variables changes again.

See also

● Chapter 1.3.1.25.3.5 “Dialog ‘Breakpoint Properties’” on page 1041

### 1.3.1.12.3 Stepping through a program

You can step through an application and navigate the code. This is useful to check the status of your code at runtime. You can examine the call process, track variable values, or locate errors.

Stepping commands are provided in the “Debug” menu for this purpose. The commands become available when you set breakpoints in online mode and then halt execution at a breakpoint: the application is in “HALT ON BP” state (debug mode). During debug mode, the current break position is highlighted in yellow and marked with the ● symbol in the text editors.
1. Download your application to a controller.
   - The application is highlighted in green. CODESYS and the editors of the POUs are in online mode.

2. In the POUs, set breakpoints at the locations in the code that you want to examine.
   - All breakpoints are listed in the “Breakpoints” view.

3. Start the application.
   - The application starts and the code is processed until the first breakpoint.

   Now the application is in debug mode. In the device tree, the application is labeled with “[halt on breakpoint]”. The status bar provides information about the operating state:

   ![Halt on BP]

   The editor was opened at the current break position. The line of code with an active breakpoint where program execution was halted is highlighted in yellow and marked by the ⚫ symbol. This statement highlighted in yellow has not been executed yet.

   ![Highlighted Code]  

   Now you can select the various stepping commands or display the call tree.

   ![Step Commands]

   Alternatively, you can first start the application and then set a breakpoint.
**Behavior of the stepping command in the 'Debug' menu**

- **Command “Step Over”**
  The statement at the breakpoint position is executed. Program execution halts before the next statement in the POU.
  If the statement contains a call (from a program, function block instance, function, method, or action), then the subordinate POU is processed completely in one step.

- **Command “Step Into”**
  The statement at the breakpoint position is executed. Program execution halts before the next statement.
  If the statement contains a call (from a program, function block instance, function, method, or action), then the program execution jumps to this subordinate POU. The first statement there is executed and the program execution halts before the next statement. The new current breakpoint position is then in the called POU.

- **Command “Step Out”**
  The command executes the POU from the current breakpoint position to the end of the POU and then jumps back to the calling POU. Program execution halts at the calling position (in the line with the call).
  If the current breakpoint position is in the main program, then the POU is run through to the end. Then the program execution jumps back to the beginning (to the program start at the first line of code in the POU) and halts there.

- **Command “Run to Cursor”**
  First set the cursor at any line of code and then execute the command. The program is executed from the current breakpoint position and halts at the current cursor position without executing the code of this line.

- **Command “Set Next Statement”**
  First set the cursor at any line of code (also before the current breakpoint position) and then execute the command. The statement marked with the cursor is executed next. All statements in between are ignored and skipped.

- **Command “Show Next Statement”**
  If you do not see the current breakpoint position, then execute the command. Then the window with the current breakpoint position comes into focus and the breakpoint position is visible.

Click “View ➔ Call Stack” to completely show the previous call tree for the breakpoint position currently reached in the program processing.

![Info](The “Call Stack” view shows the location of the block in the call structure of the program at all times, even before compiling the application.)

See also

- % Chapter 1.3.1.25.2.7.11 “Command ‘Step Into’” on page 941
- % Chapter 1.3.1.25.2.7.10 “Command ‘Step Over’” on page 941
- % Chapter 1.3.1.25.2.7.12 “Command ‘Step Out’” on page 941
- % Chapter 1.3.1.25.2.7.13 “Command ‘Run to Cursor’” on page 942
- % Chapter 1.3.1.25.2.7.14 “Command ‘Set Next Statement’” on page 942
- % Chapter 1.3.1.25.2.7.15 “Command ‘Show Next Statement’” on page 942
- % Chapter 1.3.1.25.2.3.16 “Command ‘Call tree’” on page 889

### 1.3.1.12.4 Forcing and writing variables

In CODESYS, variable values in the PLC can be changed in online mode. Here we make a distinction between forcing and writing.

When forcing, CODESYS writes the value in each cycle to hold the variable permanently at the forced value.
**NOTICE!**
Please note that forced values must be lifted explicitly by the user. A dialog will appear if forced variables still exist when logging out.

When writing, CODESYS sets the value one time. Therefore, the value can be overwritten by the program at any time.

### Forcing in the declaration part

**Requirement:** Your application includes a POU with declarations. The application is in online mode.

1. Open the POU in the editor by choosing the command “Project ➔ Edit Object”.
2. In the declaration part of the editor, double-click in column (1) “Prepared value” of a variable.
   - The field can be edited and a value can be entered.

3. Perform step 2 for other variables.
4. Click “Debug ➔ Force Values”.
   - The variable values are overwritten with the prepared values. The values are marked with the symbol 🟢.

### Forcing in the implementation part

**Requirement:** The application is in online mode.

1. Open the POU in the editor by choosing the command “Project ➔ Edit Object”.
2. In the implementation part of the editor, double-click an inline monitoring field (1).
   - The dialog “Prepare Value” opens.
3. Enter the new value in the field “Prepare a new value for the next write or force operation”. The prepared value appears in the inline monitoring field.

4. Click “Debug \(\Rightarrow\) Force Values”. The value of the variables is overwritten with the prepared values. The values are marked with the symbol \(\Box\).

---

**View and edit all forced variables**

**1 list**

**Requirement:** The application is in online mode. Several variables are forced.

1. Click “View \(\Rightarrow\) Watch \(\Rightarrow\) Watch All Forces”. The “Watch all forces” view opens. It contains all currently forced variables of the application in the form of a watch list.

2. Select all lines in the list and click “Unforce \(\Rightarrow\) Unforce and Keep All Selected Values” in the drop-down list in the upper left part of the view. The variables are unforced and they get the values that they had before forcing.

---

**Forcing a function block input in CFC**

**Requirement:** An application has a CFC POU that contains a function block, and the application is in offline mode.

---

**NOTICE!**

This kind of forcing uses a data breakpoint internally and is therefore different from forcing with the “Force Values” command or \([F7]\). Values that were forced by the command “Force FB Input” do not respond to the commands “Show All Forces” or “Unforce Values”.

---

1. Open the editor of the CFC POU by double-clicking the object in the tree.
2. Activate the forceability for the desired function block. Select the POU element in CFC and click “CFC \(\Rightarrow\) Prepare POU Element for Forcing”.
3. Login to the application on the target device. In CFC, select the input of the POU and click “Force FB Input” in the context menu. The “Force Value” dialog opens.
4. Set a new value for the input. Example in the case of a TON POU: FALSE for the Boolean input \(\text{IN}\), or \(t\#4s\) for the PT input (TIME). Click “OK” to confirm. The set value is forced immediately. A green circle is displayed at the upper left of the POU element and the name of the input in the element is highlighted in green. For a Boolean value, a small monitoring view with the value also opens at the input. In the monitoring views, the forced value is displayed, for example in the “Value” column, as in the declaration part.
5. To remove the forced value, click “Force FB Input” again. In the “Force Value” dialog, select the “Remove value” option. Forcing is canceled. The input gets the current value from the controller.
1.3.12.5 Resetting applications

Resetting the application stops the program and resets the variables to their initialization values. Depending on the type of reset, retain variables and persistent variables are also reset.

- Reset warm: All variables are reset, except `RETAIN` and `PERSISTENT` variables.
- Reset cold: All variables are reset, except `PERSISTENT` variables.
- Reset origin: All variables are reset.
- Reset origin device: All variables are reset and all applications are deleted.

The following sample program and statements clarify the functionality of the various resets.

See also

- "Lifespan of variables when calling online commands" on page 251

Sample program

Example

Declaraiton

```plaintext
VAR
  iVar: INT := 0;
END_VAR
VAR RETAIN
  iVarRetain: INT := 0;
END_VAR
VAR PERSISTENT
  iVarPersistent : INT := 0;
END_VAR
iVar := 100;
iVarRetain := 200;
iVarPersistent := 300;
```

Implementation

1. Insert the "Persistent Variables" object below the application and open it in the editor.
2. Click "Build ➔ Build".
3. Click "Declare ➔ Add All Instance Paths".
   » The instance path of the persistent variables is inserted.
4. Download the application to the controller.
Requirement: The sample program runs on the controller.

1. Click “Online ➔ Login” to switch to online mode.
2. Monitor the variables iVar, iVarRetain, and iVarPersistent.
3. Click “Online ➔ Reset Warm”.  
   ⇔ You are prompted whether you really want to execute the command.
4. Click “Yes” to confirm the dialog.
   ⇔ The application is reset. The iVar variable is set to the initialization value 0. Both of the other variables retain their values.
5. Click “Online ➔ Reset Cold”.  
   ⇔ You are prompted whether you really want to execute the command.
6. Click “Yes” to confirm the dialog.
   ⇔ The application is reset. The iVar and iVarRetain variables are set to the initialization value 0. The iVarPersistent variable retains its value.
7. Click “Online ➔ Reset Origin”.  
   ⇔ You are prompted whether you really want to execute the command.
8. Click “Yes” to confirm the dialog.  
   ⇔ The application is reset. All variables are reset to their initialization values.

See also
- Chapter 1.3.1.25.2.6.10 “Command ‘Reset Warm’” on page 931
- Chapter 1.3.1.25.2.6.9 “Command ‘Reset Cold’” on page 931
- Chapter 1.3.1.25.2.6.11 “Command ‘Reset Origin’” on page 932
- Chapter 1.3.1.25.2.6.12 “Command ‘Reset Origin Device’” on page 933

1.3.1.12.6 Flow control

With flow control, you can monitor the processing of the application program. Flow control is provided for the ST, FBD, LD, and IL language editors.

With an activated flow control, CODESYS displays the variable values and results from function calls and operations at the respective processing location and time. In this way, the exact lines of code and networks that run through the current cycle are marked in colors. Compare this to standard monitoring, in which CODESYS delivers only the value that a variable has between two processing cycles.

Flow control works in all parts of the editor view that are currently visible. “Flow control enabled” is then displayed in the status line as long as the function is active and flow control positions (processed parts of code) are visible in an editor view.

You can write and force values only in the declaration section.
NOTICE!

When you enable flow control, the cycle time of the application is prolonged.

When “Confirmed Online Mode” is activated in the communication settings, a dialog prompt appears when switching on the flow control to cancel the process.

When flow control is activated, it is not possible to use breakpoints or step through the program.

See also

- § Chapter 1.3.1.25.1.19.2 “Tab ’Communication Settings’” on page 741
- § Chapter 1.3.1.25.2.7.22 “Command ’Flow Control’” on page 947

Display of the flow control in different language editors

By default, CODESYS displays the flow control positions of the processed parts of code as green fields. Unprocessed parts of code are displayed in white.

Please note that the displayed value of an unprocessed code position is an ordinary monitoring value. This is the value between two task cycles.

```
1     i 1619 := i 1619 + 1;
2     b[0] := NOT b[0];
3     IF str "abcdefghj" = str: "" THEN
4     fi2[1.5] := fi1[1.23] &;
5     ELSE
6     fi2[1.5] := -1.5;
8     END_IF:
9     IF D[6.5E+04] <= 0.0 THEN
```

In network editors, CODESYS marks the processed networks with bars on the left edge in the flow control color.

In LD, CODESYS displays the currently processed connecting lines in green and all others in gray. The actual value of the connection is also displayed: TRUE by a bold blue line, FALSE by a bold black line, and unknown or analog values by thin black lines. Combinations of these lines are displayed as dashed lines.
In IL, CODESYS uses two fields for each statement to display the current values: one to the left of the operator with the current accumulator, and one to the right of the operands with the operand value.
1.3.1.12.7 Determining the current processing position with the call stack

You can use the call stack for determining the current position of the program flow. This function is very useful when stepping into programs.

Requirement: The application is in online mode. The program is halted at a breakpoint or you are stepping into it.

▷ Open the call stack by clicking “View ➔ Call Stack”.

▷ The call stack opens. The list shows the current location with the complete call path.

The call stack is also available in offline mode and normal online mode (without using debugging functions). In this case, it receives the last displayed location during a stepped execution, but it is displayed in gray.

See also
- § Chapter 1.3.1.25.2.3.15 “Command ‘Call Stack’” on page 888
- § Chapter 1.3.1.12.3 “Stepping through a program” on page 313

1.3.1.12.8 Checking the task deployment

The Tab “Task deployment” of the device editor indicates in an overview the tasks that process the individual inputs and outputs of the I/O mapping of your application and the priority with which they do so. You can check here whether an unintentional overwriting of values is caused, which can lead to undefined values.

1. Generate code for the application: to do this select, for example, the command “Build ➔ Generate Code”.
2. Open the device editor by double-clicking on the device object in the device tree. Select the “Task deployment” tab.

▷ You obtain a display of the inputs and outputs of your application and the assignment of the tasks and their priorities. See the description of the “Task deployment” tab for details.

See also
- § Chapter 1.3.1.25.1.19.16 “Tab ‘Task deployment’” on page 767
- § Chapter 1.3.1.8.1 “Device tree and device editor” on page 155
- § “General information about I/O mapping” on page 163

1.3.1.13 Application at runtime

1.3.1.13.1 Monitoring of values................................................................. 323
1.3.1.13.2 Changing values with recipes............................................... 331
1.3.1.13.3 Data recording with trace...................................................... 335
1.3.1.13.4 Data recording with trend..................................................... 344
1.3.1.13.5 Monitoring tasks..................................................................... 349
1.3.1.13.6 Reading the PLC log............................................................... 349
1.3.1.13.7 Analysing errors with core dump........................................... 349
1.3.1.13.8 Using PLC shell for requesting information........................... 351
1.3.1.13.9 PLC operation control via system variables............................. 352
1.3.1.13.10 Backup and restore............................................................... 353
When the application is running on the PLC, in the CODESYS Development System there are some features for monitoring and changing the values of the variables as well as for recording and storing the value charts.

Furthermore, you can poll some information from the PLC, you can have a look into the PLC-log, display a core dump and monitor the time behaviour of the tasks.

Regard also the possibility to restrict the access on the running application in critical states of the machine via online commands provided by CODESYS Development System. For this purpose some system variables are available in a module of the \texttt{ComponentManager} library.

### 1.3.1.13.1 Monitoring of values

In runtime mode, you can monitor the current variable values of a programming object at different places in a project. The following is what we refer to as monitoring:

- Online view of the programming editor of an object: inline monitoring
- Online view of the declaration editor of an object
- Object-independent, configurable watchlists

When you set the \{attribute `monitoring`\...\} pragma, you can monitor the results from function calls and the current variable values in property-type objects.

More options for recording current variable values:

- Read and save recipes
- Record values on a timeline for displaying the history immediately or later: trace and trend features

See also

- Chapter 1.3.1.24.6.2.21 “Attribute ‘monitoring’” on page 619
- Chapter 1.3.1.13.1.2 “Using watch lists” on page 330
- Chapter 1.3.1.13.2 “Changing values with recipes” on page 331
- Chapter 1.3.1.13.3 “Data recording with trace” on page 335
- Chapter 1.3.1.13.4 “Data recording with trend” on page 344

### Calling of monitoring in programming objects

When an application is running on the controller, the actual values of variables are displayed in the editors of the POU.s. This is how the values of variables are monitored.

- **Requirement:** The “Enable inline monitoring” option is activated in “Tools \rightarrow Options” in the “Text Editor” category on the “Monitoring” tab.

1. Download an application to the controller and start it.
2. Click “Debug \rightarrow Display Mode \rightarrow Decimal”.
   - The display format of the actual values is set.
3. Click a programming object in the “Devices” view or “POUs” view.
   - The respective editor opens. Actual values of the variables are refreshed continually for both the declaration and implementation.
The actual value of an expression (1) is displayed in the “Value” column (3).
You can write and force a value in the “Prepared Value” (4) column. During the forcing, the actual value is decorated with a red symbol (5).

The expression of an interface reference can be expanded. If the interface points to a global instance, then this global instance is displayed as the first entry below the reference. Afterwards, if the interface reference changes, then the displayed reference is collapsed.

Inline monitoring is the display of the current variable value in the implementation. Depending on the implementation language, the following displays are possible in the implementation part:

- Variables have a window with the current value displayed after their name: `nResult2`. If you have prepared values for variables for forcing or writing, then they are displayed in angle brackets in the inline monitoring view after the current value. After forcing, the respective values are identified by the symbol.
- Network editors and the CFC editor:
  Connecting lines are displayed in color according to their actual Boolean value (blue means TRUE, black means FALSE).
- LD editor:
  The contact and coil elements are also marked.
  For contacts and coils, a prepared value (TRUE or FALSE) is shown in a small view next to the element.
- SFC editor:
  Transitions with the value TRUE are displayed in color according to their actual Boolean value (blue means TRUE, black means FALSE).
  Active steps are displayed in blue.
  Forced transition values are displayed in red in the implementation.
- IL tabular editor:
  Current values are displayed in a separate column.
Monitoring in the LD editor

Monitoring in the SFC editor

You can deactivate the inline monitoring function in “Tools ⇒ Options”, in the “Text Editor” category on the “Monitoring” tab.

See also

- Chapter 1.3.1.24.1.3.1 “ST editor” on page 430
- Chapter 1.3.1.24.1.3.2 “ST editor in online mode” on page 431
- Chapter 1.3.1.24.1.5.2 “FBD/LD/IL editor in online mode” on page 465
- Chapter 1.3.1.24.1.4.2 “SFC editor in online mode” on page 443
- Chapter 1.3.1.24.1.6.3 “CFC editor in online mode” on page 480

Partial monitoring of an array

Limiting the monitoring range

An expanded array shows the actual values for up to 1000 elements. However, this can be confusing. In addition, an array can contain more than 1000 elements. Then it is helpful to limit the range of displayed elements. You can do this in online mode in the following way.

Requirement: An application is running. It contains a multidimensional array variable with more than 1000 elements. Example: `arrBig : ARRAY [0..100, -9..10, -19..20] OF INT;`

1. Click in the field of the “Data Type” column for the `arrBig` variable.
   - The “Monitoring Range” dialog opens.

2. Specify the value `[1, -9, -19]` for “Start”.

3. Specify the value \([1, 10, 20]\) for "End".

\[\Rightarrow\] The actual values of 800 array elements are displayed in the declaration editor. The range is limited to the elements of the index \([1, \langle i \rangle, \langle j \rangle]\) with \(i\) from -9 to 10 and \(j\) from -19 to 20.

See also
- Chapter 1.3.1.24.1.1 “Declaration editor” on page 428
- Chapter 1.3.1.12.4 “Forcing and writing variables” on page 315
- Chapter 1.3.1.25.3.9 “Dialog 'Monitoring Range'” on page 1046

Monitoring a function block
When you double-click the editor view of a function block in online mode, a dialog opens where you can choose between viewing the basic implementation or a specific instance.

If you select the basic implementation, then the code is displayed in the editor without current values. Now set a breakpoint in the basic implementation. If the execution halts there, then the current values of the instance that is processed first in the program flow are displayed. Now you can step successively through all instances.

If you select one of the instances, then the editor opens with the code of the function block instance. The current values are displayed in the declaration and, if applicable, in the implementation, and are updated continuously.

See also
- Chapter 1.3.1.25.1.28.2 “Object 'Function Block’” on page 778
- Chapter 1.3.1.12.2 “Using breakpoints” on page 310

Monitoring a property
You can monitor variables in a property object by setting a breakpoint in the function during online mode. When halted there, the current values are displayed.

In addition to your own values, the values of the variables of the superordinate instance are displayed automatically. In the declaration part of the property, the THIS pointer, which points to the superordinate instance, appears in the first line with the current data type information and values.
FUNCTION_BLOCK FB_BaseAlfa
VAR
  iBaseLocal : INT;
  sBaseLocal : STRING;
END_VAR

iBaseLocal := iBaseLocal + 1;
sBaseLocal := 'Testing Text';

FB_BaseAlfa.PorpBeta.Get
iBaseLocal := iBaseLocal + 1;
IF _iBaseLocal > 0 THEN
  PropBeta := TRUE;
END_IF

FB_BaseAlfa.PorpBeta.Set
IF PropBeta = TRUE THEN
  iBaseLocal := 0;
  sBaseLocal := 'Tested IF';
END_IF

PROGRAM PLC_PRG
VAR
  fb_BaseAlfa : FB_BaseAlfa;
END_VAR

fb_BaseAlfa();

IF fb_BaseAlfa.PropBeta = TRUE THEN
  xResult := TRUE;
END_IF
IF xReset THEN
  fb_BaseAlfa.PropBeta := TRUE;
  xReset := FALSE;
END_IF
Monitoring of property access in the superordinate program object

You can monitor the values of subordinate properties in a function block or program in addition to the variable values.

To do this, add either the pragma

\[
\text{attribute 'monitoring' = 'variable'} \quad \text{or} \quad \text{attribute 'monitoring' = 'call'}
\]

To the subordinate property object in the declaration. If you open the superordinate program instance or function block instance at runtime, then the current property values are displayed in the editor in addition to the current variable values.

See also

- § Chapter 1.3.1.25.1.28.8 “Object 'Property’” on page 791
- § Chapter 1.3.1.24.6.2.21 “Attribute 'monitoring’” on page 619

Monitoring a method

You can monitor variables in a method object by setting a breakpoint in the method during online mode. When halted there, the current values are displayed.

In addition to your own values, the values of the variables of the superordinate instance are displayed automatically. In the declaration part of the method, the \textit{THIS} pointer, which points to the superordinate instance, appears in the first line with the current data type information and values.
FUNCTION_BLOCK FB_BaseAlfa
VAR
  iBaseLocal : INT;
  sBaseLocal : STRING;
END_VAR
iBaseLocal := iBaseLocal + 1;
sBaseLocal := 'Testing Text';

METHOD MethBaseAlfa : BOOL // Method of FB_BaseAlfa
VAR_INPUT
END_VAR
VAR
  iMethLocal : INT;
END_VAR
iMethLocal := iMethLocal + 1;

PROGRAM PLC_PRG
VAR
  fb_BaseAlfa : FB_BaseAlfa;
END_VAR
fb_BaseAlfa();
fb_BaseAlfa.MethBaseAlfa();

See also
- Chapter 1.3.1.25.1.28.8 “Object ‘Property’” on page 791
- Chapter 1.3.1.25.1.28.5 “Object ‘Method’” on page 784

Monitoring a function
You can monitor variables in a function object by setting a breakpoint in the function during online mode. When halted there, the current values are displayed.

Monitoring the return value of a function call
In the ST editor of a POU, the current return value is displayed as inline monitoring at the position of the POU where a function is called.

The following conditions must be fulfilled:
- The value can be interpreted as a 4-byte numeric value. Example: INT, SINT, or LINT.
- The pragma `{attribute 'monitoring' := 'call'}` is inserted into the function.
Using watch lists

What is a watch list?
A watch list is a user-defined list of project variables that are collected in one view for the purpose of monitoring their values. In online mode, you can write and force variable values in a watch list. Monitoring, writing, and forcing are handled the same way as the declaration editor in online mode. You can customize the format of the representation of floating-point values in the options for monitoring.

There are four, ready-to-use watch lists (Watch <n>) available in a project. Click “View ➔ Watch”.

If the expression is an interface reference, then it can be expanded. If the interface points to a global instance, then this global instance is displayed as the first entry below the reference. If the interface reference changes, then the displayed reference is collapsed.

Creating and editing a watch list (offline or online mode)

Requirement: The project is in either online or offline mode. It includes an application with declared variables that you want added to one of the four possible watch lists.

1. Click “View ➔ Watch ➔ Watch <n>”.
   ⇒ The Watch <n> view opens. It contains a blank table row.
2. Double-click the field in the “Expression” column and type a variable to monitor, either manually or with the input assistant.
   Syntax: <device name>.<application name>.<object name>.<variable name>
   Example: "Dev1.App1.PLC_PRG.ivar"
   If you type the name of a structured variable, then the individual components are displayed automatically in other lines in online mode.
3. Define all successive variables that will be monitored with this list. You can change the order by using drag and drop operations.
   ⇒ The “Execution point”, “Type”, “Address”, “Comment” fields are filled in automatically according to the variables declaration. The symbol before the expression indicates the type of variable: input variable ( ), output variable ( ), or ordinary variable ( ).

   In online mode, you can also create or edit watch lists by right-clicking and choosing the “Add Watch” command.

See also
● ☰ Chapter 1.3.1.24.1.1 “Declaration editor” on page 428
● ☰ Chapter 1.3.1.25.3.13.14 “Dialog ‘Options’ - ‘Monitoring’” on page 1083

See also
● ☰ Chapter 1.3.1.25.2.3.8 “Command 'Watch' - 'Watch <n>’” on page 883
Adding variables by choosing the 'Add Watch' command (online mode)

Requirement: A project is open and running. It includes an application with declared variables that you want added to a possible watch list.

1. Click “View ➔ Watch <n>” to open the watch list.
2. Place the cursor on a variable in the declaration or implementation part of a POU and right-click to choose the “Add Watch” command.
   ⇒ This adds an entry to the list for the selected variable.
3. You can add other variables in this way or by typing directly into the list in the “Expression” field as described above.
   ⇒ The watch lists are updated immediately.

If a watch list is not open when you click “Add Watch” for a variable, then it is added automatically to the “Watch 1” list.

Writing and forcing variable values is also possible in the watch lists. In online mode, the “Prepared value” column is also available.

See also
- § Chapter 1.3.1.25.2.23.1 “Command ‘Add Watch’” on page 1037
- § Chapter 1.3.1.13.1.1 “Calling of monitoring in programming objects” on page 323
- § Chapter 1.3.1.12.4 “Forcing and writing variables” on page 315

1.3.1.13.2 Changing values with recipes

Use recipes to change or read recipes values for a specific set of variables (recipe definition) on the controller at the same time.

You define the basic settings for recipes, such as location and format, in the object “Recipe Manager”. Insert one or more recipe definitions below this object. A recipe definition comprises one or more recipes for the contained variable. The recipe consists of specific variable values.

You can save a recipe to a file or write directly from files to the PLC.

Recipes can be loaded via the CODESYS development interface, the visualization element, or the application program.

Using recipes on remote devices

The variable values from recipes are transferred automatically to and from another controller when they are data source variables and a data source exchange is configured. Reading and writing occurs synchronously. Therefore, CODESYS updates all variables in a recipe at the same time. After reading or writing, you can use the call `g_RecipeManager.LastError` to check whether or not the transfer was successful (`g_RecipeManager.LastError = 0`).
Using recipes in the CODESYS user interface

The CODESYS development interface provides commands for generating recipes as well as read/write in online mode.

See also
- Chapter 1.3.1.25.2.19 “Menu ‘Recipes’” on page 1015

Using recipes in applications

When the application is in runtime mode, you can use recipes in the user program and visualization elements.

In the user program, you use the methods for the function block RecipeManCommands from the library RecipeManagement. In the visualization, you use recipes via the input configuration (internal command of visualization elements).

During the initialization process, the recipe management reads the values of the variables that are defined in the recipe definition. This operation takes place at the end of the initialization phase of the application. At this point, all initial values of the application variables are set. This is performed to initialize missing values from recipe files correctly.

See also
- RecipeManCommands
- Input configuration Chapter 1.3.5.19.3.6 “Dialog ‘Input Configuration’” on page 1586

Creating recipes

1. Select the “Application” object in the device tree.
2. Click “Project ➔ Add Object ➔ Recipe Manager”.
   ➔ CODESYS adds the Recipe Manager to the device tree.
3. Select the “Recipe Manager” object in the device tree.
4. Click “Project ➔ Add Object ➔ Recipe Definition”.
   ➔ CODESYS adds the recipe definition below the Recipe Manager.
5. Open the editor of the recipe definition by double-clicking the object.
6. Double-click the blank field below “Variable”. Specify the name of a variable that you will define a recipe. The input assistant can be used for this: button.
7. Click “Recipes ➔ Add a New Recipe” and enter a name for the new recipe.
   ➔ A column with the new recipe name appears in the editor.
8. Enter the variable value for this recipe in this field.
9. Insert additional fields as needed.
10. Select a variable value for the recipe and click “Recipes ➔ Save Recipe”. Select a location and file name.
   ➔ CODESYS saves the recipe in the format as defined in the Recipe Manager.
Requirement: A Recipe Manager is available in the application. In a recipe definition, there is a "myRec" recipe with variable values. A myRec.txt recipe file is located on the file system and contains the entries for this recipe.

Example of the recipe file:

```
PLC_PRG.bVar:=0
PLC_PRG.iVar:=2
PLC_PRG.dwVar:=35232
PLC_PRG.stVar:='first'
PLC_PRG.wstVar:='123443245'
```

1. Double-click the "Recipe Definition" object in the device tree to open the tabular editor for the definition of the individual recipes.
   ⇒ You see the myRec column with the current values for this recipe.

2. Edit the myRec.txt file in an external text editor and replace the variable values with other values that you want to load into the recipe definition in CODESYS. Save the file.

3. Click the "myRec" column in the recipe definition and right-click for the "Load Recipe" command.
   ⇒ A dialog prompt notifies you about the possibly needing to perform an online change when logging in again. An online change is necessary when you change the current values of the recipe variables by loading the recipe.

4. Click "Yes" to close the dialog and continue. Select the myRec.txt file from the file explorer for loading.
   ⇒ The recipe values in the recipe definition are updated according to the values read in the file.

If you want to overwrite only individual recipe variables with new values, then remove the values for the other variables before loading to the recipe file. Entries without value definitions are not read, and therefore updating leaves these variables unchanged on the PLC and in the project.

For values of the data type REAL/LREAL, the hexadecimal value is also written to the recipe file in some cases. This is necessary so that the exact identical value is restored when converting back. In this case, change the decimal value and delete the hexadecimal value.

Recipe management on the controller; memory usage

When you clear the "Recipe management in the PLC" option, the Recipe Manager and recipe definitions will not use any memory on the PLC.

If you select this check box, then code is generated for the Recipe Manager and all recipe definitions, and this code is stored on the PLC. The size of the used memory primarily depends on the number of recipes and their variables, as well as the data type of the variables. Whether or not the fields of the recipe definition are filled also has an effect. The memory usage of recipes cannot be calculated. It has to be determined by experimentation at the time it is needed. The following table merely provides some guiding principles.

See also

- Chapter 1.3.1.25.1.33 “Object ‘Recipe Manager’” on page 820
- Chapter 1.3.1.25.1.34 “Object ‘Recipe Definition’” on page 823
- Chapter 1.3.1.25.2.19 “Menu ‘Recipes’” on page 1015

See also

- Chapter 1.3.1.25.2.19.4 “Command ‘Load Recipe’” on page 1016
- Chapter 1.3.1.25.2.19.8 “Command ‘Load and Write Recipe’” on page 1017
Table 7: Memory usage for CODESYS V3 SP6 with CODESYS Control Win V3 controller

<table>
<thead>
<tr>
<th>Recipe definition</th>
<th>Code Size (bytes)</th>
<th>Data Size (bytes)</th>
<th>Total (bytes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Recipe definition with 100 INT variables</td>
<td>194406</td>
<td>79400</td>
<td>267352</td>
</tr>
<tr>
<td>Recipe definition with 200 INT variables</td>
<td>238318</td>
<td>121284</td>
<td>459344</td>
</tr>
<tr>
<td>Recipe definition with 300 INT variables</td>
<td>282230</td>
<td>163084</td>
<td>543856</td>
</tr>
<tr>
<td>Recipe definition with 100 BOOL variables</td>
<td>192742</td>
<td>69884</td>
<td>343168</td>
</tr>
<tr>
<td>Recipe definition with 200 BOOL variables</td>
<td>235446</td>
<td>101568</td>
<td>436872</td>
</tr>
<tr>
<td>Recipe definition with 300 BOOL variables</td>
<td>278146</td>
<td>133284</td>
<td>510072</td>
</tr>
<tr>
<td>Recipe definition with 100 string variables</td>
<td>203278</td>
<td>870084</td>
<td>1154000</td>
</tr>
<tr>
<td>Recipe definition with 200 string variables</td>
<td>255570</td>
<td>1709784</td>
<td>2973296</td>
</tr>
<tr>
<td>Recipe definition with 300 string variables</td>
<td>307886</td>
<td>2549484</td>
<td>2964112</td>
</tr>
</tbody>
</table>

You can apply recipe values on the controller to recipe definitions in the project, even if these definitions have been modified in the project.

Requirement: The “Recipe management in the PLC” is option is selected in the Recipe Manager.

1. Create a recipe definition RecDef1 in the project, containing the variables PLC_PRG.ivar and PLC_PRG.bvar. Insert a recipe “R1”: value for PLC_PRG.ivar:33; value for PLC_PRG.bvar:TRUE.

2. Login to the controller and download the application.
   ⇒ The recipe file R1.RecDef1.txtrecipe is saved to the default directory of the controller ($PlcLogic$).

3. Logout and add another variable PLC_PRG.dwvar to the recipe definition in the project.

4. Edit the recipe definition file R1.RecDef1.txtrecipe on the device by changing the value for PLC_PRG.ivar from 33 to 34.

   Moreover, add another recipe “R2” on the device. To do this, copy the R1.RecDef1.txtrecipe and rename it to R2.RecDef1.txtrecipe. Then edit this file and change the recipe values: PLC_PRG.ivar:1, PLC_PRG.bvar:FALSE.

   ⇒ Now two recipes “R1” and “R2” are available on the device. In the project, there is only “R1”, and it also contains other values than “R1” on the device.

5. Login again to the controller with online change.

6. Click “Upload Recipes from Device” from the context menu.
   ⇒ A dialog prompt notifies you that executing the command at the next login may trigger an online change, and that the recipes on the runtime system will overwrite the recipes of the current recipe definition.

7. Confirm that you wish to continue.
   ⇒ A dialog prompt notifies you that the recipe for PLC_PRG.dwvar loaded on the device cannot yield a value from the controller.
8. Confirm that you wish to continue.
   ☰ The value of PLC_PRG.ivar in recipe “R1” of the recipe definition in the project changes to 34. The recipe “R2” with the values 1 and FALSE is also listed in the recipe definition now. PLC_PRG.dwvar remains in the recipe definition.

1.3.1.13.3 Data recording with trace

You can use a “Trace” to follow the value history of variables on the controller in a similar way as a digital sampling oscilloscope. When the application is in runtime mode with trace, all statements are executed first within the task cycle. Then, data recording starts with value storage including time stamps. These time stamps are relative and refer to the start time of the data recording. The data yields a discrete time signal and CODESYS displays its course in the trace editor.

A sample (data record) is composed of the value and the time stamp. The runtime system writes the samples to a buffer with a definable size. CODESYS requests the data, saves it in the trace editor buffer, and displays it in the trace diagram as a function of time. You can monitor the value history of the configured variables continually because CODESYS displays the latest data.

You can trigger the data recording. When this happens, the application saves the data from the time of the trigger and CODESYS displays the data at the time of the trigger.

The configuration and the display of a trace are possible in the CODESYS project by means of trace objects in the trace editor. There are the following two object types:

- “Trace”: Inserted below the IEC application in the device tree. This kind of object always contains a purely application-specific trace configuration. You can download this trace configuration to the controller and run it with the application.
- “DeviceTrace”: Inserted below the device object in the device tree. If the PLC supports a trace manager, then you can use one or more “DeviceTrace” objects to access one or more traces that are running on the controller. These can be both application-specific or controller-specific traces. For example, a controller can support traces for recording the processor load. Menu commands allow for access from the CODESYS project to the trace manager in the device.

Access to the trace manager from IEC code is possible by means of the functions from the library CmpTraceMgr.library. For more information, refer to the library documentation.

![NOTICE!]
A running data recording with trace can lead to a significant increase in the cycle time of the IEC task.

![NOTICE!]
Data recording with trace also continues running after logging out of the device.

The device description of a runtime system with trace manager includes the tracemanager entry in the TargetSettings section.

In this case, CODESYS transfers only the trace configuration when downloading the application to the PLC. When you start the trace, the application interprets the configuration on the RTS by means of the trace manager, executes the data recording, and buffers the data sets on the PLC.

The CmpTraceMgr runtime system component provides extended functionality, as compared to data recording with IEC code.
Data recording is therefore possible as follows:

- Parameters on the PLC (for example, the processor load (cpuload, plcload), or the temperature curve of a CPU or a battery). The measurement of the processor load per CPU core (cpuload) is interesting for multicore controllers.
- Device signals (for example, the current path of a drive)
- System variables of another runtime system component

You can configure parameters like IEC variables in the “Trace Configuration” dialog of the “Variable Settings”.

The display of traces that run on the controller is possible in the trace editor of a DeviceTrace object.

See also

- Chapter 1.3.1.13.3.4 “Accessing all traces on the controller” on page 342
- Chapter 1.3.1.9.17.2 “Multicore” on page 241
- Chapter 1.3.1.25.1.40 “Object ‘DeviceTrace’” on page 845
- Chapter 1.3.1.25.3.15.2 “Dialog ‘Trace Configuration’” on page 1092

Data recording after triggering

To monitor data that depends on an event or a condition, you can free the data recording that depends on a trigger. At runtime, the application checks whether the event has occurred or the condition is fulfilled, and then it buffers the data accordingly.

The trace configuration enables triggering by:

- a trigger variable that maps the event
- a condition as expression
- a combination of trigger variable and condition

Saving samples to a file

You can save samples from the development system to a file. The file can also include the trace configuration.

Table 8: Possible file formats

<table>
<thead>
<tr>
<th>File Extension</th>
<th>File type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>*.trace</td>
<td>Trace file</td>
<td>Contains the samples and the trace configuration in XML format. You can execute the “Load Trace” command to load the file to the trace editor when offline and analyze the samples without a controller.</td>
</tr>
<tr>
<td>*.txt</td>
<td>Text File</td>
<td>Contains the samples in ASCII format. You can edit the file with an external tool.</td>
</tr>
</tbody>
</table>

Table: Possible file formats

<table>
<thead>
<tr>
<th>File Extension</th>
<th>File type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>*.trace</td>
<td>Trace file</td>
<td>Contains the samples and the trace configuration in XML format. You can execute the “Load Trace” command to load the file to the trace editor when offline and analyze the samples without a controller.</td>
</tr>
<tr>
<td>*.txt</td>
<td>Text File</td>
<td>Contains the samples in ASCII format. You can edit the file with an external tool.</td>
</tr>
</tbody>
</table>
### File Extension and Description

<table>
<thead>
<tr>
<th>File Extension</th>
<th>File Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>*.trace.csv</td>
<td>“Trace dump”</td>
<td>File in CSV format includes the trace configuration and optional samples. You can create the file by clicking “Export Symbolic Trace Config”. You can transfer the file to the controller and load it to the application. Then you can execute the “Load Trace” command in CODESYS to display this in the trace editor. You can also click “Trace ➔ Save Trace” and select the *.trace.csv file format. You can transfer the file to the controller and load it with an HMI for analysis.</td>
</tr>
<tr>
<td>*.traceconfig</td>
<td>“Symbolic trace configuration”</td>
<td>Contains the trace configuration CSV format. You can create the file by clicking “Export Symbolic Trace Config”. The CmpTraceMgr runtime system component can read the file.</td>
</tr>
</tbody>
</table>

See also
- Chapter 1.3.1.25.2.21.15 “Command ‘Save Trace’” on page 1033
- Chapter 1.3.1.25.2.21.8 “Command ‘Load Trace’” on page 1029
- Chapter 1.3.1.25.2.21.7 “Command ‘Export Symbolic Trace Config’” on page 1027
- Chapter 1.3.1.25.1.36 “Object ‘Symbol Configuration’” on page 824

See also
- Chapter 1.3.1.25.1.39 “Object ‘Trace’” on page 842
- Chapter 1.3.1.25.1.40 “Object ‘DeviceTrace’” on page 845

### Getting started

**Program**

```plaintext
PROGRAM PLC_PRG
VAR
  iVar : INT;
  rSin : REAL;
  rVar : REAL;
END_VAR

iVar := iVar + 1;
iVar := iVar MOD 33;

rVar := rVar + 0.1;
rSin := 30 * SIN(rVar);
```

**Requirement:** The application is running the PLC_PRG program on the controller.

1. In the device tree, select the application and add a new trace object by clicking “Project ➔ Add Object”.
   - The respective trace editor opens with the commands available in the “Trace” menu.
2. Click “Trace ➔ Configuration”.
   - The “Trace Configuration” dialog box opens.
3. Select a task for running the trace feature. Normally this is the same task that is running in PLC_PRG.
4. Click “Add Variable” to add an entry to the tree view of the trace configuration and assign an IEC variable (for example, PLC_PRG.rSin).

5. Click “Trace ➔ Download Trace”.
   - CODESYS loads the trace configuration to the controller. The application starts recording data and transmits the data to CODESYS, where it is displayed in the trace diagram as a graph. Commands are provided for navigating through the samples and controlling the data recording.

Example

The PLC_PRG program is running on the controller. When you follow the instructions for "Getting Started", CODESYS displays the following trace diagram.

![Trace Diagram]

- (1) : “Configuration”
- (2) : “Add Variable”

See also
- ☀ Chapter 1.3.1.25.1.39 “Object 'Trace’” on page 842

Creating trace configuration

For a complete trace configuration, specify at least one task and one variable. In order to trigger the data recording, activate the trigger option and select a trigger variable or specify a recording condition.

See also
- ☀ Chapter 1.3.1.25.3.15.1 “Dialog 'Advanced Trace Settings’” on page 1091
- ☀ Chapter 1.3.1.25.3.15.2 “Dialog 'Trace Configuration’” on page 1092
- ☀ Chapter 1.3.1.25.3.17 “Dialog Box 'Advanced Trend Settings’” on page 1097
- ☀ Chapter 1.3.1.13.4.2 “Configuring trend recording” on page 346

Assigning a task

In this task, the data recording is executed in runtime mode. Usually the same task is selected where the variables are written.

1. Double-click the trace object.
   - The trace editor opens with the commands available in the “Trace” menu.
2. Click “Trace ➔ Configuration”.
   ⇒ The “Trace Configuration” dialog opens. In the tree view “Trace Record”, the top item is selected and the subdialog “Record Settings” is shown on the right.

3. Click the “arrow down” symbol (▼) in the “Task” drop-down list.
   ⇒ The drop-down list opens with all tasks that are available throughout the application.

4. Select a task for the trace.

See also
● © Chapter 1.3.1.25.3.15.2 “Dialog 'Trace Configuration’” on page 1092

Configuring a trace variable

1. Double-click the trace object.
   ⇒ The trace editor opens. The commands of the “Trace” menu are available.

2. Click “Add Variable”.
   ⇒ The “Trace Configuration” dialog opens. The subdialog “Variable Settings” is displayed on the right.

3. Click ▼ in the input field of the “Variable” setting and select a trace variable in the “Input Assistant” dialog.
   ⇒ The variable is configured for data recording. The trace record tree and the display tree were extended by the variable.

4. Click the “Add Variable” link.
   ⇒ The trace record tree and the display tree receive a new variable. The settings of the variables are available on the right.

5. Select a trace variable.

6. Click “OK” to close the dialog.
   ⇒ The variables are trace variables and are displayed in the trace variable list.

Deleting a trace variable

1. Double-click the trace object.

2. Click a variable in the trace record tree.

3. Click the “Delete Variable” command or press [Del].

4. Click “OK” to close the dialog.
   ⇒ The variable is removed from the trace variable list.

Tracing a parameter

1. Double-click the trace object.

2. Click “Add Variable”.
   ⇒ The “Trace Configuration” dialog opens. The subdialog “Variable Settings” is displayed on the right.

3. Click ▼ (right of the “Variable” setting, left of the input field).

4. Select the “Parameter” option in the drop-down list.

5. Click ▼ and select a parameter from the “Input Assistant” dialog.
6. Configure how the parameter is displayed.
7. Click “OK” to close the dialog.
   ≫ The parameter will be traced and displayed in the trace variable list.

### Configuring a trigger

1. Double-click the trace object.
   ≫ The trace editor opens with the commands available in the “Trace” menu.
2. Click “Trace ➔ Configuration”.
   ≫ The “Trace Configuration” dialog opens. The subdialog “Record Settings” is displayed on the right.
3. Select the “Enable trigger” check box.
4. Select the task in which the trend record is to be executed.
5. Select a variable from the “Trigger Variable” field.
6. Click “OK” to close the dialog.
   ≫ The data recording will be triggered.

The trigger time is displayed as a black line in the diagram in runtime mode.

1. Download the application and start it.
2. Click “Trace ➔ Download Trace”.
   ≫ The trace configuration is loaded. After triggering, the runtime system saves the value graph of the trace variables. The data is displayed in the trace editor. The trigger time is displayed as a black line in the diagram.

### Configuring the display of the time axis

1. Double-click a trace object.
2. Click the “Configuration” link above the configuration tree.
   ≫ The “Trace Configuration” dialog opens.
3. Select “Time axis” in the display tree (below “Presentation (Diagrams)”).
   ≫ The display settings of the time axis are shown on the right.
4. Edit the presets and click the “Preview” link.
   ≫ The changes are seen in the coordinate system preview.
5. Click “Y-axis” in the display tree. The “Y-axis” item is below every configured diagram. Therefore, the display of the value axis is set for each diagram.
   ≫ The subdialog “Display Settings” of the selected axis is displayed on the right.
6. Change the preset value.
   ≫ The changes are applied in the coordinate system preview.
7. Click OK to close the “Trace Configuration” dialog.
   ≫ The display changes are visible in the affected diagrams.
Configuring the display of the trace variable

1. Double-click a trace object.
2. Click the "Configuration" link.
   ⇒ The "Trace Configuration" dialog opens.
3. Select a variable below "Trace Record".
   ⇒ The subdialog "Variable Settings" of the selected variable is displayed on the right.
4. Change a setting, for example the "Line type".
5. Click "OK" to close the dialog.
   ⇒ The display changes are visible in the affected diagrams.

Configuring the buffer for data on the runtime system

1. Double-click a trace object.
2. Click "Trace ➔ Configuration".
   ⇒ The "Trace Configuration" dialog opens. The subdialog "Record Settings" is displayed on the right.
3. Click "Advanced".
   ⇒ The "Advanced Trace Settings" dialog opens.
4. Change the setting "Measure in every n-th cycle" or "Recommended runtime buffer size (samples)".
5. Click "OK" to close the dialog.
   ⇒ The buffer settings are reconfigured. It is applied after the trace configuration is loaded to the RTS the next time.

Editing the trace configuration in runtime mode

Requirement: The application is running on the controller and a trace configuration is loaded.

1. Double-click a variable in the trace record tree.
   ⇒ The "Trace Configuration" dialog opens.
2. Change the color, for example.
   ⇒ The variable is displayed in the new color in the affected diagrams without interrupting the execution of the application.

If you change essential settings, for example a trace variable, then you must download the trace configuration to the controller again.

Operating the data recording

Use menu commands for controlling how data is recorded.

Requirement: The application is loaded on the runtime system and a trace is configured.
Menu commands

- “Trace ➔ Download Trace”
- “Trace ➔ Start Trace”
- “Trace ➔ Stop Trace”
- “Trace ➔ Reset Trigger”

See also

- Chapter 1.3.1.25.2.21.6 “Command ’Download Trace’” on page 1026
- Chapter 1.3.1.25.2.21.16 “Command ’Start Trace’” on page 1033
- Chapter 1.3.1.25.2.21.17 “Command ’Stop Trace’” on page 1033
- Chapter 1.3.1.25.2.21.13 “Command ’Reset Trigger’” on page 1032

Accessing all traces on the controller

If the controller supports the runtime system component CmpTraceMgr (Trace Manager), then you can access all traces from a CODESYS project which are running on the controller. In addition to application-related traces that capture the values of IEC variables, these can also be entirely controller-specific traces (for example, for recording device signal values or the CPU load).

For each trace running on the controller that you want to present in your project, you have to insert an individual “DeviceTrace” object in the device tree.

In order to show a trace from the device in this object, the connection to the PLC has to be configured correctly (“Communication Settings”). Then use one of the following menu commands:

- “Trace ➔ Upload Trace”: Establishes the connection to the PLC and opens the “Online List” dialog for selecting a trace from the controller.
- “Trace ➔ Online List”: Available in online mode only: Also opens the “Online List” dialog.

Now the trace uploaded from the controller can be started and traced in the editor of the DeviceTrace object. The configuration of the presentation (colors, labels, etc.) is the same as with traces for application variables configured in the project.

NOTICE!
Closing the DeviceTrace editor terminates the connection to the controller.

Please note that the connections to the controller is also terminated when the last open “DeviceTrace” editor is closed. In order for device traces to be displayed again in the project, you have to reload them into the “DeviceTrace” objects.

At this time, closing the editor is also the recommended procedure for deliberately terminating the connection to the controller. Logging out is not enough for this.

See also

- Chapter 1.3.1.25.2.21.4 “Command ’Configuration’” on page 1025
- Chapter 1.3.1.25.1.40 “Object ’DeviceTrace’” on page 845
- Chapter 1.3.1.25.2.21.19 “Command ’Upload Trace’” on page 1034
- Chapter 1.3.1.25.2.21.12 “Command ’Online List’” on page 1031
Displaying the CPU load with DeviceTrace objects in the CODESYS project (example)

Requirement: The PLC device supports the Trace Manager. For the example described here, this is CODESYS Control Win V3. The device provides traces of the individual CPU loads (CpuLoad), as well as traces of the CPU load caused by the runtime system (PlcLoad). The possible display of the CPU load in the project can be helpful when using multicore functionality.

1. In the project, define the “Communication Settings” for the controller.
2. Select the PLC entry in the device tree and add a “DeviceTrace” object.
4. Set the focus in the trace editor and click “Trace ➔ Upload Trace”.
   - The connection to the controller is established and the “Online List” dialog opens.
5. Select the “PlcLoad” entry in the dialog and click “Upload”. Click OK to close the dialog.
   - Four trace views open in the trace editor to show the CPU load in the runtime system. Three of the traces are for the three CPUs and one trace for the average value. The following text appears for each: "No samples have been recorded."
6. Click “Trace ➔ Start Trace”.
   - The trace recording for the four parameters is displayed.
7. If you also want to display the traces for the CpuLoad per CPU with their average value in the project, then insert another “DeviceTrace” object into the device tree. Name it “Trace_CpuLoad” for example. Load and start the traces for “CpuLoad” in the editor as described above.
   - Now you can monitor all traces in the project:

![Trace views in trace editor](image)

8. If you want to change the appearance of the presentation, then click “Configuration” in the respective trace editor window to access the configuration dialogs. You can use these dialogs (except variable assignments) in the same way as for an IEC variable trace created in a project.
9. To disconnect from the controller, close all open DeviceTrace editor windows. If you are logged in to the device, then logging out is enough to terminate the connection.

See also
- Chapter 1.3.1.25.1.19.2 “Tab 'Communication Settings’” on page 741
- Chapter 1.3.1.9.17.2 “Multicore” on page 241

Navigating into trace data

Use menu commands to navigate the data in the trace diagram.

Requirement: The application is in online mode.

Menu commands
- “Trace ➔ Cursor”
- “Trace ➔ Mouse Zooming”
- “Trace ➔ Reset View”
- “Trace ➔ AutoFit”
- “Trace ➔ Compress”
- “Trace ➔ Stretch”
Managing trace

Use menu commands to load and save traces in various formats.

Menu commands

- “Trace ➔ Convert to Single-Channel”
- “Trace ➔ Convert to Multi-Channel”

See also

- § Chapter 1.3.1.25.2.21.5 “Command ‘Cursor’” on page 1025
- § Chapter 1.3.1.25.2.21.9 “Command ‘Mouse Zooming’” on page 1029
- § Chapter 1.3.1.25.2.21.14 “Command ‘Reset View’” on page 1032
- § Chapter 1.3.1.25.2.21.2 “Command ‘AutoFit’” on page 1025
- § Chapter 1.3.1.25.2.21.3 “Command ‘Compress’” on page 1025
- § Chapter 1.3.1.25.2.21.18 “Command ‘Stretch’” on page 1034
- § Chapter 1.3.1.25.2.21.10 “Command ‘Convert to Multi-Channel’” on page 1029
- § Chapter 1.3.1.25.2.21.11 “Command ‘Convert to Single-Channel’” on page 1030

Showing statistics

CODESYS evaluates and displays the recorded data with an option of saving the data to the clipboard. Click “Trace ➔ Statistics”.

See also

- § Chapter 1.3.1.25.2.21.20 “Command ‘Statistics’” on page 1034

1.3.1.13.4 Data recording with trend

When you want to monitor the development of data over a long period of time for the purpose of reading a trend, you can save the data with “Trend Recording”. You can configure any number of variables or parameters to save their values in a persistent database. This database is located on the PLC and is populated continually at runtime.
Trend recording comprises the following objects:

- (1): “Trend recording task” of type “Task”
- (2): Object of type “Trend Recording Manager”
- (3): Object of type “Trend Recording”

**NOTICE!**

**Timeout for trend recording**

During a trend recording, it can happen that the application task triggers a timeout that is caught with an exception when transitioning from “Running” to “Stop”. Causes can be that file operations with the SQLite database are taking too long or that too many variables are being recorded. This usually happens on a target device with weak performance.

You can avoid the occurrence of an exception:

- Configure the trend recording with less memory demand so that the amount of data that is stored is adapted to the target system.
- Reduce the number of variables.

To display the collected data, you design a visualization with a “Trend” element. This kind of visualization accesses the database for visualizing the data.

See also

- § Chapter 1.3.1.25.1.41 “Object ‘Trend Recording Manager’” on page 845
- § Chapter 1.3.1.25.1.42 “Object ‘Trend Recording’” on page 846
- § Chapter 1.3.1.25.1.43 “Object ‘Trend Recording Task’” on page 849

**Getting started with trend recording**

To execute trend recording on a runtime system, you need an application with a “Trend Recording Manager” object that contains at least one “Trend Recording” object. Then you can configure a database on the runtime system and the data buffering.
1. Add a "Trend Recording Manager" object below your application.
2. Select the "Trend Recording Manager" object and click "Add Object ➔ Trend Recording". Type a name in the "Add Trend Recording" dialog box.
   ⇒ CODESYS creates the object. The editor opens.
3. Type a task in "Record Settings".
4. Click "Add Variable".
   ⇒ CODESYS adds another variables. The blank settings open in the "Variable Settings" to the right of the tree view.
5. Select a valid IEC variable from the "Variable" field.
   ⇒ The IEC variable is configured for trend recording.
6. Build the application.
7. Download the application to the controller and click "Start".
   ⇒ The application records data in runtime mode and saves it to a database.

See also
- Chapter 1.3.1.25.1.42 “Object ‘Trend Recording’” on page 846
- Getting Started with Trend ➔ Chapter 1.3.5.11.1 “Getting started with trend visualization” on page 1165

Configuring trend recording
You can configure a database on the runtime system and the data buffering.

**NOTICE!**
Timeout for trend recording
During a trend recording, it can happen that the application task triggers a timeout that is caught with an exception when transitioning from "Running" to "Stop". Causes can be that file operations with the SQLite database are taking too long or that too many variables are being recorded. This usually happens on a target device with weak performance.

You can avoid the occurrence of an exception:
- Configure the trend recording with less memory demand so that the amount of data that is stored is adapted to the target system.
- Reduce the number of variables.

Assigning tasks In this task, the runtime system records the trend.

1. Double-click a “Trend Recording” object in the device tree.
   ⇒ The respective editor opens. In the tree view of the trend configuration, the top entry is selected, and on the right you see the current configuration in "Record Settings".
2. Click the "arrow down" symbol (▼) in the "Task" drop-down list.
   ⇒ The drop-down list opens with all tasks that are available throughout the application.
3. Select a task for trend recording.

See also
● Chapter 1.3.1.25.1.42 “Object ‘Trend Recording’” on page 846

---

**NOTICE!**
The number of variables is limited for trend recording. You can change this number in the “Trend storage” dialog.

---

**Adding IEC variables**

1. Double-click a “Trend Recording” object in the device tree.
   ➜ The respective editor opens. In the tree view of the trend configuration, the top entry is selected, and on the right you see the current configuration in “Record Settings”.
2. Right-click an entry in the tree view.
3. Click “Add Variable”.
   ➜ CODESYS adds another variables. The blank settings open in the “Variable Settings” to the right of the tree view.
4. Select a valid IEC variable from the “Variable” field.
   ➜ The IEC variable is configured for trend recording.
5. Configure how the variable is displayed in the trend diagram.
6. Configure how the alert color is displayed in the trend diagram.

See also
● Chapter 1.3.1.25.1.42 “Object ‘Trend Recording’” on page 846
● Chapter 1.3.1.25.3.16 “Dialog Box ‘Trend storage’” on page 1097

---

**Removing variables from the configuration**

1. Double-click a “Trend Recording” object in the device tree.
2. Click a variable in the tree view of the configuration.
3. Click “Delete Variable” or press [Del].

---

**Starting conditional trend recording**

You can configure conditional trend recording for execution. Configuration is not possible when depending on triggering. For that you need a “Trace” object.

1. Double-click a “Trend Recording” object in the device tree.
2. Click the top node in the tree view of the trend configuration.
   ➜ The name of the trend configuration is selected and on the right you see the current configuration in “Record Settings”.
3. Assign a Boolean variable, an access to a bit, or a property to the “Record condition” field.
   ➜ When the application is in runtime mode, data is recorded only if the value is TRUE.

See also
● Chapter 1.3.1.13.3 “Data recording with trace” on page 335
Adding parameter

1. Double-click a “Trend Recording” object in the device tree.
   → The respective editor opens. In the tree view of the trend configuration, the top entry is selected, and on the right you see the current configuration in “Record Settings”.
2. Right-click an entry in the tree view.
3. Click “Add Variable”.
   → CODESYS inserts a new variable. The blank settings open in the “Variable Settings” to the right of the tree view.
4. Click the “down” symbol (▼) to the right of the “Variable” label.
5. Select “Parameter” from the drop-down list.
6. Click and select a parameter from the “Input Assistant” dialog.
7. Configure how the parameter is displayed in the trend diagram.
8. Configure how the alert color is displayed in the trend diagram.

See also
● Chapter 1.3.1.25.1.42 “Object ’Trend Recording’” on page 846

Configuring data buffering on the RTS

1. Double-click a “Trend Recording” object in the device tree.
   → The respective editor opens. In the tree view of the trend configuration, the top entry is selected, and on the right you see the current configuration in “Record Settings”.
2. Click “Trend Storage”.
   → The “Trend Storage” dialog opens.
3. Now you can change the settings.

See also
● Chapter 1.3.1.25.3.16 “Dialog Box ’Trend storage’” on page 1097

Configuring additional buffering

1. Double-click a “Trend Recording” object in the device tree.
   → The respective editor opens. In the tree view of the trend configuration, the top entry is selected, and on the right you see the current configuration in “Record Settings”.
2. Click “Advanced”.
   → The “Advanced Trace Settings” dialog opens.
3. Now you can change the settings.

See also
● Chapter 1.3.1.25.3.17 “Dialog Box ’Advanced Trend Settings’” on page 1097
1.3.1.13.5 Monitoring tasks
In online mode, you can display some statistical values of the tasks in the runtime system. This information is very useful for testing clock cycles or solving problems in the runtime performance.
1. Switch to online mode.
2. Select the “Task Configuration” object in the device tree. Click “Project ➔ Edit Object”.
   The task configuration opens in the editor.
3. Click the “Monitor” tab.

See also
● Chapter 1.3.1.13.3.2 “Creating trace configuration” on page 338

1.3.1.13.6 Reading the PLC log
CODESYS provides the capability to display the events and error messages logged in the controller.
See also
● Chapter 1.3.1.25.1.37.3 “Tab 'Monitor’” on page 837

Reading the log Requirement: The controller is running.
1. Select the controller in the device tree.
2. Choose the command “Project ➔ Edit Object”.
   The device editor opens.
3. Choose the tab “Log”.
4. Click on to update the view.
   A connection to the controller is established. The controller in the device tree is highlighted in green.
   All controller log information are displayed.
5. Click on to delete the current list.
6. Filter the view by clicking on the desired category (for example "Information").
7. Save the log entries. Click on and choose a file name.

See also
● Chapter 1.3.1.25.1.19.8 “Tab 'Log’” on page 750

1.3.1.13.7 Analysing errors with core dump
A core dump is a memory snapshot of the application data. In case of an exception error, runtime systems that support this feature automatically store a core dump file (<application name>.core) in the application directory and on the controller.
In online mode, you can also generate a core dump explicitly if the application is currently stopped at breakpoint or if an exception has occurred. In this case, CODESYS stores the core dump file in the project directory and not on the controller.

In offline mode, you can load the core dump from the controller to the project. An online view of the application is then displayed with the data and values at the time of the exception.

In addition, you can call a project view of the controller log file at the time of generating the core dump.

**NOTICE!**

The runtime system must completely support the functionality to assure a correct display of variable values in the core dump view.

In the online view of the application that CODESYS created when loading the core dump into the project, menu commands are displayed as available but do not have any effect in this status. When selecting one of these commands, a message is displayed.

**CAUTION!**

Restriction (CDS-37572): Exception handling in the runtime system may overwrite parts of the variable values in the call stack. Therefore, the original values are lost and important information may no longer be displayed for variables in functions and methods at the time of monitoring.

When creating project archives: If you created a core dump explicitly by clicking “Create Core Dump”, then this option is available automatically in the archive configuration. On the other hand, if you copied the core dump from the PLC, then you can add it to a project archive only by selecting “Additional files”.

Important: A project archive with a core dump must necessarily include the download information. If it does not, then CODESYS cannot use the core dump.

**Loading the core dump and device log into the project for analysis**

Requirement: You have a project open with an application that has produced an exception on the controller. The runtime system is able to create core dumps. The application in the project is in offline mode.

1. Load the required core dump from the controller by clicking “Debug ➔ Load Core Dump”.
   - CODESYS shows an online view of the application. You see the variable values and call stack from the time of error (see below). This includes also the values of the I/O variables in the mapping dialog box of the device configurator and the task configuration. “Core dump loaded” appears in the status line. CODESYS copies the core dump file as <project name>.<device name>.<application name>.<application Guid>.core to the local project directory.

2. Click “Debug ➔ Load Device Log from Core Dump” to load the device log from the time of error into the project.
   - The log view opens (as in online mode of the device editor) with the events at the time of the core dump.

3. After completing the core dump analysis, click “Debug ➔ Close Core Dump”.
   - CODESYS closes the core dump view of the application. The project returns to the normal offline mode with its views.
Creating the core dump of the running application manually

Requirement: A CODESYS application is in online mode. The runtime system supports core dump functionality.

- Click “Debug ➤ Create Core Dump”.
  - CODESYS starts generating a new core dump. A progress bar appears in the status line and a button to cancel the process.
  - CODESYS saves a core dump file to the local project directory as <project name>.<device name>.<application name>.<application Guid>.core.

See also
- ☛ Chapter 1.3.1.25.2.7.23.2 “Command ‘Create Core Dump’” on page 947
- ☛ Chapter 1.3.1.25.2.7.23.1 “Command ‘Load Core Dump’” on page 947
- ☛ Chapter 1.3.1.25.2.7.23.3 “Command ‘Close Core Dump’” on page 948
- ☛ Chapter 1.3.1.25.2.7.23.4 “Command ‘Load Device Log from Core Dump’” on page 948

1.3.1.13.8 Using PLC shell for requesting information

The “PLC shell” in CODESYS is a text-based control monitor (terminal) on a tab of the device editor. There you can enter commands for the request of specific information from the controller, as well as execute actions like starting, stopping or downloading applications. Also a description on the meaning and syntax of the possible commands you can get directly via the PLC shell.

The list of available commands depends on the PLC manufacturer. It can be extended via the so-called SPS Shell Command Handler interface, either in a C-component of the runtime system, or as a function block in an IEC library or project.

The response from the PLC will be shown as a string in an output data window within the tab.

Requesting information about the application on the controller

Requirement: Your project is connected with a controller; Example: CODESYS Control Win V3, on which an application App1 is running.

1. Open the device editor double-clicking on the object CODESYS Control Win V3 in the device tree, and activate tab “PLC Shell”.
   - The tab shows an empty output data window. Below there is an entry field for a command.
2. Click button 
   - The “Insert Standard Command” dialog appears with a list of commands.
3. Choose command “?” and click button “Execute”.
   - The dialog closes and in the output data window you see a list of the supported commands and their possible parameters. Each the syntax for how to enter the command is displayed.
4. Click again [..] and choose command “pid”. In the input assistant supplement the command as follows: pid App1. Press the Enter key.

∴ In the output data window the following gets displayed (the GUIDs are just examples):

   pid App1
   Project Identification
   Application: App1
   Code GUID:0x08a893c0
   Data GUID:0x762d0e90

5. Click button ▼ in the command line.

∴ Command pid App1 is added to the history of already entered commands.

See also
● Chapter 1.3.1.25.1.19.10 “Tab 'PLC Shell’” on page 753

1.3.1.13.9 PLC operation control via system variables

CAUTION!
You are responsible for runtime system services being enabled under safe application conditions and disabled only under critical conditions.

At runtime, the state of an application or facility can become sensitive and disruptive actions can endanger the entire machine or facility. However, in this state you can suppress certain commands and prevent dangerous actions. The “PlcOperationControl” function block and “Component Manager” library are provided for this purpose.

Examples of CODESYS commands that can suppress operations when executed:

● “Online Change”, “Download”
● “Enable Breakpoint”
● “Reset Application”, “Stop Application”
● “Transmit Data”
● “Force Values”, “Write Values”

In order that a backup solution is always in place, you are not permitted to suppress the “Reset origin” and “Delete” commands.

CODESYS will notify you if a currently disabled runtime system service is required when the application is in runtime mode. Then, you can respond with an appropriate countermeasure.

Function block
PlcOperationControl for operation control

This function block is used for enabling and disabling operations.

Table 9: Property (PROPERTY)

<table>
<thead>
<tr>
<th>Name</th>
<th>Data Type</th>
<th>Initial value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>xDisableApplicationOnlineChange</td>
<td>BOOL</td>
<td>FALSE</td>
<td>TRUE: Online change is suppressed.</td>
</tr>
<tr>
<td>xDisableApplicationDownload</td>
<td>BOOL</td>
<td>FALSE</td>
<td>TRUE: Download is suppressed.</td>
</tr>
</tbody>
</table>
### Implementing operation control

**Requirement**
- Compiler version $\geq 3.4.3.0$
- In the device description, the PLC operation control is enabled by system variables.

1. Declare an instance of the `PlcOperationControl` function block (for example, `PlcOpCtrl_Inst`).
   ```plaintext```
   <> PlcOpCtrl_Inst : PlcOperationControl;
   ```plaintext```
2. Suppress a command by assigning the respective TRUE property (for example, "Stop Application").
   ```plaintext```
   <> PlcOpCtrl_Inst.xDisableApplicationStop := TRUE;
   ```plaintext```

See also
- ☰ Chapter 1.3.1.25.2.6.6 “Command ‘Online Change’” on page 928
- ☰ Chapter 1.3.1.25.2.7.16 “Command ‘Force Values’” on page 943
- ☰ Chapter 1.3.1.12.4 “Forcing and writing variables” on page 315

### 1.3.1.13.10 Backup and restore

CODESYS and the CODESYS standard runtime systems (with version 3.5.8.0 and later) support backing up application-specific files on the PLC. You can execute the required actions in the “Backup and Restore” tab of the generic device editor.

A backup consists of creating and saving a file in zip archive that contains the application-related files and an information file `meta.info`. This backup file has the extension `TBF` (="Target Backup File") and can be saved in the local file system or on the PLC.

The following applies when restoring the software status from the backup file:
- A dialog opens with a list of affected files on the PLC, and you can deactivate optional components.
- If you have to set the application to STOP mode for backup or restore, a dialog prompt will open to warn you about this- Warnings also open when incompatibilities are detected.
- The user interface is blocked when restoring to the PLC.
- Existing files are overwritten without warning.
- Existing boot applications are deactivated as soon as at least one new boot application is part of the restore.

See also
- ☰ Chapter 1.3.1.25.1.19.5 “Tab ‘Backup and Restore’” on page 748

<table>
<thead>
<tr>
<th>Name</th>
<th>Data Type</th>
<th>Initial value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>xDisableApplicationStop</td>
<td>BOOL</td>
<td>FALSE</td>
<td>TRUE: Application stop is suppressed.</td>
</tr>
<tr>
<td>xDisableApplicationBP</td>
<td>BOOL</td>
<td>FALSE</td>
<td>TRUE: Setting breakpoints is suppressed.</td>
</tr>
<tr>
<td>xDisableApplicationWrite</td>
<td>BOOL</td>
<td>FALSE</td>
<td>TRUE: Writing variables is suppressed. This can also be suppressed via PLCHandler/lecVarAccess.</td>
</tr>
<tr>
<td>xDisableApplicationForce</td>
<td>BOOL</td>
<td>FALSE</td>
<td>TRUE: Forcing variables is suppressed.</td>
</tr>
<tr>
<td>xDisableApplicationReset</td>
<td>BOOL</td>
<td>FALSE</td>
<td>TRUE: Resetting the application (not &quot;Reset origin&quot;) is suppressed.</td>
</tr>
<tr>
<td>xDisableAll</td>
<td>BOOL</td>
<td>FALSE</td>
<td>TRUE: All operations are suppressed.</td>
</tr>
</tbody>
</table>
Creating backup files

Requirement: A project is open with an application that is running on the required device. In addition, for this example an external file myExternalFile.txt is inserted as an object below the application. This file is downloaded to the PLC implicitly when downloading the application.

1. Open the device editor by double-clicking the device entry in the device tree. Click the “Backup and Restore” tab.
   ⇒ The tab opens with a menu bar including the “Backup” and “Restore” menus.
2. In the “Backup” menu, select the “Read Backup Information from Device” item.
   ⇒ If the PLC is not connected at the moment, then CODESYS opens a temporary connection to the device and reads the relevant files from the $PlcLogic$ directory of the PLC into a table in the lower part of the tabbed page. In this example, at least the following files will be listed: $PlcLogic$/Application/Application.app, $PlcLogic$/Application/Application.crc, and $PlcLogic$/Application/myExternalFile.txt. In addition, other external, project-dependent files are listed, which have been inserted below the application in the device tree. Furthermore, the source code archive file $PlcLogic$/Archive.prj is listed if you have set the project setting for this (“Implicitly at program download and online change”) as the loading time.
3. In the table, clear the check box for the $PlcLogic$/Application//myExternalFile.txt file in the “Active” column.
4. Select “Save Backup File to Device” in the “Backup” menu.
   ⇒ The “Save as” dialog opens. The file type is predefine as “Backup files (*.tbf)”. 
5. Select a location for the backup file and click “Save”.

See also
● Chapter 1.3.1.25.3.11.5 “Dialog ‘Project Settings’–‘Source Download’” on page 1063

Restoring from backup files

Requirement: A project is open with an application that is running on the required device. A backup file is saved to the local file system as described above.

1. Open the device editor by double-clicking the device entry in the device tree. Click the “Backup and Restore” tab.
   ⇒ Click “Restore ➔ Load Backup File from Disc”.
   ⇒ The default prompt opens for selecting a backup file tbf in the local file system.
2. Select the backup file and click “Open”.
   ⇒ The files are read from the backup file and shown in the table of the dialog below. The file $PlcLogic$/Application/myExternalFile.txt that was excluded in the backup is missing.
3. Click “Restore ➔ Restore Backup to Device”.
   ⇒ A dialog prompt opens with information about the actions when restoring.
4. Click “OK” to start restoring the files to the PLC file system.
   ⇒ When restore is complete, you are prompted to restart the PLC in order to activate the loaded application.

1.3.1.14 Updating an application on the PLC

CODESYS offers in principle two possibilities to transfer a modified application to the controller: download and online change.
NOTICE!
If you use a device application, please note the restrictions concerning download and online change.

A download leads to a recompilation of the application. In the process, a syntax check is performed and application code is also created and loaded to the controller. This leads to the running program being stopped. A download is the recommended method of data transfer, since a defined starting state is always created due to the program stop and the re-initialization.

In the case of an online change, only the modified parts are reloaded to the controller. A running program is not stopped for this. You should only carry out an online change in the case of minor changes to the application. In the case of extensive changes the behavior of a program cannot be safely predicted. Please read the notes in the description of the “Online Change” command regarding this.

See also
- ‡ Chapter 1.3.1.14.1 “Executing the online change” on page 355
- ‡ Chapter 1.3.1.14.2 “Execution of a download” on page 356
- ‡ Chapter 1.3.1.25.2.6.5 “Command 'Load’” on page 927
- ‡ Chapter 1.3.1.25.2.6.6 “Command 'Online Change’” on page 928

1.3.1.14.1 Executing the online change
CODESYS automatically offers you an online change if you log in with an application that is already present on the controller, but has been changed since the last download in the programming system. With this procedure only the modified parts are reloaded to the controller. A running program on the controller is not stopped during the online change.

In the view “Memory reserve for online change”, you can configure memory reserves for the online change for function blocks of a project. In this way, instance variables do not have to be moved to the memory after changes are made to a function block for an online change.

NOTICE!
When carrying out the online change, pay attention to the notes in the description of the “Online Change” command.

See also
- ‡ Chapter 1.3.1.25.2.6.6 “Command 'Online Change’” on page 928
- ‡ Chapter 1.3.1.25.2.6.2 “Command 'Login’” on page 923
- ‡ Chapter 1.3.1.11.4 “Generating application code” on page 303
- ‡ Chapter 1.3.1.9.4 “Configuring the memory reserve for an online change” on page 206

Executing the online change when logging in
Requirement: The connection settings of the controller are correctly set. The applications in the project and on the controller are identical. The project on the controller is running. The application is logged out.

1. Change your application.
2. Click “Online ➔ Login”.
   ➔ A dialog appears with the information that the application has been changed since the last download.
3. Click the “Details…” button
4. Check the details in the “Application information” tab.
   If you have not generated any code since the last change, the command “Application is not up to date. Generate code now?” appears at the bottom edge of the dialog. In this case click this command.
   ⇐ You are shown a comparison view of the objects (objects marked red are different).
5. Close the dialog.
6. Select the option “Login with Online Change” and click “OK”.
   ⇐ The change is loaded to the controller. The running program on the controller is not stopped while doing this. The application is logged in.

See also
● ☞ “View ‘Project Comparison’ - ‘Differences’” on page 906

### Execute online change in the logged-in state (online operation)

Requirement: The connection settings of the controller are correctly set. The applications in the project and on the controller are identical. The project on the controller is running. The application is logged in.

1. Select an object in the device tree. It is best to select a POU or a GVL here.
2. Click “Project ➔ Edit Object (Offline)”.
   ⇐ The object opens in the editor.
3. Change the object. For example, you can declare a new variable or change a value assignment here.
4. Click “Online ➔ Online Change”.
   ⇐ A query will appear, asking whether you really want to execute the online change.
5. Click “Yes” to confirm the dialog.
   ⇐ The change is loaded to the controller.

### 1.3.1.14.2 Execution of a download

A download of the application causes a compilation of the active application. In the process, a syntax check is performed and application code is also created and loaded to the controller. A program running on the controller is stopped during the download.

See also
● ☞ Chapter 1.3.1.25.2.6.5 “Command ‘Load’” on page 927
● ☞ Chapter 1.3.1.25.2.6.2 “Command ‘Login’” on page 923
● ☞ Chapter 1.3.1.11.4 “Generating application code” on page 303

### Downloading when logging in

Requirement: the connection settings of the controller are correctly set. The applications in the project and on the controller are identical. The project on the controller is running. The application is logged out.

---

1.3.1.14.2 Execution of a download

A download of the application causes a compilation of the active application. In the process, a syntax check is performed and application code is also created and loaded to the controller. A program running on the controller is stopped during the download.

See also
● ☞ Chapter 1.3.1.25.2.6.5 “Command ‘Load’” on page 927
● ☞ Chapter 1.3.1.25.2.6.2 “Command ‘Login’” on page 923
● ☞ Chapter 1.3.1.11.4 “Generating application code” on page 303

### Downloading when logging in

Requirement: the connection settings of the controller are correctly set. The applications in the project and on the controller are identical. The project on the controller is running. The application is logged out.

---

1.3.1.14.2 Execution of a download

A download of the application causes a compilation of the active application. In the process, a syntax check is performed and application code is also created and loaded to the controller. A program running on the controller is stopped during the download.

See also
● ☞ Chapter 1.3.1.25.2.6.5 “Command ‘Load’” on page 927
● ☞ Chapter 1.3.1.25.2.6.2 “Command ‘Login’” on page 923
● ☞ Chapter 1.3.1.11.4 “Generating application code” on page 303

### Downloading when logging in

Requirement: the connection settings of the controller are correctly set. The applications in the project and on the controller are identical. The project on the controller is running. The application is logged out.

---

1.3.1.14.2 Execution of a download

A download of the application causes a compilation of the active application. In the process, a syntax check is performed and application code is also created and loaded to the controller. A program running on the controller is stopped during the download.

See also
● ☞ Chapter 1.3.1.25.2.6.5 “Command ‘Load’” on page 927
● ☞ Chapter 1.3.1.25.2.6.2 “Command ‘Login’” on page 923
● ☞ Chapter 1.3.1.11.4 “Generating application code” on page 303

### Downloading when logging in

Requirement: the connection settings of the controller are correctly set. The applications in the project and on the controller are identical. The project on the controller is running. The application is logged out.

---

1.3.1.14.2 Execution of a download

A download of the application causes a compilation of the active application. In the process, a syntax check is performed and application code is also created and loaded to the controller. A program running on the controller is stopped during the download.

See also
● ☞ Chapter 1.3.1.25.2.6.5 “Command ‘Load’” on page 927
● ☞ Chapter 1.3.1.25.2.6.2 “Command ‘Login’” on page 923
● ☞ Chapter 1.3.1.11.4 “Generating application code” on page 303

### Downloading when logging in

Requirement: the connection settings of the controller are correctly set. The applications in the project and on the controller are identical. The project on the controller is running. The application is logged out.

---

1.3.1.14.2 Execution of a download

A download of the application causes a compilation of the active application. In the process, a syntax check is performed and application code is also created and loaded to the controller. A program running on the controller is stopped during the download.

See also
● ☞ Chapter 1.3.1.25.2.6.5 “Command ‘Load’” on page 927
● ☞ Chapter 1.3.1.25.2.6.2 “Command ‘Login’” on page 923
● ☞ Chapter 1.3.1.11.4 “Generating application code” on page 303

### Downloading when logging in

Requirement: the connection settings of the controller are correctly set. The applications in the project and on the controller are identical. The project on the controller is running. The application is logged out.

---

1.3.1.14.2 Execution of a download

A download of the application causes a compilation of the active application. In the process, a syntax check is performed and application code is also created and loaded to the controller. A program running on the controller is stopped during the download.

See also
● ☞ Chapter 1.3.1.25.2.6.5 “Command ‘Load’” on page 927
● ☞ Chapter 1.3.1.25.2.6.2 “Command ‘Login’” on page 923
● ☞ Chapter 1.3.1.11.4 “Generating application code” on page 303

### Downloading when logging in

Requirement: the connection settings of the controller are correctly set. The applications in the project and on the controller are identical. The project on the controller is running. The application is logged out.

---

1.3.1.14.2 Execution of a download

A download of the application causes a compilation of the active application. In the process, a syntax check is performed and application code is also created and loaded to the controller. A program running on the controller is stopped during the download.

See also
● ☞ Chapter 1.3.1.25.2.6.5 “Command ‘Load’” on page 927
● ☞ Chapter 1.3.1.25.2.6.2 “Command ‘Login’” on page 923
● ☞ Chapter 1.3.1.11.4 “Generating application code” on page 303

### Downloading when logging in

Requirement: the connection settings of the controller are correctly set. The applications in the project and on the controller are identical. The project on the controller is running. The application is logged out.

---

1.3.1.14.2 Execution of a download

A download of the application causes a compilation of the active application. In the process, a syntax check is performed and application code is also created and loaded to the controller. A program running on the controller is stopped during the download.

See also
● ☞ Chapter 1.3.1.25.2.6.5 “Command ‘Load’” on page 927
● ☞ Chapter 1.3.1.25.2.6.2 “Command ‘Login’” on page 923
● ☞ Chapter 1.3.1.11.4 “Generating application code” on page 303

### Downloading when logging in

Requirement: the connection settings of the controller are correctly set. The applications in the project and on the controller are identical. The project on the controller is running. The application is logged out.
1. Change your application.
2. Select the command “Online ➔ Login”
   ⇒ A dialog box appears with the information that the application has been changed since
   the last download.
3. Select the option “Login with download” and click on “OK”.
   ⇒ The running program on the controller is stopped and the change is loaded to the con-
   troller. The application is logged in.

**Downloading in the logged-in state (online mode)**

Requirement: the connection settings of the controller are correctly set. The applications in the
project and on the controller are identical. The project on the controller is running. The applica-
tion is logged in.

1. Select an object in the device tree. It is best to select a POU or a GVL here.
2. Select the command “Project ➔ Edit Object (Offline)”
   ⇒ The object opens in the editor.
3. Change the object. For example, you can declare a new variable or change a value
   assignment here.
4. Select the command “Online ➔ Download”.
   ⇒ A query will appear, asking whether you really want to execute the download.
5. Confirm the dialog box with “Yes”.
   ⇒ The running program on the controller is stopped and the change is loaded to the con-
   troller.

1.3.1.15 Copying files to/from PLC

In the generic “Files” tab of the device editor, you can copy files to and from the local file system
and the controller.

Requirement: The vendor has unlocked the tab. In the device tree, the connection to the con-
troller is configured. The device is running.

1. Double-click the PLC device object in the device tree to open the device editor.
2. Click the “Files” tab.
3. In “Host” | “Location” on the left part of the view, set the path in the local file system where
   files will be copied to and from. Example: D:\FileTransferWithPLC. If necessary,
   create a new directory by clicking the folder symbol ( ).
   ⇒ The files and directories are shown like in a file manager. Click the refresh symbol ( )
   to update the display.
4. In “Runtime” on the right side of the view, set the required directory for the data transfer in
   the same way.
   ⇒ CODESYS shows the files on the controller.
5. Select the required files from the file system tree for the file transfer (multiple selection is
   possible). You can also select a directory for transferring all files in a folder.
6. Click the left and right arrow symbols ( , ) between the two parts of the view.
   ⇒ CODESYS copies the selected files to the other file system immediately. If a file is not
   already available in the target directory, then it is created. If it is already available and
   not write-protected, then it is overwritten. Otherwise a message is shown.

See also
● ⇥ Chapter 1.3.1.25.1.19.7 “Tab ‘Files’” on page 750
### 1.3.1.16 Using device applications

Using a device application allows for the separation of globally required data from exclusively control programs:

The global data that is used by all control applications of a device are managed in their own "Device Application" object. They are compiled fully with this object and downloaded to the controller at the time of commissioning. The "Device Application" does not include control programs developed by the user. Because of this, the exclusively control applications of the device are smaller and require less time to download. Reason: The global data does not have to be included at every download or online change. However, loading the global data requires more memory and download time as when using exclusively single applications. Reason: ALL referenced data and the entire code are downloaded to the controller with the device application because it is not clear what is used later or not.

Therefore, using a device application instead of single applications makes sense when

- the device in use has enough memory (for example not an embedded system),
- and if the control tasks should be distributed to multiple applications for improved logical mapping.

**Organization in the device tree:**

- Depending on the device, you can insert the "Device Application" object in the device tree below the "PLC Logic" node of the device. By using different types of child objects, this manages the global data and code segments that are used by all exclusively control applications of the device. Examples: library manager, GVLs, network variable list. The "PLC Logic" node is hidden after the device application is inserted into the device tree.

- The exclusively control program code is contained in one or more ordinary application objects that are displayed in the tree in parallel to the "Device Application". In the hierarchy however, they are ranked below the "Device Application" and can access the global data and variables of the device application. They do not have mutual access to each other and also cannot have any child applications. However, data exchange is possible via the global data of the device application.

**Control and behavior:**

- An online change is not possible for the device application. Therefore, changes to the device application lead to a new download of the device application and the child applications (in the device tree on the same level).

- After a download or reset (warm or cold), the device application is then in status "Stop". You can login, logout, start, stop, and reset all applications of the device application, individually or collectively, via corresponding context menu commands.

- At the first login and download of an application, the device application is also downloaded automatically.

- When commissioning the device, you can use the online configuration mode for testing the I/Os. When commissioning is complete, you must then perform a new download for the applications and the device application.

- If you delete the device application from the device, then its child applications are also deleted from the device.

- When generating a boot application for an application of the device, a boot application is also generated automatically for the device application.

- By default, each application has the property that the device application stops, as well as all other applications, in case of an exception.
● Notes about task configuration:
POUs cannot be called in the task configuration of the device application. This task configuration is used by bus masters in the field in order to reach an approximation of the timing of the bus cycle task when the program is executed. Depending on the device, there are two possibilities:

– You define a task of type “External” in the task configuration of an application. As an external event for triggering the task, you specify the call of the implicitly provided bus master event. This allows for reaching an asynchronous timing compensation with the bus master.

– You define a task of type “Parent task” in the task configuration of an application. For the parent task, you select the cyclic tasks of the bus master from the provided tasks of the device application. This allows for reaching a synchronous timing compensation with the bus master.

Please note the following restriction: Whether or not synchronous calling is executed depends on the state of the device application only. If this is running and the child application is not running, then the code is called anyway.

● Notes about persistent variables:
If the device supports static memory areas for persistent and retain variables, then you can use the “Persistent Variable configuration” object to map the memory also into global and application-specific for remanent data. This makes sense so you do not have to download the device application and all applications to the controller again after a persistent variable list is modified. By default, for devices with static memory management, the entire memory is reserved in the device application for remanent data. However, as soon as you create a persistent variable list in an application, you must also provide a memory area for it there. Even an empty list requires at least a few bytes of memory. For devices without static memory allocation, an individual memory segment is allocated automatically for persistent data for each application.

● Note for symbol configuration:

– A “Symbol Configuration” as a child application of a “Device Application” is not supported.

– You can also apply the references to the symbol configuration libraries of an application in the library manager of the device application. The advantage of this is that the libraries do not have to be downloaded with the application, and therefore the applications are smaller. Refer to the option “Add library placeholder to device application” when adding a “Symbol Configuration” object, or the command with the same name in the “Tools” menu of the symbol configuration.

See also
● ° Chapter 1.3.1.25.1.2 “Object ’Device Application’” on page 712
● ° Chapter 1.3.1.25.2.4.35 “Command ’Online Config Mode’” on page 915
● ° Chapter 1.3.1.25.3.10.9 “Dialog ’Properties’ - ’Application Build Options’” on page 1051
● ° Chapter 1.3.1.25.1.4 “Object ’Persistent Variable Configuration’” on page 714
● ° Chapter 1.3.1.25.1.36 “Object ’Symbol Configuration’” on page 824

1.3.1.16.1 Creating device applications
Requirement: The device you are programming supports a device application.
1. Click “Project ➔ New Project”.
2. Select the “Standard project” template in the “New Project” dialog (“Projects” category). Specify a project name and a storage location, and then click “OK”.
⇒ The “Standard Project” dialog opens for selecting the device and the implementation language for a standard PLC_PRG POU.
3. Select a device (for example, CODESYS Control Win V3) and “Structured Text (ST)”.
4. Delete the “Application” object in the device tree.
5. Select the “PLC Logic” object and click “Add Object”.
6. Add the “Device Application” object.
7. Select the “Device Application” object (1) and click “Add Object”.
   ➢ The drop-down list offers the following objects:
     ● “Application”: The application object is inserted into the tree parallel to the device application, below the device object, but below the device application in the hierarchy. Child application of applications are not possible.
     ● “Global Variable List”
     ● “Network Variable List (Receiver)”
     ● “Network Variable List (Sender)”
     ● “Persistent Variables”
     ● “Recipe Manager”
     ● “Task Configuration”

8. Insert the necessary objects for managing the data and variables that should be available for all applications of the device.

9. Configure the task configuration of your application. As a rule, the task of the bus master is in the task configuration of the device application. In order to harmonize the application processing with the timing of the bus cycle task, you can create a task of type “External” in an application and assign the implicitly provided bus master event to it as the triggering “External event”. As an alternative, synchronous harmonization is possible by specifying the bus cycle task of the device configuration as a "Parent task".

10. To use persistent variables when the device supports static memory, you can perform a specific memory configuration in the project. Aim of this configuration: The memory for persistent variables is no longer 100% for the device application. For this purpose, the “Persistent variable configuration” object is inserted below the device application and the memory segments of the applications are configured in its editor.

In the figure, you can see the example of a device tree with (1) device application and two applications with different task areas. In addition, the tree contains an (4) EtherCAT bus master and a (3) bus task defined in IO_Application. This bus task is triggered by an asynchronous external event triggered by the bus master. Moreover, persistent variable lists and a (2) persistent variable configuration object for the memory configuration of the remanent data are created here.
Inserting into existing project with conventional application organization

To insert the “Device Application” object into an existing project with conventional application organization, proceed as follows:

- Create a new device entry in the tree.
- Delete automatically generated applications below the “PLC Logic” node.
- Insert a “Device Application” object for the selected “PLC Logic” node.
- Move the necessary objects from the old device structure to the new structure and delete the previous device entry.

See also

- Chapter 1.3.1.16 “Using device applications” on page 358
- Chapter 1.3.1.25.1.4 “Object ‘Persistent Variable Configuration’” on page 714
- Chapter 1.3.1.25.1.38.1 “Tab ‘Configuration’” on page 839
1.3.1.17 Using redundant PLC devices, CODESYS Redundancy

In the CODESYS Development System, redundancy means that an application can run on two CODESYS PLCs, where only one PLC is active at a given time. If the active PLC fails, then the other PLC takes over without any interruption of the fieldbus control. Even the inactive PLC (standby PLC) always executes the program, reads inputs, and can synchronize with the active PLC over a communication line.

When starting up the redundant system, the PLC that was started first enters standalone mode until the second PLC is started and synchronized. Then the first one switches to active state and the second one switches to passive state (standby). If both PLC devices start at the same time, then the PLC with the higher IP address becomes the active PLC.

The participating PLC devices must have a corresponding runtime system extension. By default, a "Redundancy" object is provided with the accompanying configuration editor in the user interface of the CODESYS Development System.

CODESYS Redundancy supports EtherCAT (3S-Smart Software Solutions GmbH stack) and Profibus over Hilscher CIF50 adapters.

The figure below presents a possible constellation with two redundant PLC devices (active PLC, standby PLC). Programming means the CODESYS Development System.

To operate an application on redundant PLCs, add the "Redundancy Configuration" object to the project. It provides a configuration editor where you define the memory areas of the application to be controlled, the relevant task, and a watchdog. It also provides a display of the current statuses of the both PLC devices at runtime. You can start trigger the switching of roles from passive to active either in the configuration editor or with a library function.

The boot application or the registered memory areas must be updated only on one of the two PLCs. The variable values in the registered memory areas are always synchronous for both PLC devices.

The logger of the runtime system records redundancy events on both devices.

See also

● Chapter 1.3.1.25.1.32 “Object 'Redundancy Configuration' and Redundancy Editor” on page 815
1.3.17.1 Configuring redundant PLC devices

See below for steps how to configure an application for redundant mode on two PLCs.

**NOTICE!**

**User management**

If a user management is used on the PLC, then you must configure the same user name and the same password for access to both PLC devices. Otherwise, online services, such as write variable or online change, are not transmitted to the inactive PLC.

Requirement: The runtime system components for CODESYS Redundancy are installed on both PLC devices. The PLCs are running. They are named PLC1 and PLC2. The project with the respective application is open in CODESYS Development System.

Configure the project

1. Select the application object in the device tree and insert the “Redundancy Configuration” object.
   ⇒ The “Redundancy Configuration” dialog box opens.

2. If you have already configured the communication settings for the top-level PLC, then CODESYS accepts them automatically for PLC1 and displays them in this dialog. To set it as active, open the “Communication” tab by clicking the “Set Active Path PLC 1” button.

3. Click the “Set Active Path PLC 2” button in the “Redundancy Status” tab. Define the communication path for PLC2.

4. In the “Redundancy Configuration” editor, select the “Redundancy Settings” tab and then the “Connection” tab. Specify the IP addresses of both PLC devices in “IP address PLC 1” and “IP address PLC 2”. If possible, select a separate port especially for the connection between PLC devices only. Note: The configuration settings may deviate for OEM-specific redundancy interconnections.

5. Switch to the “General” tab. Define the settings, such as the name of the application task that should control the processing (“Redundancy task”), the watchdog, the services to be executed, and the fieldbuses used.

   EtherCAT settings: The source address for PLC1 must agree with the address used in the EtherCAT master. The source address of PLC2 is the adapter that is used for EtherCAT for PLC2.

6. When both PLCs are running, click the “Write” button in the lower right of the configuration dialog in order to specify the settings in the configuration files of both PLC devices.

7. Optional: For our example, extend the application here by a programmatic query of the redundancy status. Insert a “POU” of type “Program”. An example is shown below this list of steps.

8. If necessary, stop both PLC devices because the changed settings become active only when the runtime system is restarted. Restart in the order described below.

Example of a program for retrieving the redundancy status

```
PROGRAM PLC_PRG
VAR
  state: RedundancyState;
  bResult: BOOL;
END_VAR

bResult := RDCY.GetRedundancyState(ADR(state));
```
You can always follow the status change described in the following instructions by monitoring the variable `state.eRedundancyState` in “PLC_PRG”, as well as in the “Redundancy Status” tab of the “Redundancy Configuration” editor.

### Download the application to PLC1
1. Start PLC1. (The first of the two PLCs that is started enters standalone mode.) Open the online view of “PLC_PRG” and the “Redundancy Status” tab of the “Redundancy Configuration” editor.
   - The variable `state.eRedundancyState` contains the value `RS_CYCLE_STANDALONE`. In the graphical representation, the symbol  appears at PLC1 and the symbol  appears at PLC2.
2. Select the application in the device tree. Click “Online ➔ Login”. Download the application to PLC1 and start the application by clicking “Debug ➔ Start”.
3. If the automatic creation of the boot application is not activated in the application properties, then click “Online ➔ Create Boot Application”.

### Synchronize PLC1 and PLC2
1. Start PLC2.
   - PLC2 reads the boot application from PLC1 and synchronizes the data and the task execution. The redundancy status of PLC1 switches to active. The variable `state.eRedundancyState` receives the value `RS_CYCLE_ACTIVE`. In the graphical representation, the symbol  appears at PLC1. PLC2 switches to the passive state at the same time. Both PLC devices are now running in redundancy mode.
2. To check that both PLC devices execute the same application, you can perform the following steps:
   - Log out of PLC1 by clicking “Online ➔ Logout”.
   - Open the “Communication” tab for PLC1 by double-clicking the device object in the device tree where the application is located. Change the connection settings to PLC2.
   - Click “Online ➔ Login”. There is no message regarding a download. PLC2 applies the boot application from PLC1 automatically.
   - Important: Change the communication settings back to PLC1.
   - Check the redundancy status of PLC2: The variable `state.eRedundancyState` receives the values `RS_CYCLE_STANDBY`. In the graphical representation, the symbol  appears at PLC2.
3. To simulate unexpected failures of PLC1, perform the following steps:
   - Stop PLC1. As a result, PLC1 shows status "Unknown" and PLC2 switches immediately to status "Standalone".
   - Start PLC1. PLC1 has status "Passive" and PLC2 switches to "Active".
4. You can perform the following actions by means of the buttons in the “Redundancy Status” tab:
   - “Switchover”: The active PLC is switched to status "Passive".
   - “Simulation”: The active PLC is switched to status "Simulation" (for testing purposes).
   - “Activate”: The PLC in simulation mode is switched to status "Active".
   - “Synchronize”: The PLC in simulation mode or in error status synchronizes with the other PLC, applies its data, and becomes "Passive".

See also
- § Chapter 1.3.1.25.1.19.2 “Tab ‘Communication Settings’” on page 741
- § Chapter 1.3.1.25.1.32 “Object ‘Redundancy Configuration’ and Redundancy Editor” on page 815
**1.3.1.17.2 Restrictions**

**CODESYS version**
CODESYS Redundancy requires CODESYS V3.5 SP2 or later. The same version of the runtime system must be installed on both PLC devices.

**Real-time tasks and communication**
CODESYS Redundancy synchronizes 1 task. Other tasks and applications are possible, but they run non-synchronized on both PLCs.

CODESYS Redundancy requires real-time task execution and real-time communication. Real-time task execution means that the application task (redundancy task) that is controlled by redundancy has a limited jitter. Real-time communication means that a message that is sent by means of the redundancy interconnection is received by the second PLC within a specific time. The redundancy timeout can be defined in the CODESYS Development System with the "Redundancy Configuration" editor ("Redundancy Settings"). In the runtime system configuration file CODESYSControl.cfg, it is specified by the entry StandbyWaitTime.

The value for the timeout for redundancy must be greater than the sum of the task jitter time and the maximum communication jitter time. These times depend on the system.

Real-time task execution and real-time communication are required for indicating a specific maximum timeout. If this kind of timeout occurs while waiting for a message from the other PLC, then the system assumes that the other PLC is not running anymore. As a result, the waiting PLC and the other PLC switch to standalone mode.

If the task jitter is too high, or the message transmission is delayed, then it is possible for both PLCs to switch to standalone mode. The means that there are synchronization losses and communication problems on the fieldbus.

**IEC timer**
Different execution times on both PLC devices can cause bumps (deviating output values) when switching PLCs. To prevent this, the IEC timer values are frozen during the execution of an IEC task. Calls from IEC timers (example: TON) when executing an IEC task therefore always lead to the same timer values, even if the physical time continues running. This means that it is not possible, for example, to wait actively in a loop, because the IEC timer values do not change in the current task scan.

**POINTER**
POINTER variables must not be declared in data areas that are controlled by redundancy. Reason: Redundancy-controlled values are transmitted to the other PLC during synchronization. However, pointer values are not valid on the other PLC because another memory layout may be located there.

When compiling, the redundancy feature checks that a pointer variable is located in a redundancy-controlled area. A warning is issued for each pointer variable that is detected in such an area. The check can be deactivated in the device description file with the following entry:

```
<Device>
  <Custom>
    <Redundancy DisablePointerChecks="true">
```

**EtherCAT DC**
This redundancy extension is designed more for the process industry than for factory automation. Therefore, EtherCAT drives with distributed clocks are not supported. However, EtherCAT I/Os are supported.

**Map on existing: mapping to existing variables**
The I/O mapping method "map on existing" (mapping I/Os to existing variables) is not recommended for use with CODESYS Redundancy. These variables are not stored in the input or output data areas, but where they are declared. Therefore, they are not synchronized during operation.

**Network variables**
Network variables with write access must not be used because multiple write telegrams are sent at the same time. Network variables with read access are permitted.
File access must not be used because different file data can cause bumps on the different PLCs when they are switched.

If you use files, then you must declare file handles in the data areas that do not subjected to redundancy control. The files must be opened separately on both PLCs, and the file handle of another PLC must not be used for accessing the files on the local PC.

When compiling, CODESYS Redundancy checks that handle variables (RTS_IEC_HANDLE, CAA.HANDLE) are located in a redundantly controlled area. A warning is issued for each handle variable that is detected in such an area.

Online security user management

If an online security user management is used, then you must configure both PLC devices with the same user name and password. Otherwise, online services, such as write variable or online change, are not transmitted to the passive PLC.

1.3.1.18 Using the command-line interface

You can start CODESYS.exe from the command line with the following switches and options.

Syntax:

```
<folder>CODESYS.exe> --<switch or option>
```

Paths or option parameters must be written inside straight quotation marks when they contain spaces, dashes, or slash marks.

Switch --culture (language of the user interface)

In the command line, include this switch after the development system call in order to set the language of the user interface.

Syntax:

```
--culture=<culture>
```

Example

Starting CODESYS with the user interface in English:

```
CODESYS.exe --culture=en
```

See also

- Chapter 1.3.1.25.3.13.12 “Dialog ‘Options’ – ‘International Settings’” on page 1083

Switch --profile (CODESYS profile)

In the command line, include this switch after the development system call in order to start CODESYS with a specific profile. When you start CODESYS without this switch, the “Select Profile” opens.

Syntax:

```
--profile="<profile name>"
```

Example

```
CODESYS.exe --culture=de --profile="CODESYS V3.6"
```

See also

- Chapter 1.3.1.2 “Your first CODESYS program” on page 108
Switch --compare (start project comparison)

In the command line, include this switch after the development system call in order to perform a comparison between two CODESYS projects. Type the path of the project file after the switch and then the path of the reference project. CODESYS starts and opens the “Project Comparison - Differences” view.

Syntax:

```
--compare="<path of project file>" "<path of reference project file>"
```

Example

```
CODESYS.exe --compare "D:\proj\project1.project" "D:\proj\project2.project"
```

See also

- "Chapter 1.3.1.25.2.4.18 “Command 'Compare'” on page 905"

Option --project (CODESYS project)

In the command line, include this option after the development system call in order to open the given project in CODESYS.

Syntax:

```
--project="<path of project file>"
```

Example

```
<path of project file>: File path of project
```

Open the test project:

```
CODESYS.exe --culture=de --project="D:\projects\test.project"
```

See also

- "Chapter 1.3.1.25.2.1.2 “Command ‘Open Project’” on page 853"

Option --projectarchive (CODESYS project archive)

In the command line, include this option after the development system call in order to extract the given project archive and open the project in CODESYS.

Syntax:

```
--projectarchive="<path of project archive file>"
```

Example

```
<path of project archive file>: File path of project archive
```

Extract the test.projectarchive and open the project in the development system:

```
CODESYS.exe --projectarchive="D:\projects\test.projectarchive"
```

See also

- "Chapter 1.3.1.25.2.1.9 “Command ‘Extract Archive’” on page 858"

Option --runscript (execute script)

In the command line, include this option after the development system call in order to execute a given script file from CODESYS.

Table 10: “Command Line Options for --runscript”

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>--runscript=&quot;&lt;scriptfile&gt;.py&quot;</td>
<td>CODESYS executes the &lt;scriptfile&gt;.py script file at startup. You must provide the complete path of the script file.</td>
</tr>
<tr>
<td>--scriptargs:'&lt;arg1&gt; &lt;arg2&gt; ... &lt;argn&gt;'</td>
<td>Use this option with the --runscript option. As a result, the parameters &lt;arg1&gt; ... &lt;argn&gt; are passed to the script. The arguments are forwarded to the Python variable sys.argv.</td>
</tr>
<tr>
<td>Option</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>--noUI</strong></td>
<td>Use this option with the <strong>--runscript</strong> option. The CODESYS user interface is not opened. CODESYS prints all errors, warnings, compiler reports, and command-line messages generated from the script. The script messages (1: Severity Text) can be separated from other messages (2: Severity FatalError, Error, Warning, Information) with the &quot;-&gt;&quot; operator.</td>
</tr>
<tr>
<td><strong>--enableScriptTracing</strong></td>
<td>Use this option with the <strong>--runscript</strong> option. As a result, each command of the script file is shown in the output.</td>
</tr>
<tr>
<td><strong>--textPrompts</strong></td>
<td>Use this option with the <strong>--noUI</strong> option. As a result, message service methods and default dialogs are output in the command line for user input. If you do not specify <strong>--textPrompts</strong>, then all message service prompts are confirmed automatically with default values.</td>
</tr>
<tr>
<td><strong>scriptdebugger {=&quot;&lt;debugger&gt;&quot;}</strong></td>
<td>Use this option with the <strong>--runscript</strong> option. It sets IronPython in debug mode so that external debuggers can be used to debug Python scripts. The following values are defined for &lt;debugger&gt; (uppercase/lowercase is irrelevant).</td>
</tr>
<tr>
<td></td>
<td>* auto: Automatically detects if a debugger is included in every script for the current process. At this time, only .NET-based debuggers can be detected automatically. A detected debugger overwrites the <strong>--enableScriptTracing</strong> flag.</td>
</tr>
<tr>
<td></td>
<td>* .NET: Activates debugging for .NET-based debuggers, such as &quot;Python Tools for Visual Studio&quot; (PTVS) and SharpDevelop. With this option, a debugger can also be included in running scripts, as opposed to &quot;auto&quot;.</td>
</tr>
<tr>
<td></td>
<td>Note: This is currently the default value when <strong>scriptdebugger</strong> is used without providing a value.</td>
</tr>
<tr>
<td></td>
<td>* disabled: Deactivates debugging and automatic detection.</td>
</tr>
<tr>
<td></td>
<td>* script: Switches the IronPython script engine to debug mode for activating the debugging for set-trace debuggers. The script itself must connect to and disconnect from the debugger.</td>
</tr>
<tr>
<td></td>
<td>* tracing: Activates the simple integrated script tracing mode and deactivates the automatic detection (same as the <strong>--scriptTracing</strong> switch).</td>
</tr>
<tr>
<td></td>
<td>* $absolute_path.py$: Absolute path to a Python script that initiates the connection to a Python-based debugger. The IronPython script engine is switched to debug mode for allowing the debugging for set-trace debuggers. This script is executed one time during the initialization and should define the following non-parameterized functions:</td>
</tr>
<tr>
<td></td>
<td>* scriptdebuggersetup is executed immediately before executing the user script to establish the connection to the debugger.</td>
</tr>
<tr>
<td></td>
<td>* scriptdebuggershutdown is called immediately after executing the user script or when the script engine is downloaded and should close the connection to the debugger.</td>
</tr>
</tbody>
</table>
Examples of using transfer parameters in script files with 'sys.argv'  

```python
start /b /wait CODESYS.exe  
--runscript="D:\Script\ArgvAnd__main__Test.py"  
--scriptargs:'username password 3.14 "path="C:\temp\""
```

**Script file:** ArgvAnd__main__Test.py  
```python
from __future__ import print_function
import sys
print("sys.argv: ", len(sys.argv), " elements:")

for arg in sys.argv:
    print(" - ", arg)
print()
print("__name__: ", __name__)
```

**Output result:** std out:  
```
sys.argv: 6 elements:
 - D:\TestScripts\ArgvAnd__main__Test.py  
 - username  
 - password  
 - 3.14  
 - path= "C:temp"
__name__: __main__
```

For more information about the __name__ global variable, see the Python documentation.

**Examples of the message output**  
```
start /b /wait CODESYS.exe --runscript="D:\Script\AmpelTest.py" --noUI 1>ScriptMessages.txt
```

CODESYS forwards all messages that are generated by the script to the ScriptMessages.txt file. Other messages are printed to the command line.

```
start /b /wait CODESYS.exe --runscript="D:\Script\AmpelTest.py" --noUI 2>NUL
```

CODESYS suppresses all messages, except for script messages. The script messages are printed to the command line.
The following `initdebug.py` script was tested successfully with `pydevd`-based debuggers, such as PyDev / LiClipse and PyCharm. To use this script, start CODESYS with the following command line:

```
--profile="Fanta Development Build" --scriptdebugger="D:\test\charmdebug\initdebug.py"
```

**File:** `initdebug.py`

```python
from _future_ import print_function
from _future_ import unicode_literals
import sys
sys.path.append(r"D:\test\Env2\Lib\site-packages\pycharm-debug.egg")
import pydevd
def scriptdebuggersetup():
    pydevd.settrace('localhost', port=51234, stdoutToServer=True, stderrToServer=True)
def scriptdebuggershutdown():
    pydevd.stoptrace()
```

See also

- Chapter 1.3.1.22 “Using scripts” on page 388
- [http://docs.python.org/tutorial/modules.html](http://docs.python.org/tutorial/modules.html)

**Option -- ignorewhitespace**

In the command line, add this option to the development system call after the option `--compare <project1> <project2>` in order to ignore spaces in the project comparison. Note that semantically relevant spaces, for example in STRING literals, are still taken into account.

**Syntax**

```
--compare="<path of project file>" "<path of reference project file>"
--ignorewhitespace="true"|"false"
```

**Example**

```
CODESYS.exe --compare "D:\proj\project1.project" "D:\proj \project2.project" --ignorewhitespace="true"
```

See also

- Chapter 1.3.1.25.2.4.18 “Command ’Compare’” on page 905

**Option -- ignorecomments**

In the command line, add this option to the development system call after the option `--compare <project1> <project2>` in order to ignore comments in the project comparison.

**Syntax**

```
--compare="<path of project file>" "<path of reference project file>"
--ignorecomments="true"|"false"
```

**Example**

```
CODESYS.exe --compare "D:\proj\project1.project" "D:\proj \project2.project" --ignorecomments="true"
```

See also

- Chapter 1.3.1.25.2.4.18 “Command ’Compare’” on page 905

**Option -- ignoreproperties**

In the command line, add this option to the development system call to ignore object attributes (access privileges, compile settings, directories, bitmaps, etc.) in the project comparison.
**Syntax:**
--compare="<path of project file>" "<path of reference project file>"
--ignoreproperties="true"|"false"

**Example**
CODESYS.exe --compare "D:\proj\project1.project" "D:\proj\project2.project" --ignoreproperties="true"

See also
● ⇐ Chapter 1.3.1.25.2.4.18 “Command ‘Compare’” on page 905

**Option --skipunlicensedplugins (component license missing)**

In the command line, add this option to the development system call to skip the prompt whether unlicensed components should still be loaded. If so, then CODESYS does not load these components by implication.

**Example**
CODESYS.exe --skipunlicensedplugins

See also: ⇐ Chapter 1.3.1.21 “Managing packages and licenses” on page 375

1.3.1.19 Using libraries

1.3.1.19.1 Information for library developers................................. 372
1.3.1.19.2 Adding a library to the application................................. 373
1.3.1.19.3 Adding a library to the repository................................. 373
1.3.1.19.4 Exporting library files............................................... 374

**Library repository**

The library repository is the storage location on the development system for libraries and associated metadata. You can link any installed the libraries into your project by means of a library manager. Moreover, the libraries are installed with version management for easy library updates.

You can create and edit more repositories in addition to the preinstalled System repository.

See also
● ⇐ Chapter 1.3.1.25.2.8.2 “Command ‘Library Repository’” on page 951

**Library manager**

The library manager manages the libraries that are linked into your project. It displays the libraries by the respective library types and properties. You can link more libraries from the library repository, remove them, or edit their properties.

The library manager can be inserted into the “POUs” view or the “Devices” view. In this way, a project can have one library manager per application and also one cross-application library manager in the “POUs” view. The library modules of the linked libraries in the “POUs” view can be called regardless of the application. The library modules of the linked libraries in the “Devices” view can be called in the respective application code only. Furthermore, placeholder libraries behave differently when downloading depending on their positions.

Libraries that are linked to a specific version in the project also have a placeholder for that version (placeholder library). You can define special placeholder resolutions. You can also use the placeholder resolution that is defined for a device in the device description or that is stored in the library repository for a library. The library manager notifies about the actual placeholder resolution and shows the version that is used when downloading an application (effective version).

If a library manager is linked cross-application in the “POUs” view, then you have global access to its contents. If placeholder libraries are linked, then only the placeholder resolutions in the device description or library repository are checked.
Normally, a library manager is linked in the "Devices" view. Then, only the application code below it calls the library modules. Moreover, the special placeholder resolutions are checked first for placeholder libraries. Only after that are the placeholder resolutions checked that are in the device description or that originate from the library repository.

See also

- Chapter 1.3.1.25.1.25 “Object 'Library Manager’” on page 771

1.3.1.19.1 Information for library developers

In order to avoid consistency problems and to adequately support the user, be sure to adhere to certain rules for the creation, referencing, encryption, protection, and documentation of libraries.

The following description provides only an overview of the library development possibilities. For a more detailed description of these topics, see the "LibDev-Summary" guidelines for library development.

General

- You can define categories for libraries. The libraries are then displayed in the library repository below these categories.
- You can define a namespace for a library in order to enable unambiguous access to the integrated objects. The access becomes unambiguous by adding the namespace in front of the module name:
  `<namespace>.<variable name>`
  Example: AC.Module
- You can open the POUs of unencrypted libraries (`*.library`) by double-clicking the respective entry in the Library Manager.
- You can create the following library types:
  - `*.library`: Implementation library (source code of the library)
  - `*.compiled-library, *.compiled-library-v3`: Protected library; source code no longer accessible.
  - `*.Itfs.library`: Interface library
  - `*.Cnt.library`: Container library
- You can integrate external libraries into the application. External libraries are programmed outside of CODESYS in a different programming language, for example C.

Protection of libraries

- Source code protection:
  When a library is prepared in "compiled-library" format, the source code of the library modules is no longer visible after the library is integrated into a project.
- Signing:
  In CODESYS V3 SP15 and later, a certificate is always used for the signing of library projects (`*.compiled-library-v3`). The signing can be enforced by means of a setting in the security screen. Then for generating a compiled library, you need a certificate suitable for code signing in your user profile.
  For library projects that have to be compatible with CODESYS < V3 SP15 (`*.compiled-library`), only the less safe signing is possible with a private key and a corresponding token. These deprecated methods should only be used for reasons of compatibility. Settings are configured in the “Project Information” on the “Signing” tab.
- Licensing:
  You can protect libraries by means of a license (dongle or soft container). License-protected libraries can be installed in the library repository. However, for use in the project, the valid license has to exist on the computer. Licenses are managed in the License Manager.
Library versions

- You can have several versions of a library installed on the system at the same time.
- You can have several versions of a library integrated into your project at the same time. However, we do not recommend doing this. In this case, each of the libraries must be assigned a unique namespace and access to the symbols must be qualified. Examples: V1.SendBlob, V2.SendBlob

Referenced libraries

- You can integrate a library into other libraries (referenced libraries). The nesting can be of any depth.
- You can define whether referenced libraries should be visible in the Library Manager.
- You can integrate referenced libraries via library placeholders. This way you avoid the problems that could occur due to version dependencies or the necessity to use vendor-specific libraries.

See also

- Chapter 1.3.1.3.6.1 “Retrieving and editing project information” on page 132

1.3.1.19.2 Adding a library to the application

The following instructions describe how to integrate the Util library into your application.

1. Select the library manager and open it in the editor with the command “Project ➔ Edit Object”.
   - The library manager opens in the editor.
2. Execute the command “Library ➔ Add Library”.
   - The “Add Library” dialog box opens.
3. Browse for the library by entering the string 'util' in the input field above.
   - The Util library appears in the library view.
4. Select the Util library and close the dialog box with “OK”.
   - The Util library is added in the library manager.

See also

- Chapter 1.3.1.25.2.14.1 “Command ‘Add Library’” on page 1004
- Chapter 1.3.1.19.3 “Adding a library to the repository” on page 373

1.3.1.19.3 Adding a library to the repository

The following instructions describe how to install a library in the library repository.

1. Select the command “Tools ➔ Library Repository”.
   - The dialog box “Library Repository” opens.
2. Click on the “Install” button.
3. Select the library that you wish to install. You can set a file filter here.
   - The library is added to the repository. The library can now be added in the Library Manager.
1.3.1.19.4 Exporting library files

You can export a library from the library manager of a project or from the library repository and then save it as a file to the hard disk.

**Export from the library manager**

1. Open a library manager of an application in a project.
2. Select a library in the library manager.
3. Click the export command in the context menu.
   - The “Export Library” dialog box opens.
4. If the selected library is linked in the project not only as a compiled library, but also in source format, then both file types are in the drop-down list for “File type”. Otherwise, the filter automatically shows the available type: “*.library” or “*.compiled-library”.
5. Select the file type and storage location and click “Save”.

**Export from the library manager**

1. Open the CODESYS library repository (“Tools” menu).
2. Select a library version in the window of the installed libraries.
3. Click the “Export” button.
   - The “Export Library” dialog box opens.
4. As step 4 and 5 for “Export from the library manager”.

See also

- Chapter 1.3.1.19.2 “Adding a library to the application” on page 373

1.3.1.20 Managing devices

CODESYS manages the installed devices in the device repository. A device repository is a defined location in the file system. In the default CODESYS installation, it is defined with an absolute path as the system repository. You install or uninstall devices in the “Device Repository” dialog. The system installs a device by reading the device description file. The properties of a device are defined in these files regarding configurability, programmability, and possible connections to other devices.

You can use the devices provided in the device repository by adding them to the device tree of your project.

See also

- Chapter 1.3.1.25.2.8.5 “Command ‘Device Repository’” on page 957
- Chapter 1.3.1.20.1 “Installing devices” on page 374

1.3.1.20.1 Installing devices

Install a device in the device repository in order to include it in your project.

1. Click “Tools ➔ Device Repository”.
   - The “Device Repository” dialog box opens.
2. Select the install location. “System Repository” is set by default.
3. Click "Install".
   ⇒ The "Install Device Description" dialog box opens.
4. Select the file path of the device description.
5. Select the file type filter of the required device description.
   ⇒ All device descriptions of the selected file type are listed.
6. Select the required device description and click "Open".
   ⇒ CODESYS adds the device description to the matching category of your device repository.
   
   If errors occur during installation (for example, missing files that are referenced by the device description), then CODESYS displays them in the lower part of the device repository dialog box.

See also

● Chapter 1.3.1.25.2.8.5 “Command 'Device Repository'” on page 957

1.3.1.21 Managing packages and licenses

License manager
In addition to the standard installation of CODESYS, add-ons are available, which are subject to fees and licensing. You receive licenses for these products from 3S-Smart Software Solutions GmbH in the form of ticket numbers, which you can install using the CODESYS License Manager.

The license manager stores the licensing information on an external USB dongle (CODESYS Dongle) that you must connect to your local PC.

In the license repository you can retrieve the current information for each ticket number from the central license server. Other information is also provided, such as whether a license can still be activated or must be returned.

License check when starting CODESYS
When it starts, CODESYS checks the selected profile for plug-ins that are subject to licensing.

● If the profile does not include plug-ins subject to licensing, then CODESYS starts as usual without a message.
● If the profile includes plug-ins subject to licensing and a dongle with the required licenses is connected to the USB port of the computer, then CODESYS starts without any message.
● If the profile includes plug-ins subject to licensing but no dongle is available or a required license cannot be found, then, CODESYS displays the "License Missing" dialog at start-up.
   
   Note: When starting from the command line, this dialog is skipped if you specify the --skipunlicensedplugins option in the command. In this case plug-ins without the required license are automatically not loaded.

License check while program is running
While CODESYS is running, the required licenses are checked every five minutes. If a license is missing, for example because the dongle was removed in the meantime, then the "License missing" dialog opens with the following options:

● Insert the dongle again and press "Retry": If the dongle contains the missing licenses, then you can continue working as usual.
● "Save Current Project and Exit": CODESYS saves the project and closes.
● "Close": CODESYS closes without saving the project.
If you remove the license dongle after the CODESYS plug-in has been loaded, then an error message is displayed.

To solve this problem, insert the dongle again and press the Retry button. If the dongle contains a valid license, the dialog box closes and you can continue working as usual. Clicking “Cancel” will exit CODESYS.

See also

- Chapter 1.3.1.25.2.8.3 “Command ’License Manager’” on page 953
- Chapter 1.3.1.25.2.8.4 “Command ’License Repository’” on page 956

**License manager**

With the license manager in CODESYS you can manage the licenses for add-on products on a CODESYS dongle or in a soft container. A soft container is the software counterpart to a dongle. It allows hardware licenses to be bound to a single PC even without a dongle.

During the licensing a distinction is made as a matter of principle between CODESYS licenses and licenses for individual devices.

- The licensing of CODESYS products is done by the license manager.
- The licensing of end devices is done via the PLC settings. The licensing process takes place in a similar way to the licensing of CODESYS licenses.

See also

- Chapter 1.3.1.25.2.8.3 “Command ’License Manager’” on page 953

**Package manager**

CODESYS provides the Package Manager whose functionality extends the standard installation of CODESYS with additional features and configuration settings. The concept behind the Package Manager is similar to that of a typical Windows installation mechanism.

A package is a ZIP file with the "*.package" file extension.

Possible package components

- Plug-ins
- Libraries
- Device descriptions
- Supplier descriptions
- Profiles
- Information profiles
- Profile changes
- Files
- Extensions for configuration of the menu, toolbar, keyboard shortcuts, and views
- Help modules
- Complete menu configurations for special profiles
- Complete toolbar configurations for special profiles
- Complete keyboard configurations for special profiles
- Options
- Library profiles

You can compare the “Checksum” with the package checksum from the package vendor. CODESYS shows this checksum in the “Details” dialog of the Package Manager and in the “Installation - License Agreement” dialog of the installation assistant. You do this to make sure that you have installed an original package.
In V3.5 SP13 and later, CODESYS checks whether an older version is already installed when installing the development system. If this is the case, then when you start the development system for the first time, you will see the “Import Assistant” dialog for importing the package installations available in the older version.

See also
- Chapter 1.3.1.25.2.8.1 “Command ‘Package Manager’ ” on page 949
- Chapter 1.3.1.21.1 “Installing/Uninstalling a package” on page 377
- Chapter 1.3.1.25.3.1 “Dialog ‘Import Assistant’” on page 1039

### 1.3.1.21.1 Installing/Uninstalling a package

Requirement: You have selected the command “Tools → Package Manager” and the dialog box “Package Manager” is open.

**Installing a package**
1. Click on “Install”
2. In the dialog box “Open”, select a package from the file directory and click on “Open”.
   - The setup wizard opens and guides you through the installation of the package.
   - CODESYS also displays the “checksum” of the package in the dialog box “Installation - License Agreement”.

**Uninstalling a package**
1. Select the package
2. Click on “Uninstall”
   - The wizard opens and guides you through the procedure to uninstall the package.

See also
- Chapter 1.3.1.25.2.8.1 “Command ‘Package Manager’ ” on page 949

### 1.3.1.21.2 Licensing of products

Many CODESYS products are license-protected, so you have to activate a product before you start it. An add-on product that extends the capabilities of the CODESYS Development System is usually activated with a workstation license. Licensing is performed by means of the CODESYS Development System Security Key (dongle), which manages all licenses for your workstation. As an alternative, some products will soon support licensing on a soft container.

In addition, there are the SL products that extend the scope of CODESYS Control SoftPLC systems. These systems are activated with a single device license. An activation code, which is provided as a license activation file, is installed directly on the target. License management on a device is also done by means of a soft container (if supported) in the device, or a dongle connected to the device.

In CODESYS V3.5 SP13 and later, licenses of add-on products on the local computer, as well as single-device licenses, can be managed uniformly in the CODESYS License Manager. For devices with a unique serial number, the License Manager can reactivate the license from an automatically stored license backup file if the license information on the device is lost.

---

If possible, license your products via online activation instead of offline activation.
Online activation

Activation is possible in CODESYS Development System in the “License Manager” dialog. The requirement for this is that your development system has Internet access. The target system itself does not require Internet access.

You can also activate a license on the “License Central” website.

Offline activation

If your workstation computer does not have Internet access, then you can activate products using a license activation file. You get the file from the CODESYS license server. To do this, connect to the server from any computer with Internet access and request the file. Then you transfer the file on any storage medium to your working computer. The product is activated there as usual in CODESYS Development System, both for workstation licenses as well as for single device licenses.

If you have a computer with Internet access where CODESYS Development System is not installed, you can also open the “CodeMeter Control Center” by Wibu Systems to activate the license.

Moreover, for target systems that are based on CODESYS Control Win V3 or CODESYS Control RTE V3, you can activate a single device license directly on a target. This can be useful if you do not have a development system with CODESYS Development System. Then you can do the licensing from the “CodeMeter Control Center”, or you can activate the license from a browser. CodeMeter is installed on Windows-based target systems.

The “CodeMeter Control Center” manages the CodeMeter service and is included with the CODESYS setup.

Online activation of an add-on license on the local computer

The following instructions describe how to perform licensing by means of a dongle. Licensing by means of a soft container is done in the same way.

Requirement: You have Internet access and the CODESYS Development System is installed on the computer where you want to install the license. A dongle is connected to your computer.

1. Click “Tools ➔ License Manager”.
   - The wizard starts with the “License Manager - Select Target” dialog.
2. Select “Workstation” as the target. Click “Next >”. In the “License Manager - Select Container” dialog, select the “Dongle” option. Click “Next >”.
   - The “License Manager” dialog opens and shows the product to be licensed in the “Products” window as not licensed.
3. Select the product and click the “Install Licenses” button on the bottom left.
   - The “Install Licenses on Workstation Dongle <dongle ID> - Select Operation” wizard opens.
4. Select the “Activate License” operation and click “Next >”.
   - The “Install Licenses on Workstation Dongle <dongle ID> - Activate License” dialog opens.
5. Specify the “Ticket ID” that you received from the software manufacturer. The ID consists of 5 blocks, each containing 5 alphanumeric characters.
   - Select the “License server” that provides the license for activating the product. You get the server URL from the software vendor.
6. Click “Next >”.
   - The connection to the license server (http://license.codesys.com) is established.
     - If the specified ticket contains only one license, then a dialog opens to confirm the activation after successful completion of the server action.
     - If the specified ticket contains multiple licenses, then the dialog “Install Licenses - Select Licenses” opens with a list of the licenses managed in the ticket.
7. Select the licenses that should be activated and click “Next >”.
   ⇒ After successful completion of the server action, a dialog opens with the confirmation of the activation.

Alternative

1. Open a browser and select the License Central Internet page.
   ⇒ The “License Manager” dialog opens.
2. Enter your license number.
3. Click “Search”.
4. Click “Next >”
   ⇒ The connection to the license server (http://license.codesys.com) is established.

**Offline activation of an add-on license on the local computer**

Requirement: You want to install the license for an add-on package. The CODESYS Development System is installed on the computer where you want to install the license. The computer does not have Internet access.

For this method, first you create a license request file on the target computer in the CODESYS. Then you copy this file to a computer with Internet access. The “CodeMeter Control Center” must be installed on this computer. You use the CODESYS license server to generate a license activation file for activation on the target computer.

The following instructions describe how to perform licensing by means of a dongle. Licensing by means of a soft container is done in the same way.

1. Click “Tools ➔ License Manager”.
   ⇒ The “License Manager” wizard starts.
2. Select “Workstation” as the target. Click “Next >” and select “Dongle” as the container. Click “Next” again.
   ⇒ The dialog opens for selecting a product and container.
3. Select the dongle where you want to install the license. Example: “Local:CmStick [2-2404696]”.
4. Click “Install Licenses”.
   ⇒ The “Install Licenses on Workstation Dongle <dongle ID> - Select Operation” wizard opens.
5. Select the option “Request License” and click “Next >”.
   ⇒ The “Install Licenses on <computer> - Request License” wizard opens.
6. In the “Software vendor” input field, specify the firmcode from the software vendor that provided the license for activating the product. You can also select the name of the software vendor from the drop-down list.
7. Specify the location and filename of the “Context file”, for example CmStick [2-2404696].WibuCmRaC.
8. Click “Finish”.
   ⇒ The context file (*.WibuCmRaC) is created.
9. Copy this license request file to a computer with Internet access.
10. On this computer, open the CODESYS license server in a browser (http://license.codesys.com).

   ➞ CODESYS License Server:

11. Enter the number of the license ticket and click “Next”.

12. In the subsequent dialog, click “Activate Licenses”.

13. In the next dialog, select the binding for the license. Click the graphic with the CODESYS Runtime Key (dongle).

   ➞ The following dialog opens:
14. In the dialog, select the license container, for example “2-2404696”. Select “Offline license transfer”.
   ⇒ The following dialog opens:

   ![Select license request file (.WibuCmRaC)](image)

15. Select the license request file that you created on the other computer, and click “Upload Request and Continue Now”.
   ⇒ The following dialog opens:

   ![Download License Update File](image)

16. Load the license update file to your computer. Select “Download License Update File Now”.

17. Copy the license update file to the target computer.

18. In the “License Manager” dialog, click the “Install Licenses” button and select the “Install License” option.
   ⇒ The “Install Licenses on Workstation Dongle <dongle ID> - Install License” dialog opens.

19. Enter the path of the license update file in the input field (example: D:\Lic\CmStick [2-2404696].WibuCmRaU).

20. Click “Finish”.
   ⇒ The license update file is installed on the selected dongle.

---

**Offline activation of a windows-based runtime system**

You want to install the license for a runtime system that is based on Windows.

For this method, first you create a license request file on the target computer in the “CodeMeter Control Center”. Then you copy this file to a computer with Internet access. The “CodeMeter Control Center” must be installed on this computer. You use the CODESYS license server to generate a license activation file which you activate on the target computer.
Initial situation: The CODESYS Development System is not installed on the computer where you want to install the license. The computer does not have Internet access.

1. Open the “CodeMeter Control Center”. Double-click in the toolbar.
2. Select the license memory “3S-Smart Software Solutions Softlicenses” in the license overview.
3. Click the “Activate License” button.
   ⇒ The “CmFAS” wizard opens.
4. Click “Next”.
   ⇒ A list of possible actions is displayed.
5. Select “Generate License Request” and click “Next”.
6. Specify a name for the license request file and save it by clicking “Apply”. Close the dialog.
   ⇒ The license request file is generated (example: 32767-4241603201.WibuCmRaC).
7. Copy this license request file to a computer with Internet access.
8. On this computer, open the CODESYS license server in a browser.

   → CODESYS License Server:

![CODESYS License Server](image)

9. Enter the number of the license ticket and click “Next”.
10. In the subsequent dialog, click “Activate Licenses”.
11. In the next dialog, select the binding for the license. Click the graphic with the CODESYS Software Key (soft container).

   → The following dialog opens:

![Select license container](image)
12. In the dialog, select the license container, for example “32767-4241603201 (3S-Smart Software Solutions Soft Licenses). Click “Offline license transfer”.

⇒ The following dialog opens:

```
Select license request file (*.WibuCmRaC)
Durchsuchen... CmStick [2-2404696].WibuCmRaC

Upload request and continue now
```

13. Select the license request file that you created on the other computer, and click “Upload Request and Continue Now”.

⇒ The following dialog opens:

```
Download License Update File

Upload request ✓ Download update Upload receipt

To transfer your licenses offline - Second step "Download Update”:
1. Click "Download license update file now" and save the file on your computer.
2. Import this license update file to the license container with Serial 128-9754372. This file can be imported with the CODESYS. How it works ➔
3. After you have successfully transferred the license update file to the license container, click “Next” to confirm the license transfer.

Download license update file now Next ➔
```

14. Load the license update file to your computer. Click “Download License Update File Now”.

⇒ The dialog opens for saving the file.

15. Save the license update file and copy it to the target computer.

16. Open the “CodeMeter Control Center”.

17. Select the license memory “3S-Smart Software Solutions Softlicenses” in the license overview.

18. Click the “Activate License” button.

⇒ The “CmFAS” wizard opens.

19. Click “Next”.

⇒ A list of possible actions is displayed.

20. Select “Save License Update” and click “Next”.

21. Select the license update file, for example 128-9754375.WibuCmRaU, and click “Apply”.

⇒ The license update file is installed on the selected dongle.
You want to install a single license for a Linux-based runtime system.

☑️ Requirements: Your development system with CODESYS Development System does not have Internet access, but it is connected to the device.

1. Open CODESYS.
2. Create a standard project.
3. Update the device.
4. In CODESYS, click “Tools ➔ License Manager”.
   ⇒ The wizard starts with the “License Manager - Select Target” dialog.
5. Select “Device” as the target. Click “Next >”. In the “License Manager - Select Container” dialog, select the “Softcontainer” option or “Dongle”. Click “Next >”.
   ⇒ The “License Manager” dialog opens. The product to be licensed is displayed in the “Products” view.
6. Click the “Install Licenses” button. In the “Install Licenses on <device> - Select Operation” dialog, select the “Request License” option. Click “Next >”.
   ⇒ The “Install Licenses on <device> - Request License” dialog opens.
7. In the “Software vendor” input field, specify the firmcode from the software vendor that provided the license for activating the product. You can also select the name of the software vendor from the drop-down list.
8. Specify the location and filename of the “Context file”, for example CmStick [2-2404696].WibuCmRaC.
9. Click “Finish”.
   ⇒ The context file (*.WibuCmRaC) is created.
10. Generate the update file (*.WibuCmRaU) in the browser.
11. Copy this license request file to a computer with Internet access. The “CodeMeter Control Center” must be installed on this computer.
12. On this computer, open the CODESYS license server in a browser (http://license.codesys.com).

   CODESYS License Server:

13. Enter the number of the license ticket and click “Next”.
14. In the subsequent dialog, click “Activate Licenses”.
15. In the next dialog, select the binding for the license. Click the graphic with the CODESYS Runtime Key (dongle).

   The following dialog opens:
16. In the next dialog, select the license container, for example “2-2404696”. Click “Offline license transfer”.
   ➤ The following dialog opens:

![Select license request file (.WibuCmRaC)](image)

17. Select the license request file that you created on the other computer, and click “Upload Request and Continue Now”.
   ➤ The following dialog opens:

![Download License Update File](image)

18. Load the license update file to your computer. Click “Download License Update File Now”.
19. Copy the license update file to the target computer.
20. In the “License Manager” dialog, click the “Install Licenses” button.
   ➤ The “Install Licenses on <device> - Select Operation” wizard opens.
21. Select the “Activate License” option, click “Next >”, and specify the ticket number.
22. Click “Next” to transfer the license to the software key (soft container).
   ➤ The license is active.

**Restoring a license**

In CODESYS V3.5 SP13 and later, a license backup file (*.WibuCmRaC) is created automatically when activating workstation licenses for devices with a unique serial number. The file is stored on your computer and on the license server. In case the license files on the device get lost, you can use the License Manager to restore the license from this file.
You have activated a device license.

1. In CODESYS, click “Tools ➔ License Manager”. Follow the wizard with the corresponding entries for the target: Device and container (soft container or dongle), as well as the selection of the corresponding device.

2. In the “License Manager” dialog, click the “Additional Functions” button and select the “Restore License” command.

   The “Restore Licenses” dialog opens.

3. Enter the “Ticket ID” for the device license and click “Restore”.

   CODESYS searches for the saved license backup file, first on the local computer and then on the License Central server. If the file matching the device is found, then the license is restored and activated.

Requirements:

- The CODESYS Development System (V3.5 SP15 or later) is open.
- The gateway and the controller whose information should be read are running.
- No applications exist on the controller.

1. Click “Tools ➔ Device Reader”.

   The “Select Device” dialog opens.

2. Double-click the desired gateway to scan the network.

   If no gateways are displayed, then click “New Gateway” and select the desired gateway in the “Gateway” dialog.

3. Select a controller.

4. Click “OK” to confirm the selection.

   Note: If an application exists on the controller, then a dialog opens to prompt whether or not all applications should be removed from the controller. When you click “No” to this dialog, the license information of the controller cannot be read. The “Device Reader” command is aborted.

   CODESYS creates the list with the license information for the products and device features of the selected controller and displays the information in the “Device Reader” dialog.

See also

- License Central
- Chapter 1.3.1.25.2.8.3 “Command ‘License Manager’” on page 953
- Chapter 1.3.1.25.2.8.4 “Command ‘License Repository’” on page 956
- Chapter 1.3.1.25.2.8.12 “Command ‘Device Reader’” on page 960

1.3.1.22  Using scripts

With the scripting feature in CODESYS, you can automate commands or complex program operations that you would otherwise have to do manually with mouse clicks and text input in the CODESYS user interface. You can start these scripts from the CODESYS user interface (command or configured toolbar) or from the Windows command line.
Examples of use cases

- Integration of CODESYS in automatic build server environments:
  - Continuous Integration (CI)
  - Continuous Delivery (CD)
  - Continuous Testing
- Integration with third-party software, for example:
  - Code generators
  - Creation of projects that are custom tailored to a specific machine configuration
- Creation of documentation
- Updating of libraries:
  - Setting of project information during the release process
- Automatic testing:
  - Mostly in connection with the CODESYS Test Manager
- Outputting variables via monitoring APIs

See also
- Continuous Delivery (CD)
- Continuous Integration (CI)
- Continuous Testing

Scripting language

The CODESYS scripting language is modular and based on IronPython. For this purpose, the CODESYS "ScriptEngine" component combines the IronPython interpreter with the CODESYS development environment. Then you can use the extensive Python framework libraries, which includes file access in networks and much more.

CODESYS does not yet include its own Python editor. Create your scripts with any text editor or the Python editor.

See also
- Chapter 1.3.1.22.2 “Creating a python script” on page 396

Architecture of the ScriptEngine, extension possibilities

The (Iron)Python scripting language used in CODESYS allows for access to the CODESYS scripting APIs for controlling CODESYS operations. Moreover, it lets users effectively apply both the Python standard library and third-party Python modules, as well as third-party .NET framework libraries and .NET assemblies.

Users can execute the scripts from menu commands or configured toolbars in the CODESYS interface or from the Windows command line. Add-ons such as the CODESYS Test Manager also provide ways to execute scripts.
There is not an integrated Python editor in CODESYS. Use your favorite text editor or the Python development environment.

With the Automation Platform APIs, the ScriptEngine APIs can be extended. Examples for this are CODESYS Test Manager and CODESYS SVN. Both provide their own objects and methods as an extension to the scripting APIs. In addition, the CODESYS Test Manager allows for the execution of scripts in a test case. For more information, refer to the respective API documentation of the add-on. Registered Automation Platform users will find more information in the CODESYS Developer Network.

See also
- Chapter 1.3.1.25.2.8.6 “Command ‘Scripting’ - ‘Execute Script File’” on page 958
- Chapter 1.3.1.22 “Using scripts” on page 388
- API Reference Documentation for the ScriptEngine:

1.3.1.22.1 Executing a script

1.3.1.22.1.1 Calling scripts from menu commands ........................................ 390
1.3.1.22.1.2 Starting scripts from the command line ........................................... 391
1.3.1.22.1.3 Calling scripts from toolbar icons...................................................... 392

You can execute Python script files (<filename>.py), which contain a sequence of commands for activating CODESYS functionalities, in the following way:
- From the CODESYS user interface by means of commands in the menu “Tools ➔ Scripting”
- From the CODESYS user interface by means of a customized, configured toolbar
- From the Windows command-line

Calling scripts from menu commands

Requirement: A valid Python script file <filename>.py is located in the file system. The CODESYS user interface is open.
1. (Optional) To monitor the processing of individual commands used in the script, click “Tools "Scripting" "Activate Script Tracing".

2. In CODESYS, click “Tools "Scripting" "Execute Script".
   
   ⇒ The statements in the script are executed and, if script tracing is activated, listed in the message view.

See also

● Chapter 1.3.1.25.2.8.6 “Command 'Scripting' - 'Execute Script File’” on page 958
● Chapter 1.3.1.25.2.8.7 “Command 'Scripting' - 'Enable Script Tracing’” on page 958
● Chapter 1.3.1.22 “Using scripts” on page 388

---

**Starting scripts from the command line**

In automated environments such as CI (Continuous Integration) servers or if scripts in CODESYS have to be controlled by other programs, menu commands are not appropriate for executing scripts. For these kinds of requirements, you can use the Windows command line to start CODESYS and execute scripts.

**Requirement:** A valid Python script file `<filename>.py` is located in the file system.

1. Create a CMD file that starts with `start CODESYS` and executes the script file with the option `--runscript`. Other options are possible, such as `--noUI` if the CODESYS user interface should not be opened.

2. Open the Windows “Command prompt” and execute the CMD file.

You can pass arguments with additional information to the script. Python scripts can access arguments with the `sys.argv[]` list. The first element (Index 0) is always the name or path of the Python script that is executed, followed by the actual parameters. (This is similar to `argc/ argv` in C.) In addition, scripts can also access environment variables that are set before CODESYS is started with the corresponding Python or .NET APIs.

**Example**

A CMD batch file `argvtestbat.cmd` has the following contents (all in one line):

```
"C:\Program Files (x86)\3S CODESYS V3.5 SP10\CODESYS\Common \CODESYS.exe" --profile="CODESYS V3.5 SP10" -- runscript="D:\Documents\Scripting\ArgvTestScript.py" -- scriptargs: 'username password 3.14 "path="C:\temp\""' --noUI
```

A matching script file `ArgvTestScript.py`: 
from __future__ import print_function

import sys

print("sys.argv: ",
         len(sys.argv),
         " elements:")

for arg in sys.argv:
    print(" - ", arg)

Now when you execute the CMD file, CODESYS starts and executes the script without opening the CODESYS main window. Then CODESYS is exited:

```
C:\Users\user>\G:\Program Files (x86)\CODESYS\CODESYS.exe --runscript=D:\\Misc\trunk\ScriptEngine\ArgTestScript.py --scriptargs:username password 3.14
```

For a complete reference of all possible command line parameters, refer to the help page for the command-line interface in CODESYS in the section for "--runscript".

See also
- Documentation for Python API
- Information about .NET API
- Chapter 1.3.1.18 “Using the command-line interface” on page 366

### Calling scripts from toolbar icons

You can provide your own toolbar in the CODESYS user interface with up to 32 icons for calling script files. For this you need an ICO file where the icon is stored, and a PY file where the Python script to be called is stored.

You create the configuration file `config.json` in the installation directory or in the program data directory (CODESYS\Script Commands). Specify the call information outline for each icon in the file. Here you can configure a maximum of 16 icons. You can also store the ICO and PY files in the same directory.

Storage locations
- `<CODESYS installation directory>\CODESYS\Script Commands`
- `%PROGRAMDATA%\CODESYS\Script Commands`

**Example**

**Default installation on Windows 7**

C:\Program Files (x86)\CODESYS 3.5.14.0\CODESYS\Script Commands\  
C:\ProgramData\CODESYS\Script Commands\  

If you store a `config.json` file with different call information at each of the storage locations, then you can configure up to 32 different icons.
Table 11: Call information

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
</table>
| Name    | Required; displayed as icon tooltip  
Example: "Name": "Pause"  |
| Desc    | Optional; comment for icon  
Example: "Desc": "Operation pause"  
Note: Not yet displayed in the user interface |
| Icon    | Required; file path <directory path>\<icon name>.ico of the icon  
Example: "Icon": "pause.ico"  
Hint: If the file is in the same folder as the config.json file, then the file name is enough. |
| Path    | Required; path of the Python script <directory path>\<script name>.ico  
Example: "Path": "stop.py"  
Hint: If the file is in the same folder as the config.json file, then the file name is enough. |
| Params  | Optional; only if the script requires parameters  
Example: "Params": [ "file1", "file2" ] |
Example

File config.json

```json
[
  {
    "Name": "Start",
    "Desc": "Starts processing",
    "Icon": "start.ico",
    "Path": "goon.py",
    "Params": [
      "--continue"
    ]
  },
  {
    "Name": "Pause",
    "Desc": "Pause operation",
    "Icon": "pause.ico",
    "Path": "stop.py",
    "Params": [
      "delay:-1"
    ]
  },
  {
    "Name": "Processing",
    "Desc": "Process again",
    "Icon": "VarStatSmall.ico",
    "Path": "process.py",
    "Params": [
      "exit"
    ]
  }
]
```

The following files are located in the Script Commands:

- config.json
- goon.py
- stop.py
- process.py
- start.ico
- pause.ico
- VarStatSmall.ico

Creating script calls for a toolbar button

1. Create the Script Commands folder in one of the storage locations.
   ➔ C:\ProgramData\CODESYS\Script Commands

2. Create executable Python files there.
   ➔ Example:

   File `copy.py`
   ```python```
   print("The script COPY.PY is executed")
   ```

   File `delete.py`
   ```python```
   print("The script DELETE.PY is executed")
   ```

3. Create the ICO files for the scripts.
   ➔ Example: Copy_before.ico, Copy_below.ico, CopyAll.ico
4. Create a configuration file `config.json` there.
   ⇒ The folder `C:\ProgramData\CODESYS\Script Commands` has the following contents:

   ![Image showing configuration file contents]

5. Open `config.json` and add the outlined call information.
   ⇒
   ```json
   [
   {
     "Name": "Copy Before",
     "Desc": "Copy something",
     "Icon": "Copy_before.ico",
     "Path": "copy.py",
     "Params": [
       "before"
     ]
   },
   {
     "Name": "Copy Below",
     "Desc": "Copy something",
     "Icon": "Copy_below.ico",
     "Path": "copy.py"
   },
   {
     "Name": "Copy All",
     "Desc": "Copy something",
     "Icon": "CopyAll.ico",
     "Path": "copy.py",
     "Params": [
       "all"
     ]
   },
   {
     "Name": "Delete",
     "Desc": "Delete something",
     "Icon": "Delete.ico",
     "Path": "delete.py"
   }
   ]
   ```

6. Start CODESYS.
   ⇒ The script files, configuration file, and symbol files are read and provided in the “Tools ➔ Customize” dialog in the “Command Icons” tab, “ScriptEngine Commands” category.

7. Click “Tools ➔ Customize” and then click the “Toolbars” tab.
8. Select the empty toolbar there and click the “Add toolbar” button.
   ⇒ A line editor opens at the empty toolbar.

9. Type in a name (example: User defined toolbar).
   ⇒ The custom toolbar is displayed in the CODESYS window.

10. Add the recently imported commands and close the dialog.

11. Click one of the icons.
   ⇒ The following output is displayed in the message view.

See also

- Chapter 1.3.1.22 “Using scripts” on page 388
- Chapter 1.3.1.1.2.2 “Customizing toolbars” on page 104

1.3.1.22.2 Creating a python script

| 1.3.1.22.2.1 | Getting started with python for CODESYS................................. 397 |
| 1.3.1.22.2.2 | Tips for python programmers about .NET API documentation........ 397 |
| 1.3.1.22.2.3 | Basic syntax of python (with examples)................................... 398 |
| 1.3.1.22.2.4 | Python control structures (with examples)................................. 406 |
| 1.3.1.22.2.5 | Using scripts to access CODESYS functionalities....................... 411 |
| 1.3.1.22.2.6 | Transitioning from python 2 to python 3.................................. 420 |
| 1.3.1.22.2.7 | Comparison of IronPython and cPython.................................. 420 |

Python is a dynamic language. You can start in a simple linear programming style (batch files) and later add the necessary and more powerful means, such as conditions, loops, functions, exceptions, classes, and modules. The focus of the language is on easy and expressive code. Python is more typical in runtime mode and uses an automatic garbage collector to protect the programmer from accidental damage to the entire system.

IronPython is an implementation of Python for .NET and allows for full access to the .NET framework and classes. The implementation of the IronPython interpreter is based on Python Version 2.7.

There are a variety of free manuals and help pages on the internet. See the following links for an introduction and detailed introduction about IronPython.
  – Especially CODESYS-specific questions.
  – Also includes some examples.
• **https://docs.python.org/2/tutorial/index.html**: Python tutorial in the official Python documentation.
• **http://docs.python.org/release/2.7/**: Official documentation for Python 2.7
• **http://wiki.python.org/moin/GermanLanguage**: Collection of links for German help pages.
• **http://stackoverflow.com/**: General community for programming. For general question about (Iron)Python, not CODESYS-specific.
• **http://ironpython.net/**: IronPython homepage
• **http://ironpython.net/support/**: Mailing list, FAQ, etc.
• **https://gitter.im/IronLanguages/ironpython**: Chat channel for IronPython developers.

### Version incompatibility to python V3.x

The Python programming language will soon be available in the new version V3.x. Some of the older program modules have been removed. 3S-Smart Software Solutions GmbH is planning an update to this new version. Scripting developers should take this into consideration and design their scripts accordingly, for example by using the expression:

```python
from __future__ import print_function
```

You can find more information about this topic at

- [http://wiki.python.org/moin/Python2orPython3](http://wiki.python.org/moin/Python2orPython3)
- [http://docs.python.org/release/3.1.2/whatsnew/3.0.html](http://docs.python.org/release/3.1.2/whatsnew/3.0.html)

### Getting started with python for CODESYS

See below for a simple application of a Python script in CODESYS:

1. In any text editor, create a text file `hello.py` with the following contents:
   ```python
   print("Hello, automation!")
   ```
2. Start CODESYS and click "Tools ➔ Scripting ➔ Execute Script File". Select the file `hello.py` in the file system.
   ➔ See the result in the message view:

   ![Message view with output "Hello, automation!"](image)

   For more detailed examples of Python scripts for different use cases with CODESYS, refer to the following help pages:

   See also

   - ![Chapter 1.3.1.22.2.3 “Basic syntax of python (with examples)” on page 398](image)
   - ![Chapter 1.3.1.22.2.4 “Python control structures (with examples)” on page 406](image)
The current prerelease of the script interface documentation has been generated automatically from the underlying .NET and C# sources. Therefore, the documentation includes some expressions that are not familiar to Python programmers. The following overview provides some tips about how these expressions can be understood from the Python perspective.

- An interface is the contract that tells an instance of a class that implements the interface which members (methods, properties) it has to prepare. In IronPython, you can implement one or more .NET interfaces in one class by inheriting from a superclass. If a method is needed for the interface but is not available in the class definition, then an exception is thrown. (The DeviceImportFromSvn.py example shows a class that implements the ImportReporter interface.)

- Each parameter and each method in .NET is typed strictly. The type of parameter is separated by one space character before the parameter name, and the type of return value from a method by one space character before the method name. You can use instances from subclasses when you define a class (or interface). A method without a return value is marked void.

- You can overload methods, as multiple methods of the name can exist in one class. However, the number or the types of parameters must be different. IronPython automatically takes care of the most appropriate method overloading being called.

- The data type `int` corresponds to an integer from -2,147,483,648 to 2,147,483,647. The data type `bool` corresponds to the Python type `True` and `False`. The data type `string` corresponds to the Python types `str` or `unicode`, which are identical in IronPython. The `IDictionary<Object, Object>` data type corresponds to an ordinary Python dictionary. IronPython automatically converts between Python and .NET data types.

- When a `T` type is inherited from `IBaseObject<T>`, it means that this type can be extended by other plug-ins for additional members. The actual use of this extended type as a parameter or return value is marked by `IExtendedObject<T>`. The `IEnumerable<T>` interface to a `T` type means that you can use every Python sequence (generators, lists, tuples, …) that returns `T` type values (or a subclass). Returns the sequence of incompatible objects, and throws an exception at runtime.

- The `IList<T>` interface to a `T` type identifies a typified list that guarantees to include only elements of type `T` (or a subclass). An exception is thrown at runtime when any attempt is made to add an incompatible object.

- The `params T[] name` id for a parameter of type `T` corresponds to the Python mechanism `*name` for variable argument lists.

- In Python, enumerations (enum) do not exist as language constructs. Its purpose is to define an exact number of constant values for a specific purpose, for example the days of the week. Access to .NET enumerations from IronPython works by using "Name.Member", for example via `OnlineChangeOption.Try`. (There are many ways of emulating enums in Python. For examples, see `http://pypi.python.org/pypi/enum` or `http://www.ironpython.info/index.php/Enumerations`.)

- The syntax `T name { get; set; }` defines a property with `name` as the name and `T` as the type. If `set;` is missing, then the property is read-only. In Python, the respective construct is `@property` decorator.

See also

- `Documentation for Python API`
- `Information about .NET API`
- Chapter 1.3.1.22.2.5 “Using scripts to access CODESYS functionalities” on page 411

**Basic syntax of python (with examples)**

Python is similar to the languages of the C family, but there are some significant differences and unique properties.

The most obvious syntactic difference between Python and language such as C and ST is that the Python parser recognizes block structures by their indentation. There is no `BEGIN/END` or braces `{}` to identify the blocks of `IF/ELSE` conditions, `FOR` and `WHILE` loops, or functions.
Comments start with # and extend to the end of the line. In the first and second line of the source code, you can set a special marker to declare the encoding of the file. We recommend that you use UTF-8 as the encoding if ASCII characters are not required.

For debugging purposes, you use print for easy output. With the % operator, you achieve functionality similar to the C function printf(). The output is displayed in the message view of CODESYS.

```
Example:
defining a function with the parameter i
def do_something(i):
    # if branch
    if i>0:
        print("The value is: \%i\" \% i)
        sum += i
        print("The new sum is: \%i\" \% sum)
    # else if (optional, there can be none or several elif branches)
    elif i=0:
        print("The sum did not change: \%i\" \% sum)
    # and the final else branch (also optional).
    else:
        handle_error()
    # an endless while loop
    while True:
        print("I got stuck forever!")
```

Everything that belongs to the same block has to be indented the same distance. The size of the indentation is irrelevant. Elements such as brackets and braces have a higher priority than indentations. Therefore, the following code segment is completely correct, even if it is written in a poor programming style:

```
Example:
# warning: bad style below. Kids, don't try this at home!
if foo >= bar:
    print("foobart")
else:
    print("barfoo")
```

To avoid ambiguity, you should not mix tabs and spaces in a file.

At this time, mixing tabs and spaces girt in Python 3 qualifies as a syntax error.
The official Python Style Guide recommends indentation of four spaces and includes some examples of good and poor style. The Python tutorial provides a summary of coding style.

Python is case-sensitive, similar to and in contrast to ST. Keywords, such as def, if, else, and while, have to be lowercase (in contrast to the ST rule: keywords are uppercase). Two identifiers, such as "i" and "I", also identify two different variables.
the following keywords are reserved in Python and not permitted for use as identifiers for variables, functions, etc.: and | as | assert | break | class | continue | def | del | elif | else | except | exec | finally | for | from | global | if | import | in | is | lambda | not | or | pass | print | raise | return | try | while | with | yield.

Python 3 defined four other keywords: False | None | True | nonlocal. While the first three are really new, the first three were already predefined constants in Python 2 and should not be used for any other purposes.

See also
- Python Style Guide
- Python Tutorial

Variables and data types

Python is a powerful, dynamically typed language -- all type information is evaluated at runtime. Variables hold references to objects, and the object knows its type, not the variable. When a programmer attempts to execute an operation that is not possible (for example, adding an integer and a string), Python throws an exception at runtime.

Consequently, there are no declarations of variables and their types. In Python, variables are created only to assign values to them. This is completely different in C and ST where types are strong and static. Every variable is declared with a type, and at compile time the compiler checks that the type and operators are permitted.

Refer to the following examples for dealing with variables:

Example: Variables

```python
# assign the integer 1 to the variable i (also "creates" the variable")
i = 1

# assign the string "foobar" to the variable s
s = "foobar"

# Add 5 to the integer i - this is equivalent to i = i + 5
i += 5 # i now holds the integer 6.

# Try to add i and s - this will throw an exception when executed
# TypeError: unsupported operand type(s) for +: 'int' and 'str'
result = i + s

# variables can also be "undeclared" by deleting them.
# Further access to the variable i will throw a NameError exception,
# as the variable does not exist any more.
del i

i += 5 # now throws an exception: NameError: name 'i' is not defined
```

All existing variables reference one value only. There is not any unassigned or uninitialized variables in Python. To express the absence of a value, Python provides a special object: None. In C or ST, you would use a null pointer. Its only purpose is to express "no value here", although None is actually an existing instance of the class NoneType.

Numeric types and floating points

In contrast to the dozens of integer types in IEC or C, there is only one integer type in Python. Integer types in Python do not have a fixed size. Instead, they grow as needed and are limited only by available memory.
Example: Integers.py

```python
from __future__ import print_function

i = 1
print(i)

j = 0x1234   # hex number, is 16#1234 in IEC and 4660 in decimal
k = 0o123    # octal number, is 8#123 in IEC and 83 decimal
l = 0b101010 # binary number, is 2#101010 in IEC and 42 in decimal
print(j, k, l)

m = (2 + 3)*10 # k is 50 now
print(m)

n = 10 ** 100 # 10 to the power of 100
print(n)
```

Resulting output:

```
1
1234 123 42
50
10000000000000000000000000000000
```

There is also only one floating-point type in Python which is similar to the IEC data type `LREAL`. It provides 64-bit IEEE floating point arithmetic.

The syntax is like C-based languages for the most part:

Example: Floating-point types

```python
# A simple float...
a = 123.456

# A float containing the integral value 2
b = 2.

# Leading zeros can be left off
c = .3 # same as 0.3

# Exponential / scientific representation
d = -123e-5
```

Two special cases are `True` and `False`, two constants that define the Boolean truth values. They behave similar to the integer values `0` and `1`, except when they are converted into strings and return their names.
Example:
Booleans.py

```python
# booleans behave like integers, except when converted to strings.
# The built-in function "type" can be used to query the type of a value.
print("True: ", True, type(True))
print("False: ", False, type(False))
print("1: ", 1, type(1))
print("False + 0: ", False + 0, type(False + 0))
print("True * 5: ", True * 5, type(True * 5))
```

Resulting output:

<table>
<thead>
<tr>
<th>True:</th>
<th>True &lt;type 'bool'&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>False:</td>
<td>False &lt;type 'bool'&gt;</td>
</tr>
<tr>
<td>1:</td>
<td>1 &lt;type 'int'&gt;</td>
</tr>
<tr>
<td>False + 0:</td>
<td>0 &lt;type 'int'&gt;</td>
</tr>
<tr>
<td>True * 5:</td>
<td>5 &lt;type 'int'&gt;</td>
</tr>
</tbody>
</table>

Strings

In IronPython, strings are always in Unicode and any length. It does not make any difference if they are enclosed in ' or " Strings can also have triple quotation marks """ or ''' which allows for multiline string literals.

Similar to C, special characters can be excluded by means of backslashes (\): As a comparison, the dollar sign ($) is used in IEC for this purpose.

There are also raw strings that have other rules for the backslash. This is practical when the string should have literal backslashes. Example: Windows file paths or regular expressions.
Example:

```
# encoding:utf-8
from __future__ import print_function

a = "a simple string"
b = 'another string'
c = "strings may contain 'quotes' of the other type."
d = "multiple string literals" ' are concatenated ' "by the parser"
e = "Escaping: quotes: " \" backslash: \ newline: \r\n ascii code: \x40"
f = """"triple-quoted strings may contain newlines, "single" 'quotes' and ""multiquotes"" of the other type"
g = "Unicode is also possible: 北京, Москва, Αθήνα, القاهرة"
h = r"c:\raw\strings\retain\backslashes.txt"

# we iterate over a sequence of all the variables defined above:
for i in (a,b,c,d,e,f,g,h):
    print(i) # prints the contents of the variable
```

Resulting output:

```
 ascii code: @
 triple-quoted strings may contain newlines, "single"
 'quotes' and ""multiquotes"" of the other type
 Unicode is also possible: 北京, Москва, Αθήνα, القاهرة
c:\\raw\\strings\\retain\\backslashes.txt
```

Python does not have characters types. Characters are expressed by the use of strings with a length of 1. In this way, iteration via a string, or indexing in a string, returns a single-character string.

See also

- [Python Documentation, Strings](https://docs.python.org/3/library/string.html)

### Lists and tuples (data sets)

Lists and tuples basically correspond to arrays in C and IEC, but there are some noticeable differences:

- Index access is always checked. Accessing a list or a tuple with an invalid index throws an exception.
- Both lists and tuples can contain elements of different types (also other lists and tuples). In contrast in C and IEC, arrays can contain only elements of a single type.
- Lists are dynamic, and elements can be added, removed, or replaced at any time.
- Tuples are not changeable: Once a tuple is created, it cannot be modified anymore.

Lists are created with the `list()` constructor. As an alternative, you can use brackets `[]`. Tuples are created with the `tuple()` constructor or parentheses `()`.
Example:
list_tuples.py

```python
from __future__ import print_function

print("Testing tuples and lists")

# We define a tuple with the numbers from 1 to 10:
t = (1, 2, 3, 4, 5, 6, 7, 8, 9, 10)
print("Tuple:", t)

# We can access the 6th element of the tuple.
# As in C, index counting starts with 0.
print("Element 5:", t[5])

# Subscription is more powerful using the range syntax:
print("Range[2:5]:", t[2:5]) # lower bound is inclusive, upper bound is exclusive.
print("Range[2::2]:", t[2::2]) # start with 3rd element, and print every 2nd element.
print("Range[-3:-1]:", t[-3:-1]) # Start with the 3rd last element, end just before the last element (upper bound is exclusive)
print("Range[::]:", t[::]) # negative step with - print backwards

# lists are similar to tuples...
l = [11, 12, 13, "8", t] # contains mixed types: 3 integers, a string, and the tuple defined above.
print("List:", l)

# ... but elements can be added or removed dynamically.
l.append(9) # Add a 9 to the list.
print("List with 9:", l)

del l[1] # remove the element at index 1, the 12.
print("Removed[1]:", l)
del l[1:3] # Remove the elements at index 1 and 2, the 13 and the '8'.
print("Removed[1:3]:", l)
```

Resulting output:

```
List: [11, 12, 13, '8', (1, 2, 3, 4, 5, 6, 7, 8, 9, 10)]
List with 9: [11, 12, 13, '8', (1, 2, 3, 4, 5, 6, 7, 8, 9, 10), 9]
List Range[3:6:2]: ['8', 9]
Removed[1]: [11, 13, '8', (1, 2, 3, 4, 5, 6, 7, 8, 9, 10), 9]
Removed[1:3]: [11, (1, 2, 3, 4, 5, 6, 7, 8, 9, 10), 9]
```

Dictionary

Python also has a hash table type (also "hashmap"). In contrast to the list, it can be indexed with any elements, for example strings. Its constructor is `dict()` and its literals are declared with braces `{}`.

The sample script `dictionaries.py` creates the output displayed below. In the last line, the script is terminated with a "KeyError" exception:
Example:
dictionaries

```python
from __future__ import print_function
print("Testing dictionaries")

# Declare a dictionary with three entries, the third being a list
d = {1: "a", 2: "b", "my list": [1, 2, 3]}
print(d)

# print the value of the key 1
print(d[1])

# remove the value with the key "my list"
del d["my list"]

# Add a value 4 with the key 3
d[3] = 4
print(d)

# The "get" method returns the second argument if the key cannot be
# found.
print(d.get(1, 42))
print(d.get(23, 42))

# print all keys in the dictionary
for key in d:
    print(key)

# index access for unknown keys will throw a "KeyError" exception!
print(d[23])
```

Resulting output:

And then in the last line, the script terminates:
You can view the stack trace by clicking the "Details" button. Here you find out about line number 27 and the unknown key 23.

Python control structures (with examples)

Loops
As opposed to C and ST, for loops in Python do not count loop variables, but iterate over a sequence. This kind of sequence can be a dictionary, a list, a tuple, the characters in a string, or lines in a file.

The following example shows some for loops:
from __future__ import print_function

print("Enumerating over a simple list:")
for i in (1,2,3,4):
    print(i, end="", "") # end= replaces the newline with ", "
print() # but we still need a newline at the end of this case.

print("Enumerating over the characters in a string:")
for i in "CODESYS": # characters are representet as strings of length 1.
    print(i, end="", "")
print()

print("Enumerating over the integers 1 to 4:")
for i in range(1, 5): # upper bound is exclusive.
    print(i, end="", "")
print()

print("Enumerating using xrange:")
for i in xrange(5): # xrange is similar to range, but needs less memory for large ranges.
    print(i, end="", "")
print()

print("Enumerating including the item number:")
for i, v in enumerate("CODESYS"):
    print(i, v)

Resulting output:

If you require an index or number in addition to the item, then you should use enumerate as shown in the last case of the sample script. The following code is considered as poor style:
**Example: Poor style**

```python
text = "CODESYS"

for i in range(len(text)):  # BAD STYLE!
    v = text[i]             # DON'T TRY THIS AT HOME!
    print(i, v)
```

Besides `for` loops, Python also has `while` loops which are very similar to those in C and ST:

**Example: "while" loops**

```python
i = 0
while i < 3:
    print(i)
    i += 1
```

Note: This example is not very practical. You would more likely use a `for` loop with a range.

**IF / ELSE**

The `if/else` construct is similar to those in other programming languages. Here is a short example:

**Example: "if_else.py"**

```python
from __future__ import print_function
i = int(system.ui.query_string("Please enter an integral number..."))
if i < 0:
    print("Your number was negative.")
elif i > 0:
    print("Your number was positive.")
else:
    print("It seems your number was zero.")
```

The `else` branch is optional and there can be zero, one, or many `elif` branches.

**Functions, classes, and methods**

Python allows for defining functions and classes with methods. A class with methods is basically similar to a function block in ST, or classes in languages such as C++, Java, or C#. However, Python does not support interfaces.

For detailed information, refer to the Python documentation for defining functions and classes.
# defining a function with name sum and two parameters a and b:
def sum(a, b):
    return a + b  # we return the sum of a and b.

# we can now call the function defined above:
print(sum(5, 7))

# Now we define a class Foo:
class Foo:
    # The class gets a method "bar".
    # Note: for methods, the first parameter is always "self" and
    # points to the current instance. This is similar to "this" in
    # ST and other languages.
    def bar(self, a, b):
        print("bar(%s,%s)" % (a, b))

# We create an instance of the class:
f = Foo()

# We call the method bar on the instance.
f.bar("some", "params")

See also
* Python Documentation, Defining Functions
* Python Documentation, Classes

In IEC, you can import libraries for reuse by other written code. As a pendant, there is the possibility in Python of importing modules.

The Python standard library contains many modules for different purposes, such as:
* String processing
* Date and time handling
* Collections
* Threading
* Mathematical functions
* File handling
* Persistence
* Compression and archiving
* Database access
* Encryption services
* Network and Internet access
* Sending of emails

To create your own modules, write a Python file that defines the functions and classes that you want to provide. Save this file to the same directory as our sample script. If you name the file mymodule.py, then you can import it with import mymodule.

Here is an example of importing and using the cosine function and the pi constant from the math module:
from math import cos, pi
print(pi)  # prints 3.14159265359
print(cos(pi))  # prints -1.0

The following contains more examples that access information about the operating system, the Python version, and the interpreter:

import os
print(os.environ["OS"])  

from sys import platform, version, executable
print(platform)
print(version)
print(executable)

There is a special module __future__ for activating new language features. Above all, it is used when Python developers introduce new functionalities that are backward compatible. These kinds of functionalities have to be activated with special "__future__ imports". One example that we use in most of our sample scripts here is the activation of the new power syntax of print as a function instead of a statement.

# make print() a function instead of a statement
from __future__ import print_function

The Python documentation provides a complete list of all __future__ imports.

In addition to the normal Python modules, IronPython code can also access .NET assemblies as if they were Python modules. This opens the access to the .NET framework class library and third-party libraries. Here is an example how to open a dialog box by means of the library Windows Forms:

import clr
clr.AddReference("System.Windows.Forms")
from System.Windows.Forms import MessageBox
MessageBox.Show("Hello")

See also
Using scripts to access CODESYS functionalities

All objects and commands that CODESYS provides for scripting are also in the Python module "scriptengine". Whenever a script is started, an implicit <code>from scriptengine import *</code> results. This allows for easy access to CODESYS. However, if your script imports modules that require access CODESYS APIs, then these modules have to import the module scriptengine themselves.

In the following table, you will find the main objects (categories) that can be used in Python scripts as entry points. For comprehensive documentation about entry points, see the API reference documentation for the CODESYS ScriptEngine.

<table>
<thead>
<tr>
<th>Objects</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>system</td>
<td>Access to general CODESYS functionalities, such as:</td>
</tr>
<tr>
<td></td>
<td>● Exiting CODESYS</td>
</tr>
<tr>
<td></td>
<td>● Handling the general user interface</td>
</tr>
<tr>
<td></td>
<td>● Access to the message memory (including compiler messages)</td>
</tr>
<tr>
<td></td>
<td>● Control of delay and progress bars.</td>
</tr>
<tr>
<td>projects</td>
<td>Access to the CODESYS project as an object tree that combines the three navigator views (devices, POUs, modules) in one project tree. Also allows for the loading, creating, saving, and closing of projects.</td>
</tr>
<tr>
<td></td>
<td>For most objects in a project, there are special methods with detailed functionality, for example compiling, access to ST POUs, export, import, device configuration, etc.</td>
</tr>
<tr>
<td>online</td>
<td>Access to online functionalities, such as:</td>
</tr>
<tr>
<td></td>
<td>● Login to devices and applications</td>
</tr>
<tr>
<td></td>
<td>● Management of access data (user name, password)</td>
</tr>
<tr>
<td></td>
<td>● Performance of network scans</td>
</tr>
<tr>
<td></td>
<td>● Gateway management</td>
</tr>
<tr>
<td>librarymanager</td>
<td>Permits the management of library repositories and viewing, installation, and removal of libraries.</td>
</tr>
<tr>
<td>device_repository</td>
<td>Handling of device repositories; import and export of device descriptions.</td>
</tr>
<tr>
<td>modulerepository</td>
<td>Management of CODESYS Application Composer modules and CODESYS Application Composer repositories.</td>
</tr>
</tbody>
</table>

See the following specific sample scripts for ways to access CODESYS functionalities. For detailed information, see the API reference documentation for the CODESYS ScriptEngine.

See also

✓ Chapter 1.3.1.22.2.2 “Tips for python programmers about .NET API documentation” on page 397
✓ Offline help: API Reference Documentation for the ScriptEngine:
✓ Online help: API Reference Documentation for the ScriptEngine:

Example: 
Printing the device tree of the current project

The script PrintDeviceTree.py is an example for navigating in a project. It creates the output of a hierarchical display of all devices in the open project.

Load a project that contains some device objects and execute the script.
Example: PrintDeviceTree.py

```python
# encoding:utf-8
# We enable the new python 3 print syntax
from __future__ import print_function

# Prints out all devices in the currently open project.
print("--- Printing the devices of the project: ---")

# Define the printing function. This function starts with the
# so called "docstring" which is the recommended way to document
# functions in python.
def print_tree(treeobj, depth=0):
    """ Print a device and all its children 
    Arguments:
    treeobj -- the object to print
    depth -- The current depth within the tree (default 0).
    The argument 'depth' is used by recursive call and
    should not be supplied by the user.
    """
    if treeobj.is_device:
        name = treeobj.get_name(False)
        deviceid = treeobj.get_device_identification()
        print("{0}- {1} {2}".format("--"*depth, name, deviceid))
    # we recursively call the print_tree function for the child
    objects.
    for child in treeobj.get_children(False):
        print_tree(child, depth+1)

    # We iterate over all top level objects and call the print_tree
    function for them.
    for obj in projects.primary.get_children():
        print_tree(obj)

print("--- Script finished. ---")
```

The device tree (from the "Devices" view) is displayed in the message view and all non-device
objects are left out:

---

Example: Reading of variables

The script ReadVariable.py logs in to the device and starts the application if necessary.
Then the value of the variable PLC_PRG.iVar1 is read and output. To try the script, you have
to modify the project path and variable names.
Example:

ReadVariable.py

```python
# encoding:utf-8
from __future__ import print_function

# close open project if necessary:
if projects.primary:
    projects.primary.close()

# opens project
proj = projects.open(r"D:\data\projects\Ampel.project")

# set "Ampel.project" to active application
app = proj.active_application
onlineapp = online.create_online_application(app)

# login to device
onlineapp.login(OnlineChangeOption.Try, True)

# set status of application to "run", if not in "run"
if not onlineapp.application_state == ApplicationState.run:
    onlineapp.start()

# wait 1 second
system.delay(1000)

# read value of iVar1
value = onlineapp.read_value("PLC_PRG.iVar1")

# display value in message view or command line
print(value)

# log out from device and close "Ampel.project"
onlineapp.logout()
proj.close()
```

In the extension of the script ReadVariable.py, the script MailVariables.py loads variables and expressions from a recipe file and reads their current values from the controller. Then these values are written back to the same file. In addition, it uses the Python SMTP library to send an email with an attachment containing a list of all variables.

To use the script, you have to modify the paths, email address, and the name of the SMTP server to your environment.
Example:

```
# encoding:utf-8
from __future__ import print_function

# Close current project if necessary and open "ScriptTest.project"
if not projects.primary == None:
    projects.primary.close()
project = projects.open("D:\Data\projects\scriptTest.project")

# retrieve active application
application = project.active_application

# create online application
online_application = online.create_online_application(application)

# login to application.
online_application.login(OnlineChangeOption.Try, True)

# start PLC if necessary
if not online_application.application_state == ApplicationState.run:
online_application.start()

# wait 2 seconds
system.delay(2000)

# open recipe file to read values.
recipe_input_file = open("D:\Data\projects\RecipeInput.txt", "r")
watch_expressions = []
for watch_expression in recipe_input_file:
    watch_expressions.append(watch_expression.strip())
print watch_expressions

# read values from the controllerd
watch_values = online_application.read_values(watch_expressions)
print watch_values

# open output file to write values
recipe_output_file = open("D:\Data\projects\RecipeOutput.txt", "w")
for i in range(len(watch_expressions)):
    recipe_output_file.write(watch_expressions[i])
    recipe_output_file.write(" = ")
    recipe_output_file.write(watch_values[i])
    recipe_output_file.write("\n")

# Close files
recipe_input_file.close()
recipe_output_file.close()

# send Email
# import respective libraries
import smtplib
from email.mime.text import MIMEText

#open output file
recipe_output_file = open("D:\Data\projects\RecipeOutput.txt", "r")
mail = MIMEText(recipe_output_file.read())
```
recipe_output_file.close()

# email address sender and recipient
fromm = "info@example.com"
to = "info@example.com"

# set sender and recipient
mail["Subject"] = "Attention value has changed"
mail["From"] = fromm
mail["To"] = to

# send email
smtp = smtplib.SMTP("name of smtp server")
smtp.sendmail(fromm, [to], mail.as_string())
smtp.quit()

# logout and close application
online_application.logout()
project.close()

Example: Creating and editing of POU's

The script CreateDut.py creates the objects MyStruct, MyAlias, and MyUnion in the CODESYS project. The folder DataTypes already has to be present.
CreateDut.py

```python
# encoding=utf-8
from __future__ import print_function

STRUCT_CONTENT = """
    a : BOOL;
    b : BIT;
    c : BIT;
"""

UNION_WHOLE = """
TYPE MyUnion :
    UNION
        Zahl : INT;
        Prozent : MyAlias;
        Bits : MyStruct;
    END_UNION
END_TYPE
"""

proj = projects.primary
folder = proj.find('DataTypes', recursive = True)[0]

# Create a struct DUT and insert the list of variables just into the right
# place in line two, row 0 (line numbering starts with line 0)
struktur = folder.create_dut('MyStruct') # DutType.Structure is the default
struktur.textual_declaration.insert(2, 0, STRUCT_CONTENT)

# Alias types get their "content" via the base type, which will just end up
# as one line in the declaration part:
# TYPE MyAlias : INT (0..100); END_TYPE
bereich = folder.create_dut('MyAlias', DutType.Alias, "INT (0..100)"

# Instead of injecting the variables into the existing declaration, one can also just replace the complete declaration part, including the
# boilerplate code.
union = folder.create_dut('MyUnion', DutType.Union)
union.textual_declaration.replace(UNION_WHOLE)
```

Example: User interface / Interaction with the user

In some cases, scripts have to interact with the user. We provide some simple APIs for the most common interactions. The sample script System_UI_Test.py shows all of the possible functions in this regard.
from __future__ import print_function

"""Performs some tests on the messagestore and UI."""

print("Some Error, Warning and Information popups:")
system.ui.error("Fatal error: Everything is OK. :-)")
system.ui.warning("Your bank account is surprisingly low")
system.ui.info("Just for your information: 42")

print("Now, we ask the user something.")
res = system.ui.prompt("Do you like this?", PromptChoice.YesNo, PromptResult.Yes);
print("The user selected '%s'" % res)

print("Now, the user can choose between custom options:")
res = system.ui.choose("Please choose:", ("First", 2, 7.5, "Something else"))
print("The user selected option '%s'" % str(res)) # res is a tuple

print("Now, the user can choose several options:")
res = system.ui.select_many("Please select one or more options", PromptChoice.OKCancel, PromptResult.OK, ("La Premiere", "The Second", "Das Dritte")
print("The returned result is: '%s'" % str(res)) # res is a tuple

print("Now, the user can select files and directories")
res = system.ui.open_file_dialog("Choose multiple files:", filter="Text files (*.txt)|*.txt|Image Files (*.BMP;*.JPG;*.GIF)| *.BMP;*.JPG;*.GIF|All files (*.*)|*.*", filter_index = 0, multiselect=True)
print("The user did choose: '%s'" % str(res)) # res is a tuple as multiselect is true.

res = system.ui.save_file_dialog("Choose a file to save:", filter="Text files (*.txt)|*.txt|Image Files (*.BMP;*.JPG;*.GIF)| *.BMP;*.JPG;*.GIF|All files (*.*)|*.*", filter_index = 0)
print("The user did choose: '%s'" % res)

res = system.ui.browse_directory_dialog("Choose a directory", path="C:\")
print("The user did choose: '%s'" % res)

print("Now we query a single line string")
res = system.ui.query_string("What's your name?")
print("Nice to meet you, dear %s." % res)

print("Now we query a multi line string")
res = system.ui.query_string("Please tell me a nice story about your life!", multi_line=True)
if (res):
    print("Huh, that has been a long text, at least %s characters!" % len(res))
else:
    print("Hey, don't be lazy!")

print("Username and password prompts...")
res = system.ui.query_password("Please enter your favourite password!", cancellable=True)
if res:
    print("Huh, it's very careless to tell me your favourite password!")
In the script `ProjectInfoExample.py`, we set some information in the “Project Information” object. The most important information items, such as “Title” and “Version”, have explicit properties. However, you can read and write any other information fields by means of the dictionary syntax. For example, those that are recommended for the properties of a library project.

The example below may seem somewhat unrealistic, but similar code is used in build servers that create, test, and possibly release automatic library projects and other projects. The ScriptEngine is one of the key elements for creating CI (Continuous Integration) and CD (Continuous Delivery) systems.

```python
# encoding: utf-8
from __future__ import print_function
proj = projects.load("D:\Some.library")
info = proj.get_project_info()

# Set some values
info.company = "Test Library Ltd"
info.title = "Script Test Project"
info.version = (0, 8, 15, 4711)
info.default_namespace = "testlibrary"
info.author = "Python von Scriptinger"

# some values recommended in the library toolchain
info.values["DefaultNamespace"] = "testlibrary"
info.values["Placeholder"] = "testlibrary"
info.values["DocFormat"] = "reStructuredText"

# now we set a custom / vendor specific value.
info.values["SpecialDeviceId"] = "PLC0815_4711"

# Enable generation of Accessor functions, so the IEC application can display the version in an info screen.
info.change_accessor_generation(True)

# And set the library to released
info.released = True;
proj.save()
```

See also

- § Chapter 1.3.1.19.1 “Information for library developers” on page 372
The sample script `DeviceImportFromSVN.py` gets a PLCOpenXML file from an external program (in this case a SVN client) and imports it into a new created CODESYS project. To use the script, you have to modify the paths to your environment.

```python
# encoding:utf-8
# Imports a Device in PLCOpenXML from Subversion via command line svn client.
# We enable the new python 3 print syntax
from __future__ import print_function
import sys, os

# some variable definitions:
SVNEXE = r"C:\Program Files\Subversion\bin\svn.exe"
XMLURL = "file:///D:/testrepo/testfolder/TestExport.xml"
PROJECT = r"D:\test.project"

# clean up any open project:
if projects.primary:
    projects.primary.close()

# Fetch the plcopenxml data from Subversion.
# We'll catch the output of the program into the xmldata variable.
# The 'with' construct automatically closes the open pipe for us.
with os.popen('"' + SVNEXE + '" cat ' + XMLURL, 'r') as pipe:
    xmldata = pipe.read()

# create a new project:
proj = projects.create(PROJECT)

# import the data into the project.
proj.import_xml(xmldata, False)

# and finally save. :-)
proj.save()

print("--- Script finished. ---")
```

**Example:**
Calling external commands and importing PLCOpenXML files

The following sample script can perform the calling and installing of a library as part of a CT (Continuous Testing) environment so that they can be tested. In addition to standard-CODESYS, the add-on CODESYS SVN also has to be installed with a valid license.

**Advanced example:**
Calling a library from SVN and installing it in CODESYS
import tempfile

if projects.primary:
    projects.primary.close()

tempdir = tempfile.mkdtemp()
URL = "svn://localhost/testrepo/trunk/SvnTestLibrary/"

proj = svn.checkout(URL, tempdir, "testlibrary", as_library=True)
proj.save()

repo = librarymanager.repositories[0]
librarymanager.install_library(proj.path, repo, True)

proj.close()

---

Transitioning from python 2 to python 3

With Python version 3, Python developers introduced some incompatible changes and removed some obsolete functionalities. At this time, the Python community is still in the transitional phase from version 2 to version 3.

IronPython does not yet support Python 3, but it is being worked on. As the Python community no longer supports Python 2, we intend to upgrade to Python 3 as soon as it is supported by IronPython. Although we strive for a smooth transition, script writers should take care that their scripts are created in a future-proof style. For example, by using the expression `from __future__ import print_function`.

See also
- Python 2 or Python 3?
- New in Python 3

Comparison of IronPython and cPython

There are some small differences and Incompatibilities between IronPython and standard Python ("cPython"). Some are direct errors in IronPython and should be removed in future versions. However, others are considered "implementation details" and will remain. Some of them are very challenging topics.

The difference that is most obvious for users is the handling of strings. Original cPython has two different string types for byte strings and Unicode strings. This concept is similar to the data types STRING and WSTRING in IEC. IronPython simply uses .NET strings that are always Unicode-capable and use UTF-16 internally. However, IronPython implements a trick to hide the difference to cPython from the programmer. (Interesting is that the developers completely reengineered string handling for the new Python version 3. A module was developed that is much closer to that of IronPython. Afterwards, only Unicode strings are used and there is an individual data type for handling raw bytes.)

Python modules that are written in C cannot be imported into IronPython because cPython uses internal data structures that are completely different from IronPython. Most of the standard library modules were reimplemented in IronPython. However, some modules (such as the TK interface) are not available as long as they are not ported to IronPython explicitly. On the other hand, IronPython provides access to .NET assemblies including the .NET framework (as shown above), which more than compensates for this feature.
While cPython uses reference counting and a deterministic garbage collector for cleaning up cyclic garbage, IronPython relies on the non-deterministic .NET garbage collector. In most cases, this difference does not matter. But when you open files or other resources from the Python standard library or the .NET framework, you should make sure to close them later. It is best to use the `with` statement with the Python context manager or .NET `IDisposable` instances.

See also
- *Information about Context Managers*
- *Information about .NET IDisposable*

### 1.3.1.23 Security

<table>
<thead>
<tr>
<th>Section</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.3.1.23.1</td>
<td>General information</td>
</tr>
<tr>
<td>1.3.1.23.2</td>
<td>Security for the development system</td>
</tr>
<tr>
<td>1.3.1.23.3</td>
<td>Security for the runtime system / PLC</td>
</tr>
<tr>
<td>1.3.1.23.4</td>
<td>FAQ</td>
</tr>
</tbody>
</table>

Due to the increased networking of controllers and plants, potential threats are also quickly rising. Therefore, you should carefully consider all possible security measures.

Security measures are absolutely necessary to protect data and communication channels from unauthorized access.

On the following help pages, you can learn more about the safety functions of CODESYS and the controller.

### 1.3.1.23.1 General information

Below is some general information about security measures. This information applies regardless of the use in CODESYS or on in a controller connected to CODESYS.

**Access protection with user management**

As a means of protecting against unauthorized access to data, it is necessary to configure user accounts with specific access rights. Only a user with the credentials has access to the data or functions.

The following types of user management are roughly distinguished:

- **Simple user management:**
  To access data, only a password or the valid combination of user name and password has to be entered. This means that access can be only granted or denied. Graduated rights cannot be configured.

- **Group-based user management:**
  The access rights are assigned to user groups. Users who belong to a group can access the data or functions after entering the credentials with precisely these assigned and different rights.

**Encryption, signature**

Encryption:

Encryption of data means the following: Data is converted into an unreadable form and can only be made readable again with a matching key. In the simplest case, the key is a password or a key pair.

There are two types of encryption methods:
• Symmetrical method: (the only type of encryption until the mid-1970s)
  Characteristic: Use of a secret key
  Advantages: Fast, easy encoding
  Disadvantages: The key must be shared secretly.

• Asymmetric method:
  Characteristic: Use of a key pair (private/secret key and public key)
  Advantages: The public key can be made accessible to anyone, and authentication possible with it.
  Disadvantages: Slow (approx. 1,000 to 10,000 times slower than symmetric methods); complex encoding; long key lengths

Key exchange is usually performed by asymmetric methods; encryption and decryption by symmetric methods.

Signature:
In order for the irrefutable ownership and integrity of a message to be verifiable, it should be provided with a signature. The following steps are common:

• Sender: Determines a unique hash value over the data (H)
• Sender: Encrypts the hash value with private key (He)
• Recipient: Also calculates the hash value and decrypts the He with the public key and compares the two values. This allows the sender to be identified uniquely and verifies that the sender owns the private key.

In the case of asymmetric encryption, a public key contained in a certificate is first exchanged between the sender and the recipient. In addition, each participant needs a private key with which they can decrypt the data if they have the certificate. So if you want to access a certificate, you need a certificate AND a private key.

Hash methods are necessary for this:

• Hash method:
  Characteristic: Unique fingerprint of the data (for example, checksum of the data)
  As low a collision as possible (it is very difficult to find / construct two different data for a single hash value)

Certificates
In order to assign the public key to an identity, it is usually embedded in a certificate.

In certificate-based systems, each user receives a digital certificate. The certificate is used for digital identification. It contains information about the identity and the public key of the user. Each certificate is authenticated by an issuing authority, which in turn may be authenticated by higher authorities. The trust system of this PKI (Public Key Infrastructure) is strictly hierarchical. The common trust anchor is a root certificate.

Contents of a certificate:

• Version
• Serial number
• Algorithm ID
• Issuer (authority or company)
• Validity from (not before) to (not after)
• Certificate owner (subject)
• Certificate owner key information (subject public key)
  – Public key algorithm
  – Public key of the certificate owner
• Unique ID of the issuer (optional)
• Unique ID of the owner (optional). The owner owns the private key that matches the public key.
• Extensions
  – Purpose (extended key usage)
  – ...

The certificate consists of 2 parts / files:
● Public X.509 certificate (can be issued to anyone)
● Private key that matches the certificate or its public key only (must be kept secret).

For information on how to manage the certificates in your local "Windows Certificate Store" you can also see in the current Help:

● Chapter 1.3.1.6 “Protecting and saving a project” on page 139

1.3.1.23.2 Security for the development system

In CODESYS, you can apply access protection to projects, libraries, as well as individual applications. In addition to a simple write protection for a project, a user management (credentials, access rights) and encryption using certificates should be used.

See current help:

● Chapter 1.3.1.6 “Protecting and saving a project” on page 139

1.3.1.23.3 Security for the runtime system / PLC

Communication with the controller connected in the CODESYS project should be protected against unauthorized access in the following ways:

● Enabling user management: simple or group-based
● Certificate-based encryption of communication with the controller

NOTICE!

With CODESYS V3.5 SP14, encrypted communication and user management are not enforced yet by default for SL runtime systems. However, you can explicitly set the communication to safe mode.

Enabling security features

First switch the communication to encryption so that you do not reveal any credentials to other users in the network when transferring the user management.

Enforcing encrypted communication

● On the controller:
  − Runtime version >= 3.5 SP14: Encryption can be enabled for “Communication Policy” and enforced for all clients.
  − Runtime version < 3.5 SP14: The controller can communicate with or without encryption. Encryption is not enforced.

See the current help regarding this:

● Chapter 1.3.1.11.2 “Encrypting communication, changing security settings” on page 296

● In CODESYS:
  − Encrypted communication can be selected as an option in the device editor on the “Communication Settings” tab (command or “Change Communication Policy” dialog) or in the “Security Screen” view.

See the current help regarding this:

● Chapter 1.3.1.11.2 “Encrypting communication, changing security settings” on page 296

If the CODESYS Security Agent is installed, then see the help for CODESYS Security Agent.
Enforcing a user management

- On the controller:
  - Runtime version >= 3.5 SP14: User management can be enabled for “Communication Policy” and enforced.
  - Runtime version < 3.5 SP14: Simple user management or one for users/groups with respective rights can be enabled.

- In CODESYS:
  - See the current help regarding this:
    - Chapter 1.3.1.11.2 “Encrypting communication, changing security settings” on page 296
    - Chapter 1.3.1.11.3 “Handling of device user management” on page 300

**OPC UA server**  
See current help:
- Chapter 1.3.3.1 “OPC UA server” on page 1100

### 1.3.1.23.4 FAQ

<table>
<thead>
<tr>
<th>Number</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.3.1.23.4.1</td>
<td>Certificate expired .................................................................................. 424</td>
</tr>
<tr>
<td>1.3.1.23.4.2</td>
<td>New certificate (while the current one is still valid).............................. 424</td>
</tr>
<tr>
<td>1.3.1.23.4.3</td>
<td>Client does not support security feature.................................................. 425</td>
</tr>
<tr>
<td>1.3.1.23.4.4</td>
<td>CA-signed certificates preferred (PLC shell)............................................. 425</td>
</tr>
<tr>
<td>1.3.1.23.4.5</td>
<td>Problems at login...................................................................................... 426</td>
</tr>
<tr>
<td>1.3.1.23.4.6</td>
<td>Disabling user management.......................................................................... 426</td>
</tr>
<tr>
<td>1.3.1.23.4.7</td>
<td>Permitting encrypted communication again............................................... 427</td>
</tr>
</tbody>
</table>

#### Certificate expired

If the certificate from the controller for encrypted communication has expired (valid from “not before” until “not after”), you get a prompt with a corresponding message in CODESYS when you attempt to access the controller. For example, to renew the expired certificate, you can accept the expired certificate and connect to the controller.

You will see this message again every time you try to login until a valid certificate is installed on the controller.

If you have created or imported a new certificate on the controller, then this new certificate will be available for you to accept the next time you login.

See also
- Chapter 1.3.1.11.2 “Encrypting communication, changing security settings” on page 296

#### New certificate (while the current one is still valid)

A new certificate can be issued before the existing certificate expires. This makes it possible for the encrypted communication to continue seamlessly. As soon as a new certificate is available on the controller parallel to the one currently used, the new certificate will be offered by the controller at the next login attempt. All you have to do is accept it.
See also
● “Installing a controller certificate for encrypted communication via the PLC shell of the device editor” on page 297

### Client does not support security feature

#### User management

The following CODESYS clients do not support user management yet:

- WebServer < V3.5.14.0

In order for these clients to be able to establish a connection to the controller, the user management must not be enabled.

#### Encrypted communication

The following CODESYS clients do not support encrypted communication yet:

- Data servers with compiler version <= V3.5.9.0
- WebVisu < V3.5.14.0 or in the case of enabled file transfer
- WebServer < V3.5.14.0
- Remote TargetVisu
- Data source ApplicationV3
- OPC Server V3
- PLCHandler < V3.5.14.0

In order for these clients to be able to establish a connection to the controller, the encrypted communication can be set as optional. Therefore clients can establish either an encrypted or an unencrypted connection.

> Do not use the same user or password for encrypted and unencrypted communication.

See also
● “Chapter 1.3.1.11.2 “Encrypting communication, changing security settings” on page 296

### CA-signed certificates preferred (PLC shell)

Using CA-signed certificates is not conveniently supported yet in CODESYS. However, you can still request and use these types of certificates. In the device editor, on the “PLC Shell” tab, you export the required CSR files to the file system and import from there the CER files sent from the certification authority.

#### Requesting and providing a CA-signed certificate

- You are connected to the controller.

1. First you generate certificate signing requests (CSR) of all server certificates.
   For this purpose, click the “PLC Shell” tab of the controller and type the command `cert-createcsr` in the input line.

2. Click the “Log” tab and then the refresh button (⟳).
   In the log entries, you can see that the CSR files were generated.
3. Click the "Files" tab and open the file path `cert/export` in the right side of the "Runtime" dialog.
   ⇒ The `export` folder contains the generated CSR files, for example `0_CmpsecureChannel.csr, 1_CmpApp.csr, 2_CmpWebServer.csr`.
4. Select a file path where you wish to insert the CSR files in the left side of the "Host" dialog, mark the CSR files in the right side of the dialog, and click `<<`.
   ⇒ The CSR files are copied to the required folder.
5. These requests can be signed for certification signing by a certificate authority (CA), and then you receive a signed certificate from the certification authority.
6. In the steps that follow, you import these signed server certificates to your controller.

| NOTICE! |
| Caution: Self-signed certificates of the server must be deleted before importing the CA-signed certificates. |

7. Select the "Path" `cert/import` in the right side of the "Runtime" dialog.
8. In the left side of the "Host" dialog, select the path in the file system where you saved the signed certificates and selected the certificates.
9. Click `>>`.
   ⇒ The certificates are copied to the `cert/import` folder.
10. Click the "PLC Shell" tab.
11. Type the command `cert-import own <file name of the certificate.cer>` in the input line of the tab and press the `[Enter]` key.
   ⇒ The signed certificates are available to the runtime system servers.

See also
- `Chapter 1.3.1.25.1.19.10 “Tab 'PLC Shell'” on page 753`
- `Chapter 1.3.1.25.1.19.8 “Tab 'Log'” on page 750`
- `Chapter 1.3.1.25.1.19.7 “Tab 'Files'” on page 750`

Problems at login

If you have entered an incorrect password when logging in to the user management of the controller, then the login dialog reappears immediately afterwards. After three incorrect attempts, the controller is locked for a defined period of time. However, stricter policies on the controller can lead to the user being locked out and only authorized again by an administrator.

See also
- `Chapter 1.3.1.25.2.6.2 “Command 'Login'” on page 923`

Disabling user management

| NOTICE! |
| After disabling the user management, your controller is accessible again for everyone in the network of the controller. Therefore, you should only do this in justified exceptional cases or if the clients used do not support any user management. |
To delete the user management of the controller after any unintentional activation, proceed as follows:

1. If the security policy for device user management is set to "Enforced", first set it back to "Optional".
2. Execute the "Reset Origin Device" command. This deletes the user management and you can then reconnect to the controller without having to enter user credentials.

See also

- § Chapter 1.3.1.11.2 “Encrypting communication, changing security settings” on page 296
- § Chapter 1.3.1.25.2.6.12 “Command ‘Reset Origin Device’” on page 933
- § Chapter 1.3.1.11.3 “Handling of device user management” on page 300
- § Chapter 1.3.1.6 “Protecting and saving a project” on page 139

Permitting encrypted communication again

| Remember that not every controller supports the deactivation of encrypted communication. |
| NOTICE! |
| We strongly advise against disabling encrypted communication. Especially in connection with an enabled user management, encrypted communication should be enabled so that credentials do not fall into the wrong hands. |

To disable encrypted communication with the controller again, proceed as follows:

1. If the communication policy for encrypted communication is set to "Enforced", first set it back to "Optional".
2. In the device editor, on the "Communication Settings" tab in the "Device" menu, disable "Encrypted communication". If you have installed the CODESYS Security Agent, then you can also change the setting in the "Security Screen".
   - CODESYS establishes unencrypted communication again with the controller. Other clients can also communicate again without encryption.

See also

- § Chapter 1.3.1.11.2 “Encrypting communication, changing security settings” on page 296
- § Chapter 1.3.1.25.2.3.18 “Command ‘Security Screen’” on page 891
1.3.1.24 Reference, Programming

1.3.1.24.1 Programming languages and editors

1.3.1.24.1.1 Declaration editor.................................................. 428
1.3.1.24.1.2 Common functions in graphical editors.................... 429
1.3.1.24.1.3 Structured text and extended structured text (ExST).... 430
1.3.1.24.1.4 Sequential function chart (SFC)............................ 442
1.3.1.24.1.5 Function block diagram / Ladder diagram / Instruction list (FBD/LD/IL).................................................. 461
1.3.1.24.1.6 Continuous function chart (CFC) and page-oriented CFC 476

You program a POU in each case in the editor for the implementation language that you selected when creating the POU. CODESYS offers a text editor for ST and graphic editors for SFC, FBD/LD/IL and CFC.

The editor opens with a double-click on the POU in the device tree or in the “POUs” view.

Each of the programming language editors consists of two sub-windows:

- In the upper part you make declarations in the “declaration editor”, in text or tabular form depending on the setting.
- In the lower part you insert the implementation code in the respective language.

You can configure the display and the behavior of each editor project-wide on the associated tab of the CODESYS options.

Declaration editor

The declaration editor is used for declaring variables in the variable lists and POUs.

If the declaration editor is used with an implementation language editor, then is opens in a view above the implementation language editor.

The declaration editor offers two possible views: textual and tabular. In the dialog in “Tools \ Options \ Declaration Editor”, you define whether only the textual view or only the tabular view is available, or whether you can switch between both views by means of the buttons on the right side of the editor view.

A rectangle selection is possible in the textual view of the declaration editor. You can find the key combination for the rectangle selection in the help page for the ST editor.

See also

- % Chapter 1.3.1.9.2.1 “Using the declaration editor” on page 173
- % Chapter 1.3.1.9.2 “Declaration of variables” on page 172
- % Chapter 1.3.1.24.1 “Programming languages and editors” on page 428
- % Chapter 1.3.1.25.3.13.6 “Dialog ‘Options’ – ‘Declaration Editor’” on page 1078
- % Chapter 1.3.1.24.1.3.1 “ST editor” on page 430
### Declaration editor in online mode

In online mode, you see the tabular view of the editor. The header always contains the current object path: `<device name>.<application name>.<object name>`. In contrast to offline mode, the table also contains the columns “Value” and “Prepared Value”.

The “Value” column shows the actual value on the PLC, offering monitoring functionality. If the expression is an array, then you can define the range of the array indices to monitor. You do this, double-click the “Data Type” column to open the “Monitoring Area” dialog and type in the “Start Index” and the “End Index” of the required range. You can use a scrollbar to define a series of 1000 elements easily.

The “Prepared value” column contains the value that you prepared for forcing or writing.

If you double-click a “Prepared Value” field, then you can specify a value explicitly for writing or forcing. In the case of enumerations, a combo box opens from which you can select a value. In the case of a Boolean variable you can toggle the prepared value with the help of the [Enter] key or the [Space] bar. If an expression (variable) is of a structured data type, for example the instance of a function block or an array variable, then a plus or a minus sign is placed in front.

You can customize the format of the representation of floating-point values in the options for monitoring.

See also

- Chapter 1.3.1.9.2.1 “Using the declaration editor” on page 173
- Chapter 1.3.1.25.3.7 “Dialog Box ‘Prepare Value’” on page 1043
- “Forcing in the declaration part” on page 316
- Chapter 1.3.1.25.3.13.14 “Dialog ‘Options’ - ‘Monitoring’” on page 1083

### Common functions in graphical editors

The implementation part of the graphical editors for FBD, LD, CFC, and SFC contains a toolbar in the lower right corner.

| ![Keyboard] | Return to normal editing mode: The mouse pointer changes back to the shape of the default arrow. You can select and edit elements in the editor view. |
| ![Panning] | Panning tool: The mouse pointer changes to the shape of two crossed arrows. You can click and drag anywhere in the editor view to move the visible area of the FBD/LD/IL editor or also pivot a CFC chart. |
| ![Magnification] | Magnification tool: A magnified window opens in the lower right corner of the editor view and the mouse pointer changes to the shape of a cross. As you move the mouse pointer over your diagram, the magnification tool shows the area of the diagram under the cross at 100% magnification. Note: If you click in the view, then the magnification tool closes and the part of the diagram that the tool contained is displayed at 100% magnification. If you want to retain the set zoom factor, then you should use the default arrow ( ![Keyboard]) for returning to the default editing mode. |
| ![Zooming] | Zooming tool: This opens a drop-down list with a selection of zoom factors. Clicking more selections ( . . . ) will open the “Zoom” dialog for typing other values. The current zoom factor is always shown to the left of the symbol. |

Zooming with the scroll wheel: By holding down the [Ctrl] key and moving the scroll wheel, you can change the zoom factor in steps of 10%.

Every graphical editor has its own “ToolBox” view that is located on the right of the editor view by default. The toolbox contains elements that you can drag to insertion points in the editor view. CODESYS highlights the insertion points with gray position flags in the shape of diamonds, triangles, or arrows. These flags are green when you move the mouse pointer over them. When you release the mouse button, CODESYS inserts the element at the selected position.

It is also possible to use the mouse for moving elements in the editor.
You can drag function block declarations in the FBD, LD, and CFC graphical editors to the editor view. To do this, select the full declaration (variable name and data type) and drag it to a suitable position in the editor view. In the ladder diagram, you can also drag Boolean declarations to the editor and insert them as contacts.

See also
- Chapter 1.3.1.24.1.4.1 “SFC editor” on page 442
- Chapter 1.3.1.24.1.5.1 “FBD/LD/IL editor” on page 461
- Chapter 1.3.1.24.1.6.1 “CFC editor” on page 477

### Structured text and extended structured text (ExST)

<table>
<thead>
<tr>
<th>Section</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.3.1.24.1.3.1</td>
<td>ST editor</td>
</tr>
<tr>
<td>1.3.1.24.1.3.2</td>
<td>ST editor in online mode</td>
</tr>
<tr>
<td>1.3.1.24.1.3.3</td>
<td>ST expressions</td>
</tr>
<tr>
<td>1.3.1.24.1.3.4</td>
<td>Assignments</td>
</tr>
<tr>
<td>1.3.1.24.1.3.5</td>
<td>Statements</td>
</tr>
</tbody>
</table>

### ST editor

The ST editor is a textual editor used for the implementation of code in Structured Text (ST) and Extended Structured Text (ExST).

The line numbering is displayed on the left side of the editor. When inputing programming elements, the "List components" functionality (activated in the CODESYS options, "SmartCoding" category) and the input assistant [F2] are also helpful. When the cursor is placed over a variable, CODESYS shows a tooltip with information for declaring variables.

The box selection can be made with the following key combinations:
- [Shift] + [Alt] + [Arrow right]: The selected area is extended one position to the right.
- [Shift] + [Alt] + [Arrow left]: The selected area is extended one position to the left.
- [Shift] + [Alt] + [Arrow up]: The selected area is extended one position up.
- [Shift] + [Alt] + [Arrow down]: The selected area is extended one position down.

The behavior (for example parentheses, mouse actions, tabs) and appearance of the editor are configured in the "Text Editor" category of the CODESYS options.

For an incremental search for strings in the editor, open an input field at the bottom edge of the editor by means of the key combination [Ctrl] + [Shift] + [i]. As soon as you start typing in characters, the corresponding search locations are highlighted in color in the editor. The number of found matches is shown to the right of the input field. You can set the cursor at the search location by using the arrow keys or the key combinations [Alt] + [Page up] or [Alt] + [Page down].

When you place the cursor on a symbol name, all occurrence locations of the symbol within the editor are highlighted in color. The search locations correspond to the hits in the cross-reference list. For very large projects, this can cause input delays. In this case, you can disable the function in the options of the text editor.

CODESYS identifies syntax errors already when inputing in the editor and shows the corresponding messages in the message view ("Precompile" category). If the corresponding option is selected in the CODESYS options ("SmartCoding" category), then the error locations in the text are also underlined with a wavy red line.

See also
- Chapter 1.3.1.9.3.3.1 “Programming structured text (ST)” on page 201
- Chapter 1.3.1.24.1.3.3 “ST expressions” on page 431
- Chapter 1.3.1.24.1.3.5.11 “ST comments” on page 442
- Chapter 1.3.1.24.1.3 “Structured text and extended structured text (ExST)” on page 430
- Chapter 1.3.1.25.3.13.20 “Dialog 'Options' - 'Text Editor'” on page 1086
- Chapter 1.3.1.25.2.3.5 “Command 'Messages'” on page 882
ST editor in online mode

In online mode CODESYS displays the variables and expressions in the ST editor. The writing and forcing of the variables and expressions as well as debugging functions (breakpoints, single step execution) are also possible.

If you use assignments as expressions in ST programming, no further breakpoint positions are created within a line.

See also
- Chapter 1.3.1.13.1.1 “Calling of monitoring in programming objects” on page 323
- Chapter 1.3.1.12.4 “Forcing and writing variables” on page 315
- Chapter 1.3.1.13.1.2 “Using watch lists” on page 330
- Chapter 1.3.1.12.2 “Using breakpoints” on page 310
- Chapter 1.3.1.12 “Testing and debugging” on page 309
- Chapter 1.3.1.12 “Testing and debugging” on page 309
- Chapter 1.3.1.12.3 “Stepping through a program” on page 313
- Chapter 1.3.1.12.6 “Flow control” on page 319
- Chapter 1.3.1.12.7 “Determining the current processing position with the call stack” on page 322

ST expressions

An expression is a construct that returns a value following its evaluation.

Expressions are composed of operators and operands. In Extended Structured Text (ExST) you can also use assignments as expressions. An operand can be a constant, a variable, a function call or a further expression.

<table>
<thead>
<tr>
<th>Examples</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>2014</td>
<td>(* Constant *)</td>
</tr>
<tr>
<td>lvar</td>
<td>(* Variable *)</td>
</tr>
<tr>
<td>fct(a,b)</td>
<td>(* Function call *)</td>
</tr>
<tr>
<td>(x*y)/z</td>
<td>(* Expression *)</td>
</tr>
<tr>
<td>real_var2 := int.var;</td>
<td>(* in ExST: Assignment *)</td>
</tr>
</tbody>
</table>

See also
- “ExST - Extended structured text” on page 201

Evaluation of expressions

The evaluation of an expression takes place by processing the operators according to certain rules of binding. CODESYS processes the operator with the strongest binding first. Operators with the same binding strength are processed from left to right.

<table>
<thead>
<tr>
<th>Operation</th>
<th>Symbol</th>
<th>Binding strength</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parenthesize</td>
<td>(Expression)</td>
<td>Strongest binding</td>
</tr>
<tr>
<td>Function Call</td>
<td>Function name (parameter list)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>all operators with syntax: &lt;operator&gt;</td>
<td></td>
</tr>
<tr>
<td></td>
<td>()</td>
<td></td>
</tr>
<tr>
<td>Exponentiate</td>
<td>EXPT</td>
<td></td>
</tr>
<tr>
<td>Negate</td>
<td>–</td>
<td></td>
</tr>
<tr>
<td>Complementation</td>
<td>NOT</td>
<td></td>
</tr>
<tr>
<td>Operation</td>
<td>Symbol</td>
<td>Binding strength</td>
</tr>
<tr>
<td>-----------------</td>
<td>--------------</td>
<td>------------------</td>
</tr>
<tr>
<td>Multiplication</td>
<td>*</td>
<td></td>
</tr>
<tr>
<td>Division</td>
<td>/</td>
<td></td>
</tr>
<tr>
<td>Modulo</td>
<td>MOD</td>
<td></td>
</tr>
<tr>
<td>Addition</td>
<td>+</td>
<td></td>
</tr>
<tr>
<td>Subtraction</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>Comparison</td>
<td>&lt;,&gt;,&lt;=,&gt;=</td>
<td></td>
</tr>
<tr>
<td>Equality</td>
<td>=</td>
<td></td>
</tr>
<tr>
<td>Inequality</td>
<td>&lt;&gt;</td>
<td></td>
</tr>
<tr>
<td>Bool AND</td>
<td>AND</td>
<td></td>
</tr>
<tr>
<td>Bool XOR</td>
<td>XOR</td>
<td></td>
</tr>
<tr>
<td>Bool OR</td>
<td>OR</td>
<td>Weakest binding</td>
</tr>
<tr>
<td></td>
<td>OR_ELSE</td>
<td></td>
</tr>
</tbody>
</table>

See also
- % Chapter 1.3.1.24.3 “Operators” on page 505

## Assignments

1.3.1.24.1.3.4.1 ST assignment operator......................................................... 432
1.3.1.24.1.3.4.2 ST assignment operator for outputs......................................... 432
1.3.1.24.1.3.4.3 ExST assignment ‘S=’.............................................................. 433
1.3.1.24.1.3.4.4 ExST assignment ‘R=’............................................................ 434
1.3.1.24.1.3.4.5 ExST – Assignment as expression........................................... 435
1.3.1.24.1.3.4.6 Assignment operator ‘REF’................................................... 435

### ST assignment operator

**Syntax:**

<operand> := <expression>

This assignment operator executes the same function as the `MOVE` operator.

See also
- % Chapter 1.3.1.24.3.6 “Operator ‘MOVE’” on page 512

### ST assignment operator for outputs

The assignment operator => assigns the output of a function, a function block, or a method to a variable. The position on the right side of the operator can also be blank.

**Syntax**

<output> => <variable>
FBcomp_Output1 and FB_Output2 are outputs of a function block. The value of FBcom_Output1 is assigned to the variable bVar1.

**ExST assignment 'S='**

When the operand of the Set assignment switches to TRUE, then TRUE is assigned to the variable to the left of the operator. The variable is set.

<variable name> S= <operand name> ;

The variables and the operand have the data type BOOL.

**Example**

```plaintext
PROGRAM PLC_PRG
VAR
  xOperand: BOOL := FALSE;
  xSetVariable: BOOL := FALSE;
END_VAR

xSetVariable S= xOperand;
```

When the operand xOperand switches from FALSE to TRUE, then TRUE is also assigned to the variable xSetVariable. But then the variable keeps this state, even if the operand continues to change its state.

---

**Multiple assignments**

In the case of multiple assignments within a code line, the individual assignments are not processed from right to left, but all assignments refer to the operands at the end of the code line.

**Example**

```plaintext
FUNCTION funCompute : BOOL
VAR_INPUT
  xIn : BOOL;
END_VAR
IF xIn = TRUE THEN
  funCompute := TRUE;
RETURN;
END_IF

PROGRAM PLC_PRG
VAR
  xSetVariable: BOOL;
  xResetVariable: BOOL := TRUE;
  xVar: BOOL;
END_VAR

xSetVariable S= xResetVariable R= funCompute(xIn := xVar);
```

xResetVariable gets the R= assignment of the return value of funCompute. xSetVariable gets the S= assignment of its return value of funCompute, but not from xResetVariable.
ExST assignment 'R='

When the operand of the Reset assignment switches to \texttt{TRUE}, then \texttt{FALSE} is assigned to the variable to the left of the operator. The variable is reset.

\begin{verbatim}
<variable name> R= <operand name> ;
\end{verbatim}

The variables and the operand have the data type \texttt{BOOL}.

**Example**

\begin{verbatim}
VAR
  xOperand: BOOL := FALSE;
  xResetVariable: BOOL := TRUE;
END_VAR

xResetVariable R= xOperand;
\end{verbatim}

When the operand \texttt{xOperand} switches from \texttt{FALSE} to \texttt{TRUE}, then \texttt{FALSE} is also assigned to the variable \texttt{xResetVariable}. But then the variable keeps its state, even if the operand continues to change its state.

**NOTICE!**

In the case of multiple assignments within a code line, the individual assignments are not processed from right to left, but all assignments refer to the operands at the end of the code line.

**Example**

\begin{verbatim}
FUNCTION funCompute : BOOL
  VAR_INPUT
    xIn : BOOL;
  END_VAR
  IF xIn = TRUE THEN
    funCompute := TRUE;
    RETURN;
  END_IF
END_FUNCTION

PROGRAM PLC_PRG
  VAR
    xSetVariable: BOOL;
    xResetVariable: BOOL := TRUE;
    xVar: BOOL;
  END_VAR
  xSetVariable S= xResetVariable R= funCompute(xIn := xVar);
  xResetVariable gets the \texttt{R=} assignment of the return value of \texttt{funCompute}.
  xSetVariable gets the \texttt{S=} assignment of its return value of \texttt{funCompute}, but not from \texttt{xResetVariable}.
\end{verbatim}
ExST – Assignment as expression

In ExST, as an extension to the IEC 61131-3 standard, CODESYS permits the use of assignments as expressions.

### Examples

<table>
<thead>
<tr>
<th>Assignment</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>int_var1 := int_var2 := int_var3 + 9;</code></td>
<td>(* <code>int_var1</code> and <code>int_var2</code> receive the value of <code>int_var3 + 9</code> *)</td>
</tr>
<tr>
<td><code>real_var1 := real_var2 := int_var;</code></td>
<td>(* <code>real_var1</code> and <code>real_var2</code> receive the value of <code>int_var</code> *)</td>
</tr>
<tr>
<td><code>int_var := real_var1 := int_var;</code></td>
<td>(* Incorrect assignment, the data types do not correspond! *)</td>
</tr>
<tr>
<td><code>IF b := (i = 1) THEN</code>&lt;br&gt;<code>i := i + 1; END_IF</code></td>
<td></td>
</tr>
</tbody>
</table>

See also

- ↪ “ExST - Extended structured text” on page 201
- ↪ Chapter 1.3.1.24.1.3.4.3 “ExST assignment ‘S=’” on page 433

Assignment operator 'REF'

This operator generates a reference (pointer) to a value.

**Syntax:**

```
REF= <variable>
```

**Example**

```
A : REFERENCE TO DUT;
B : DUT;
C : DUT;

A REF= B; // corresponds to A := ADR(B);
A := C; // corresponds to A^ := C;
```

See also

- ↪ Chapter 1.3.1.24.5.13 “Reference” on page 579
Statements

1.3.1.24.1.3.5.1  ST statement 'IF'................................................................. 436
1.3.1.24.1.3.5.2  ST instruction 'FOR'............................................................. 437
1.3.1.24.1.3.5.3  ST instruction 'CASE'........................................................... 438
1.3.1.24.1.3.5.4  ST instruction 'WHILE'.......................................................... 439
1.3.1.24.1.3.5.5  ST instruction 'REPEAT'......................................................... 439
1.3.1.24.1.3.5.6  ST statement 'RETURN'....................................................... 440
1.3.1.24.1.3.5.7  ST instruction 'JMP'............................................................... 440
1.3.1.24.1.3.5.8  ST instruction 'EXIT'............................................................. 441
1.3.1.24.1.3.5.9  EXST instruction 'CONTINUE'................................................ 441
1.3.1.24.1.3.5.10 ST function block call....................................................... 441
1.3.1.24.1.3.5.11 ST comments........................................................................ 442

ST statement 'IF'

The IF statement is used for checking a condition and, depending on this condition, for executing the subsequent statements.

A condition is coded as an expression that returns a Boolean value. If the expression returns TRUE, then the condition is fulfilled and the corresponding statements after THEN are executed. If the expression returns FALSE, then the following conditions, which are identified with ELSIF, are evaluated. If an ELSIF condition returns TRUE, then the statements are executed after the corresponding THEN. If all conditions return FALSE, then the statements after ELSE are executed.

Therefore, at most one branch of the IF statement is executed. ELSIF branches and the ELSE branch are optional.

Syntax

IF <condition> THEN  
<statements>   
( ELSIF <condition> THEN  
<statements> )*  
( ELSE  
<statements> )?  
END_IF;  
// { ... }* None, once or several times  
// { ... }? Optional
Example

PROGRAM PLC_PRG
VAR
  iTemp: INT;
  xHeatingOn: BOOL;
  xOpenWindow: BOOL;
END_VAR

IF iTemp < 17 THEN
  xHeatingOn := TRUE;
ELSIF iTemp > 25 THEN
  xOpenWindow := TRUE;
ELSE
  xHeatingOn := FALSE;
END_IF;

The program is run as follows at runtime:
For the evaluation of the expression \( iTemp < 17 = \text{TRUE} \), the subsequent statement is executed and the heating is switched on. For the evaluation of the expression \( iTemp < 17 = \text{FALSE} \), the subsequent ELSIF condition \( iTemp > 25 \) is evaluated. If this is true, then the statements in ELSIF are executed and the view is opened. If all conditions are \text{FALSE}, then the statement in ELSE is executed and the heating is switched off.

See also

- Chapter 1.3.1.24.1.3.3 “ST expressions” on page 431

ST instruction ‘FOR’

The FOR loop is used to execute instructions with a certain number of repetitions.

Syntax:

FOR <counter> := <start value> TO <end value> {BY <increment> } DO
  <instructions>
END_FOR;

The section inside the curly parentheses {} is optional.
CODESYS executes the <instructions> as long as the <counter> is not greater, or - in case of negative increment - is not smaller than the <end value>. This is checked before the execution of the <instructions>.

Every time the instructions <instructions> have been executed, the counter <counter> is automatically increased by the increment <increment>. The increment <increment> can have any integral value. If you do not specify an increment, the standard increment is 1.

Example

FOR iCounter := 1 TO 5 BY 1 DO
  iVar1 := iVar1*2;
END_FOR;

If you have pre-configured iVar1 with 1, iVar1 has the value 32 after the FOR loop.
CAUTION!
The end value <end value> may not attain the same value as the upper limit of the data type of the counter.

If the end value of the counter is equal to the upper limit of the data type of the counter, an endless loop results. For example, an endless loop results in the above example if iCounter is of the data type SINT and the <end value> equals 127, since the data type SINT has the upper limit 127.

As an extension to the IEC 61131-3 standard you can use the CONTINUE instruction within the FOR loop.

See also
- § Chapter 1.3.1.24.5.2 “Integer data types” on page 570
- § Chapter 1.3.1.24.1.3.5.9 “EXST instruction ‘CONTINUE’” on page 441

ST instruction ‘CASE’

Use this dialog box for pooling several conditional instructions containing the same condition variable into a construct.

Syntax:
CASE <Var1> OF
  <value1>:<instruction1>
  <value2>:<instruction2>
  <value3, value4, value5>:<instruction3>
  <value6 ... value10>:<instruction4>
  ...
  <value n>:<instruction n>
{ELSE <ELSE-instruction>}
END_CASE;

The section within the curly brackets {} is optional.

Processing scheme of a CASE instruction.
- If the value of the variable <Var1> is <value i>, then the instruction <instruction i> is executed.
- If the variable <Var1> has non of the given values, then the <ELSE-instruction> is executed.
- If the same instruction is executed for several values of the variable, then you can write the values in sequence, seperated by commas.

Example

CASE iVar OF
  1, 5: bVar1 := TRUE;
      bVar3 := FALSE;
  2: bVar2 := FALSE;
      bVar3 := TRUE;
  10..20: bVar1 := TRUE;
          bVar3 := TRUE;
ELSE
  bVar1 := NOT bVar1;
  bVar2 := bVar1 OR bVar2;
END_CASE;
**ST instruction 'WHILE'**

The **WHILE** loop is used like the **FOR** loop in order to execute instructions several times until the abort condition occurs. The abort condition of a **WHILE** loop is a boolean expression.

**Syntax:**

```st
WHILE <boolean expression> DO
  <instructions>
END_WHILE;
```

CODESYS repeatedly executes the `<instructions>` for as long as the `<boolean expression>` returns **TRUE**. If the boolean expression is already **FALSE** at the first evaluation, then CODESYS never executes the instructions. If the boolean expression never adopts the value **FALSE**, then the instructions are repeated endlessly, as a result of which a runtime error results.

**Example**

```st
WHILE iCounter <> 0 DO
  Var1 := Var1*2
  iCounter := iCounter-1;
END_WHILE;
```

**NOTICE!**

You must ensure by programming means that no endless loops are caused.

In a certain sense the **WHILE** and **REPEAT** loops are more powerful than the **FOR** loop, since you don't need to already know the number of executions of the loop before its execution. In some cases it is thus only possible to work with these two kinds of loop. If the number of executions of the loop is clear, however, then a **FOR** loop is preferable in order to avoid endless loops.

As an extension to the IEC 61131-3 standard you can use the **CONTINUE** instruction within the **WHILE** loop.

See also

- § Chapter 1.3.1.24.1.3.5.2 “ST instruction ‘FOR’” on page 437
- § Chapter 1.3.1.24.1.3.5.9 “EXST instruction ‘CONTINUE’” on page 441

**ST instruction 'REPEAT'**

The **REPEAT** loop is used like the **WHILE** loop, but with the difference that CODESYS only checks the abort condition after the execution of the loop. The consequence of this behavior is that the **REPEAT** loop is executed at least once, regardless of the abort condition.

**Syntax:**

```st
REPEAT
  <instructions>
UNTIL <boolean expression>
END_REPEAT;
```

CODESYS executes the `<instructions>` until the `<boolean expression>` returns **TRUE**. If the boolean expression already returns **TRUE** at the first evaluation, CODESYS executes the instructions precisely once. If the boolean expression never adopts the value **TRUE**, then the instructions are repeated endlessly, as a result of which a runtime error results.
Example

```plaintext
REPEAT
  Var1 := Var1*2;
iCounter := iCounter-1;
UNTIL
iCounter = 0
END_REPEAT;
```

In a certain sense the **WHILE** and **REPEAT** loops are more powerful than the **FOR** loop, since the number of executions of the loop doesn't already need to be known before its execution. In some cases you can only work with these two kinds of loop. If the number of executions of the loop is clear, however, then a **FOR** loop is preferable in order to avoid endless loops.

As an extension to the IEC 61131-3 standard you can use the **CONTINUE** instruction within the **WHILE** loop.

See also

- [Chapter 1.3.1.24.1.3.5.4 “ST instruction 'WHILE'” on page 439](#)
- [Chapter 1.3.1.24.1.3.5.2 “ST instruction 'FOR'” on page 437](#)
- [Chapter 1.3.1.24.1.3.5.9 “EXST instruction 'CONTINUE'” on page 441](#)

**ST statement 'RETURN'**

Use the **RETURN** statement in order to exit from a function block. You can make this dependent on a condition, for example.

Example

```plaintext
IF xIsDone = TRUE THEN
  RETURN;
END_IF;

iCounter := iCounter + 1;
```

If the value of `xIsDone` is equal to **TRUE**, then the function block is exited immediately and the statement `iCounter := iCounter + 1;` is not executed.

See also

- [Chapter 1.3.1.24.1.3.5.1 “ST statement 'IF'” on page 436](#)

**ST instruction 'JMP'**

The **JMP** instruction is used to execute an unconditional jump to a program line that is marked by a jump label.

**Syntax:**

```plaintext
<label>: <instructions>
JMP <label>;
```

The jump label `<label>` is any unique identifier that you place at the beginning of a program line. On reaching the **JMP** instruction, a return to the program line with the `<label>` takes place.
Example

iVar1 := 0;
_label1: iVar1 := iVar1+1;
(*instructions*)

IF (iVar1 < 10) THEN
  JMP _label1;
END_IF;

NOTICE!

You must ensure by programming means that no endless loops are caused. For example, you can make the jump conditional.

ST instruction 'EXIT'

The EXIT instruction is used in a FOR, WHILE or REPEAT loop in order to end the loop regardless of other abort conditions.

See also

- § Chapter 1.3.1.24.1.3.5.2 “ST instruction 'FOR'” on page 437
- § Chapter 1.3.1.24.1.3.5.4 “ST instruction 'WHILE'” on page 439
- § Chapter 1.3.1.24.1.3.5.5 “ST instruction 'REPEAT'” on page 439

EXST instruction 'CONTINUE'

CONTINUE is an instruction of the Extended Structured Text (ExST).

The instruction is used inside FOR, WHILE and REPEAT loops in order to jump to the beginning of the next execution of the loop.

Example

FOR Counter:=1 TO 5 BY 1 DO
  INT1:=INT1/2;
  IF INT1=0 THEN
    CONTINUE; (* to provide a division by zero *)
    END_IF
  Var1:=Var1/INT1; (* executed, if INT1 is not 0 *)
END_FOR;

Erg:=Var1;

See also

- § Chapter 1.3.1.24.1.3.5.2 “ST instruction 'FOR'” on page 437
- § Chapter 1.3.1.24.1.3.5.4 “ST instruction 'WHILE'” on page 439
- § Chapter 1.3.1.24.1.3.5.5 “ST instruction 'REPEAT'” on page 439

ST function block call

Syntax

<FB-instance>(<FB input variable>:=<value or address>, <other FB input variables>);
The timer function block **TON** is instanced in **TMR:TON** and called with assignments for the parameters **IN** and **PT**. The output **Q** is addressed with **TMR.Q** and assigned to the variable **varA**.

See also
- Chapter 1.3.1.25.1.28.2 “Object ‘Function Block’” on page 778

**ST comments**

<table>
<thead>
<tr>
<th>Comment</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Single-line</td>
<td>Starts with // and ends at the end of the line.</td>
<td>// This is a comment</td>
</tr>
<tr>
<td>Multi-line</td>
<td>Starts with (* and ends with *).</td>
<td>(* This is a multi-line comment *)</td>
</tr>
<tr>
<td>Nested</td>
<td>Starts with (* and ends with <em>). There may be further (</em> . . . *) comments inside this comment.</td>
<td>(* a:=inst.out; (* 1st comment <em>) b:=b+1; (</em> 2nd comment *) *)</td>
</tr>
</tbody>
</table>

**Sequential function chart (SFC)**

- SFC editor
- SFC editor in online mode
- Processing order in SFC
- Qualifiers for actions in SFC
- Implicit variables
- SFC flags
- Elements

**SFC editor**

The SFC editor is graphical editor. A new SFC POU includes an Init step and a subsequent transition.

In the SFC editor, you can insert individual elements into the diagram by means of commands in the “SFC” menu, the context menu, or the “ToolBox” view.

When inserting by means of a menu command, the elements that can be inserted at the currently selected position are available.

Before inserting branches parallel to multiple actions and transitions, you must highlight these actions and transitions in a multiple selection.

You can also drag SFC elements from the “ToolBox” view to the diagram. When you drag an element over the editor, CODESYS marks all possible insertion points with gray boxes. If you move the mouse over a gray box, then the color of the box changes to green. When you release the mouse button, the object is inserted at that location.
If you drag a branch into the diagram, then you must set the beginning and the end of the branch using the mouse pointer. You set the beginning of the branch by releasing the mouse button at an insertion point. The color of the box then changes to red. You set the end of the branch by clicking the second insertion point. Then CODESYS inserts a branch around the objects between the beginning and end markers.

For copying step and transition elements that call action objects or transition objects, two different duplication modes can be set. Either the references are copied at the same time, or the referenced objects are embedded and duplicated when copying.

You define the look and feel of the editor in the CODESYS options ("SFC Editor").

See also
- "Common functions in graphical editors" on page 429
- "Sequential function chart (SFC)" on page 442
- "Programming the sequential function chart (SFC)" on page 202
- "Dialog 'Options' - 'SFC Editor'" on page 1074

**SFC editor in online mode**

In the SFC editor, you can display the variables and expressions in use on the controller in online mode. You can also write and force variables and expressions. Debugging functions, such as breakpoints and step-by-step execution, are not yet available.

You can set the online display of the SFC elements and attributes in the CODESYS options ("SFC Editor").

If you have declared SFC flags explicitly, then they are displayed in the declaration section in online mode. They are not displayed in offline mode.

*Please note the processing order of elements in an SFC diagram.*

In online mode, CODESYS displays active steps in blue.
See also
● § Chapter 1.3.1.24.1.4.5 “Implicit variables” on page 446
● § Chapter 1.3.1.9.2.1 “Using the declaration editor” on page 173
● § Chapter 1.3.1.24.1.4.3 “Processing order in SFC” on page 444
● § Chapter 1.3.1.25.3.13.1 “Dialog ‘Options’ - ‘SFC Editor’” on page 1074

Processing order in SFC

Basic element behavior

- Active step: An active step includes actions currently being executed. In online mode, CODESYS displays active steps in blue.
- Initial step: In the first cycle after calling a POU in SFC, the initial step is activated automatically and the step action is executed.
- CODESYS executes IEC actions at least two times: the first time is when the step is activated, and the second time when the step is deactivated (but not until the next cycle).
- Alternative branches: If the step before the branch is active, then CODESYS passes the first transition of each alternative branch line from left to right. CODESYS activates the subsequent step in the first branch line with a transition yielding TRUE.
- Parallel branches: If the step before the branch (horizontal double line) is active and the transition before the branch yields TRUE, then CODESYS activates the first steps in every branch line. The branch lines are then processed at the same time. The step after the end of the branch is activated when every last step in each branch line is active and the transition after the double line yields TRUE.

1. Reset IEC actions
CODESYS resets the internal action control flags of the action qualifiers (N, R, S, L, D, P, SD, DS, SL). These flags control IEC actions. However, flags are not reset when they are called within actions.

2. Execute exit actions
CODESYS verifies whether all steps fulfill the condition for executing the exit action for each step. The order of verification follows the layout in the SFC diagram, from top to bottom and from left to right.
CODESYS executes an exit action when the step is deactivated (after any entry and step actions have been executed in the preceding cycle and the condition for the subsequent step yields TRUE).

3. Execute entry actions
CODESYS verifies whether all steps fulfill the condition for executing the entry action for each step. The order of verification follows the layout in the SFC diagram, from top to bottom and from left to right. If the conditions are fulfilled, then CODESYS executes the entry actions.
CODESYS executes an entry action as soon as the transition of the preceding step has been processed and yields TRUE, thus indicating that the step has been activated.

4. Time check / Execute step actions
CODESYS performs the following check for each step in the order of the SFC layout:
- CODESYS copies the elapsed time of the active step to the respective implicit step variable <step name>.t. (not yet implemented)
- If a timeout occurs, then CODESYS sets the respective error flags. (not yet implemented)
- For non-IEC steps: CODESYS executes the step action.

5. Execute IEC actions
CODESYS executes the IEC actions in alphabetical order, passing through the list of actions two times. In the first pass, CODESYS executes the IEC actions for each step that was deactivated in the preceding cycle. In the second pass, the IEC actions are executed for each active step.
6. Transition check / Activate next steps

The transitions are passed as follows: If a step is active in the current cycle and the subsequent transition yields `TRUE` and any defined minimum time of the step has elapsed, then the subsequent step is activated.

**NOTICE!**

Please note when executing actions:

An action can be executed multiple times within the same cycle if you use it in multiple SFC diagrams. For example, if a sequential function chart includes two IEC actions A and B, both of which are programmed in SFC and call an IEC action C, then the IEC action C is called two times.

If you use the same IEC action at the same time in different levels of an SFC diagram, then this can lead to unpredictable results when processing. For this reason, CODESYS issues a corresponding error message. This error message can appear for projects that have been created in an earlier version of the development system.

Please note: It is possible to use implicit variables to monitor the processing status of steps and actions and to control processing.

See also

- Chapter 1.3.1.24.1.4.5 “Implicit variables” on page 446
- Chapter 1.3.1.24.1.4.4 “Qualifiers for actions in SFC” on page 445

Qualifiers for actions in SFC

You assign qualifiers to IEC steps. Qualifiers describe how a step action is processed.

Qualifiers are processed by the `SFCActionControl` function block in the library `IecSfc.library`. The library is automatically integrated into the project by the SFC plug-in.

<table>
<thead>
<tr>
<th>Qualifier</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>N</td>
<td>Non-stored</td>
</tr>
<tr>
<td>R0</td>
<td>overriding Reset</td>
</tr>
<tr>
<td>S0</td>
<td>Set (Stored)</td>
</tr>
<tr>
<td>L</td>
<td>time Limited</td>
</tr>
<tr>
<td>D</td>
<td>time Delayed</td>
</tr>
<tr>
<td>P</td>
<td>Pulse</td>
</tr>
<tr>
<td>SD</td>
<td>Stored and time Delayed</td>
</tr>
<tr>
<td>DS</td>
<td>Delayed and Stored</td>
</tr>
<tr>
<td>-----</td>
<td>-------------------</td>
</tr>
<tr>
<td>CODESYS begins executing the action only after the given delay time has elapsed following step activation and the step is still active. The action is executed until it receives a reset.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SL</th>
<th>Stored and time Limited</th>
</tr>
</thead>
<tbody>
<tr>
<td>CODESYS executes this action as soon as the step is active. The action is executed until the given time has elapsed or it receives a reset.</td>
<td></td>
</tr>
</tbody>
</table>

You must enter the times for the L, D, SD, DS, and SL qualifiers in the format of a TIME constant.

When an IEC action is deactivated, it is executed one more time. This means that CODESYS executes this kind of action at least two times. This also applies to actions with the P qualifier.

See also

- ☞ Chapter 1.3.1.9.3.4.1 “Programming the sequential function chart (SFC)” on page 202

Implicit variables

Every SFC object supplies implicit variables for you to monitor the status of steps and IEC actions at runtime. These implicit variables are declared automatically by CODESYS for each step and each IEC action.

The implicit variables are structure instances of the type `SFCStepType` for steps and type `SFCActionType` for actions. The variables have the same names as their elements, for example "step1" variable name for "step1" step name. The structure members describe the status of a step or action or the currently elapsed time in an active step.

In the element properties, you can define whether CODESYS should export a symbol definition for this flag to the symbol configuration.

See also

- ☞ Chapter 1.3.1.24.1.4.7.6 “SFC element properties” on page 459

**Step and action status**

Syntax for the implicit variable declaration:

```
<step name>:SFCStepType;
_<action name>:SFCActionType;
```

**Table 13: The following implicit variables are available for step or IEC action status:**

<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&lt;step name&gt;.x</code></td>
<td>Shows the activation status in the current cycle. When <code>&lt;step name&gt;.x</code> = TRUE, CODESYS processes the step in the current cycle.</td>
</tr>
<tr>
<td><code>&lt;step name&gt;._x</code></td>
<td>Shows the activation status for the next cycle. When <code>&lt;step name&gt;._x</code> = TRUE and <code>&lt;step name&gt;.x</code> = FALSE, CODESYS processes the step in the next cycle. This means that <code>&lt;step name&gt;._x</code> is copied to <code>&lt;step name&gt;.x</code> at the beginning of a cycle.</td>
</tr>
</tbody>
</table>
The flag `t` yields the current elapsed time since the step was activated. This applies only to steps, regardless of whether a minimum time has been defined or not in the step properties. Also see SFC flag SFCError.

For internal use only

| IEC action | 
| --- | --- |
| `<action name>._x` | TRUE when the action is being executed. |
| `<action name>._x` | TRUE when the action is active. |

**NOTICE!**

You can use the above variables to force a specific status value to a step (activate a step). However, note that this can cause an unstable status in the SFC.

See also
- Chapter 1.3.1.24.1.4.6 “SFC flags” on page 447

**Access to implicit variables**

Syntax for access:

Assign the implicit variable directly in the POU: `<variable name>:=<step name>..<implicit variable> or <variable name>:=<action name>..<implicit variable>`

**Example**

```plaintext
status:=step1._x;
```

From another POU, with the POU name: `<variable name>:=<POU name>..<step name>..<implicit variable> or <variable name>:=<POU name>..<action name>..<implicit variable>`

**Example**

```plaintext
status:=SFC_prog.step1._x;
```

**Symbol generation**

In the element properties of a step or action, you define whether CODESYS should add a symbol definition for the step or action flag. In the “Properties” view, you have to select the necessary access rights in the “Symbol” column.

See also
- Chapter 1.3.1.24.1.4.7.6 “SFC element properties” on page 459

**SFC flags**

SFC flags are implicitly generated variables with predefined names used for controlling the processing of an SFC diagram. You can use these flags, for example, to display timeouts or reset step chains. In addition, you can activate tip mode specifically to activate transitions. You have to declare and activate these variables in order to have access to them.

**Implicit generation of SFC flags**

CODESYS declares SFC flags automatically when you activate the respective options. You can set this option in the “SFC Settings” tab of the properties dialog box for each POU, or in the “SFC” project settings dialog box for each SFC POU in the project.
The SFC settings for the SFC flags of individual POUs are effective only if you have not selected the "Use defaults" check box. If you select this option, then the settings apply that were defined in the project settings.

SFC flags that you declare in the SFC settings dialog box are visible only in the online view of the SFC block.

See also

- Chapter 1.3.1.25.3.10.13 “Dialog 'Properties' - 'SFC Settings’” on page 1055

Explicit generation of SFC flags

Manual declaration, which was necessary in CoDeSys V2.3, is now only required to enable write access from another block. In this case, you should note that when you declare the flag in a global variable list, you must deactivate its "Declare" setting in the SFC settings dialog box. If you do not do this, then a local SFC flag is implicitly declared that CODESYS uses instead of the global variable.
You have created an SFC block named sfc1, which contains the s1 step. You have defined timeouts in the step properties. (See "Online view of SFC block sfc1" below.)

If for any reason the s1 step remains active longer than its time properties have permitted (timeout), then CODESYS sets the SFCError flag to permit access by the application.

To permit access, you have to declare and activate the SFC flag in the SFC settings. If you have only declared it, then the SFC flag is only displayed in the online view of sfc1 in the declaration section, but it has no function.

Now the SFC flag can be referenced within the POU, for example in an action (2) or outside of the block (1).

Online view of the SFC block sfc1
SFCError is TRUE as long as a timeout occurs within *sfc2*.

### Access to the flags

**Syntax for access**

Within the POU, you assign the flag directly: `<variable name>:=<SFC flag>`

**Example**

```plaintext
checkerror:=SFCerror;
```

**From another block with POU name:** `<variable name>:=<POU-Name>.<SFC flag>`

**Example**

```plaintext
checkerror:=SFC_prog.SFCerror;
```

If you need write access from another block, then you also have to declare the SFC flag explicitly as a VAR_INPUT variable in the SFC block or globally in a GVL.
### Example

Local declaration:

```plaintext
PROGRAM SFC_prog
VAR_INPUT
  SFCinit:BOOL;
END_VAR

Global declaration in a global variable list:

```plaintext
VAR_GLOBAL
  SFCinit:BOOL;
END_VAR

PROGRAM PLC_PRG
VAR
  setinit: BOOL;
END_VAR
```

```plaintext
SFC_prog.SFCinit:=setinit;  // write access to SFCinit in SFC_prog
```

### SFC flags

<table>
<thead>
<tr>
<th>Name</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SFCInit</td>
<td>BOOL</td>
<td>TRUE: CODESYS resets the sequence to the initial step. The other SFC flags are also reset (initialization). While the variable is TRUE, the initial step remains set (active), but its actions are not executed. Only when you reset SFCInit to FALSE does the block processing continue.</td>
</tr>
<tr>
<td>SFCReset</td>
<td>BOOL</td>
<td>This functions similar to SFCInit. However, CODESYS continues processing after the initialization of the initial step. For example, in the initial step, you could immediately reset the SFCReset flag to FALSE.</td>
</tr>
<tr>
<td>SFCError</td>
<td>BOOL</td>
<td>TRUE if a timeout occurs in an SFC diagram. If second timeout occurs in the program, it is not registered unless you previously reset the variable SFCError. The declaration of SFCError is a requirement for other flag variables to function for controlling the chronological sequence (“SFCErrorStep”, SFCErrorPOU, SFCQuitError).</td>
</tr>
<tr>
<td>SFCEnableLimit</td>
<td>BOOL</td>
<td>Used specifically for activating (TRUE) and deactivating (FALSE) the timeout control in steps using SFCError. If you declare and activate this variable (SFC settings), then you must set it to TRUE for SFCError to work. If you do not, then the timeouts are ignored. This is useful, for example, at start-up or in manual operation. If you do not declare the variable, then SFCError will work automatically. The requirement is the declaration of SFCError.</td>
</tr>
<tr>
<td>SFCErrorStep</td>
<td>String</td>
<td>Stores the name of the step that caused a timeout, which was registered by SFCError. The requirement is the declaration of SFCError.</td>
</tr>
<tr>
<td>SFCErrorPOU</td>
<td>String</td>
<td>Stores the name of the block in which a timeout occurred and was registered by SFCError. The requirement is the declaration of SFCError.</td>
</tr>
<tr>
<td>SFCQuitError</td>
<td>BOOL</td>
<td>As long as this boolean variable is TRUE, CODESYS pauses the processing of the SFC diagram and any timeout in the variable SFCError is reset. If you reset the variable to FALSE, then all previous times in the active steps are reset. The requirement is the declaration of SFCError.</td>
</tr>
<tr>
<td>SFCPause</td>
<td>BOOL</td>
<td>As long as this variable is TRUE, CODESYS pauses the processing of the SFC diagram.</td>
</tr>
<tr>
<td>SFCTrans</td>
<td>BOOL</td>
<td>TRUE if a transition is active.</td>
</tr>
</tbody>
</table>
### Name | Data Type | Description
--- | --- | ---
SFCCurrentStep | String | Shows the name of the active step, regardless of the time monitoring. In parallel branches, the name of the step of the rightmost branch line is always stored.  
SFCTip, SFCTipMode | BOOL | Controls the tip mode of the SFC block. If you enable this flag with SFCTipMode=TRUE, then you can activate the next step only by setting SFCTip to TRUE. While SFCTipMode is set to FALSE, transitions can also be used to continue activation.

**Elements**

1.3.1.24.1.4.7.1 SFC elements 'Step' and 'Transition' .................................................. 452  
1.3.1.24.1.4.7.2 SFC element 'Action' .............................................................................. 454  
1.3.1.24.1.4.7.3 SFC element 'Branch' ............................................................................... 457  
1.3.1.24.1.4.7.4 SFC element 'Jump' .................................................................................. 458  
1.3.1.24.1.4.7.5 SFC element 'Macro' ................................................................................ 458  
1.3.1.24.1.4.7.6 SFC element properties .......................................................................... 459

**SFC elements 'Step' and 'Transition'**

As a rule, CODESYS inserts steps and transitions as combinations. Inserting a step without a transition or a transition without a step causes an error when compiling. You can modify this by double-clicking the name.

![Step symbol; Transition symbol](image)

**NOTICE!**

Step names must be unique within the scope of the parent block. Consider this especially when using actions that were also programmed in SFC.

Please note that you can convert a step into an initial step by clicking "Init step" or by setting the respective property in the SFC properties.

All steps are defined by the step properties, which you can display and edit in the "Properties" view, depending on the set options.

You have to add those actions to the step which are to be executed when the step is active. A distinction is made between IEC actions and step actions. Details for this are found in the chapter about the SFC element "Action".

A transition must include the condition for the subsequent step to be active as soon as the value of the condition yields TRUE. Therefore, a transition condition must yield TRUE or FALSE. It can be defined in one of two ways:

- **(1) Inline condition (direct):** You replace the default transition name with either the name of a Boolean variable, a Boolean address, a Boolean constant, or a statement with a Boolean result, for example (i<100) AND b. You cannot specify programs, function blocks, or assignments here.

- **(2) Multi-use condition (separate transition or property object):** You replace the default transition name with the name of a transition or property object (condition). You create these objects by clicking "Project ➔ Add Object". This allows multiple use of transitions, for example "condition_xy" in the figures below. Like an inline condition, the object can contain a Boolean variable, Boolean address, Boolean constant, or an statement with a Boolean result. In addition, it can also contain multiple statements with any code.
**NOTICE!**

The user is responsible for assigning the required expression to a transition variable if the transition includes multiple instructions.

Transitions that reference a transition or property object are marked with a small triangle in the upper right corner of the transition box.

As opposed to CoDeSys V2.3, now CODESYS treats a transition condition like a method call. The entry has the following syntax:

```
<transition name>::=<transition condition>
```
You will find an example (condition_xy) in the figure above.

See also

- Chapter 1.3.1.9.3.4.1 “Programming the sequential function chart (SFC)” on page 202
- Chapter 1.3.1.25.2.11.6 “Command ‘Insert Step-Transition’” on page 969
- Chapter 1.3.1.24.1.4.7.2 “SFC element ‘Action’” on page 454
- Chapter 1.3.1.25.2.11.1 “Command ‘Init Step’” on page 967
- Chapter 1.3.1.24.1.4.7.6 “SFC element properties” on page 459
- Chapter 1.3.1.9.23.4 “Calling methods” on page 267

**SFC element ‘Action’**

Symbol: 

An action includes one or more statements in one of the valid implementation languages. You can assign an action to a step.

Actions that you use in SFC steps must be created as POUs in the project.

Exception: In the case of IEC actions, which you add to a step as action association, you can also specify a Boolean variable instead of an action object. The value of these variables is switched between FALSE and TRUE each time the "action" is executed.

**NOTICE!**

You must define unique step names within the scope of the parent block. An action written in SFC cannot contain a step with a name identical to the step to which the action is assigned.

A distinction is made between IEC actions and step actions:

1. **IEC actions**

IEC actions comply with the IEC1131-3 standard. They are executed according to their qualifiers.

IEC actions are executed two times: first when the step is activated and second when the step is deactivated. If you assign multiple actions to one step, then the action list is processed from top to bottom.

Each action box includes the qualifier in the first column and the action name in the second column, both of which can be edited directly.
NOTICE!
If the same global Boolean variable is associated as an IEC action in different SFC blocks, then this can result in undesired overwriting effects.

In contrast to step actions, you can use different qualifiers for IEC actions. Moreover, each IEC action is provided with a control flag. This directs CODESYS to execute an action only one time at any moment, even if the action is called by another step at the same time. This cannot be guaranteed for step actions.

You assign IEC actions to steps by clicking “SFC ➔ Insert Action Association”.

See also
- ⇐ Chapter 1.3.1.25.2.11.14 “Command ’Insert Action Association’” on page 972
- ⇐ Chapter 1.3.1.24.1.4.4 “Qualifiers for actions in SFC” on page 445

2. Step actions
You can use these step actions to extend the IEC standard.

- **Entry action:**
  CODESYS executes this action after the step is activated and before the main action is executed.
  You reference a new action, or an action created below the SFC object, from a step by means of the “Entry action” element property (2). You can also add a new action to the step by means of the “Add Entry Action” command. The entry action is marked with an ⧫ in the lower left corner of the step box.

- **Main action:**
  CODESYS executes this action when the step is active and any entry actions have already been processed. However, in contrast to IEC actions (see above), these step actions are not executed a second time when the step is deactivated. Moreover, you cannot use qualifiers here.
  You add an existing action to a step by means of the “Main action” element property (1). You can create and add a new action by clicking the step element. A main action is marked with a filled triangle in the upper right corner of the step box.

- **Exit action:**
  CODESYS executes this action one time when the step is deactivated. Please note, however, that an exit action is not executed in the same cycle, but at the beginning of the next cycle.
  You reference a new action, or an action created below the SFC object, from a step by means of the “Exit action” element property (3). You can also add a new action to the step by means of the “Add Exit Action” command. The exit action is marked with an ✗ in the lower right corner of the step box.
The main difference between step actions and IEC actions with a qualifier N is that an IEC action is always executed two times: when the step is activated and when the step is deactivated. See the following example:

You have attached the Action_AS1 action to the AS1 step as a step action (left) and as an IEC action with qualifier N (right). Because two transitions are activated in each case, the time to reach the initial step again is two PLC cycles. This is true as long as the iCounter counter variable was initialized at 0 and then incremented in the Action_AS1 action. After the Init step is reactivated, iCounter returns a value of 1 in the example on the left. In the example on the right, a value of 2 is returned because the IEC action is executed a second time due to the deactivation of AS1.

Another difference is that step actions can be pseudo-embedded. In this case, they can be called only from the related step. If you copy this step, CODESYS creates new action objects automatically and copies the respective implementation code. You define whether or not a step action is embedded, either when the first action is inserted into the step, or later in the "Duplicate when copying" element property. In general, this behavior can also be preset in the SFC options.

Finally, for IEC actions a Boolean variable can be specified instead of an action object. This is not possible for step actions.
SFC element 'Branch'

Symbol

Use branches to program parallel or alternative sequences in the sequential function chart.

For alternative branches, CODESYS processes just one of the branch lines at a time, depending on the preceding transition condition. Parallel branches are processed at the same time.

See also
●  Chapter 1.3.1.24.1.4.3 “Processing order in SFC” on page 444
●  Chapter 1.3.1.9.3.4.1 “Programming the sequential function chart (SFC)” on page 202
●  Chapter 1.3.1.25.2.11.13 “Command ‘Insert Branch Right’” on page 971

Parallel branch

For parallel branches, the branch lines must begin and end with steps. Parallel branch lines can contain additional branches.

The horizontal lines before and after the branch are double lines.

Processing in online mode: If the preceding transition (t2 in the example) yields TRUE, then the first steps in all parallel branch lines are active (Step11 and Step21). CODESYS processes the individual branch lines at the same time and the subsequent transition is passed afterwards (t3).

The "Branch<n>" jump marker is added automatically to the horizontal line that indicates the beginning of a branch. You can define this marker as the jump destination.

Please note that you can convert a parallel branch into an alternative branch by clicking "Alternative".

See also
●  Chapter 1.3.1.25.2.11.11 “Command ‘Alternative’” on page 971

Alternative branch

The horizontal line before and after the branch is a single line.

In an alternative branch, the branch lines must begin and end with transitions. The branch lines can contain additional branches.
If the step before the branch is active, then CODESYS passes the first transition of each alternative branch line from left to right. For the first transition that yields **TRUE**, the associated branch line opens, thus activating the step following the transition.

Please note that you can convert an alternative branch into a parallel branch by clicking "Parallel".

See also

- § Chapter 1.3.1.25.2.11.10 “Command 'Parallel'” on page 970

**SFC element 'Jump'**

Symbol $
$  

Use a jump to define which actions in a step should be executed next as soon as the transition preceding the jump is **TRUE**. Jumps may become necessary, as execution paths cannot cross or lead upwards.

Excluding the required jump at the end of a diagram, you can generally insert jumps only at the end of a branch.

The destination of a jump is defined by the added text string, which you can edit directly. The jump destination can be a step name or the marker for a parallel branch.

See also

- § Chapter 1.3.1.9.3.4.1 “Programming the sequential function chart (SFC)” on page 202
- § Chapter 1.3.1.25.2.11.16 “Command 'Insert Jump’” on page 973

**SFC element 'Macro'**

Symbol $
$
A macro includes part of the SFC diagram, but it is not displayed in detail in the main view of the editor.

Using macros does not influence the processing flow. Macros are used for hiding specific parts of the diagram, for example to increase overall clarity.

You open the macro editor by double-clicking the macro box or by clicking “SFC ➔ Zoom Into Macro”. You can program here just like in the main view of the SFC editor. To close the macro editor, click “SFC ➔ Zoom Out of Macro”.

Macros can also include other macros. The caption of the macro editor always shows the path of the open macro within the diagram, for example:

```
Macro1 ➔ Macro2
```

See also

- % Chapter 1.3.1.9.3.4.1 “Programming the sequential function chart (SFC)” on page 202
- % Chapter 1.3.1.25.2.11.20 “Command ’Zoom Into Macro’” on page 974
- % Chapter 1.3.1.25.2.11.21 “Command ’Zoom Out of Macro’” on page 974

**SFC element properties**

You edit the properties of an SFC element in the “Properties” view. Click “View ➔ Element Properties” to open this view. The properties to be displayed depend on the currently selected element.
The properties that are displayed in the SFC diagram next to the element depend on the settings in the “View” tab of the SFC editor options.

### General

<table>
<thead>
<tr>
<th>Property</th>
<th>Value description</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Name”</td>
<td>Element name, by default &quot;&lt;element&gt;&lt;consecutive number&gt;&quot;, for example step name &quot;Step0&quot;, &quot;Step1&quot;, branch name &quot;Branch0&quot;, etc.</td>
</tr>
<tr>
<td>“Comment”</td>
<td>Element comment in text, for example “counter reset”. You can insert line breaks by pressing [Ctrl]+[Enter].</td>
</tr>
</tbody>
</table>
| “Symbol” | For each SFC element, CODESYS declares an implicit variable with the same name as the element. The configuration determines whether this flag variable should be exported to the symbol configuration and which access rights for the symbol should be applied in the PLC.  
  - “No access”: The symbol is exported to the symbol configuration but cannot be accessed from the PLC.  
  - “Read”: The symbol is exported to the symbol configuration and can be read from the PLC.  
  - “Write”: The symbol is exported to the symbol configuration and can be written from the PLC.  
  - “Read/Write”: Combination of read and write.  
  - Empty: A symbol is not exported to the symbol configuration. |

### Specific

<table>
<thead>
<tr>
<th>Property</th>
<th>Value description</th>
</tr>
</thead>
</table>
| “Init step”            | [✔]: This option is activated only for the defined initial step. By default, this is the first step in an SFC diagram.  
  Note: If you activate this property for another step, then it must be deactivated in the previous step to prevent compilation errors. |
| “Duplicate when copying”| [✔]: When copying the step or transition, a new object is created for each called action or transition. It contains a copy of the implementation code of the copied object.  
  [☐]: When copying a step or transition, the link to the called object is retained for the respective action or transition. No new objects are generated. The source and the copies of the step or transition call the same action or transition. |
**Property** | **Value description**
---|---
"Times" | Minimum time that the step is active, even when the subsequent transition is TRUE.
  ● "Minimum active" | Maximum time that the step can be active. If this time is exceeded, then CODESYS sets the \texttt{SFCError} implicit variable to TRUE.
  ● "Maximum active" | Times according to IEC syntax (for example t#8s) or the \texttt{TIME} variable; default: t#0s.

"Actions" | \textbf{● "Entry action"}: CODESYS executes these actions after activating the step.
  ● "Step action" | \textbf{● "Step action"}: CODESYS executes this action when the step is active and any entry actions have already been processed.
  ● "Exit action" | \textbf{● "Exit action"}: CODESYS executes this action in the subsequent cycle when the step is deactivated.

**When using the respective implicit SFC variables and flags, you receive information about the status of a step or an action or about timeouts.**

See also
- § Chapter 1.3.1.25.3.13.1 “Dialog ‘Options’ - ‘SFC Editor’” on page 1074
- § Chapter 1.3.1.24.1.4.5 “Implicit variables” on page 446
- § Chapter 1.3.1.24.1.4.7.2 “SFC element ‘Action’” on page 454

**Function block diagram / Ladder diagram / Instruction list (FBD/LD/IL)**

1.3.1.24.1.5.1 FBD/LD/IL editor................................................................. 461
1.3.1.24.1.5.2 FBD/LD/IL editor in online mode........................................ 465
1.3.1.24.1.5.3 Modifiers and operators in IL............................................. 466
1.3.1.24.1.5.4 Elements........................................................................ 470

**FBD/LD/IL editor**

The FBD/LD/IL editor is a combined editor of the programming languages FBD, LD and IL.

*If necessary, IL can be activated in the CODESYS options.*

There is a common set of commands and elements and CODESYS automatically converts the 3 programming languages into one another internally.

The code in the implementation part is structured in all three languages with the aid of networks.

The "FBD/LD/IL" menu provides the commands for working in the editor.

In offline and online modes, you can switch editors at any time by using the menu command in "View".

The behavior of the FBD/LD/IL editor is defined by the settings in "Tools ‣ Options" (category "FBD, LD and IL").
NOTICE!
There are some special elements that CODESYS cannot convert and thus it displays only in the applicable language. There are also constructs that are not clearly convertible between IL and FBD and are therefore ‘normalized’, i.e. nullified, when converted back to FBD. This concerns: negation of expressions and explicit/implicit assignment of function block inputs and outputs.

An error-free conversion between the languages requires syntactically correct code. Otherwise parts of the implementation can be lost.

See also
- Chapter 1.3.1.9.3.1 “FBD/LD/IL” on page 182 (programming)
- Chapter 1.3.1.25.2.13 “Menu ‘FBD/LD/IL’” on page 992 (commands)
- Chapter 1.3.1.25.1.13.8 “Dialog ‘Options’ - ‘FBD, LD, and IL’” on page 1079
- Chapter 1.3.1.24.1.2 “Common functions in graphical editors” on page 429

### FBD and LD editor

**Inserting and arranging elements**

You can drag elements with the mouse from the view “Tools” (toolbox) into the implementation part of the editor. Alternatively you can use the commands of the context menu or the “FBD/LD/IL” menu.

Settings for the display and interface are defined in the CODESYS options, category “FBD/LD/IL”.

If you drag an element with the mouse over a network in the editor, all possible insertion positions are displayed with gray diamond-shaped, triangular or arrow-shaped position marks. As soon as the mouse pointer is located over one of these marks, the mark turns green. If the mouse button is now released, CODESYS inserts the element at this position.

![Example](image)

If you drag a function block or an operator from the toolbox or a network at the left-hand side of the network onto one of the two arrows, then CODESYS automatically creates a new network and inserts the element there.

In order to replace an element, drag a suitable other element onto its position with the mouse. Elements that you can replace by the new element are marked by CODESYS in the editor with text fields, for example “Replace”, “Attach input”.

You can use the usual commands in the menu “Edit” for cutting, copying, pasting and deleting elements. Copying also works with drag-and-drop by holding down the [Ctrl] key.

**NOTICE!**
The operators with EN/ENO functionality can only be inserted in the FBD and IL editors.

### Selecting elements

A box or a connecting line in the editor is selected by clicking on it with the mouse so that it has the focus. Multiple selection is possible by keeping the [Ctrl] key pressed. A selected element is shaded red.

**Tooltip**
If the cursor points to certain elements, for example to a variable or to an input, a tooltip appears showing information about this element.

In the case of elements underlined with a wavy red line, the tooltip shows the pre-compile error message of the error that occurs with this element.

**Navigating in the editor**

<table>
<thead>
<tr>
<th>Table 14: Navigating in the editor</th>
</tr>
</thead>
<tbody>
<tr>
<td>With the help of the keys and commands described below, you can place the focus within the editor on a different cursor position. The change between the positions is also network-spanning in function.</td>
</tr>
<tr>
<td>[←]</td>
</tr>
<tr>
<td>[→]</td>
</tr>
<tr>
<td>[↑][↓]</td>
</tr>
<tr>
<td>[Ctrl] + [Home]</td>
</tr>
<tr>
<td>[Ctrl] + [End]</td>
</tr>
<tr>
<td>[Page Up]</td>
</tr>
<tr>
<td>[Page Down]</td>
</tr>
<tr>
<td>Command “Go to…”</td>
</tr>
</tbody>
</table>

**Opening function blocks**

If a function block is inserted in the editor, then you can open its implementation by a double-click or with the context menu command “Browse for symbol ➔ Go to definition”.

See also

- “Function block diagram (FBD)” on page 182
- “Ladder diagram (LD)” on page 183
- Chapter 1.3.1.9.3.1.1 “Programming function block diagrams (FBD)” on page 184
- Chapter 1.3.1.9.3.1.2 “Programming ladder diagrams (LD)” on page 186
- Chapter 1.3.1.24.1.5.4 “Elements” on page 470
- Chapter 1.3.1.25.3.13.8 “Dialog 'Options' - 'FBD, LD, and IL'” on page 1079
- Chapter 1.3.1.24.1.5.2 “FBD/LD/IL editor in online mode” on page 465
- Chapter 1.3.1.25.2.13.44 “Command ‘Go to’” on page 1004

**IL editor**

**Inserting and arranging elements:**

You can insert elements with the help of the commands of the menu “FBD/LD/IL” in the context menu. You can also drag a new network from the tool box into the implementation part of the editor by drag-and-drop.

You can use the usual commands in the menu “Edit” for cutting, copying, pasting and deleting elements. Copying also works with drag-and-drop by holding down the [Ctrl] key.

**NOTICE!**

Please note that operators with EN/ENO functionality can only be inserted in the FBD and IL editors.
Each program line is entered in a table row.

**Table 15: Structure of networks in the IL editor**

<table>
<thead>
<tr>
<th>Column</th>
<th>Contents</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Operator</td>
<td>Contains the IL operator (LD, ST, CAL, AND, OR, etc.) or a function name. If you call a function block, you must additionally specify the corresponding parameters here; in the preceding field you must enter in this case := or =&gt;.</td>
</tr>
<tr>
<td>2</td>
<td>Operand</td>
<td>Contains precisely one operand or the name of a jump label. In the case of several operands you must enter them in several rows and when doing so insert a comma directly behind the individual operands. (See example below)</td>
</tr>
<tr>
<td>3</td>
<td>Address</td>
<td>Contains the address of the operand as defined in its declaration. non-editable You can activate/deactivate the display via the option “Display symbol address”. To do this, select the command “Tools ➔ Options” and the “General” tab in the category “FBD, LD and IL”.</td>
</tr>
<tr>
<td>4</td>
<td>Symbol comment</td>
<td>Contains the comment that was specified for the operand if necessary in the declaration. non-editable You can activate/deactivate the display via the option “Display symbol comment” if you select the command “Tools ➔ Options” and the “General” tab in the category “FBD, LD and IL”.</td>
</tr>
<tr>
<td>5</td>
<td>Operand comment</td>
<td>Comment for the current program line. You can activate/deactivate the display via the option “Operand comment” if you select the command “Tools ➔ Options” and the “General” tab in the category “FBD, LD and IL”.</td>
</tr>
</tbody>
</table>

**Example**

```
CAL     tonInst1()
    IN:= bVar,
    FT:= t1,
    ET=> t0ut11
LD      tonInst1.0
ST      tonInst2.IN
CAL     tonInst2()
    FT:= t2,
    0=> bReady,
    ET=> t0ut2)
```

Spalte: 1: CAL, tonInst1(), IN:= bVar, FT:= t1, ET=> t0ut1
Spalte: 2: is TRUE, FT seconds after IN had a r.
Spalte: 3: tonInst1.0
Spalte: 4: starts timer with rising edge, reset.
Spalte: 5: for tonInst2

---

**Example**

```
PLC Automation with V3 CPUs
Programming with IEC 61131-3 editor > CODESYS Development System
```

2020/12/10 3ADR010583, 1, en_US 464
### Table 16: Navigating in the editor

<table>
<thead>
<tr>
<th>Key(s)/command</th>
<th>Cursor movement</th>
</tr>
</thead>
<tbody>
<tr>
<td>[↑], [↓]</td>
<td>Jumps to the field located above/below.</td>
</tr>
<tr>
<td>[Tab]</td>
<td>Jumps one field to the right within the row.</td>
</tr>
<tr>
<td>[Shift + Tab]</td>
<td>Jumps to the left to the preceding field within the row.</td>
</tr>
<tr>
<td>[Space]</td>
<td>Opens the editing frame for the selected field. Alternatively you can click with the mouse on the field. If applicable the button for the input assistant dialog box is available.</td>
</tr>
<tr>
<td>[Ctrl] + [Enter]</td>
<td>Inserts a new row below the current row.</td>
</tr>
<tr>
<td>[Del]</td>
<td>Deletes the current row.</td>
</tr>
<tr>
<td>[Ctrl] + [Home]</td>
<td>Sets the focus at the start of the document and marks the first network.</td>
</tr>
<tr>
<td>[Ctrl] + [End]</td>
<td>Sets the focus at the end of the document and marks the last network.</td>
</tr>
<tr>
<td>[Page Down]</td>
<td>Scrolls up by one page and marks the top rectangle.</td>
</tr>
<tr>
<td>[Page Up]</td>
<td>Scrolls down by one page and marks the top rectangle.</td>
</tr>
</tbody>
</table>

See also
- “Instruction list (IL)” on page 183
- Chapter 1.3.1.9.3.1.3 “Programming in instruction list (IL)” on page 187
- Chapter 1.3.1.24.1.5.3 “Modifiers and operators in IL” on page 466
- Chapter 1.3.1.25.3.13.8 “Dialog ‘Options’ - ‘FBD, LD, and IL’” on page 1079
- Chapter 1.3.1.24.1.5.2 “FBD/LD/IL editor in online mode” on page 465

### FBD/LD/IL editor in online mode

In online mode the current value of each variable is displayed behind the variable in the editor. Writing/forcing and the setting of breakpoints is possible.

If the variable is presently forced, this is indicated directly in front of the forced value by $\mathbf{\text{F}}$. If a value has been prepared for writing or forcing, this value is displayed directly behind the current value in square brackets $\langle \text{value}\rangle$.

#### Example

**Forced variable:**

$B\text{Var1} \mathbf{\text{F}} \langle \text{TRUE} \rangle$

**Prepared value**

$i\text{Var1} \langle 0<12> \rangle$

In the online view of a ladder diagram (LD) the connecting lines are marked in color: connections with the value $\text{TRUE}$ are displayed as a thick blue line, connections with the value $\text{FALSE}$ as a thick black line. Conversely, connections with an unknown or analog value are displayed normally (thin black line).

![NOTICE!](image)

Note that values of the connections are calculated from the monitored variables. This is not a genuine flow control.

### Breakpoints
Possible positions for breakpoints are in principle the positions at which values of variables can change (instructions), at which the program branches or at which another box is called.

Possible breakpoint positions:
- On the entire network: causes the breakpoint to be set at the first possible position in the network.
- On a box, if the box contains an assignment. Not possible with operator boxes, for example ADD, DIV.
- On assignments.
- At the end of the box at the position of the return to the calling box. In online mode an empty network automatically appears here; it is marked by 'RET' in place of a network number.

**NOTICE!**
At present you cannot directly set a breakpoint on the first box in the network. However, if you set a breakpoint on the entire network, this breakpoint marking is transferred automatically to the first box in online mode.

**NOTICE!**
Breakpoints in methods: CODESYS automatically sets a breakpoint in all methods that can be called. Therefore, if a method managed by an interface is called, breakpoints are set in all methods that occur in function blocks that implement this interface as well as in all derived function blocks that use the method. If a method is called by a pointer to a function block, CODESYS sets the breakpoints in the method of the function block and in all derived function blocks that use the method.

See also
- § Chapter 1.3.1.12.4 “Forcing and writing variables” on page 315
- § Chapter 1.3.1.12.2 “Using breakpoints” on page 310

**Modifiers and operators in IL**

**Table 17: Modifiers**

<table>
<thead>
<tr>
<th>Modifier</th>
<th>Combined with operator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>C</td>
<td>JMP, CAL, RET</td>
<td>The command is only executed if the result of the preceding expression is TRUE.</td>
</tr>
<tr>
<td>N</td>
<td>JMPC, CALC, RETC</td>
<td>The command is only executed if the result of the preceding expression is FALSE.</td>
</tr>
<tr>
<td>N</td>
<td>otherwise</td>
<td>negation of the operand (not of the accumulator).</td>
</tr>
</tbody>
</table>

**Table 18: Operators with the possible modifiers**

<table>
<thead>
<tr>
<th>Operator</th>
<th>N</th>
<th>Meaning</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>LD</td>
<td>N</td>
<td>Loads the (negated) the value of the operand into the accumulator.</td>
<td>LD iVar</td>
</tr>
<tr>
<td>ST</td>
<td>N</td>
<td>Stores the (negated) content of the accumulator in the operand.</td>
<td>ST iErg</td>
</tr>
<tr>
<td>S</td>
<td></td>
<td>Sets the operand (type BOOL) to TRUE if the content of the accumulator is TRUE.</td>
<td>S bVar1</td>
</tr>
<tr>
<td>Operator</td>
<td>N</td>
<td>Meaning</td>
<td>Example</td>
</tr>
<tr>
<td>----------</td>
<td>---</td>
<td>---------</td>
<td>---------</td>
</tr>
<tr>
<td>R</td>
<td></td>
<td>Sets the operand (type BOOL) to FALSE if the content of the accumulator is TRUE.</td>
<td>R bVar1</td>
</tr>
<tr>
<td>AND</td>
<td>N, (</td>
<td>Bitwise AND of the accumulator value and (negated) operand</td>
<td>AND bVar2</td>
</tr>
<tr>
<td>OR</td>
<td>N, (</td>
<td>Bitwise OR of the accumulator value and (negated) operand</td>
<td>OR xVar</td>
</tr>
<tr>
<td>XOR</td>
<td>N, (</td>
<td>Bitwise exclusive OR of the accumulator value and (negated) operand</td>
<td>XOR N, (bVar1,bVar2)</td>
</tr>
<tr>
<td>NOT</td>
<td></td>
<td>Bitwise negation of the accumulator value</td>
<td></td>
</tr>
<tr>
<td>ADD</td>
<td></td>
<td>Addition of the accumulator value and the operand; result is written into the accumulator.</td>
<td>ADD iVar1</td>
</tr>
<tr>
<td>SUB</td>
<td></td>
<td>Subtraction of the operand from the accumulator value; result is written into the accumulator.</td>
<td>SUB iVar2</td>
</tr>
<tr>
<td>MUL</td>
<td></td>
<td>Multiplication of accumulator value and operand; result is written into the accumulator.</td>
<td>MUL iVar2</td>
</tr>
<tr>
<td>DIV</td>
<td></td>
<td>Division of the accumulator value by the operand; result is written into the accumulator.</td>
<td>DIV 44</td>
</tr>
<tr>
<td>GT</td>
<td></td>
<td>Checks whether the accumulator value is greater than the operand value; result (BOOL) is written into the accumulator; &gt;</td>
<td>GT 23</td>
</tr>
<tr>
<td>GE</td>
<td></td>
<td>Checks whether the accumulator value is greater than or equal to the operand value; result (BOOL) is written into the accumulator.</td>
<td>GE iVar2</td>
</tr>
<tr>
<td>EQ</td>
<td></td>
<td>Checks whether the accumulator value is equal to the operand value; result (BOOL) is written into the accumulator.</td>
<td>EQ iVar2</td>
</tr>
<tr>
<td>NE</td>
<td></td>
<td>Checks whether the accumulator value is not equal to the operand value; result (BOOL) is written into the accumulator;</td>
<td>NE iVar1</td>
</tr>
<tr>
<td>LE</td>
<td></td>
<td>Checks whether the accumulator value is smaller than or equal to the operand value; result (BOOL) is written into the accumulator.</td>
<td>LE 5</td>
</tr>
<tr>
<td>LT</td>
<td></td>
<td>Checks whether the accumulator value is smaller than the operand value; result (BOOL) is written into the accumulator.</td>
<td>LT cVar1</td>
</tr>
<tr>
<td>JMP</td>
<td>CN</td>
<td>Unconditional (conditional) jump to the specified jump label</td>
<td>JMPN next</td>
</tr>
<tr>
<td>CAL</td>
<td>CN</td>
<td>(Conditional) call of a program or a function block (if the accumulator value is TRUE)</td>
<td>CAL progl</td>
</tr>
<tr>
<td>RET</td>
<td></td>
<td>Exit the box and return to the calling box.</td>
<td>RET</td>
</tr>
<tr>
<td>RET</td>
<td>C</td>
<td>If the accumulator value is TRUE: exit the box and return to the calling box.</td>
<td>RETC</td>
</tr>
<tr>
<td>RET</td>
<td>CN</td>
<td>If the accumulator value is FALSE: exit the box and return to the calling box.</td>
<td>RETCN</td>
</tr>
<tr>
<td>)</td>
<td></td>
<td>Evaluation of the reset operation</td>
<td></td>
</tr>
</tbody>
</table>
### Example

<table>
<thead>
<tr>
<th></th>
<th>AND</th>
<th>TRUE</th>
<th>( \text{load TRUE to accumulator} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>AND</td>
<td>bVar1</td>
<td>( \text{execute AND with negated value of bVar1} )</td>
</tr>
<tr>
<td></td>
<td>JMP</td>
<td>b1</td>
<td>( \text{if accu. is TRUE, jump to label &quot;b1&quot;} )</td>
</tr>
<tr>
<td>3</td>
<td>LD</td>
<td>bVar2</td>
<td>( \text{store negated value of bVar2...} )</td>
</tr>
<tr>
<td></td>
<td>ST</td>
<td>bRes</td>
<td>( \text{... in bRes} )</td>
</tr>
<tr>
<td></td>
<td>b1</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>LD</td>
<td>bVar2</td>
<td>( \text{store value of bVar2...} )</td>
</tr>
<tr>
<td></td>
<td>ST</td>
<td>bRes</td>
<td>( \text{... in bRes} )</td>
</tr>
</tbody>
</table>

### Application

<table>
<thead>
<tr>
<th>Several operands for 1 operator</th>
</tr>
</thead>
<tbody>
<tr>
<td>Options</td>
</tr>
<tr>
<td>- You enter the operands into consecutive rows, separated by commas in the 2nd column.</td>
</tr>
<tr>
<td>- You repeat the operator in consecutive rows.</td>
</tr>
</tbody>
</table>

### Description

- **Variant 1:**
  - LD 2
  - ADD 3, 4, 6
  - ST 1VAR

- **Variant 2:**
  - LD 2
  - ADD 3
  - ADD 4
  - ADD 6
  - ST 1VAR

### Examples

<table>
<thead>
<tr>
<th>Complex operands</th>
</tr>
</thead>
<tbody>
<tr>
<td>For a complex operand, you enter the opening parenthesis ( in the first column. You enter the closing parenthesis in the first column in a separate row following the operand entries of the following rows.</td>
</tr>
<tr>
<td>A string is rotated by a character each cycle:</td>
</tr>
<tr>
<td>LD stRotate</td>
</tr>
<tr>
<td>RIGHT (stRotate</td>
</tr>
<tr>
<td>LEN 1</td>
</tr>
<tr>
<td>)</td>
</tr>
<tr>
<td>CONCAT (stRotate</td>
</tr>
<tr>
<td>LEFT 1</td>
</tr>
<tr>
<td>)</td>
</tr>
<tr>
<td>ST stRotate</td>
</tr>
<tr>
<td>Application</td>
</tr>
<tr>
<td>-----------------------------</td>
</tr>
</tbody>
</table>
| Function block call, program call | Column 1: Operator `CAL` or `CALC`  
Column 2: Name of the function block instance or the program and opening parenthesis ( . If no parameters follow, the closing parenthesis ) is entered here.  
Rows following that:  
Column 1: parameter name followed by `:=` for input parameter or `=>` for output parameter  
Column 2: parameter value followed by a comma , if further parameters follow. The closing parenthesis ) is input after the last parameter.  
As a limitation according to the IEC standard, complex expressions cannot be used here. You must assign such constructs to the function block or the program before the call. | ![Example](image) |
| Function Call               | Row 1: Column 1: `LD`  
Column 2: input variable  
Row 2: Column 1: Function name Column 2: further input parameters separated by commas.  
CODESYS writes the return value into the accumulator.  
Row 3: Column 1: `ST` Column 2: variable into which the return value is written | ![Example](image) |
| Action call                 | Like function block call or program call.  
The action name is appended to the name of the FB instance or the program.                                                                                                                                     | ![Example](image) |
### Jump

<table>
<thead>
<tr>
<th>Application</th>
<th>Description</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jump</td>
<td>Column 1: operator JMP or JMPC. Column 2: Name of the jump label of the destination network. In the case of an unconditional jump, the preceding instruction sequence must end with one of the following commands: ST, STN, S, R, CAL, RET, JMP. In the case of a conditional jump the execution of the jump depends on the loaded value.</td>
<td><strong>LD</strong> BVar1</td>
</tr>
</tbody>
</table>

See also
- % Chapter 1.3.1.24.1.5.1 “FBD/LD/IL editor” on page 461
- % “Instruction list (IL)” on page 183
- % Chapter 1.3.1.9.3.1.3 “Programming in instruction list (IL)” on page 187

## Elements

| 1.3.1.24.1.5.4.1 | FBD/LD/IL element 'Network' | 470 |
| 1.3.1.24.1.5.4.2 | FBD/LD/IL element 'Box' | 471 |
| 1.3.1.24.1.5.4.3 | FBD/LD/IL element 'Assignment' | 471 |
| 1.3.1.24.1.5.4.4 | FBD/LD/IL element 'Box with EN/ENO' | 471 |
| 1.3.1.24.1.5.4.5 | FBD/LD/IL element 'Input' | 472 |
| 1.3.1.24.1.5.4.6 | FBD/LD/IL element 'Label' | 472 |
| 1.3.1.24.1.5.4.7 | FBD/LD/IL element 'Jump' | 472 |
| 1.3.1.24.1.5.4.8 | FBD/LD/IL element 'Return' | 472 |
| 1.3.1.24.1.5.4.9 | FBD/LD/IL element 'Branch' | 472 |
| 1.3.1.24.1.5.4.10 | FBD/LD/IL element 'Execute' | 473 |
| 1.3.1.24.1.5.4.11 | LD element 'Contact' | 473 |
| 1.3.1.24.1.5.4.12 | LD element 'Coil' | 474 |
| 1.3.1.24.1.5.4.13 | LD element 'Branch Start/End' | 474 |
| 1.3.1.24.1.5.4.14 | Closed branch | 475 |

## FBD/LD/IL element 'Network'

**Symbol: **

A network is the base unit of an FBD or LD program. In the FBD/LD/IL editor, the networks are arranged in a list. Each network is provided with a sequential network number on the left side and can include: logical and arithmetic expressions, program/function/function block calls, jumps, or return statements.

An IL program consists of at least one network. This network can include all IL statements of the program.

You can provide each network with a title, comment, or label. In the CODESYS options (category "FBD, LD, and IL", you can define whether network title, comment, and separator between individual networks are displayed in the editor.

Click the first line of the network to enter a network title. Click the second line of the network to enter a network comment.
FBD/LD/IL element 'Box'

Symbol: ☐

A box and its call can represent additional functions, for example IEC function blocks, IEC functions, library function blocks, operators.

A box can have any number of inputs and outputs.

If the box also provides an image file, the box icon is displayed inside the box. The requirement is that the option “Show box symbol” is activated in the CODESYS options, category "FBD, LD and IL".

If you have changed the box interfaces, you can update the box parameters with the command “FBD/LD/IL Update Parameters” without having to re-insert the box.

See also

- Chapter 1.3.1.25.2.13.38 “Command ‘Update Parameters’” on page 1002

FBD/LD/IL element 'Assignment'

Symbol: ☐

The FBD editor shows a newly inserted assignment as a line with 3 question marks after it. The LD editor shows a newly inserted assignment as a coil with 3 question marks located above it.

After insertion you can replace the placeholder ??? by the name of the variable to which the signal coming from the left is to be assigned. The input assistant is available to you for this.

In IL an assignment is programmed via the operators LD and ST.

FBD/LD/IL element 'Box with EN/ENO'

Symbol: ☐

The element is available only in the FBD and LD editors.

The box generally corresponds to the FBD/LD/IL element “Box”; however, this box additionally contains an EN input and an ENO output. EN and ENO have the data type BOOL.

Function of the EN input and ENO output: if the input EN has the value FALSE at the time of the calling the box, the operations defined in the box are not executed. Otherwise, i.e. if EN has the value TRUE, these operations are executed. The ENO output has the same value as the EN input.

See also

- Chapter 1.3.1.25.2.13.6 “Command ‘Insert Box with EN/ENO’” on page 994
- Chapter 1.3.1.24.1.5.4.2 “FBD/LD/IL element ‘Box’” on page 471
FBD/LD/IL element 'Input'

Symbol: "44"

The maximum number of inputs depends on the type of box.

A newly added input is first marked with "??". You can replace the string "??" by a variable or a constant.

See also
- Chapter 1.3.1.25.2.13.13 “Command 'Insert Input’' on page 995
- Chapter 1.3.1.24.1.5.1 “FBD/LD/IL editor” on page 461

FBD/LD/IL element 'Label'

The label is an optional identifier for a network in FBD and LD, which you can specify as a destination for a jump.

If you insert a jump label in a network, it will be added as an editable field “Label:” in the network.

See also
- Chapter 1.3.1.24.1.5.1 “FBD/LD/IL editor” on page 461
- Chapter 1.3.1.25.2.13.11 “Command 'Insert Label’' on page 995

FBD/LD/IL element 'Jump'

Symbol →

In FBD or LD a jump is inserted either directly before an input, directly after an output or at the end of the network, depending on the current cursor position.

You enter a jump label as the jump destination directly behind the jump element.

In IL you program a jump with the instruction JMP.

See also
- Chapter 1.3.1.24.1.5.1 “FBD/LD/IL editor” on page 461
- Chapter 1.3.1.25.2.13.10 “Command 'Insert Jump’' on page 995
- Chapter 1.3.1.24.1.5.4.6 “FBD/LD/IL element 'Label’' on page 472

FBD/LD/IL element 'Return'

This element immediately interrupts the execution of the box if the input of the RETURN element goes TRUE.

In an FBD or LD network you can place the Return instruction parallel to or after the preceding elements.

In IL the RET instruction is available to you for this purpose.

See also
- Chapter 1.3.1.25.2.13.12 “Command 'Insert Return’' on page 995
- Chapter 1.3.1.24.1.5.3 “Modifiers and operators in IL” on page 466

FBD/LD/IL element 'Branch'

Symbol: 〒
The element is available in the LD and FBD editor and represents an open line branch. A line branch splits the processing line from the current cursor position onwards into 2 subnetworks, which are executed in succession from top to bottom. You can branch each subnetwork further, as a result of which multiple branches are created within a network.

Each subnetwork is given a marker symbol (rectangle) at the branch point, which you can select in order to execute further commands.

- The commands “Copy”, “Cut” and “Paste” are not available for subnetworks.

In order to delete a subnetwork, you must first delete all elements of the network and then the marker symbol of the subnetwork.

See also
- Chapter 1.3.1.25.2.13.33 “Command 'Insert Branch’” on page 1001
- Chapter 1.3.1.25.2.13.34 “Command 'Insert Branch Above’” on page 1001
- Chapter 1.3.1.25.2.13.35 “Command 'Insert Branch Below’” on page 1001

**FBD/LD/IL element 'Execute'**

Symbol: 

The element is a box that enables you to directly enter ST code in the FBD and LD editors. You can drag the “Execute” element with the mouse from the “Tools” view into the implementation part of your POU. If you click on “Enter ST code here...”, an input field opens where you can input multiple-line ST code.

- Chapter 1.3.1.24.1.5.1 “FBD/LD/IL editor” on page 461

**LD element 'Contact'**

Symbol: 

The element is available only in the LD editor.
A contact passes on the signal TRUE (ON) or FALSE (OFF) from left to right until the signal finally reaches a coil in the right-hand part of the network. For this purpose a boolean variable containing the signal is assigned to the contact. To do this, replace the placeholder ??? above the contact with the name of a boolean variable.

You can arrange several contacts both in series and in parallel. In the case of two parallel contacts, only one needs to obtain the value TRUE in order for ON to be passed on to the right. If contacts are connected in series, all of them must obtain the value TRUE in order for ON to be passed on to the right by the last contact in the series. Hence, you can program electrical parallel and series connections with LD.

A negated contact forwards the signal TRUE if the variable value is FALSE. You can negate an inserted contact with the help of the command “FBD/LD/IL → Negation” or insert a negated contact from the “Tools” view.

If you place the mouse pointer on a contact with the left mouse button pressed and with a network selected, the button “Convert to coil” appears in the network. If you now move the mouse pointer onto this button, still with the mouse button pressed, and then release the mouse button over this button, CODESYS converts the contact into a coil.

See also

- Chapter 1.3.1.25.2.13.17 “Command 'Insert Contact’” on page 996
- Chapter 1.3.1.25.2.13.22 “Command ‘Insert Negated Contact’” on page 998
- Chapter 1.3.1.25.2.13.18 “Command ‘Insert Contact (Right)’” on page 997
- Chapter 1.3.1.25.2.13.20 “Command ‘Insert Contact in Parallel (Above)’” on page 997
- Chapter 1.3.1.25.2.13.19 “Command ‘Insert Contact in Parallel (Below)’” on page 997
- Chapter 1.3.1.24.1.5.1 “FBD/LD/IL editor” on page 461

LD element ‘Coil’

Symbol: in the editor

The element is available only in the LD editor.

A coil adopts the value supplied from the left and saves it in the boolean variable assigned to the coil. Its input can have the value TRUE (ON) or FALSE (OFF).

Several coils in a network can only be arranged in parallel.

In a negated coil the negated value of the incoming signal is stored in the boolean variable that is assigned to the coil.

Set coil, Reset coil

Symbol: , in the editor

Set coil: If the value TRUE arrives at a set coil, the coil retains the value TRUE. As long as the application is running, the value can no longer be overwritten here.

Reset coil: If the value TRUE arrives at a reset coil, the coil retains the value FALSE. As long as the application is running, the value can no longer be overwritten here.

You can define an inserted coil as a set or reset coil with the help of the command “FBD/LD/IL → Set/Reset” or insert it as an element “Set Coil” and “Reset Coil” from the “Tools” view.

See also

- Chapter 1.3.1.25.2.13.14 “Command ‘Insert Coil’” on page 996
- Chapter 1.3.1.25.2.13.16 “Command ‘Insert Reset Coil’” on page 996
- Chapter 1.3.1.25.2.13.29 “Command ‘Negation’” on page 1000
- Chapter 1.3.1.25.2.13.31 “Command ‘Set/Reset’” on page 1000

LD element ‘Branch Start/End’

Symbol: →
The element serves the closed line branch.

See also

- Chapter 1.3.1.24.1.5.4.14 “Closed branch” on page 475
- Chapter 1.3.1.24.1.5.1 “FBD/LD/IL editor” on page 461
- Chapter 1.3.1.25.2.13.36 “Command 'Set Branch Start Point'” on page 1001
- Chapter 1.3.1.25.2.13.37 “Command 'Set Branch End Point'” on page 1002

Closed branch

A closed branch is available in LD only, and it contains a starting point and an end point. It is used for implementing parallel analyses of logical elements.

Inserting a closed branch

- Command “FBD/LD/IL ➔ Insert Contact Parallel (Below) ”
- Command “FBD/LD/IL ➔ Insert Contact Parallel (Above) ”
- Command “FBD/LD/IL ➔ Set Branch Start/End Point”

Closed branch at a contact

When you select one or more contacts and then execute the command “Insert Contact in Parallel”, a parallel branch is added with a single vertical line. For this kind of branching, the signal flow passes through both branches. This is an OR construct of both branches.

Closed branch at a block, OR evaluation, or short-circuit evaluation

New: When you select a box and execute the command “Insert Contact in Parallel”, a parallel branch is inserted with a double vertical line. This indicates that a short-circuit evaluation (SCE) is implemented. SCE allows for the execution of a function block with a Boolean output to be bypassed if a specific condition is TRUE. The condition can be displayed in the LD editor as a branch connected parallel to the function block branch. The short circuit condition is defined by one or more contacts in this branch that are interconnected parallel or sequentially.

Functional principle:

The branches that do not include the function block are processed first. If CODESYS detects the value TRUE for one of these branches, then the function block is not called in the parallel branch. In this case, the value at the input of the function block is sent directly to the output. If CODESYS determines FALSE for the SCE condition, then the box will be called and the Boolean result of its processing is passed on. If all branches contain function blocks, they are analyzed from top to bottom and their outputs are logically ORed. If there are no branches with function blocks, normal OR operations are performed.
The function block instance \( x_1 \) (TON) has a Boolean input and a Boolean output. The execution of \( x_1 \) is skipped if TRUE is determined for the condition in the parallel line branch. The condition value results from the OR and AND operations that connect contacts \( \text{cond1}, \text{cond2} \) and \( \text{cond3} \).

\( x_1 \) is executed if the condition value from the connection of the contacts \( \text{cond1}, \text{cond2} \) and \( \text{cond3} \) is FALSE.

\[ \begin{align*}
\text{P_IN} & := b_1 \text{ AND } b_2; \\
\text{IF} \ ((\text{P_IN AND cond1}) \text{ AND } (\text{cond2 OR cond3})) \text{ THEN} & \quad \text{P_OUT} := \text{P_IN}; \\
\text{ELSE} & \quad \text{x1(IN := P_IN, PT := {p 10}t\#2s);} \\
& \quad \text{tElapsed := x1.ET;} \\
& \quad \text{P_OUT := x1.Q;} \\
\text{END_IF} & \\
\text{bRes} & := \text{P_OUT AND b3;}
\end{align*} \]

See also
- Chapter 1.3.1.24.1.5.1 “FBD/LD/IL editor” on page 461
- Chapter 1.3.1.25.2.13.36 “Command 'Set Branch Start Point'” on page 1001
- Chapter 1.3.1.25.2.13.37 “Command 'Set Branch End Point'” on page 1002
- Chapter 1.3.1.25.2.13.20 “Command 'Insert Contact in Parallel (Above)’” on page 997
- Chapter 1.3.1.25.2.13.19 “Command 'Insert Contact in Parallel (Below)’” on page 997
- Chapter 1.3.1.25.2.13.21 “Command 'Toggle Parallel Mode’” on page 998
From an external point of view, a function block diagram consists of inputs and outputs, with data being processed between them. From an internal point of view, a function block diagram consists of POUs and their connections which represent data (signals) and act as assignment operators in ST. The overall behavior is composed of the behavior of the inserted POUs which call other POUs or library POUs.

Code in the "Continuous Function Chart (CFC)" implementation language mainly illustrates the data flow through the system. Therefore, a continuous function chart is also referred to as a "signal flow chart".

In the page-oriented CFC editor, you can wire POUs to each other and create well-structured function block diagrams distributed over multiple pages. The page-oriented editor behaves like the CFC editor, but provides the following functionality:

The page-oriented editor behaves like the CFC editor, but provides additional functionality. The editor supports you with the following functions:

- Creating pages
- Setting the page size
- Copying and inserting pages in the page navigator
- Copying the implementation of a POU in the CFC implementation language and inserting into a page
- Well-structured and space-saving arranging of inputs, outputs, and connection marks in the border areas
- Connection over pages with connection marks

CFC editor

Configuring the editor

You can configure the appearance, behavior, and printing for the entire project in the CODESYS options in the “CFC Editor” category. For example, on the “View” tab, you can configure the color of the connecting lines depending on the data type.

Editing

| Cursor symbol: ✈️  | Requirement: "Pointer" is selected in the “ToolBox” view. The symbol indicates that you can edit in the editor. Select elements or connections to move them or to execute commands. |
| Cursor symbol: ➔  | Requirement: An element is selected in the “ToolBox” view. Clicking in the editor inserts the selected element. You can also drag an element to the editor. |
| Dragging a function block instance from the declaration in the editor | Requirement: A line is selected in the declaration of the CFC. The instance is inserted as a POU with name, type, and all pins. |
| Dragging a variable from the declaration to a POU pin in the editor | The variable is inserted as an input or output with a connection to the POU pin in focus. Hint: The cursor indicates when your focused location is valid for a variable: |
**[Ctrl] + click in the programming area**

Requirement: An element is selected in the “ToolBox” view.
As long as you hold down the [Ctrl] key, a selected element is created each time you click in the programming area.

**[Ctrl]+[Right Arrow]**

Requirement: In the CFC program, **exactly** one output pin is selected for an element.
The selection is moved so that the input pin at the end of the connecting line is selected. In the case of multiple pins, all are selected.

![Diagram](image1.png)

**[Ctrl]+[Left Arrow]**

Requirement: In the CFC program, **exactly** one input pin is selected for an element.
The selection is moved so that the output pin at the beginning of the connecting line is selected. In the case of multiple pins, all are selected.

Example:

![Diagram](image2.png)

See also

- Chapter 1.3.1.24.1.2 “Common functions in graphical editors” on page 429

## Connecting

You can insert connecting lines between element connections. Connecting lines are inserted by means of auto-routing so that connecting lines are automatically optimal and as short as possible. The connecting lines are checked for collisions.

<table>
<thead>
<tr>
<th>Dragging a pin to another</th>
<th>A connecting line is inserted between the two element pins.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Command “Connect Selected Pins”</td>
<td>Requirement: Multiple pins are selected. The pins are marked in red.</td>
</tr>
</tbody>
</table>
Move an inserted element so that it touches the pin of another element.

Requirement: The “Enable AutoConnect” option is selected. The touching pins are connected automatically.

The connection icon is located in the upper right corner of the editor. A green icon indicates collision-free connections. A red icon indicates collisions. Clicking the icon opens a menu with commands for collision processing, for example the “Show Next Collision” command.

Requirement: A connection is selected and the “Connection Mark” command is executed. Instead of a long connecting line, a connection is represented by connection marks.

See also
- "Chapter 1.3.1.25.2.12.22 “Command ‘Show Next Collision’” on page 986"

Commands when editing
See also
- "Chapter 1.3.1.25.2.12 “Menu ‘CFC’” on page 977"

See also
- "Chapter 1.3.1.24.1.2 “Common functions in graphical editors” on page 429"
- "Chapter 1.3.1.9.3.2.2 “Programming in the CFC editor” on page 193"
- "Chapter 1.3.1.9.3.2.1 “Automatic execution order by data flow” on page 189"
- "Chapter 1.3.1.25.2.12 “Menu ‘CFC’” on page 977"
- "Chapter 1.3.1.25.3.10.12 “Dialog ‘Properties’ - ‘CFC Execution Order’” on page 1054"

CFC editor, page-oriented

POUs generated in the “Continuous Function Chart (CFC) - page-oriented” cannot be converted into “Continuous Function Chart (CFC)” POUs or back.
Editing

You can drag a “Page” element from the “ToolBox” view to the page navigation. Then an additional page is inserted.

You can select existing pages in the page navigation and duplicated them by clicking “Edit ➔ Copy” and “Edit ➔ Paste”.

The size of the page is changed by means of the “Edit Page Size” command.

Connections over multiple pages are established by means of the “Connection Mark - Source” and “Connection Mark - Sink” elements. When you drag a connecting line from an input pin or an output pin to the border area, a new connection mark is created automatically. The advantage is that the "List components" input assistance provides all previously defined connection mark sources.

If you have selected an element in the editor, then you can use the arrow keys to move the selection from one element to the next to navigate through the circuit. If you then select a connection mark and press another arrow key, even the corresponding connection mark of the next/previous page will be selected.

You can transfer networks from a CFC POU to the program area of a page-oriented CFC by clicking “Edit ➔ Copy” and “Edit ➔ Paste” (from the clipboard). You can also use drag&drop.

Execution order

The execution order is determined automatically according to the order of the pages as they are sorted in the page navigator of the editor. Within a page, a page-oriented CFC object behaves like a CFC object. Therefore, you can switch between “Auto Data Flow Mode” and “Explicit Execution Order Mode”.

Additional commands in “CFC page-oriented”

See also

- Chapter 1.3.1.25.2.12.2 “Command ‘Edit Page Size’” on page 978
- Chapter 1.3.1.25.2.12.1 “Command ‘Edit Worksheet’” on page 977

See also

- Chapter 1.3.1.24.1.2 “Common functions in graphical editors” on page 429
- Chapter 1.3.1.9.3.2.2 “Programming in the CFC editor” on page 193
- Chapter 1.3.1.9.3.2.1 “Automatic execution order by data flow” on page 189
- Chapter 1.3.1.25.2.12 “Menu ‘CFC’” on page 977
- Chapter 1.3.1.25.3.10.12 “Dialog ‘Properties’ - ‘CFC Execution Order’” on page 1054

CFC editor in online mode

In online mode, you can monitor and change variable values of the controller. In addition, debugging features are provided such as breakpoints and stepping.

Monitoring

As usual, you can monitor values in the declaration part as well as in the implementation part (with inline monitoring).

Inline monitoring of a function block is possible only when an instance of the function block is open. No values are displayed in the basic implementation view.
The connections between Boolean variables are displayed in color according to their actual value: **TRUE** in blue and **FALSE** in black. The element pins are decorated with the actual value.

**Example**

An application contains a CFC POU. An internal Boolean variable is switched there. With each cycle, the variable `iToggle` switches its state from **TRUE** to **FALSE**.

In the case of scalar variables, the element pins are decorated with the actual values.
Example

In online mode in the declaration editor, you can prepare a value for forcing or writing a monitored variable.

When you select the “Prepare values in implementation part” check box in the “CFC Editor” category of the CODESYS options, you can also prepare values in the implementation part.

To do this, open the “Prepare Value” dialog by double-clicking an element or the monitoring box next to an element. No dialog appears for Boolean variables. However, with each mouse click on the value displayed next to the variable, the values TRUE and FALSE are toggled.

Prepared values are displayed in angle brackets. After executing a write or a force, a red "F" is shown in the monitoring box.

You can write input parameters of function block instances of type VAR_INPUT CONSTANT in online mode and modify the parameters in this way. After logging out, you save these parameters by clicking “Save Prepared Parameters to Project”.

Forcing and writing of variables

Changing of constant input parameters of function block instances
Requirement: A CFC editor is active. An instantiated function block has VAR_INPUT CONSTANT variables in its declaration.

1. In the editor, open the POU by calling the function block instance.
   ⇢ The declaration of FB_DoIt has been supplemented by the constant MAXIMUM.

   ```
   FUNCTION_BLOCK FB_DoIt
   VAR_INPUT
     iAlfa : INT;
     iBravo: INT;
     sCharlie : STRING := 'Charlie';
     xItem : BOOL;
     iDelta : INT;
   END_VAR
   VAR_INPUT CONSTANT
     MAXIMUM : INT := 12;
   END_VAR
   VAR_OUTPUT
     iResult : INT;
     sResult : STRING;
     xResult : BOOL;
   ```

2. Login to the controller.
3. Click the "Parameters" button of the function block instance.
   ⇢ The "Edit Parameters" dialog opens.
4. Click the "Value" column in an inline monitoring field of a parameter.
   ⇢ The "Prepare Value" dialog opens.
5. Type 20 in the "Prepare a new value for the next write or force operation" field.
6. Click "OK" to confirm the entry.
   ⇢ The prepared value is shown in angle brackets next to the current value (for example, <20>).
7. Click “Debug ➔ Write Values”.
   ⇒ The prepared value is written. The parameter is changed and displayed in the project in brackets after the value.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Value</th>
<th>Initial value</th>
<th>Min</th>
<th>Max</th>
<th>Unit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>⋯</td>
<td>MAXIMUM</td>
<td>INT 20</td>
<td>20</td>
<td>12</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The difference between both values is shown by a red cross next to the parameter field of the function block instance.

8. Click “Edit Parameters” to close the dialog. Logout.
9. Click “CFC ➔ Save Prepared Parameters to Project”.
   ⇒ The change parameter values are saved to the project. The asterisk next to the parameter field disappears.

**Breakpoint locations**

Possible position of a breakpoint
- Element “Output”
  Variables are described.
- Element “Box”
  POU is called.
- Element “RETURN”
  The program flow is branched.
- Element “Selector”
  Structure elements are described.

Click “Debug ➔ Toggle Breakpoint” to set a new breakpoint or delete an existing breakpoint. A red circle in the block diagram represents an active breakpoint.
NOTICE!

A breakpoint is set automatically in all methods that can be called. Therefore, if a method is called that is defined over an interface, then breakpoints are set in all methods of function blocks that implement this interface. This also applies to all inherited function blocks that define methods.

You can process a POU in steps in debug mode. A called POU is supplemented internally by a `RETURN` at the beginning before the element with the number 0 and at the end after the last element. When stepping, these are started automatically.

Commands in online mode

See also

- Chapter 1.3.1.25.2.12.35 “Command ‘Force FB Input’” on page 989
- Chapter 1.3.1.25.2.12.34 “Command ‘Prepare Function Block Element for Forcing’” on page 989
- Chapter 1.3.1.25.2.12.18 “Command ‘Edit Parameters’” on page 984
- Chapter 1.3.1.25.2.12.19 “Command ‘Save Prepared Parameters to Project’” on page 985

See also

- Chapter 1.3.1.13.1 “Monitoring of values” on page 323
- Chapter 1.3.1.12.4 “Forcing and writing variables” on page 315
- “Forcing a function block input in CFC” on page 317
- Chapter 1.3.1.12.2 “Using breakpoints” on page 310
- Chapter 1.3.1.12.3 “Stepping through a program” on page 313
Elements

1.3.1.24.1.6.4.1  CFC element 'Page'................................................. 486
1.3.1.24.1.6.4.2  CFC element 'Control Point'.................................... 486
1.3.1.24.1.6.4.3  CFC element 'Input'.............................................. 486
1.3.1.24.1.6.4.4  CFC element 'Output'............................................. 486
1.3.1.24.1.6.4.5  CFC element 'Box'.................................................. 487
1.3.1.24.1.6.4.6  CFC element 'Jump'................................................ 487
1.3.1.24.1.6.4.7  CFC element 'Label'.............................................. 487
1.3.1.24.1.6.4.8  CFC element 'Return'............................................. 487
1.3.1.24.1.6.4.9  CFC element 'Composer'....................................... 487
1.3.1.24.1.6.4.10 CFC element 'Selector'......................................... 488
1.3.1.24.1.6.4.11 CFC element 'Comment'....................................... 488
1.3.1.24.1.6.4.12 CFC element 'Connection Mark - Source/Sink'........... 488
1.3.1.24.1.6.4.13 CFC element 'Input Pin'........................................ 489
1.3.1.24.1.6.4.14 CFC element 'Output Pin'..................................... 489

CFC element 'Page'

Symbol: □

The element inserts a new page into the editor. It is available only in the page-oriented CFC editor. The number of the page is automatically assigned in accordance with its position. You can enter the name and the description of the page into the orange header. The page size is adapted with the “Edit Page Size” command.

See also

- ☐ Chapter 1.3.1.25.2.12.2 “Command ‘Edit Page Size’” on page 978

CFC element 'Control Point'

Symbol: ↕

Use a control point in order to fix points of a connection before you adapt the line routing. To do this, drag the element to the desired position on a connecting line. Connecting lines with control points are no longer routed automatically.

See also

- ☐ Chapter 1.3.1.9.3.2.2 “Programming in the CFC editor” on page 193
- ☐ Chapter 1.3.1.25.2.12.30 “Command ‘Create Control Point’” on page 988
- ☐ Chapter 1.3.1.25.2.12.29 “Command ‘Remove Control Point’” on page 988

CFC element 'Input'

Symbol: ➞

CODESYS inserts an input element by default with the text “???” . You can directly edit this field by clicking on it and entering a constant value or a variable name. Alternatively you can open the input assistant in order to select a variable by clicking on ▼▼.

CFC element 'Output'

Symbol: ⇥

CODESYS inserts an output element by default with the text “???” . You can directly edit this field by clicking on it and entering a constant value or a variable name. Alternatively you can open the input assistant in order to select a variable by clicking on ▼▼.
CFC element 'Box'

Symbol: 📦

You use the element in order to insert an operator, a function, a function block or a program. By default CODESYS inserts the element with the name "???". You can directly edit this field by clicking on it and entering a function block name. Alternatively you can open the input assistant and select a function block by clicking on it.

In the case of a function block, CODESYS additionally displays an input field ("???") above the function block symbol. You must replace this name by the name of the function block instance. If you instance a function block with constant input parameters, the function block element displays the 'Parameter...' field in the bottom left corner. You edit the parameters by clicking on this field.

In order to replace an existing box, you replace only the currently inserted identifier with the desired new name. When doing this, please note that CODESYS adapts the number of input and output pins in accordance with the definition of the POU and that existing assignments may therefore be deleted.

See also
● ✔ Chapter 1.3.1.25.2.12.18 “Command ‘Edit Parameters’” on page 984

CFC element 'Jump'

Symbol: ⇔

You use the element in order to define a position at which program execution is to continue. You must define this target position by a label. To do this, enter the name of the mark in the input field "???". If you have already inserted the corresponding label, you can also select it via the input assistant (⋯).

See also
● ✔ Chapter 1.3.1.24.1.6.4.7 “CFC element 'Label'” on page 487

CFC element 'Label'

Symbol: ⬤

A label defines a position to which program execution jumps with the help of a jump element.

In online mode CODESYS automatically inserts a RETURN flag at the end of a CFC function block.

See also
● ✔ Chapter 1.3.1.24.1.6.4.6 “CFC element 'Jump'” on page 487

CFC element 'Return'

Symbol: ⇩

Use the element in order to exit the function block.

Please note that in online mode in the CFC editor a return element is automatically inserted before the first line and after the last element. In single-step execution CODESYS automatically jumps to the return element at the end before exiting the function block.

CFC element 'Composer'
The composer element is for handling structural components. The individual components of a structure are made available to you as an input. For this purpose you must name the composer element like the structure concerned (replace the “???”).

The composer element is the counterpart to the selector element.

See also

- § Chapter 1.3.1.24.1.6.4.10 “CFC element 'Selector'” on page 488

CFC element 'Selector'

Symbol: 

The selector element is for handling structural components. The individual components of a structure are made available to you as an output. For this purpose you must name the selector element like the structure concerned (replace the “???”).

The selector element is the counterpart to the composer element.

See also

- § Chapter 1.3.1.24.1.6.4.9 “CFC element 'Composer'” on page 487

CFC element 'Comment'

Symbol: 

With this element you input a comment in the CFC editor. Replace the placeholder text in the element by the comment text. A line break can be inserted with the aid of the shortcut \[Ctrl\] + [Enter].

CFC element 'Connection Mark - Source/Sink'

Symbol: ↔, ↔

You can use connection marks instead of a connecting line between elements. That helps you to display complex diagrams more clearly.

For a valid connection you must connect an element “Connection Mark - Source” with the output of an element and an element “Connection Mark - Sink” with the input of another element. Both marks must bear the same name. The names are not case-sensitive.

Notes on naming

- The standard name for connection marks is C-<nr>. <nr> is a sequential number starting with 1.
- You can rename the standard name. In doing so, you must make sure that the connection mark - source and connection mark - sink have the same name.
- If you change the name of the connection mark - source, the destination name is automatically renamed.
- If you change the name of the connection mark - sink, the source name is retained.

Observe the command “Connection Mark” for the automatic transformation of an existing connection.

See also

- § Chapter 1.3.1.25.2.12.31 “Command 'Connection Mark'” on page 988
- §
CFC element 'Input Pin'

Symbol: 

Depending on the type of function block you can add further inputs to an inserted function block element. To do this you must select the function block element and drag the function block input element onto the body of the function block.

Please note: You can drag an input or output connection to another position on the function block with the [Ctrl] key pressed.

See also

- Chapter 1.3.1.24.1.6.4.14 “CFC element 'Output Pin’” on page 489

CFC element 'Output Pin'

Symbol: 

Depending on the type of function block you can add further outputs to an inserted function block element. To do this you must select the function block element and drag the function block output element onto the body of the function block.

Please note: You can drag an input or output connection to another position on the function block with the [Ctrl] key pressed.

See also

- Chapter 1.3.1.24.1.6.4.13 “CFC element 'Input Pin’” on page 489

### 1.3.1.24.2 Variables

<table>
<thead>
<tr>
<th>Section</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.3.1.24.2.1</td>
<td>Local variables - VAR............................................................ 489</td>
</tr>
<tr>
<td>1.3.1.24.2.2</td>
<td>Input variables - VAR_INPUT.................................................... 490</td>
</tr>
<tr>
<td>1.3.1.24.2.3</td>
<td>Output variables - VAR_OUTPUT.................................................. 490</td>
</tr>
<tr>
<td>1.3.1.24.2.4</td>
<td>Input/Output variable (VAR_IN_OUT)........................................ 491</td>
</tr>
<tr>
<td>1.3.1.24.2.5</td>
<td>Global variables - VAR_GLOBAL.................................................. 495</td>
</tr>
<tr>
<td>1.3.1.24.2.6</td>
<td>Temporary variable - VAR_TEMP.................................................. 495</td>
</tr>
<tr>
<td>1.3.1.24.2.7</td>
<td>Static variables - VAR_STAT..................................................... 495</td>
</tr>
<tr>
<td>1.3.1.24.2.8</td>
<td>External variables - VAR_EXTERNAL........................................... 496</td>
</tr>
<tr>
<td>1.3.1.24.2.9</td>
<td>Instance variables - VAR_INST.................................................... 496</td>
</tr>
<tr>
<td>1.3.1.24.2.10</td>
<td>Configuration variables - VAR_CONFIG........................................ 497</td>
</tr>
<tr>
<td>1.3.1.24.2.11</td>
<td>Constant variables - VAR_CONSTANTE........................................... 497</td>
</tr>
<tr>
<td>1.3.1.24.2.12</td>
<td>Persistent variable - PERSISTENT................................................ 498</td>
</tr>
<tr>
<td>1.3.1.24.2.13</td>
<td>Retain variable - RETAIN.......................................................... 500</td>
</tr>
<tr>
<td>1.3.1.24.2.14</td>
<td>SUPER....................................................................................... 501</td>
</tr>
<tr>
<td>1.3.1.24.2.15</td>
<td>THIS........................................................................................... 502</td>
</tr>
</tbody>
</table>

The scope of a variable defines how and where you can use a variable. You define the scope in the variable declaration.

**Local variables - VAR**

Local variables are declared between the keywords VAR and END_VAR in the declaration part of programming objects.

You have read-only access to local variables by using the instance path.

You can extend local variables with an attribute keyword.
Example

```c
VAR
 iVar1 : INT;
END_VAR
```

See also

- § Chapter 1.3.1.9.20 “Data persistence” on page 249

Input variables - VAR_INPUT

Input variables are used at the inputs of function blocks.

**VAR_INPUT** variables are declared between the keywords **VAR_INPUT** and **END_VAR** in the declaration part of programming objects.

You can extend input variables with an attribute keyword.

Example

```c
VAR_INPUT
 iIn1 : INT; (* 1st input variable *)
END_VAR
```

See also

- § Chapter 1.3.1.9.20 “Data persistence” on page 249

Output variables - VAR_OUTPUT

Output variables are used at the outputs of function blocks.

**VAR_OUTPUT** variables are declared between the keywords **VAR_OUTPUT** and **END_VAR** in the declaration part of programming objects. CODESYS returns the values of this variable to the calling POU. There you can retrieve the values and continue using them.

You can extend output variables with an attribute keyword.

Example

```c
VAR_OUTPUT
 iOut1 : INT; (* 1st output variable *)
END_VAR
```

See also

- § Chapter 1.3.1.9.20 “Data persistence” on page 249

Output variables in functions and methods

According to the IEC 61131-3 standard, functions and methods have additional outputs. You have to assign these additional outputs when calling the function, as shown below.

Example

```c
fun(iIn1 := 1, iIn2 := 2, iOut1 => iLoc1, iOut2 => iLoc2);
```
Input/Output variable (VAR_IN_OUT)

A VAR_IN_OUT variable is an input/output variable, which is part of a POU interface and serves as a formal pass-by-reference parameter.

Syntax declaration

```plaintext
<keyword> <POU name>
VAR_IN_OUT
   <variable name> : <data type> ( := <initialization value> )? ;
END_VAR
<keyword> : FUNCTION | FUNCTION_BLOCK | METHOD | PRG
```

You can declare an input/output variable in the VAR_IN_OUT declaration section in the POUs PRG, FUNCTION_BLOCK, METHOD, or FUNCTION. As an option, a constant of the declared data type can be assigned as an initialization value. The VAR_IN_OUT variable can be read and written.

Usage

- **Call**: When a POU is called, the formal VAR_IN_OUT variable receives the actual variable (pass-by-reference variable) as the argument. In runtime mode, no copies are generated when parameters are passed. Instead, the formal variable receives a reference to the actual variable passed remotely. The referential variables contain a memory address internally as a value to the actual value (pass as pointer, call-by reference). It is not possible to specify a constant (literal) or a bit variable directly as an argument.
- **Read/write access within the POU**: If the variable is written to within the POU, then this affects the passed variable. When the POU is exited, the performed changes are retained. This means that a POU uses its VAR_IN_OUT variables just like the calling POU uses its variables. Read access is always permitted.
- **Read/write access remotely**: VAR_IN_OUT variables cannot be directly read or written remotely via `<function block instance name>`.`<variable name>`. This works only for VAR_INPUT and VAR_OUTPUT variables.
- **Passing string variables**: If a string variable is passed as an argument, then it is recommended when the actual variable and the formal variable have the same length. Otherwise, the passed string can be manipulated accidentally. This problem does not occur in the case of VAR_OUTPUT CONSTANT parameters.
- **Passing bit variables**: A bit variable cannot be passed directly to a VAR_IN_OUT variable because it needs an intermediate variable.

> When a string is passed as a variable or a constant to a formal VAR_IN_OUT CONSTANT variable, the string is automatically passed completely. You do not have to check the string length.

See also

- Chapter “Transfer variable VAR_IN_OUT CONSTANT” on page 493
Example

Passing arrays

```plaintext
TYPE DUT_A :
STRUCT
  xA: BOOL;
  iB: INT;
END_STRUCT
END_TYPE

FUNCTION_BLOCK FB_SetArray
VAR_IN_OUT
  aData_A : ARRAY[0..1] OF DUT_A; // Formal variable
END_VAR
aData_A[0].xA := TRUE;
aData_A[0].iB := 100;

PROGRAM PLC_PRG
VAR
  fbSetA : FB_SetArray;
aSpecialData : ARRAY[0..1] OF DUT_A; // Actual variable
END_VAR
fbSetA(aData_A := aSpecialData);
```

Passing strings

```plaintext
{attribute 'qualified_only'}
VAR_GLOBAL
  g_sDEV_STATUS : STRING(25) := 'Device_A';
END_VAR

FUNCTION_BLOCK FB_SetStatus
VAR_IN_OUT
  sDeviceStatus : STRING(25); // Formal parameter
END_VAR
sDeviceStatus := CONCAT(sDeviceStatus, ' Activ');

PROGRAM PLC_PRG
VAR
  fbDoB : FB_SetStatus;
END_VAR
fbDoB(sDeviceStatus := GVL.g_sDEV_STATUS); //Call with actual parameter
```

The variable `sDeviceStatus` is part of the POU interface of `FB_B`. When calling `fbDoB`, first a device name is assigned to the string and then the string is manipulated.

Passing bit variables

```plaintext
VAR_GLOBAL
  xBit0 AT %MX0.1 : BOOL;
  xTemp : BOOL;
END_VAR
```
FUNCTION_BLOCK FB_DoSomething
VAR_INPUT
  xIn : BOOL;
END_VAR
VAR_IN_OUT
  xInOut : BOOL;
END_VAR
IF xIn THEN
  xInOut := TRUE;
END_IF

PROGRAM PLC_PRG
VAR
  xIn : BOOL;
  DoSomething_1  : FB_DoSomething;
  DoSomething_2  : FB_DoSomething;
END_VAR

// The following line of code causes a compiler error:
// C0201: Typ 'BIT' is not equal to type 'BOOL' of VAR_IN_OUT 'xInOut'
DoSomething_1(xIn := xIn, xInOut := xBit0);

// Workaround
xTemp := xBit0;
DoSomething_2(xIn := xIn, xInOut := xTemp);
xBit0 := xTemp;

The program calls the function block instances DoSomething_1 and DoSomething_2. As a result of the direct assignment of the bit variable xBit0 to the VAR_IN_OUT input, a compiler error is generated when the DoSomething_1 instance is called. In contrast, calling the DoSomething_2 instance with the assignment of an intermediate variable is correct code.

Transfer variable
VAR_IN_OUT CONSTANT

Syntax declaration
<keyword> <POU name>
VAR_IN_OUT CONSTANT
  <variable name> : <data type>; // formal parameter
END_VAR
<keyword> : FUNCTION | FUNCTION_BLOCK | METHOD | PRG

VAR_IN_OUT CONSTANT variables are declared without assigning an initialization value.

Usage:
- When calling the POU, a constant variable or a literal is passed. All data types are permitted for this. Consequently, write access is not permitted.
- Passing parameters of a string constant: The string length of the constants can be any size, and the string length does not depend on the string length of the VAR_IN_OUT CONSTANT variables.

If the “Replace constants” option is selected in “Project ➔ Project Settings” (category “Compile Options”), then the passing of the parameter of a constant or constant variable with a base data type causes a compiler error to be generated.
This variable is supported in compiler version >= 3.5.2.0.

Example
Passing parameters of string constants and string variables

FUNCTION funManipulate : BOOL
VAR_IN_OUT
sReadWrite : STRING(16); (* Can be read or written here in POU *)
dwVarReadWrite : DWORD; (* Can be read or written here in POU *)
END_VAR
VAR_IN_OUT CONSTANT
c_sReadOnly : STRING(16); (* Constant string variable can only be read here in POU *)
END_VAR

sReadWrite := 'String_from_POU';
dwVarReadWrite := STRING_TO_DWORD(c_sReadOnly);

PROGRAM PRG_A
VAR
sVarFits : STRING(16);
sValFits : STRING(16) := '1234567890123456';
dwVar : DWORD;
END_VAR

// The following line of code causes the compiler error C0417:
// C0417: VAR_IN_OUT parameter 'sReadWrite' needs a variable with write access as input.
funManipulate(sReadWrite := '1234567890123456',
c_sReadOnly := '1234567890123456',
dwVarReadWrite := dwVar);

// Correct code
funManipulate(sReadWrite := sValFits, c_sReadOnly := '23',
dwVarReadWrite := dwVar);
funManipulate(sReadWrite := sVarFits, c_sReadOnly := sValFits,
dwVarReadWrite := dwVar);

In the code, strings are passed to the funManipulate function via different VAR_IN_OUT variables. When passing a literal, a compiler error is output to a VAR_IN_OUT variable. When passing a constant variable to a VAR_IN_OUT CONSTANT variable, correct code is generated even for passing string variables.

See also
- Chapter 1.3.1.9.2 “Declaration of variables” on page 172
- Chapter 1.3.1.25.3.11.3 “Dialog Box 'Project Settings' - 'Compileoptions'” on page 1062
- Chapter 1.3.1.25.1.28.3 “Object 'Function'” on page 781
- Chapter 1.3.1.25.1.28.2 “Object 'Function Block'” on page 778
- Chapter 1.3.1.25.1.28.5 “Object 'Method'” on page 784
- Chapter 1.3.1.25.1.28.4 “Object 'Interface'” on page 783
- Chapter 1.3.1.25.1.28.6 “Object 'Interface Method'” on page 788
- Chapter 1.3.1.24.2.11 “Constant variables - VAR CONSTANT” on page 497
Global variables - VAR_GLOBAL

Global variables are ordinary variables, constants, external or remanent variables that are recognized within the entire project.

You declare global variables in global variable lists or in the declaration section of programming objects between the keywords VAR_GLOBAL and END_VAR.

The system recognizes a global variable when you prepend the variable name with a dot (for example, .iGlobVar1).

**NOTICE!**
If a local variable that is declared in a block has the same name as a global variable, then it has precedence within the block.

**NOTICE!**
For compiler version 3.2.0.0 and later, CODESYS always initializes global variables before the local POU variables.

**Example**

```
VAR_GLOBAL
  iVarGlob1 : INT;
END_VAR
```

See also
- § Chapter 1.3.1.25.1.21 “Object 'GVL' - Global Variable List” on page 769
- § Chapter 1.3.1.24.3.67 “Operator - Global namespace” on page 555

Temporary variable - VAR_TEMP

This function is an extension of the IEC 61131-3 standard.

You declare temporary variables locally between the keywords VAR_TEMP and END_VAR.

VAR_TEMP declarations are possible only in program blocks and function blocks.

CODESYS initializes temporary variables each time the block is called.

The application can access the temporary variables only in the implementation section of a program block or a function block.

**Example**

```
VAR_TEMP
  iVarTmp1 : INT; (*1st temporary variable *)
END_VAR
```

Static variables - VAR_STAT

This function is an extension of the IEC 61131-3 standard.

You declare static variables locally between the keywords VAR_STAT and END_VAR. CODESYS initializes static variables the first time each block is called.

You can access static variables only from within the namespace where the variables are declared (like static variables in C). But static variables retain their values when the application leaves the block. For example, you can use static variables as counters for function calls.
You can extend static variables with an attribute keyword.

**Example**

```
VAR_STAT
  iVarStat1 : INT;
END_VAR
```

See also
-  `$ Chapter 1.3.1.9.20 “Data persistence” on page 249`

**External variables - VAR_EXTERNAL**

External variables are global variables that are imported into a block.

You declare these variables between the keywords `VAR_EXTERNAL` and `END_VAR`. If the global variable does not exist, then an error message is printed.

**NOTICE!**

CODESYS does not require you to declare a global variable as external in order to use it in a POU. The keyword exists only for maintaining compliance with IEC 61131-3.

**Syntax**

```
<POU keyword> <POU name>
VAR_EXTERNAL
  <variable name> : <data type>;
END_VAR
```

Initialization is not permitted.

**Example**

```
FUNCTION_BLOCK FB_DoSomething
  VAR_EXTERNAL
  iVarExt1 : INT; (* 1st external variable *)
END_VAR
```

See also
-  `$ Chapter 1.3.1.25.1.21 “Object 'GVL' - Global Variable List” on page 769`

**Instance variables - VAR_INST**

CODESYS does not save a `VAR_INST` method variable in a method stack, but in the stack of the function block instance. This means that the `VAR_INST` variable functions like other Variables of the function block instance, and it is not reinitialized each time the method is called.

`VAR_INST` variables are permitted in methods only and you can access these variables only within the method. The variable values of instance variables are monitored in the declaration section of the method.

You can extend instance variable with an attribute keyword.
Example

```plaintext
METHOD meth_last : INT
VAR_INPUT
  iVar : INT;
END_VAR
VAR_INST
  iLast : INT := 0;
END_VAR
meth_last := iLast;
iLast := iVar;
```

Configuration variables - VAR_CONFIG

Use configuration variables for assigning complete addresses to variables that are declared in function blocks with incomplete addresses and will be mapped on device I/Os.

Declare the variables in a global variables list between VAR_CONFIG and END_VAR. The global variables list is termed "variables configuration", where you type the configuration variables with a complete instance path and the correct address.

Example

Declaration of the variable xLocIn with incomplete address %I* in a function block:

```plaintext
FUNCTION_BLOCK locio

VAR
  xLocIn AT %I* : BOOL := TRUE;
END_VAR

The locio function block is used in the PLC_PRG program:

```plaintext
PROGRAM PLC_PRG

VAR
  locioVar1 : locio;
END_VAR
```

The correct variables configuration in the global variable list is as follows:

```
VAR_CONFIG
  PLC_PRG.locioVar1.xLocIn AT %IX1.0 : BOOL;
END_VAR
```

See also

- Chapter 1.3.1.9.12.1 “Variables configuration - VAR_CONFIG” on page 227

Constant variables - VAR CONSTANT

Constant variables are declared in global variable lists or in the declaration part of programming objects. In implementations, constant variables can be accessed as read-only via the instance path.

Syntax

```plaintext
<scope>  CONSTANT
    <identifier> : <data type> := <initialization> ;
END_VAR
```

```plaintext
<scope> : VAR | VAR_INPUT | VAR_STAT | VAR_GLOBAL
```

2020/12/10
Always assign an initialization value when declaring a constant variable. Then the constant cannot be written any more.

**Example**

**Declaration**

```plaintext
VAR CONSTANT
  c_rTAXFACTOR : REAL := 1.19;
END_VAR
```

**Call**

```plaintext
rPrice := rValue * c_rTAXFACTOR;
```

In implementations, you can access constant variables exclusively as read-only. Constant variables are located to the right of the assignment operator.

See also

- Chapter 1.3.1.24.2.4 “Input/Output variable (VAR_IN_OUT)” on page 491

### Persistent variable - PERSISTENT

Persistent variables are declared in the declaration section `VAR_GLOBAL RETAIN PERSISTENT` in the persistent global variable list. For variables that are marked with the `PERSISTENT` keyword outside of the persistence editor, instance paths are added there.

As of CODESYS version 3.3.0.1, a variable declaration with `PERSISTENT RETAIN` has the same effect as with `RETAIN PERSISTENT` or `PERSISTENT`.

#### Syntax of the declaration in the global persistent variable list

```plaintext
VAR_GLOBAL PERSISTENT RETAIN
  <identifier>: <data type> (:= <initialization>)?;
  <instance path to POU variable>
END_VAR
```

#### Syntax of the declaration in POU's

```plaintext
<scope> PERSISTENT RETAIN
  <identifier>: <data type> ( := <initialization> )?; // ( ... )? :
Optional
END_VAR
<scope> : VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT | VAR_STAT | VAR_GLOBAL
```

An assignment of inputs, outputs, or memory addresses with the `AT` keyword is not permitted.

**Never use the POINTER TO data type in persistent variable lists. If the application is downloaded again, their addresses could change. The corresponding compiler warnings are shown in the message window.**

**If you frequently change the names or data types of remanent variables, then it is better to declare them as retain variables with the RETAIN keyword only.**
NOTICE!
Declare variables directly in the list of persistent variables and avoid inserting instance paths. As a result, this takes up twice as much memory and also increases the cycle time.

Example

Declaration in the persistent variable list

```plaintext
{attribute 'qualified_only'}
VAR_GLOBAL PERSISTENT RETAIN
g_iCounter : INT;
// Generated instance path of persistent variable
PLC_PRG.fb_A.iPersistentCounter_A: INT;
END_VAR
```

Declaration in the function block FB_A:

```plaintext
FUNCTION_BLOCK FB_A
VAR_INPUT
END_VAR
VAR_OUTPUT
END_VAR
VAR PERSISTENT
   iPersistentCounter_A : INT;
END_VAR
```

Declaration in the program PLC_PRG:

```plaintext
VAR
   fb_A1 : FB_A;
END_VAR
```

Possible declaration locations

<table>
<thead>
<tr>
<th>Locality</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Directly in the persistent global variable list</td>
<td>The variable is persistent and lies in the protected memory area.</td>
</tr>
<tr>
<td>Locally in a program with an instance path in the persistent variable list</td>
<td>The variable is persistent and lies in the protected memory area and in the memory (double allocation).</td>
</tr>
<tr>
<td>Locally in a function block with an instance path in the persistent variable list</td>
<td>This variable is not persistent. A warning is shown in the message window. Hint: Click “Declarations ➔ Add All Instance Paths” to import the variables into the persistent variable list.</td>
</tr>
<tr>
<td>Only locally in a program</td>
<td>This variable is not persistent. A warning is shown in the message window. Hint: Click “Declarations ➔ Add All Instance Paths” to import the variables into the persistent variable list.</td>
</tr>
<tr>
<td>Only locally in a function block</td>
<td>This declaration does not have any effect. This variable is not persistent.</td>
</tr>
<tr>
<td>Locally in a function</td>
<td></td>
</tr>
</tbody>
</table>

In the persistence editor, click “Add All Instance Paths” if local variables are marked with PERSISTENT.
Whenever possible, avoid marking variables, which are declared in a function block, with `PERSISTENT`. This is because the function block instance is stored entirely in remanent memory and not just the marked variable.

See also

- Chapter 1.3.1.9.20 “Data persistence” on page 249
- Chapter 1.3.1.9.20.5 “Declaring VAR PERSISTENT variables ” on page 256
- Chapter 1.3.1.25.2.17.4 “Command ‘Add all instance paths’” on page 1012
- Chapter 1.3.1.9.20.1 “Preserving data with persistent variables” on page 252

**Retain variable - `RETAIN`**

Retain variables are declared by the keyword `RETAIN` is added in programming objects in the scope `VAR`, `VAR_INPUT`, `VAR_OUTPUT`, `VAR_IN_OUT`, `VAR_STAT`, or `VAR_GLOBAL`.

**Syntax in the declaration**

```plaintext
<scope> RETAIN
   <identifier>: <data type> { := <initialization> }? // ( ... )? : Optional
END_VAR
<scope> : VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT | VAR_STAT | VAR_GLOBAL
```

An assignment of inputs, outputs, or memory addresses with the `AT` keyword is not permitted.

**Example**

In a POU:

```plaintext
VAR RETAIN
   iVarRetain: INT;
END_VAR
```

In a GVL:

```plaintext
VAR_GLOBAL RETAIN
   g_iVarRetain: INT;
END_VAR
```

**Possible declaration locations**

<table>
<thead>
<tr>
<th>Locality</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Locally in a program</td>
<td>Only the variable lies within the retain memory area.</td>
</tr>
<tr>
<td>Globally in a global variable list</td>
<td>Only the variable lies within the retain memory area.</td>
</tr>
<tr>
<td>Locally in a function block</td>
<td>The entire instance of the function block with all of its data lies within the retain memory area. Only the declared retain variable is protected.</td>
</tr>
<tr>
<td>Locally in a function</td>
<td>Even the variable does not lie within the retain memory area. This declaration does not have any effect.</td>
</tr>
<tr>
<td>Locally and persistently in a function</td>
<td>Even the variable does not lie within the retain memory area. This declaration does not have any effect.</td>
</tr>
</tbody>
</table>
Whenever possible, avoid marking variables of a function block with RETAIN.

See also
- Chapter 1.3.1.9.20 “Data persistence” on page 249
- Chapter 1.3.1.9.20.5 “Declaring VAR PERSISTENT variables” on page 256
- Chapter 1.3.1.25.2.17.4 “Command 'Add all instance paths'” on page 1012
- Chapter 1.3.1.9.20.2 “Preserving data with retain variables” on page 254

SUPER

SUPER is a special variable and is used for object-oriented programming.

SUPER is the pointer of a function block to the basic function block instance from which the function block was generated. The SUPER pointer thus also permits access to the implementation of the methods of the basic function block (basic class). A SUPER pointer is automatically available for each function block.

You can use SUPER only in methods and in the associated function block implementations.

Dereferencing of the pointer: SUPER^.

Use of the SUPER pointer: with the help of the keyword SUPER you call a method that is valid in the instance of the basic class or parent class.

Examples

ST:

SUPER^.METH_DoIt();

FBD/CFC/LD

---

THIS is not yet implemented for the instruction list (IL).
Use of **SUPER** and **THIS** pointers

```plaintext
FUNCTION_BLOCK FB_Base
  VAR_OUTPUT
    iCnt : INT;
  END_VAR

  METHOD METH_DoIt : BOOL
    iCnt := -1;
  END_METHOD

  METHOD METH_DoAlso : BOOL
    METH_DoAlso := TRUE;
  END_METHOD

FUNCTION_BLOCK FB_1 EXTENDS FB_Base
  VAR_OUTPUT
    iBase : INT;
  END_VAR

  THIS^.METH_DoIt();  //Call of the methods of FB_1
  THIS^.METH_DoAlso();

  SUPER^.METH_DoIt();   //Call of the methods of FB_Base
  SUPER^.METH_DoAlso();
  iBase := SUPER^.iCnt;

  METHOD METH_DoIt : BOOL
    iCnt := 1111;
    METH_DoIt := TRUE;
  END_METHOD

PROGRAM PLC_PRG
  VAR
    myBase : FB_Base;
    myFB_1 : FB_1;
    iTHIS : INT;
    iBase : INT;
  END_VAR

  myBase();
  iBase := myBase.iCnt;
  myFB_1();
  iTHIS := myFB_1.iCnt;
```

See also

- % Chapter 1.3.1.24.5 “Data types” on page 569
- % Chapter 1.3.1.24.2.15 “THIS” on page 502

**THIS**

**THIS** is a special variable and is used for object-oriented programming.

**THIS** is the pointer of a function block to its own function block instance. A **THIS** pointer is automatically available for each function block.

You can use **THIS** only in methods and in function blocks. **THIS** is available for the implementation in the input assistant in the category “Keywords”.

Dereferencing of the pointer: **THIS**^
Use of the `THIS` pointer

- If a local variable obscures a function block variable in a method, you can set the function block variable with the `THIS` pointer. See example below (1)
- If the pointer to the function block’s own function block instance is referenced for use in a function. (See example below (2))

```
ST:
THIS^.METH_DoIt();

FBD/CFC/LD:
```

```
THIS^  
  METH_DoAlso  
  METH_DoAlso  
```

*`THIS` is not yet implemented for the instruction list (IL).*
Examples

(1) The local variable iVarB obscures the function block variable iVarB.

```plc
FUNCTION_BLOCK fbA
VAR_INPUT
  iVarA: INT;
END_VAR
iVarA := 1;

FUNCTION_BLOCK fbB EXTENDS fbA
VAR_INPUT
  iVarB: INT := 0;
END_VAR
iVarA := 11;
iVarB := 2;

METHOD DoIt : BOOL
VAR_INPUT
END_VAR
VAR
  iVarB: INT;
END_VAR
iVarB := 22;    // The local variable iVarB is set.
THIS^.iVarB := 222;    // The function block variable iVarB is set even though iVarB is obscured.
```

```plc
PROGRAM PLC_PRG
VAR
  MyfbB: fbB;
END_VAR
MyfbB(iVarA:=0, iVarB:= 0);
MyfbB.DoIt();
```

(2) A function call requires the reference to its own instance.

```plc
FUNCTION funA
VAR_INPUT
  pFB: fbA;
END_VAR
...

FUNCTION_BLOCK fbA
VAR_INPUT
  iVarA: INT;
END_VAR
...

FUNCTION_BLOCK fbB EXTENDS fbA
VAR_INPUT
  iVarB: INT := 0;
END_VAR
iVarA := 11;
iVarB := 2;

METHOD DoIt : BOOL
VAR_INPUT
END_VAR
VAR
  iVarB: INT;
END_VAR
iVarB := 22;    // The local variable iVarB is set.
funA(pFB := THIS^);    // funA is called via THIS^.
```

PROGRAM PLC_PRG
VAR
  MyfbB: fB;
END_VAR
MyfbB(iVarA:=0 , iVarB:= 0);
MyfbB.DoIt();

See also
- ☞ Chapter 1.3.1.24.5.12 “Pointers” on page 577
- ☞ Chapter 1.3.1.24.2.14 “SUPER” on page 501

1.3.1.24.3 Operators

CODESYS V3 supports all IEC-61131-3 operators. These operators are recognized implicitly throughout the project. In addition to these IEC operators, CODESYS also supports some non-IEC 61131-3 operators.

Operators are used in blocks, such as functions.

For information about the processing order (binding strength) of the ST operators, please refer to the section on ST expressions.

CAUTION!
For operations with floating-point data types, the computational result depends on the applied target system hardware.

CAUTION!
For operations with overflow or underflow in the data type, the computational result depends on the applied target system hardware.

Overflow/underflow in the data type

The CODESYS compiler generates code for the target device and computes temporary results always with the native size that is defined by the target device. For example, computation is performed at least with 32-bit temporary values on x86 and ARM systems and always with 64-bit temporary values on x64 systems. This provides considerable advantages in the computation speed and often also produces the desired result. But this also means that an overflow or underflow in the data type is not truncated in some cases.
Examples

Example 1

The result of this addition is not truncated and the result in \texttt{dwVar} is 65536.

\begin{verbatim}
VAR
wVar : WORD;
dwVar : DWORD;
END_VAR

wVar := 65535;
dwVar := wVar + 1;
\end{verbatim}

Example 2

The overflow and underflow in the data type is not truncated and the results \((bVar1, bVar2)\) of both comparisons are \texttt{FALSE} on 32-bit and 64-bit hardware.

\begin{verbatim}
VAR
wVar1 : WORD;
wVar2 : WORD;
bVar1 : BOOL;
bVar2 : BOOL;
END_VAR

wVar1 := 65535;
wVar2 := 0;
bVar1 := (wVar1 + 1) = wVar2;
bVar2 := (wVar2 - 1) = wVar1;
\end{verbatim}

Example 3

By the assignment to \texttt{wVar3}, the value is truncated to the target data type \texttt{WORD} and the result \texttt{bVar1} is \texttt{TRUE}.

\begin{verbatim}
VAR
wVar1 : WORD;
wVar2 : WORD;
wVar3 : WORD;
bVar1 : BOOL;
END_VAR

wVar1 := 65535;
wVar2 := 0;
wVar3 := (wVar1 + 1);
bVar1 := wVar3 = wVar2;
\end{verbatim}

Example 4

In order to force the compiler to truncate the temporary results, a conversion can be inserted. The type conversion makes sure that both comparisons are 16-bit only and the results \((bVar1, bVar2)\) of both comparisons are each \texttt{TRUE}.

\begin{verbatim}
VAR
wVar1 : WORD;
wVar2 : WORD;
bVar1 : BOOL;
bVar2 : BOOL;
END_VAR

wVar1 := 65535;
wVar2 := 0;
bVar1 := \texttt{TO\_WORD}(wVar1 + 1) = wVar2;
bVar2 := \texttt{TO\_WORD}(wVar2 - 1) = wVar1;
\end{verbatim}
Arithmetic operators
- Chapter 1.3.1.24.3.1 “Operator ‘ADD’” on page 509
- Chapter 1.3.1.24.3.3 “Operator ‘SUB’” on page 510
- Chapter 1.3.1.24.3.2 “Operator ‘MUL’” on page 510
- Chapter 1.3.1.24.3.4 “Operator ‘DIV’” on page 511
- Chapter 1.3.1.24.3.5 “Operator ‘MOD’” on page 512
- Chapter 1.3.1.24.3.6 “Operator ‘MOVE’” on page 512
- Chapter 1.3.1.24.3.7 “Operator ‘INDEXOF’” on page 513
- Chapter 1.3.1.24.3.8 “Operator ‘SIZEOF’” on page 513

Bitstring operators
- Chapter 1.3.1.24.3.10 “Operator ‘AND’” on page 513
- Chapter 1.3.1.24.3.11 “Operator ‘OR’” on page 514
- Chapter 1.3.1.24.3.12 “Operator ‘XOR’” on page 514
- Chapter 1.3.1.24.3.9 “Operator ‘NOT’” on page 513
- Chapter 1.3.1.24.3.13 “Operator ‘AND_THEN’” on page 515
- Chapter 1.3.1.24.3.14 “Operator ‘OR_ELSE’” on page 515

Bitshift operators
- Chapter 1.3.1.24.3.15 “Operator ‘SHL’” on page 516
- Chapter 1.3.1.24.3.16 “Operator ‘SHR’” on page 516
- Chapter 1.3.1.24.3.17 “Operator ‘ROL’” on page 517
- Chapter 1.3.1.24.3.18 “Operator ‘ROR’” on page 518

Selection operators
- Chapter 1.3.1.24.3.19 “Operator ‘SEL’” on page 519
- Chapter 1.3.1.24.3.20 “Operator ‘MAX’” on page 520
- Chapter 1.3.1.24.3.21 “Operator ‘MIN’” on page 520
- Chapter 1.3.1.24.3.22 “Operator ‘LIMIT’” on page 521
- Chapter 1.3.1.24.3.23 “Operator ‘MUX’” on page 521

Comparison operators
A comparison operator is a Boolean that compares two inputs (first and second operand).
- Chapter 1.3.1.24.3.24 “Operator ‘GT’” on page 522
- Chapter 1.3.1.24.3.25 “Operator ‘LT’” on page 522
- Chapter 1.3.1.24.3.26 “Operator ‘LE’” on page 522
- Chapter 1.3.1.24.3.27 “Operator ‘GE’” on page 523
- Chapter 1.3.1.24.3.28 “Operator ‘EQ’” on page 523
- Chapter 1.3.1.24.3.29 “Operator ‘NE’” on page 523

Address operators
- Chapter 1.3.1.24.3.30 “Operator ‘ADR’” on page 524
- Chapter 1.3.1.24.3.31 “Operator ‘Content Operator’” on page 524
- Chapter 1.3.1.24.3.32 “Operator ‘BITADR’” on page 525

Call operators
- Chapter 1.3.1.24.3.33 “Operator ‘CAL’” on page 525
Implicit conversion from a larger type to a smaller type is not possible (for example, from INT to BYTE or from DINT to WORD). You must use a special type conversion to convert from a larger type to a smaller type. As a rule, you can convert any elementary type into any other elementary type.

**Typed conversion**:
<elementary type1>_TO_<elementary type2>

**Overflow conversion**:
TO_<elementary type2>

---

**NOTICE!**

In the case of ...TO_STRING conversions, CODESYS generates the string left-aligned. If the string length is defined too short, then it will be truncated on the right.

---

**NOTICE!**

If the input value of a type conversion operator is outside the value range of the output data type, then the result of the operation is undefined and dependent on the platform. This is the case, for example, if the input value for the conversion from LREAL to DINT is negative.

See also
- Chapter 1.3.1.24.3.34 “Operator ‘BOOL_TO’” on page 526
- Chapter 1.3.1.24.3.35 “Operator ‘TO_BOOL’” on page 527
- Chapter 1.3.1.24.3.37 “Operator ‘<INT Type>_TO_<INT Type>’” on page 528
- Chapter 1.3.1.24.3.38 “Operator ‘REAL_TO_/LREAL_TO_’” on page 529
- Chapter 1.3.1.24.3.39 “Operator ‘TIME_TO/TIME_OF_DAY_TO_’” on page 530
- Chapter 1.3.1.24.3.40 “Operator ‘DATE_TO/DT_TO’” on page 531
- Chapter 1.3.1.24.3.41 “Operator ‘STRING_TO’” on page 532
- Chapter 1.3.1.24.3.42 “Operator ‘TRUNC’” on page 533
- Chapter 1.3.1.24.3.43 “Operator ‘TRUNC_INT’” on page 534
- Chapter 1.3.1.24.3.36 “Operator ‘TO_<xxx>’” on page 527

---

### Numeric Operators

- Chapter 1.3.1.24.3.44 “Operator ‘ABS’” on page 534
- Chapter 1.3.1.24.3.45 “Operator ‘SQR’” on page 534
- Chapter 1.3.1.24.3.46 “Operator ‘LN’” on page 535
- Chapter 1.3.1.24.3.47 “Operator ‘LOG’” on page 535
- Chapter 1.3.1.24.3.48 “Operator ‘EXP’” on page 535
- Chapter 1.3.1.24.3.49 “Operator ‘EXPT’” on page 536
- Chapter 1.3.1.24.3.50 “Operator ‘SIN’” on page 536
- Chapter 1.3.1.24.3.53 “Operator ‘ASIN’” on page 537
- Chapter 1.3.1.24.3.51 “Operator ‘COS’” on page 537
- Chapter 1.3.1.24.3.52 “Operator ‘TAN’” on page 537
- Chapter 1.3.1.24.3.54 “Operator ‘ACOS’” on page 538
- Chapter 1.3.1.24.3.55 “Operator ‘ATAN’” on page 538

---

### Namespace operators

Namespace operators are extended from IEC 61131-3 operators and make it possible for you to provide unique access to variables and modules, even when you use the same name multiple times for variables or modules in a project.

- Chapter 1.3.1.24.3.67 “Operator - Global namespace” on page 555
- Chapter 1.3.1.24.3.68 “Operator - Namespace for global variables lists” on page 555
Working with different tasks requires the synchronization of these tasks. This is especially true when working on multicore platforms. Some special operators are provided in CODESYS to support this synchronization.

These operators are extensions of IEC-61131-3. The operators `TEST_AND_SET` and `__COMPARE_AND_SWAP` are used for similar tasks.

See also
- Chapter 1.3.1.24.3.66 “Operator ‘TEST_AND_SET’” on page 554
- Chapter 1.3.1.24.3.64 “Operator ‘__COMPARE_AND_SWAP’” on page 552
- Chapter 1.3.1.24.3.65 “Operator ‘__XADD’” on page 553

Other operators
- Chapter 1.3.1.24.3.56 “Operator ‘__DELETE’” on page 538
- Chapter 1.3.1.24.3.57 “Operator ‘__ISVALIDREF’” on page 541
- Chapter 1.3.1.24.3.58 “Operator ‘__NEW’” on page 541
- Chapter 1.3.1.24.3.59 “Operator ‘__QUERYINTERFACE’” on page 544
- Chapter 1.3.1.24.3.60 “Operator ‘__QUERY_POINTER’” on page 545
- Chapter 1.3.1.24.3.71 “Operator ‘INI’” on page 556
- Chapter 1.3.1.24.3.61 “Operators ‘__TRY’, ‘__CATCH’, ‘__FINALLY’, ‘__ENDTRY’” on page 546

Operator ‘ADD’

This IEC operator is used for adding variables.

Permitted data types: BYTE, WORD, DWORD, LWORD, SINT, USINT, INT, UINT, DINT, UDINT, LINT, ULINT, REAL, LREAL, TIME, TIME_OF_DAY (TOD), DATE, DATE_AND_TIME (DT)

Possible combinations for TIME data types: TIME+TIME = TIME, TOD+TIME = TOD, DT+TIME = DT

Feature in the FBD/LD editor: You can extend the ADD operator to function block inputs. The number of additional function block inputs is limited.
Examples

**ST:**

```
var1 := 7+2+4+7;
```

**FBD:**

---

### Operator 'MUL'

This IEC operator is used for multiplying variables.

**Permitted data types:** BYTE, WORD, DWORD, LWORD, SINT, USINT, INT, UINT, DINT, UDINT, LINT, ULINT, REAL, LREAL, TIME

Feature in the FBD/LD editor: You can extend the MUL operator to additional function block inputs. The number of additional function block inputs is limited.

---

**Examples**

**ST:**

```
var1 := 7*2*4*7;
```

**FBD:**

---

### Operator 'SUB'

This IEC operator is used for subtracting variables.
Permitted data types: BYTE, WORD, DWORD, LWORD, SINT, USINT, INT, UINT, DINT, UDINT, LINT, ULINT, REAL, LREAL, TIME, TIME_OF_DAY (TOD), DATE, DATE_AND_TIME (DT)

Possible combinations for TIME data types: TIME-TIME = TIME, DATE-DATE = TIME, TOD-TOD = TOD, TOD-TOD = TIME, DT-TIME = DT, DT-DT = TIME

Negative TIME values are undefined.

Examples

<table>
<thead>
<tr>
<th>ST:</th>
<th>FBD:</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>var1 := 7-2;</code></td>
<td><img src="image" alt="SUB" /></td>
</tr>
</tbody>
</table>

Operator 'DIV'

This IEC operator is used for dividing variables.

Permitted data types: BYTE, WORD, DWORD, LWORD, SINT, USINT, INT, UINT, DINT, UDINT, LINT, ULINT, REAL, LREAL, TIME

NOTICE!
Division by zero may have different results depending on the target system.

Examples

<table>
<thead>
<tr>
<th>ST:</th>
<th>FBD:</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>var1 := 8/2;</code></td>
<td><img src="image" alt="FBD" /></td>
</tr>
</tbody>
</table>

1. Series of DIV blocks, 2. Single DIV block, 3. DIV blocks with EN/ENO parameters
Please note that it is possible to monitor division by zero at runtime by using the implicit monitoring functions **CheckDivInt**, **CheckDivLint**, **CheckDivReal**, and **CheckDivLReal**.

See also

- % Chapter 1.3.1.25.1.29.2 “POU 'CheckDivInt'” on page 801
- % Chapter 1.3.1.25.1.29.3 “POU 'CheckDivLint'” on page 801
- % Chapter 1.3.1.25.1.29.4 “POU 'CheckDivReal'” on page 802
- % Chapter 1.3.1.25.1.29.5 “POU 'CheckDivLReal'” on page 803

**Operator 'MOD'**

This IEC operator is used for modulo division.
The result of the function is the integer remainder of division.

**Permitted data types:** BYTE, WORD, DWORD, LWORD, SINT, USINT, INT, UINT, DINT, UDINT, LINT, ULINT

**NOTICE!**
Division by zero may have different results depending on the target system.

**Examples**

Result in Var1: 1

**ST:**

```
var1 := 9 MOD 2;
```

**FBD:**

![MOD diagram]

**Operator 'MOVE'**

This IEC operator is used for assigning a variable to another variable of a corresponding type.

Because the **MOVE** block is available in the CFC, FBD, and LD editors, you can also use the **EN/ENO** functionality for variable assignment.

**CFC with EN/ENO function:**

CODESYS assigns the value of **var1** to **var2** only if "en_i" yields **TRUE**.

**ST:**

```
ivar2 := MOVE(ivar1);
```

This corresponds to:

```
ivar2 := ivar1;
```
Operator 'INDEXOF'

This operator is an extension of the IEC 61131-3 standard.

Instead of the INDEXOF operator, the ADR operator is provided in CODESYS V3 for obtaining a pointer at the index of a block.

See also
● § Chapter 1.3.1.24.3.30 “Operator ‘ADR’” on page 524

Operator 'SIZEOF'

This operator is an extension of the IEC 61131-3 standard.

This operator is used for defining the number of bytes that are required by the variable \( x \). The SIZEOF operator always yields an unsigned value. The type of return variable adapts to the detected size of the variable \( x \).

<table>
<thead>
<tr>
<th>Return Value of ( \text{SIZEOF}(x) )</th>
<th>Data type of the constant (CODESYS uses this implicitly for detected size)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 &lt;= size of ( x ) &lt; 256</td>
<td>USINT</td>
</tr>
<tr>
<td>256 &lt;= size of ( x ) &lt; 65536</td>
<td>UINT</td>
</tr>
<tr>
<td>65536 &lt;= size of ( x ) &lt; 4294967296</td>
<td>UDINT</td>
</tr>
<tr>
<td>4294967296 &lt;= size of ( x )</td>
<td>ULINT</td>
</tr>
</tbody>
</table>

Examples

Result in \( \text{Var1}:10 \)

ST:

\[
\text{arr1 : ARRAY}[0..4] \text{ OF INT;}
\]
\[
\text{Var1 : INT;}
\]
\[
\text{var1 := SIZEOF(arr1);} \quad (* \text{var1 := USINT#10}; *\)
\]

Operator 'NOT'

This IEC operator is used for the bitwise \text{NOT} of a bit operand.

When the respective input bit yields 0, the output bit also yields 1, and vice-versa.

Permitted data types: BOOL, BYTE, WORD, DWORD, LWORD

Examples

Result in \( \text{var1:2#0110_1100} \)

ST:

\[
\text{var1 := NOT 2#1001_0011;}
\]

FBD:

![Diagram of NOT operator]

Operator 'AND'

This IEC operator is used for the bitwise \text{AND} of bit operands.
When the input bits all yield 1, the output bit also yields 1; otherwise 0.

Permitted data types: BOOL, BYTE, WORD, DWORD, LWORD

### Examples

<table>
<thead>
<tr>
<th>Result in</th>
<th>ST:</th>
<th>FBD:</th>
</tr>
</thead>
<tbody>
<tr>
<td>var1</td>
<td>var1 := 2#1001_0011 AND 2#1000_1010;</td>
<td>var1 := 2#1001_0011 AND 2#1000_1010;</td>
</tr>
</tbody>
</table>

#### Operator 'OR'

This IEC operator is used for the bitwise OR of bit operands.

When at least one of the input bits yields 1, the output bit also yields 1; otherwise 0.

Permitted data types: BOOL, BYTE, WORD, DWORD, LWORD

<table>
<thead>
<tr>
<th>Result in</th>
<th>ST:</th>
<th>FBD:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Var1</td>
<td>Var1 := 2#1001_0011 OR 2#1000_1010;</td>
<td>Var1 := 2#1001_0011 OR 2#1000_1010;</td>
</tr>
</tbody>
</table>

#### Operator 'XOR'

This IEC operator is used for the bitwise XOR of bit operands.

When only one of the two input bits yields 1, the output bit also yields 1. When both inputs yield 1 or 0, then the output yields 0.

Permitted data types: BOOL, BYTE, WORD, DWORD, LWORD

---

**NOTICE!**

Please note the following behavior of the XOR block in extended form (more than two inputs): CODESYS compares the inputs in pairs and then the corresponding results (according to the standard, but not necessarily according to expectations).
Examples

Result in var1: 2#0001_1001

ST:
var1 := 2#1001_0011 XOR 2#1000_1010;

FBD:

Operator 'AND_THEN'

This operator is an extension of the IEC 61131-3 standard.

The AND_THEN operator is permitted only for programming in structured text with the AND operation of BOOL and BIT operands with short-circuit evaluation. This means that:

When all operands yield TRUE, the result of the operands also yield TRUE; otherwise FALSE.

However, CODESYS also executes the expressions on other operands only if the first operand of the AND_THEN operator is TRUE. This can prevent problems with null pointers, for example in conditions such as IF (ptr <> 0 AND_THEN ptr^ = 99) THEN....

In contrast, CODESYS always evaluates all operands when using the AND IEC operator.

See also
● “Chapter 1.3.1.24.3.10 “Operator ‘AND’ on page 513

Operator 'OR_ELSE'

This operator is an extension of the IEC 61131-3 standard.

The OR_ELSE operator is permitted only for programming in structured text: OR operation of BOOL and BIT operands; with short-circuit evaluation. This means:

When at least one of the operands yields TRUE, the result of the operation also yields TRUE; otherwise FALSE.

In contrast to using the OR IEC operator, for OR_ELSE the expressions on all other operators are not evaluated as soon as one of the operands is evaluated as TRUE.

Example

VAR
    bEver: BOOL;
    bX: BOOL;
    dw: DWORD := 16#000000FF;
END_VAR
bEver := FALSE;
bX := dw.8 OR_ELSE dw.1 OR_ELSE dw.1 OR_ELSE (bEver := TRUE);

See also
● “Chapter 1.3.1.24.3.11 “Operator ‘OR’ ” on page 514
Operator 'SHL'

This IEC operator is used for bitwise shift of an operand to the left.

\[
\text{erg} := \text{SHL} \left( \text{in}, n \right)
\]

\text{in}: \text{Operand that is shifted to the left}

\text{n}: \text{Number of bits to shift \text{in} to the left}

\begin{center}
\textbf{NOTICE!}

If \(n\) overwrites the data type width, then it depends on the target system how the \text{BYTE}, \text{WORD}, \text{DWORD}, \text{and LWORD} operands are padded. The target systems cause padding with zeros or \(n \mod \text{\text{<tab width>}}\).
\end{center}

\begin{center}
\textbf{NOTICE!}

Please note the number of bits that CODESYS uses for this operation as defined by the data type of the input variable \text{in}.
\end{center}

\textbf{Examples}

The results for \text{erg} \_ \text{byte} and \text{erg} \_ \text{word} are different, although the values of the \text{in} \_ \text{byte} and \text{in} \_ \text{word} input variables are the same and the data types of the input variables are different.

\textbf{ST}:

\begin{verbatim}
PROGRAM shl_st
VAR
  in_byte : BYTE := 16#45; (* 2#01000101 )
  in_word : WORD := 16#0045; (* 2#0000000001000101 )
  erg_byte : BYTE;
  erg_word : WORD;
  n: BYTE := 2;
END_VAR

  erg_byte := SHL(in_byte,n); (* Result is 16#14, 2#00010100 *)
  erg_word := SHL(in_word,n); (* Result is 16#0114, 2#0000000100010100 *)
\end{verbatim}

\textbf{FBD}:

\begin{center}
\begin{tikzpicture}
  \node [pin={[pin distance=1cm]below:2}, text width=0.25cm, text centered] at (0,0) {in};
  \node [text width=0.25cm, text centered] at (0.5,0) {2};
  \node [text width=0.25cm, text centered] at (1,0) {erg_byte};
  \node [text width=0.25cm, text centered] at (0.5,0.5) {\textbf{SHL}};
\end{tikzpicture}
\end{center}

Operator 'SHR'

This IEC operator is used for bitwise shift of an operand to the right.

\[
\text{erg} := \text{SHR} \left( \text{in}, n \right)
\]

\text{in}: \text{Operand that is shifted to the right}

\text{n}: \text{Number of bits for shifting \text{in} to the right}
NOTICE!
If \( n \) overwrites the data type width, then it depends on the target system how the BYTE, WORD, DWORD, and LWORD operands are padded. The target systems cause padding with zeros or \( n \) MOD <tab width>.

**Examples**

```st
PROGRAM shr_st
VAR
  in_byte : BYTE:=16#45; (* 2#01000101 *)
  in_word : WORD:=16#0045; (* 2#0000000001000101 *)
  erg_byte : BYTE;
  erg_word : WORD;
  n: BYTE :=2;
END_VAR

  erg_byte := SHR(in_byte,n); (* Result is 16#11, 2#00010001 *)
  erg_word := SHR(in_word,n); (* Result is 16#0011,
  2#0000000000010001 *)
```

**FBD:**

```
in_byte --- Shr --- erg_byte

2
```

**Operator 'ROL'**

This IEC operator is used for bitwise rotation of an operand to the left.

**Permitted data types:** BYTE, WORD, DWORD, LWORD

```st
erg := ROL (in, n)
```

CODESYS moves \( in \) n-times one bit to the left and adds the bit to the leftmost position from the right.

NOTICE!
Please note the number of bits that CODESYS uses for this operation as defined by the data type of the input variable \( in \). If this is a constant, then CODESYS uses the smallest possible data type. The data type of the output variables still does not influence this operation.
The results for `erg_byte` and `erg_word` are different depending on the data type of the input variables, although the values of the `in_byte` and `in_word` input variables are the same.

**ST:**

```plaintext
PROGRAM rol_st

VAR
  in_byte : BYTE := 16#45;
  in_word : WORD := 16#45;
  erg_byte : BYTE;
  erg_word : WORD;
  n : BYTE := 2;
END_VAR

erg_byte := ROL(in_byte, n); (* Result: 16#15 *)
erg_word := ROL(in_word, n); (* Result: 16#0114 *)
```

**FBD:**

![FBD diagram](image)

**IL:**

```
LD  in_byte
ROL n
ST  erg_byte
```

**Operator 'ROR'**

This IEC operator is used for bitwise rotation of an operand to the right.

Permitted data types: BYTE, WORD, DWORD, LWORD

```plaintext
erg := ROR(in, n)
```

CODESYS moves `in` n-times one bit to the right and adds the bit to the rightmost position from the left.

Please note the number of bits that CODESYS uses for this operation as defined by the data type of the input variable `in`. If this is a constant, then CODESYS uses the smallest possible data type. The data type of the output variables still does not influence this operation.
The results for `erg_byte` and `erg_word` are different depending on the data type of the input variables, although the values of the `in_byte` and `in_word` input variables are the same.

**ST:**

```plaintext
PROGRAM ror_st

VAR
  in_byte : BYTE := 16#45;
  in_word : WORD := 16#45;
  erg_byte : BYTE;
  erg_word : WORD;
  n: BYTE := 2;
END_VAR

erg_byte := ROR(in_byte,n); (* Result: 16#51 *)
erg_word := ROR(in_word,n); (* Result: 16#4011 *)
```

**FBD:**

![FBD diagram](image)

---

**Operator 'SEL'**

The IEC operator is used for bitwise selection.

```plaintext
OUT := SEL(G, IN0, IN1) means:
OUT := IN0; if G = FALSE
OUT := IN1; if G = TRUE
```

Permitted data types:

IN0, ..., INn and OUT: Any identical data type. Make sure that variables of the identical type are used at all three positions, especially when using user-defined data types. The compiler checks for type identity and returns any compile errors. The assignment of function block instances to interface variables is specifically not supported.

G: BOOL

---

**NOTICE!**

When G is TRUE, CODESYS does not compute an expression that precedes IN0. When G is FALSE, CODESYS does not compute an expression that precedes IN1.

Caution: In the case of graphical programming languages, the expressions at IN0 and IN1 are computed independently of the G input when a “Box”, “Jump”, “Return”, “Line Branch”, or “Edge Detection” precedes.
Examples

ST:
Var1 := SEL(TRUE, 3, 4); (* Result: 4 *)

FBD:

---

Operator 'MAX'

This IEC operator is used for the maximum function. It yields the largest value of two values.

\[
\text{OUT} := \text{MAX(IN0, IN1)}
\]

Permitted data types: all

Examples

ST:
Result: 90
Var1 := MAX(30, 40);
Var1 := MAX(40, MAX(90, 30));

FBD:
Result: 90

---

Operator 'MIN'

This IEC operator is used for the minimum function. It yields the smallest value of two values.

\[
\text{OUT} := \text{MIN(IN0, IN1)}
\]

Permitted data types: all
Examples | Result: 30
---|---
ST: | 
Var1:=MIN(90,30); 
Var1 := MIN(MIN(90,30),40); 
FBD: 

![FBD Diagram](image)

**Operator 'LIMIT'**

This IEC selection operator is used for limiting.

\[
\text{OUT} := \text{LIMIT}(\text{Min}, \text{IN}, \text{Max})
\]

**Means:** \(\text{OUT} := \text{MIN} (\text{MAX} (\text{IN}, \text{Min}), \text{Max})\)

Max is the upper limit and Min is the lower limit for the result. If the IN value is above the Max upper limit, then LIMIT yields Max. If the value of IN is below the Min lower limit, then the result is Min.

Permitted data types for IN and OUT: all

Examples | Result in Var1 is 80
---|---
ST: | 
Var1 := LIMIT(30,90,80); 

**Operator 'MUX'**

This IEC operator is used as a multiplexer.

\[
\text{OUT} := \text{MUX}(\text{K}, \text{IN0}, \ldots, \text{INn})
\]

**Means:** \(\text{OUT} = \text{IN}_K\)

Permitted data type for K: BYTE, WORD, DWORD, LWORD, SINT, USINT, INT, UINT, DINT, LINT, ULINT, OUDINT.

IN0, ..., INn, and OUT: Any identical data type. Make sure that variables of the identical type are used at all three positions, especially when using user-defined data types. The compiler checks for type identity and returns any compile errors. The assignment of function block instances to interface variables is specifically not supported.

MUX selects the K-th value from a set of values. The first value is K=0. If K is greater than the number of other inputs (n), then CODESYS passes on the last value (INn).

**NOTICE!**

For runtime optimization, CODESYS computes only the expression that precedes \(\text{IN}_K\). However, CODESYS computes all branches in simulation mode.
### Examples

Result in \( \text{Var1} \) is 30.

**ST:**

\[ \text{Var1} := \text{MUX}(0, 30, 40, 50, 60, 70, 80); \]

### Operator 'GT'

This IEC operator is used for the "greater than" function.

Permitted data types of the operands: any basic data type.

If the first operand is greater than the second operand, then the operator yields the result **TRUE**; otherwise **FALSE**.

**Examples**

Result: **FALSE**

**ST:**

\[ \text{VAR1} := 20 > 30; \]

**FBD:**

![FBD diagram for GT](image)

### Operator 'LT'

This IEC operator is used for the "less than" function.

Permitted data types of the operands: any basic data type.

If the first operand is less than the second operand, then the operator yields the result **TRUE**; otherwise **FALSE**.

**Examples**

Result: **TRUE**

**ST:**

\[ \text{Var1} := 20 < 30; \]

**FBD:**

![FBD diagram for LT](image)

### Operator 'LE'

This IEC operator is used for the "less than or equal to" function.

Permitted data types of the operands: any basic data type.

If the first operand is less than or equal to the second operand, then the operator yields the result **TRUE**; otherwise **FALSE**.
Examples

Result in Var1: TRUE

ST:
Var1 := 20 <= 30;

Operator 'GE'

This IEC operator is used for the "greater than or equal to" function.
Permitted data types of the operands: any basic data type.
If the first operand is greater than or equal to the second operand, then the operator yields the result TRUE; otherwise FALSE.

Examples

Result: TRUE

ST:
VAR1 := 60 >= 40;

Operator 'EQ'

This IEC operator is used for the "equals" function.
Permitted data types of the operands: any basic data type, depending on target system and compiler version: structure data type.
If the operands are equal, then then the operator yields the result TRUE, otherwise FALSE.

Examples

Result: TRUE

ST:
VAR1 := 40 = 40;

Operator 'NE'

This IEC operator is used for the "does not equal" function.
Permitted data types of the operands: any basic data type, depending on target system and compiler version: structure data type.
If the operands are not equal, then the operator yields the result **TRUE**; otherwise **FALSE**.

If the target system supports the data type, then as from compiler version >= 3.5.7.0 also operands of type **STRUCT** (structure) can be compared. Example: IF (stStruct1 := stStruct2) THEN....

### Examples

**Result in Var1 is FALSE**

**ST:**

Var1 := 40 <> 40;

**FBD:**

![Diagram of Var1]

---

### Operator ‘ADR’

This operator is an extension of the IEC 61131-3 standard.

ADR yields the address of its argument in a **DWORD**. You can pass this address to the manufacturer functions or assign them to a pointer in the project.

---

**NOTICE!**

As opposed to CoDeSys V2.3, you can use the ADR operator with function names, program names, function block names, and method names. Therefore, **ADR** replaces the **INDEXOF** operator. When using function pointers, please note that you can pass a function pointer to external libraries, but it is not possible to call a function pointer from within CODESYS. To enable a system call (runtime system), you must set the respective object property (**Build** tab) for the function object.

---

**CAUTION!**

When you apply an online change, address contents can shift, causing **POINTER** variables to reference invalid memory ranges. To avoid problems, make sure that CODESYS updates pointer values in every cycle.

---

**CAUTION!**

Do not return **Pointer-TO** variables of functions and methods to the caller or assign them to global variables.

### Examples

**ST:**

`dwVar := ADR(bVAR);`

---

### Operator ‘Content Operator’

This operator is an extension of the IEC 61131-3 standard.

You can use this operator to dereference pointers by appending the operator as `^` to the pointer identifier.
CAUTION!
When using pointers to addresses, please note that applying an online change can shift address contents.

Example

ST:

```
pt : POINTER TO INT;
var_int1 : INT;
var_int2 : INT;
pt := ADR(var_int1);
var_int2 := pt^;
```

Operator 'BITADR'

This operator is an extension of the IEC 61131-3 standard.

```
BITADR yields the bit offset within a segment in a DWORD.
```

NOTICE!
The offset depends on whether the byte addressing check box is selected or cleared in the target system settings.

The highest value nibble (4 bits) in this DWORD defines the memory range:

- **Flag:** 16x40000000
- **Input:** 16x80000000
- **Output:** 16xC0000000

CAUTION!
When using pointers to addresses, please note that applying an online change can shift address contents.

Examples

ST:

```
VAR
  Var1 AT %IX2.3 : BOOL;
  bitoffset: DWORD;
END_VAR

bitoffset := BITADR(var1); (* Result if byte addressing = TRUE: 16x80000013, if byte addressing = FALSE: 16x80000023 *)
```

Operator 'CAL'

This IEC operator is used for calling function blocks.

In IL, **CAL** calls the instance of a function block.

```
CAL <function block> (<input variable1> := <value>, <input variableN> := <value>)
```
Example

Call of the \texttt{Inst} instance of a function block with assignment of the input variables \texttt{Par1} and \texttt{Par2} with 0 or \textsc{true}.

\texttt{CAL Inst(Par1 := 0, Par2 := \textsc{true});}

---

**Operator ‘BOOL\_TO’**

The IEC operator is used for converting the \texttt{BOOL} data type to another data type.

\texttt{BOOL\_TO\_<data\ type>}

For number types the result is 1, when the operand is \textsc{true}, and 0, when the operand is \textsc{false}.

For the \texttt{STRING} data type, the result is \textsc{true} or \textsc{false}.

\begin{itemize}
  \item \textbf{NOTICE!} If the input value of a type conversion operator is outside the value range of the output data type, then the result of the operation is undefined and dependent on the platform. This is the case, for example, if the input value for the conversion from \texttt{LREAL} to \texttt{DINT} is \textsc{negative}.
\end{itemize}

---

**Examples**

<table>
<thead>
<tr>
<th>ST code</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>\texttt{i := BOOL_TO_INT(TRUE);}</td>
<td>1</td>
</tr>
<tr>
<td>\texttt{str := BOOL_TO_STRING(TRUE);}</td>
<td>\textsc{true}</td>
</tr>
<tr>
<td>\texttt{t := BOOL_TO_TIME(TRUE);}</td>
<td>T#1ms</td>
</tr>
<tr>
<td>\texttt{tof := BOOL_TO_TOD(TRUE);}</td>
<td>TOD#00:00:00.001</td>
</tr>
<tr>
<td>\texttt{dat := BOOL_TO_DATE(FALSE);}</td>
<td>D#1970</td>
</tr>
<tr>
<td>\texttt{dandt := BOOL_TO_DT(TRUE);}</td>
<td>DT#1970-01-01-00:00:01</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>FBD code</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRUE – \texttt{BOOL_TO_INT} – i</td>
<td>1</td>
</tr>
<tr>
<td>TRUE – \texttt{BOOL_TO_STRING} – str</td>
<td>\textsc{true}</td>
</tr>
<tr>
<td>TRUE – \texttt{BOOL_TO_TIME} – t</td>
<td>T#1ms</td>
</tr>
<tr>
<td>TRUE – \texttt{BOOL_TO_TOD} – tof</td>
<td>TOD#00:00:00.001</td>
</tr>
<tr>
<td>FALSE – \texttt{BOOL_TO_DATE} – t</td>
<td>D#1970-01-01</td>
</tr>
<tr>
<td>TRUE – \texttt{BOOL_TO_DT} – dandt</td>
<td>DT#1970-01-01-00:00:01</td>
</tr>
</tbody>
</table>
Operator 'TO_BOOL'

The IEC operator is used for converting from another variable type into a BOOL variable.

_<data type>_TO_BOOL

The result yields TRUE when the operand does not equal 0. The result yields FALSE when the operand equals 0.

For the STRING type, the result is true if the operand is TRUE; otherwise it is FALSE.

**NOTICE!**
If the input value of a type conversion operator is outside the value range of the output data type, then the result of the operation is undefined and dependent on the platform. This is the case, for example, if the input value for the conversion from LREAL to DINT is negative.

### Examples

<table>
<thead>
<tr>
<th>ST code</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>b := BYTE_TO_BOOL(2#11010101);</td>
<td>TRUE</td>
</tr>
<tr>
<td>b := INT_TO_BOOL(0);</td>
<td>FALSE</td>
</tr>
<tr>
<td>b := TIME_TO_BOOL(T#5ms);</td>
<td>TRUE</td>
</tr>
<tr>
<td>b := STRING_TO_BOOL('TRUE');</td>
<td>TRUE</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>FBD code</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>BYTE_TO_BOOL</td>
<td>TRUE</td>
</tr>
<tr>
<td>INT_TO_BOOL</td>
<td>FALSE</td>
</tr>
<tr>
<td>TIME_TO_BOOL</td>
<td>TRUE</td>
</tr>
<tr>
<td>STRING_TO_BOOL</td>
<td>TRUE</td>
</tr>
</tbody>
</table>

See also

- **“Type conversion operators” on page 508**

Operator 'TO_<xxx>,'

The IEC operator is used for converting variables of one data type into another data type, where the input type is not explicitly given (overflow conversion).

<data type>_TO_<data type>

The function depends on what is described for each typed conversion.
NOTICE!
If the input value of a type conversion operator is outside the value range of the output data type, then the result of the operation is undefined and dependent on the platform. This is the case, for example, if the input value for the conversion from LREAL to DINT is negative.

Examples

ST implementation language:

```st
VAR
  iVar : INT;
  bVar : BOOL;
  sVar : STRING;
  rVar : REAL;
END_VAR
iVar := TO_INT(4.22);            (* Result:  4 *)
bVar := TO_BOOL(1);            (* Result: TRUE *)
sVar := TO_STRING(342);        (* Result: '342' *)
rVar := TO_WORD('123');        (* Result: 123 *)
```

See also
● “Type conversion operators” on page 508

Operator `<INT Type>_TO_<INT Type>`

Conversion of one integer data type into another integer data type.
`<INT data type>_TO_<INT data type>`

NOTICE!
Information can be lost when converting from larger data types to smaller data types. If the number to be converted exceeds the range limit, then CODESYS ignores the first bytes of the number.

NOTICE!
If the input value of a type conversion operator is outside the value range of the output data type, then the result of the operation is undefined and dependent on the platform. This is the case, for example, if the input value for the conversion from LREAL to DINT is negative.

Examples

Result in si:127

ST:
  si := INT_TO_SINT(4223);
If you save the integer 4223 (16#107f in hexadecimal notation) as a SINT variable, then this variable is assigned the value 127 (16#7f in hexadecimal notation).

FBD:
Operator 'REAL_TO_/ LREAL_TO'  

The IEC operator is used for converting the REAL and LREAL data types into another data type.

```plaintext
REAL_TO_<data type>  
LREAL_TO_<data type>
```

CODESYS rounds the real value of the operand up or down to an integer value and then converts it into the respective type. Exceptions are the STRING, BOOL, REAL, and LREAL data types.

**NOTICE!**  
The rounding logic for borderline cases depends on the target system or the FPU (Floating Point Unit) of the target system. For example, a value of −1.5 can be converted differently on different controllers.

**NOTICE!**  
When converting data types from REAL or LREAL into SINT, USINT, INT, UINT, DINT, UDINT, LINT, or ULINT, if the value of the REAL/LREAL number is outside the value range of the integer, then you will receive an undefined result depending on the target system. Then an exception is also possible. To get code that is independent of the target system, you must use the application to catch value range overflows. If the REAL/LREAL number is within the range, then the conversion operates the same way on all systems.

When converting the STRING type, note that the total length of the decimal number is limited to 16 characters. If the (L)REAL number has more decimal places, then the sixteenth place is rounded and represented as a string. If the STRING is defined too short for the number, then CODESYS truncates it from the right.

**NOTICE!**  
Information can be lost when converting from larger data types to smaller data types.

**NOTICE!**  
If the input value of a type conversion operator is outside the value range of the output data type, then the result of the operation is undefined and dependent on the platform. This is the case, for example, if the input value for the conversion from LREAL to DINT is negative.
Examples

<table>
<thead>
<tr>
<th>ST code</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>i := REAL_TO_INT(1.5);</td>
<td>2</td>
</tr>
<tr>
<td>j := REAL_TO_INT(1.4);</td>
<td>1</td>
</tr>
<tr>
<td>i := REAL_TO_INT(-1.5);</td>
<td>-2</td>
</tr>
<tr>
<td>j := REAL_TO_INT(-1.4);</td>
<td>-1</td>
</tr>
</tbody>
</table>

FBD code

<table>
<thead>
<tr>
<th>FBD code</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.5 REAL_TO_INT 1</td>
<td>2</td>
</tr>
</tbody>
</table>

See also

- “Type conversion operators” on page 508

Operator 'TIME_TO/TIME_OF_DAY_TO'

The IEC operator is used for converting the TIME and TIME_OF_DAY data types into another data type.

<TIME data type>_TO_<data type>

Internally, CODESYS saves the time (in milliseconds) to a DWORD (for TIME_OF_DAY since 00:00). CODESYS converts this value.

For the STRING data type, the result is the time constant.

NOTICE!

Information can be lost when converting from larger data types to smaller data types.

NOTICE!

If the input value of a type conversion operator is outside the value range of the output data type, then the result of the operation is undefined and dependent on the platform. This is the case, for example, if the input value for the conversion from LREAL to DINT is negative.
### Examples

<table>
<thead>
<tr>
<th>ST code</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>str := TIME_TO_STRING(T#12ms);</code></td>
<td>T#12ms</td>
</tr>
<tr>
<td><code>dw := TIME_TO_DWORD(T#5m);</code></td>
<td>300000</td>
</tr>
<tr>
<td><code>si := TOD_TO_SINT(TOD#00:00:00.012);</code></td>
<td>12</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>FBD code</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>T#12ms</code></td>
<td><code>str</code></td>
</tr>
<tr>
<td><code>TIME TO STRING</code></td>
<td></td>
</tr>
<tr>
<td><code>T#5m</code></td>
<td><code>dw</code></td>
</tr>
<tr>
<td><code>TIME TO DWORD</code></td>
<td>30000</td>
</tr>
<tr>
<td><code>TOD#00:00:00.012</code></td>
<td><code>si</code></td>
</tr>
<tr>
<td><code>TOD_TO_SINT</code></td>
<td>12</td>
</tr>
</tbody>
</table>

See also
- ☞ “Type conversion operators” on page 508

### Operator ‘DATE_TO/DT_TO’

The IEC operator is used for converting the `DATE` and `DATE_AND_TIME` data types into another data type.

`<DATE data type>_TO_<data type>`

Internally, CODESYS saves the date in a `DWORD` (in seconds since 01 January 1970). CODESYS converts this value.

For the `STRING` data type, the result is the date constant.

**NOTICE!**

Information can be lost when converting from larger data types to smaller data types.

**NOTICE!**

If the input value of a type conversion operator is outside the value range of the output data type, then the result of the operation is undefined and dependent on the platform. This is the case, for example, if the input value for the conversion from `LREAL` to `DINT` is negative.
Examples

<table>
<thead>
<tr>
<th>ST code</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>b := DATE_TO_BOOL(D#1970-01-01);</td>
<td>FALSE</td>
</tr>
<tr>
<td>i := DATE_TO_INT(D#1970-01-15);</td>
<td>29952</td>
</tr>
<tr>
<td>i := DT_TO_BYTE(DT#1970-01-15-05:05:05);</td>
<td>129</td>
</tr>
<tr>
<td>str := DT_TO_STRING(DT#1998-02-13-14:20);</td>
<td>DT#1998-02-13-14:20</td>
</tr>
</tbody>
</table>

FBD code

<table>
<thead>
<tr>
<th>FBD code</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>D#1970-01-01 DATE_TO_BOOL b</td>
<td>FALSE</td>
</tr>
<tr>
<td>D#1970-01-15 DATE_TO_INT i</td>
<td>29952</td>
</tr>
<tr>
<td>D#1970-01-15-05:05:05 DATE_TO_BYTE i</td>
<td>129</td>
</tr>
<tr>
<td>D#1998-02-13-14:20 DATE_TO_STRING str</td>
<td>DT#1998-02-13-14:20</td>
</tr>
</tbody>
</table>

See also
- ☑ “Type conversion operators” on page 508

Operator ‘STRING_TO’

The IEC operator is used for converting the STRING data type into another data type.

**STRING_TO_.<data type>**

You must define the STRING operand according to the IEC 61131-3 standard. The value has to be a valid constant (literal) of the target type. This affects any given exponential values, infinite values, prefixes, grouping characters (.), and commas. Additional characters after the digits of a number are permitted (for example, 23xy). Additional characters before a number are not permitted.

The operand must be a valid value of the target data type.

**NOTICE!**

If the data type of the operand does not match the target data type or if the value is outside the range of the target data type, then the result output depends on the processor type and is therefore undefined. Information can be lost when converting from larger data types to smaller data types.
NOTICE!
If the input value of a type conversion operator is outside the value range of the output data type, then the result of the operation is undefined and dependent on the platform. This is the case, for example, if the input value for the conversion from LREAL to DINT is negative.

| Examples |
|-----------------|----------------|
| **ST code**     | **Result**     |
| b := STRING_TO_BOOL('TRUE'); | TRUE         |
| w := STRING_TO_WORD('abc34'); | 0            |
| w := STRING_TO_WORD('34abc'); | 34           |
| t := STRING_TO_TIME('T#127ms'); | T#127ms     |
| r := STRING_TO_REAL('1.234'); | 1,234        |
| bv := STRING_TO_BYTE('500'); | 244          |

<table>
<thead>
<tr>
<th>FBD code</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="STRING_TO_BOOL" /></td>
<td>TRUE</td>
</tr>
</tbody>
</table>

See also
- ☞ “Type conversion operators” on page 508

Operator 'TRUNC'
The IEC operator is used for converting the REAL data type into the DINT data type. CODESYS takes only the integer part of the number.

In CoDeSys V2.3, the TRUNC operator converts REAL into INT. If you import a V2.3 project, then CODESYS automatically replaces TRUNC with TRUNC_INT.

If CODESYS cannot represent the input value by a DINT or INT, then the result of this function is undefined. The behavior of such input values is platform-dependent.

NOTICE!
If the input value of a type conversion operator is outside the value range of the output data type, then the result of the operation is undefined and dependent on the platform. This is the case, for example, if the input value for the conversion from LREAL to DINT is negative.

<table>
<thead>
<tr>
<th>Examples</th>
<th>Result in diVar: l</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ST</strong></td>
<td></td>
</tr>
<tr>
<td>diVar := TRUNC(1.9); (* Result: 1 *)</td>
<td></td>
</tr>
<tr>
<td>diVar := TRUNC(-1.4); (* Result: -1 *)</td>
<td></td>
</tr>
</tbody>
</table>
See also

- “Type conversion operators” on page 508

Operator 'TRUNC_INT'

The IEC operator is used for converting the REAL data type into the INT data type. CODESYS takes only the integer part of the number.

TRUNC_INT corresponds to the TRUNC operator in CoDeSys V2.3, and it is used automatically at this point when importing V2.3 projects. Note the change function of TRUNC.

If CODESYS cannot represent the input value by a DINT or INT, then the result of this function is undefined. The behavior of such input values is platform-dependent.

NOTICE!

If the input value of a type conversion operator is outside the value range of the output data type, then the result of the operation is undefined and dependent on the platform. This is the case, for example, if the input value for the conversion from LREAL to DINT is negative.

Examples

Result in iVar: 1

ST:

iVar := TRUNC_INT(1.9); (* Result: 1 *)
iVar := TRUNC_INT(-1.4); (* Result: -1 *)

See also

- “Type conversion operators” on page 508

Operator 'ABS'

This IEC operator yields the absolute value of a number.

Permitted data types for input and output variables and numeric constants: any numeric basic data type

Examples

Result in i: 2

ST:

i := ABS(-2);

FBD:

Operator 'SQRT'

This IEC of course yields the square root of a number.

Permitted data types for input variables: any numeric basic data type
Permitted data types for output variables: REAL or LREAL

Examples

Result in q: 4

ST:

```
q := SQRT(16);
```

FBD:

Operator 'LN'

This IEC operator yields the natural logarithm of a number.
Permitted data types for input variables: any numeric basic data type
Permitted data types for output variables: REAL and LREAL

Examples

Result: 3.80666

ST:

```
q := LN(45);
```

FBD:

Operator 'LOG'

This IEC operator yields the base-10 logarithm of a number.
The input variable can be any numeric basic data type, but the output variable must be the data type REAL or LREAL.

Examples

Result in q: 2.49762

ST:

```
q := LOG(314.5);
```

FBD:

Operator 'EXP'

This IEC operator yields the exponential function.
Permitted data types for input variables: any numeric basic data type
Permitted data types for output variables: REAL and LREAL
### Example

**Power function with literals**

Var1 := EXPT(7,2);

FBD:

```
  EXPT

```

Return value: Var1 = 49

---

### Example

**Power function with variables**

PROGRAM PLC_PRG
VAR
  lrPow : LREAL;
  iBase : INT := 2;
  iExponent : INT := 7;
END_VAR

lrPow := EXPT(iBase, iExponent);

Return value: lrPow = 128

---

### Operator 'EXPT'

This IEC operator raises a number to a higher power and returns the power of the base raised to the exponent: \( \text{power} = \text{base}^{\text{exponent}} \). The input values (parameters) are the base and the exponent. The power function is undefined if the base is zero and the exponent is negative. However, the behavior depends on the platform in this case.

Syntax:

\[
\text{EXPT}(<\text{base}>,<\text{exponent}>)
\]

Permitted data types for the input values: Numeric base data types (SINT, USINT, INT, UINT, DINT, UDINT, LINT, ULINT, REAL, LREAL, BYTE, WORD, DWORD, and LWORD)

Permitted data types for the return value: Floating-point number types (REAL and LREAL)

---

### Operator 'SIN'

This IEC operator yields the sine value of a number.

Permitted data types for input variables that measure the angle in radians: any numeric basic data type

Permitted data types for output variables: REAL and LREAL

---

Result in q: 7.389056099

ST:

\[ q := \text{EXP}(2); \]

FBD:

```
  EXP

```

2 \rightarrow q

Examples
### Operator 'COS'

This IEC operator yields the cosine value of a number.

Permitted data types for input variables that measure the angle in radians: any numeric basic data type

Permitted data types for output variables: \texttt{REAL} and \texttt{LREAL}

**Examples**

Result in \( q \): 0.479426

ST:

\[
q := \text{SIN}(0.5);
\]

FBD:

![SIN block](image)

**Examples**

Result in \( q \): 0.877583

ST:

\[
q := \text{COS}(0.5);
\]

FBD:

![COS block](image)

### Operator 'TAN'

This IEC operator yields the tangent value of a number.

Permitted data types for input variables that measure the angle in radians: any numeric basic data type

Permitted data types for output variables: \texttt{REAL} and \texttt{LREAL}

**Examples**

Result in \( q \): 0.546302

ST:

\[
q := \text{TAN}(0.5);
\]

FBD:

![TAN block](image)

### Operator 'ASIN'

This IEC operator yields the arcsine value of a number.

Permitted data types for input variables: any numeric basic data type

Permitted data types for output variables: \texttt{REAL} and \texttt{LREAL}
Examples

Result in q: 0.523599

ST:
q := ASIN(0.5);

FBD:

Operator 'ACOS'

This IEC operator yields the arccosine value of a number. The value is computed in radians.
Permitted data types for input variables that measure the angle in radians: any numeric basic data type
Permitted data types for output variables: REAL and LREAL

Examples

Result in q: 1.0472

ST:
q := ACOS(0.5);

FBD:

Operator 'ATAN'

This IEC operator yields the arctangent value of a number. The value is computed in radians.
Permitted data types for input variables that measure the angle in radians: any numeric basic data type
Permitted data types for output variables: REAL and LREAL

Examples

Result in q: 0.463648

ST:
q := ATAN(0.5);

FBD:

Operator '__DELETE'

This operator is an extension of the IEC 61131-3 standard.
NOTICE!
For compatibility, the compiler version must be >= 3.3.2.0.

The operator releases the memory of instances that the "__NEW" operator generated dynamically. The __DELETE operator does not have a return value and the operand is set to zero after this operation.

Requirement: In the properties dialog of the application, the “Use dynamic memory allocation” check box is selected in the “Application Build Options” tab.

__DELETE (<pointer>)

NOTICE!
Two tasks should not call __DELETE simultaneously. Either you use a semaphore (SysSemEnter) or comparable method to prevent any concurrent calling of __DELETE, or you use __DELETE in one tasks only (recommended).

You can use a semaphore (SysSemEnter) to prevent two tasks from allocating memory at the same time. As a consequence, the extensive use of __DELETE causes higher jitter.

If Pointer references a function block, then CODESYS calls the associated FB_EXIT method before the pointer is set to zero.
FUNCTION_BLOCK FBDynamic

VAR_INPUT
  in1, in2 : INT;
END_VAR

VAR_OUTPUT
  out : INT;
END_VAR

VAR
  test1 : INT := 1234;
  _inc : INT := 0;
  _dut : POINTER TO DUT;
  neu : BOOL;
END_VAR

out := in1 + in2;

METHOD FB_Exit : BOOL

VAR_INPUT
  bInCopyCode : BOOL;
END_VAR

__Delete(_dut);

METHOD FB_Init : BOOL

VAR_INPUT
  bInitRetains : BOOL;
  bInCopyCode : BOOL;
END_VAR

_dut := __NEW(DUT);

METHOD INC : INT

VAR_INPUT
END_VAR

_inc := _inc + 1;
INC := _inc;

PLC_PRG(PRG)

VAR
  pFB : POINTER TO FBDynamic;
  bInit: BOOL := TRUE;
  bDelete: BOOL;
  loc : INT;
END_VAR

IF (bInit) THEN
  pFB := __NEW(FBDynamic);
  bInit := FALSE;
END_IF
IF (pFB <> 0) THEN
    pFB^.(in1 := 1, in2 := loc, out => loc);
    pFB^.INC();
END_IF

IF (bDelete) THEN
    __DELETE(pFB);
END_IF

Operator ‘__ISVALIDREF’

This operator is an extension of the IEC 61131-3 standard.

The operator is used for checking whether a reference refers to a valid value. For a description of use and an example, refer to the description for the REFERENCE data type.

See also
- § Chapter 1.3.1.24.5.13 “Reference” on page 579

Operator ‘__NEW’

The operator is an extension of the IEC 61131-3 standard.

The __NEW operator reserves dynamic memory to instantiate function blocks, user-defined data types, or arrays of standard types. The operator returns a matching typed pointer.

Requirement: In the properties dialog of the parent application, on the “Application Build Options” tab, the “Use dynamic memory allocation” option is selected.

Syntax

<pointer name> := __NEW( <type> ( , <size> )? );
__DELETE( <pointer name> );

<type> : <function block> | <data unit type> | <standard data type>

The operator generates an instance of the type <type> and returns a pointer to this instance. Then the initialization of the instance is called. If <type> is a scalar standard data type, then the optional operand <size> is also evaluated. Then the operator generates an array of type <standard data type> and size <size>. If the attempt to allocate memory fails, then __NEW returns the value 0.

Use the operator within the assignment ":=". Otherwise an error message is displayed.

A function block or a user-defined data type whose instance is created dynamically with __NEW uses a fixed memory area. Here it is required that you mark the objects with the pragma (attribute 'enable_dynamic_creation'). It is not required for function blocks that are part of a library.

If you change the data layout of the function block in online mode, then you cannot execute a login with an online change afterwards. This is because the memory area of the function block instance has been invalidated. You change the data layout when you add new variables to the function block, delete existing variables, or change the data types of variables.
Example

Array (DWORD):

```plaintext
PROGRAM PLC_PRG
VAR
  pdwScalar : POINTER TO DWORD; // Typed pointer
  xInit : BOOL := TRUE;
  xDelete : BOOL;
END_VAR

IF (xInit) THEN
  pdwScalar := __NEW(DWORD, 16); // Allocates memory (16 dwords) and assigns them to pointer pdwScalar
END_IF
IF (xDelete) THEN
  __DELETE(pdwScalar); // Frees memory of pointer
END_IF
```

Function block:

```plaintext
{attribute 'enable_dynamic_creation'}
FUNCTION_BLOCK FBComputeGamma
VAR_INPUT
  iAlpha : INT;
  iBeta : INT;
END_VAR
VAR_OUTPUT
  iGamma : INT;
END_VAR
VAR
END_VAR

iGamma := iAlpha + iBeta;
```

User-defined data type (DUT):

```plaintext
{attribute 'enable_dynamic_creation'}
TYPE ABCDATA :
  STRUCT
    iA, iB, iC, iD : INT;
  END_STRUCT
END_TYPE

PROGRAM PLC_PRG
VAR
  pABCDData : POINTER TO ABCDATA; // Typed pointer
  xInit : BOOL := TRUE;
END_VAR
```
PROGRAM PLC_PRG
VAR
pbDataAlpha : POINTER TO BYTE;
pDataBeta : POINTER TO BYTE;
xInit : BOOL := TRUE;
xDelete : BOOL;
usiCnt : USINT;
bTestC: BYTE;
END_VAR

IF (xInit) THEN
    pbDataAlpha := __NEW(BYTE, 16); // Allocates 16 bytes for pbDataAlpha
    pDataBeta := __NEW(BYTE); // Allocates memory for pDataBeta
    xInit := FALSE;
    FOR usiCnt := 0 TO 15 DO
        pbDataAlpha[usiCnt] := usiCnt; // Writes to new array
    END_FOR
    pDataBeta^:= 16#FF; // Writes to new data
END_IF

bTestC := pbDataAlpha[12]; // Reads new array by index access

IF (xDelete) THEN // Frees memory
    ___DELETE(pbDataAlpha);
    ___DELETE(pDataBeta);
END_IF

Array (BYTE):

NOTICE!

We do not recommend the simultaneous execution of two tasks that both call the __NEW operator. You use either a semaphore (SysSemEnter) or a comparable technique to prevent a concurrent call of __NEW. However, this results in a higher jitter when __NEW is applied extensively.

We recommend that you call __NEW operators in one task only.

See also
- Chapter 1.3.1.25.3.10.9 “Dialog 'Properties' - 'Application Build Options’” on page 1051
- Chapter 1.3.1.25.1.14 “Object 'DUT’” on page 735
- Chapter 1.3.1.25.1.28.2 “Object 'Function Block’” on page 778
- Chapter 1.3.1.24.6.2.39 “Attribute 'enable_dynamic_creation’” on page 637
- Chapter 1.3.1.24.11.133 “Compiler error C0509” on page 709
Operator '__QUERYINTERFACE'

This operator is an extension of the IEC 61131-3 standard.

At runtime, the operator executes a type conversion of an interface reference into another type. The operator returns a BOOL result. TRUE means that CODESYS has performed the conversion successfully.

__QUERYINTERFACE(<ITF_Source>,<ITF_Dest>);

1. Operand: Interface reference or FB interface
2. Operand: Interface reference with required target type

The requirement for the explicit conversion is that both the ITF_Source and ITF_Dest are derived from Interface __System.IQueryInterface. This interface is implicitly available does not require a library.
Example

INTERFACE ItfBase EXTENDS __System.IQueryInterface
METHOD mbase : BOOL
END_METHOD

INTERFACE ItfDerived1 EXTENDS ItfBase
METHOD mderived1 : BOOL
END_METHOD

INTERFACE ItfDerived2 EXTENDS ItfBase
METHOD mderived2 : BOOL
END_METHOD

FUNCTION_BLOCK FB1 IMPLEMENTS ItfDerived1
METHOD mbase : BOOL
mbase := TRUE;
END_METHOD
METHOD mderived1 : BOOL
mderived1 := TRUE;
END_METHOD
END_FUNCTION_BLOCK

FUNCTION_BLOCK FB2 IMPLEMENTS ItfDerived2
METHOD mbase : BOOL
mbase := FALSE;
END_METHOD
METHOD mderived2 : BOOL
mderived2 := TRUE;
END_METHOD
END_FUNCTION_BLOCK

PROGRAMM POU
VAR
inst1 : FB1;
inst2 : FB2;
itfbase1 : ItfBase := inst1;
itfbase2 : ItfBase := inst2;
itfderived1 : ItfDerived1 := 0;
itfderived2 : ItfDerived2 := 0;
xResult1, xResult2, xResult3, xResult4: BOOL;
END_VAR

xResult1 := __QUERYINTERFACE(itfbase1, itfderived1); // xResult = TRUE, itfderived1 <>0
                  // references
the instance inst1
xResult2 := __QUERYINTERFACE(itfbase1, itfderived2); // xResult = FALSE, itfderived2 = 0
xResult3 := __QUERYINTERFACE(itfbase2, itfderived1); // xResult = FALSE, itfderived1 = 0
xResult4 := __QUERYINTERFACE(itfbase2, itfderived2); // xResult = TRUE, itfderived2 <> 0
                  // references
the instance inst2

Operator '___QUERYPOINTER'
This operator is an extension of the IEC 61131-3 standard.
At runtime, the operator makes it possible to convert the type of an interface reference of a function block to a pointer. The operator returns a BOOL result. TRUE means that CODESYS has performed the conversion successfully.

**NOTICE!**

For compatibility, the definition of the pointer to be converted must be an extension of the base interface "__SYSTEM.IQueryInterface".

`__QUERYPOINTER (<ITF_Source>, <Pointer_Dest>)`

The operator receives an interface reference or a FB instance with the required target types as the first operand and a pointer as the second operand. After processing `__QUERYPOINTER`, Pointer_Dest receives the pointer to the reference or instance of a function block that the ITF_Source interface reference currently refers to. Pointer_Dest is not typed and can be cast to any type. You have to make sure of the type. For example, the interface could offer a method that returns a type code.

**Operators '__TRY', '__CATCH', '__FINALLY', '__ENDTRY'**

These operators are extended from the IEC 61131-3 standard and they are used for specific exception handling in IEC code.

**Syntax**

```
__TRY
  <statements_try>
__CATCH(exec)
  <statements_catch>
__FINALLY
  <statements_finally>
__ENDTRY
  <statements_next>
```

When a statement in the __Try operator throws an exception, the application does not stop. Instead, the application executes the statements in __Catch, starts the exception handling, and then executes the statements in __FINALLY. The exception handling ends with __ENDTRY, and the application executes the subsequent statements.

An IEC variable for an exception has the data type `__System.ExceptionCode`. 

---

PLC Automation with V3 CPUs
Programming with IEC 61131-3 editor > CODESYS Development System

2020/12/10

3ADR010583, 1, en_US

546
If the statement in \_TRY throws an exception, then program execution is not stopped. Instead, the statement in \_CATCH is executed. Therefore, in this example, the application executes the exc function, then the statement in \_FINALLY, and finally the statement in \_ENDTRY.

FUNCTION Tester : UDINT
VAR_INPUT
  count : UDINT;
END_VAR
VAR_OUTPUT
  strExceptionText : STRING;
END_VAR
VAR
  exc : __SYSTEM.ExceptionCode;
END_VAR
\_TRY
  Tester := tryFun(count := count, testcase := g_testcase);  // This statement is tested. If it throws an exception, then the statement in \_CATCH is executed first, and then the statement in \_FINALLY.
  \_CATCH(exc)
    HandleException(exc, strExceptionText => strExceptionText);
  \_FINALLY
    GVL.g_count := GVL.g_count + 2;
\_ENDTRY

See also
● § Chapter 1.3.1.25.2.6.17 “Command ‘Stop Execution on Handled Exceptions’” on page 936

Data Type
'__System.ExceptionCode'

TYPE ExceptionCode :
  (RTSEXCPT_UNKNOWN := 16#FFFFFFFF,
   RTSEXCPT_NOEXCEPTION := 16#00000000,
   RTSEXCPT_WATCHDOG := 16#00000010,
   RTSEXCPT_HARDWAREWATCHDOG := 16#00000011,
   RTSEXCPT_IO_CONFIG_ERROR := 16#00000012,
   RTSEXCPT_PROGRAMCHECKSUM := 16#00000013,
   RTSEXCPT_FILEBUS_ERROR := 16#00000014,
   RTSEXCPT_IOUPDATE_ERROR := 16#00000015,
   RTSEXCPT_CYCLE_TIME_EXCEED := 16#00000016,
   RTSEXCPT_ONLCHANGE_PROGRAM_EXCEEDED := 16#00000017,
   RTSEXCPT_UNRESOLVED_EXTREFS := 16#00000018,
   RTSEXCPT_DOWNLOAD_REJECTED := 16#00000019,
   RTSEXCPT_BOOTPROJECT_REJECTED_DUE_RETAIN_ERROR := 16#0000001A,
   RTSEXCPT_LOADBOOTPROJECT_FAILED := 16#0000001B,
   RTSEXCPT_OUT_OF_MEMORY := 16#0000001C,
   RTSEXCPT_RETAIN_MEMORY_ERROR := 16#0000001D,
   RTSEXCPT_BOOTPROJECT_CRASH := 16#0000001E,
   RTSEXCPT_BOOTPROJECTTARGETMISMATCH := 16#00000021,
   RTSEXCPT_SCHEDULEERROR := 16#00000022,
   RTSEXCPT_FILE_CHECKSUM_ERR := 16#00000023,
   RTSEXCPT_RETAIN_IDENTITY_MISMATCH := 16#00000024,
   RTSEXCPT_IEC_TASK_CONFIG_ERROR := 16#00000025,
   RTSEXCPT_APP_TARGET_MISMATCH := 16#00000026,
   RTSEXCPT_ILLEGAL_INSTRUCTION := 16#00000050,
   RTSEXCPT_ACCESS_VIOLATION := 16#00000051,
RTSEXCPT_PRIV_INSTRUCTION := 16#00000052,
RTSEXCPT_IN_PAGE_ERROR := 16#00000053,
RTSEXCPT_STACK_OVERFLOW := 16#00000054,
RTSEXCPT_INVALID_DISPOSITION := 16#00000055,
RTSEXCPT_INVALID_HANDLE := 16#00000056,
RTSEXCPT_GUARD_PAGE := 16#00000057,
RTSEXCPT_DOUBLE_FAULT := 16#00000058,
RTSEXCPT_INVALID_OPCODE := 16#00000059,
RTSEXCPT_MISALIGNMENT := 16#000000100,
RTSEXCPT_ARRAYBOUNDS := 16#000000101,
RTSEXCPT_DIVIDEBYZERO := 16#000000102,
RTSEXCPT_OVERFLOW := 16#000000103,
RTSEXCPT_NONCONTINUABLE := 16#000000104,
RTSEXCPT_PROCESSORLOAD_WATCHDOG := 16#000000105,
RTSEXCPT_FPU_ERROR := 16#000001001,
RTSEXCPT_FPU_DENORMAL_OPERAND := 16#000001002,
RTSEXCPT_FPU_DIVIDEBYZERO := 16#000001003,
RTSEXCPT_FPU_INEXACT_RESULT := 16#000001004,
RTSEXCPT_FPU_INVALID_OPERATION := 16#000001005,
RTSEXCPT_FPU_OVERFLOW := 16#000001006,
RTSEXCPT_FPU_STACK_CHECK := 16#000001007,
RTSEXCPT_FPU_UNDERFLOW := 16#000001008,
RTSEXCPT_VENDOR_EXCEPTION_BASE := 16#000020001,
RTSEXCPT_USER_EXCEPTION_BASE := 16#000100001
) UDINT ;
END_TYPE

Operator ‘__VARINFO’

This operator is an extension of the IEC 61131-3 standard.
The operator yields information about a variable. You can save the information as data structure in a variable of data type __SYSTEM.VAR_INFO.

Syntax in the declaration:

<name of the info variable> : __SYSTEM.VAR_INFO; // Data structure for info variable

Syntax for the call:

<name of the info variable> := __VARINFO( <variable name> ); // Call of the operator
FUNCTION_BLOCK FB_Velocity
VAR_INPUT
  rVelocity: REAL := 1.2;
END_VAR
VAR_OUTPUT
END_VAR
VAR
  infoVelocity: __SYSTEM.VAR_INFO; //Info of Velocity
END_VAR

infoVelocity := __VARINFO(rVelocity); // Gets the info of Velocity locally

PROGRAM PLC_PRG
VAR
  iCounter : INT := 0; // Counts the calls
  infoCounter : __SYSTEM.VAR_INFO; //Info of Counter
  arrA : ARRAY [1..2, 1..2, 1..2] OF INT := [0, 1, 2, 3, 4, 5, 6, 7]; // Stores the A data
  infoA : __SYSTEM.VAR_INFO; //Info of A
  fbVel : FB_Velocity;
END_VAR

iCounter := iCounter + 1;
infoCounter := __VARINFO(iCounter);
infoA := __VARINFO(arrA);
fbVel();

The iCounter and arrA variables are recognized in the application code. The variable information is saved in the infoCounter and infoA variables. Moreover, the FB_Velocity function block is instantiated.

<table>
<thead>
<tr>
<th>Name</th>
<th>Data type</th>
<th>Initialization</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ByteAddress</td>
<td>DWORD</td>
<td>0</td>
<td>Address of the variable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Example: 16#072E35EC</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Note: For bit access of a variable &lt;variable name&gt;.&lt;bit index&gt;, the address of the variable that contains the bit is given.</td>
</tr>
<tr>
<td>ByteOffset</td>
<td>DWORD</td>
<td>0</td>
<td>Offset of the variable address (in bytes).</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Example: 13936 bytes.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Note: If the variable is global, then the offset is relative to the beginning of the area. If the variable is a local variable in a function or method, then the offset is relative to the current stack frame. If the variable is a local variable in a function block, then the offset is relative to the function block instance.</td>
</tr>
<tr>
<td>Name</td>
<td>Data type</td>
<td>Initialization</td>
<td>Description</td>
</tr>
<tr>
<td>--------------</td>
<td>-----------</td>
<td>----------------</td>
<td>----------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Area</td>
<td>DINT</td>
<td>0</td>
<td>Memory area number $\text{Area}$ in the runtime system.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Example: $-1$: Means that the variable is not global in the memory, but relative to an instance or on the stack.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Note: The memory areas are device-dependent.</td>
</tr>
<tr>
<td>BitNr</td>
<td>INT</td>
<td>0</td>
<td>Number of bits (in bytes)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Example: $\text{16#00FF}$ bytes</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Note: If the variable is not an integer data type, then $\text{BitNr} = -1 = \text{16#FFFF}$.</td>
</tr>
<tr>
<td>BitSize</td>
<td>INT</td>
<td>0</td>
<td>Memory size of the variable (in bits)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Example: 16 bits</td>
</tr>
<tr>
<td>BitAddress</td>
<td>UDINT</td>
<td>0</td>
<td>Bit address of the variable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Requirement: The variable is located in the input memory area $\text{I}$, output memory area $\text{Q}$, or marker memory area $\text{M}$.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Otherwise the value is undefined.</td>
</tr>
<tr>
<td>TypeClass</td>
<td>TYPE_CLASS</td>
<td></td>
<td>Data type class of the variable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Example: $\text{TYPE_INT, TYPE_ARRAY}$</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Note: For user-defined data types or function block instances, $\text{TYPE_USERDEF}$ is output as the data type class.</td>
</tr>
<tr>
<td>TypeName</td>
<td>STRING(79)</td>
<td>*</td>
<td>Date type name of the variable as $\text{STRING(79)}$</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Note: For user-defined data types, the function block name or the DUT name is output.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Example: '\text{INT}', 'ARRAY'</td>
</tr>
<tr>
<td>NumElements</td>
<td>UDINT</td>
<td>0</td>
<td>Number of array elements</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Requirement: The variable has the data type ARRAY.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Example: 8</td>
</tr>
<tr>
<td>BaseTypeClass</td>
<td>TYPE_CLASS</td>
<td></td>
<td>Elementary basic data type of the array elements.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Requirement: The variable has the data type ARRAY.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Example: $\text{TYPE_INT}$ for $\text{arrA : ARRAY [1..2,1..2,1..2] OF INT;}$</td>
</tr>
<tr>
<td>ElemBitSize</td>
<td>UDINT</td>
<td>0</td>
<td>Memory size of the array element (in bits)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Requirement: The variable has the data type ARRAY.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Example: $16$ bits for $\text{arrA : ARRAY [1..2,1..2,1..2] OF INT;}$</td>
</tr>
</tbody>
</table>
### MemoryArea

<table>
<thead>
<tr>
<th>Name</th>
<th>Data type</th>
<th>Initialization</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MemoryArea</td>
<td>MEMORY_AREA</td>
<td>MEM_MEMORY</td>
<td>Information about the memory area</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- MEM_GLOBAL: Global memory area</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>For example in Area 0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- MEM_LOCAL: Local memory area</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>in Area -1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- MEM_MEMORY: Marker memory area</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>%M For example in 16#10 in Area 1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- MEM_INPUT: Input memory area</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>%I For example in 16#04 in Area 2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- MEM_OUTPUT: Output memory area</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>%Q For example in 16#08 in Area 3</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- MEM_RETAIN: Retain memory area</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>For example in 16#20 in Area 0</td>
</tr>
</tbody>
</table>

Example: MEM_GLOBAL

Note: The memory area configuration is device-dependent.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Data type</th>
<th>Initialization</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Symbol</td>
<td>STRING(39)</td>
<td>&quot;</td>
<td>Variable name as STRING(39)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Example: 'iCounter', 'arrA'</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Comment</th>
<th>Data type</th>
<th>Initialization</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Comment</td>
<td>STRING(79)</td>
<td>&quot;</td>
<td>Comment of the variable declaration</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Example: 'Counts the calls' or 'Stores the A data'</td>
</tr>
</tbody>
</table>

---

**Operator '__CURRENTTASK'**

This operator is an extension of the IEC 61131-3 standard.

In runtime mode, the operator provides information about the IEC task that is currently running.

The operator is supported only on target systems in which the target system setting memory-layout\max-stack-size is set to a value > 0.

The operator allows for access to a structure with two variables:

- **TaskIndex**: Zero-based index that identifies the task
- **pTaskInfo**: Detailed information about the currently running task. It can be assigned to a POINTER TO Task_Info2 from the library CmpIecTask.

The operator cannot be used in the declaration of a POU. This would result in an error message. If the current task cannot be determined, then the TaskIndex -1 and the pTaskInfo are zero.
Example

//Declaration
VAR
   idx : INT;
   pInfo : POINTER TO Task_Info2;
END_VAR

//Program code
idx := __CURRENTTASK.TaskIndex;
pInfo := __CURRENTTASK.pTaskInfo;

Operator '__COMPARE_AND_SWAP

The multicore operator is an extension of the IEC 61131-3 standard.
The operator can be used for implementing a semaphore, for example to guarantee exclusive access to a variable written to by different tasks.

__COMPARE_AND_SWAP gets a pointer to a data type __XWORD variable, an old value, and a new value as its input (example: bMutex := __COMPARE_AND_SWAP(ADR(dwSynch), dwOld, dwNew)); The old and new values can also be data type __XWORD variables. The referenced __XWORD variable is compared with the old value and if both are equal, then the new value is written. The result of the function is TRUE when the new value could be written.

The compiler automatically replaces the data type __XWORD with DWORD on 32-bit systems and LWORD on 64-bit systems.

This operation is atomic, so it cannot be interrupted by another task, even on multicore platforms.
Example

The following example shows a typical usage. Exclusive access to a type STRING variable, which is addressed via the pstrOutput pointer, should be implemented.

The access to a string is not atomic. If multiple tasks write to the same string at the same time, then the contents may be inconsistent.

With this function, it is now possible to write the same STRING variable in different tasks.

```plaintext
FUNCTION ExclusiveStringWrite : BOOL
VAR_INPUT
    strToWrite : STRING;
    pstrOutput : POINTER TO STRING;
END_VAR
VAR_STAT
    dwSynch : __XWORD;
END_VAR
VAR
    bMutex: BOOL;
END_VAR

bMutex:= __COMPARE_AND_SWAP(ADR(dwSynch), 0, 1); (* compare with 0 and write 1 as atomic operation *)
IF bMutex THEN                         // bMutex is TRUE if write could be done
    pstrOutput^ := strToWrite;         // Now you can write safely on the string
    dwSynch := 0;                      // The __XWORD variable must be reset.
    ExclusiveStringWrite := TRUE;      // Writing was successful
ELSE
    ExclusiveStringWrite := FALSE;     // Writing was not successful
END_IF
```

See also

- ¶ “Multicore operators” on page 509
- ¶ Chapter 1.3.1.24.3.66 “Operator 'TEST_AND_SET’” on page 554

Operator ‘__XADD’

The multicore operator is an extension of the IEC 61131-3 standard.

The operator can be used for implementing an atomic counter. If an integer variable is incremented by means of ordinary addition, for example `iTest := iTest + 1;`, then this operation is not executed atomically. Another access to the variable could take place between reading and writing the variable.

If the counter is incremented in multiple tasks, then the counter result can be less than the number of counting cycles. So if two tasks execute the above code one time and the variable previously had the value 0, then the variable can then have the value 1. This is especially problematic if arrays are being processed in multiple tasks and a unique index is required for the array in each processing cycle.

When the __XADD operator is called, it gets a pointer to a type DINT variable as the first summand and a type DINT value as the second summand. __XADD returns the old value of the first summand and in the same step adds the second summand to the first summand.

For example, the function call can look like this: `diOld := __XADD(ADR(diVar), deAdd);`
The following example shows a typical usage. An array should be populated from two tasks. In the process, all positions in the array should be used and no position should be overwritten. With this function, multiple tasks can populate a Boolean array.

```plaintext
FUNCTION WriteToNextArrayPosition : BOOL
VAR_EXTERNAL
  g_diIndex : DINT;  // Index and array are globally defined and used by multiple tasks
  g_boolArray : ARRAY [0..1000] OF BOOL;
END_VAR
VAR_INPUT
  bToWrite : BOOL;
END_VAR
VAR
  diIndex : DWORD;
END_VAR

diIndex := __XADD(ADR(g_diIndex), 1);        // Returns a unique index
WriteToNextArrayPosition := FALSE;
IF (diIndex >= 0 AND diIndex <= 1000) THEN
  g_boolArray[diIndex] := bToWrite;         //Writes to unique index
  WriteToNextArrayPosition := TRUE;         // TRUE: Array was not full yet
END_IF
```

See also

● "Multicore operators" on page 509

**Operator 'TEST_AND_SET'**

The multicore operator is an extension of the IEC 61131-3 standard.

The operator can be used for implementing a semaphore, for example to guarantee exclusive access to a variable written to by different tasks.

`TEST_AND_SET` gets a type DWORD variable as its input. Write access to this variable must be possible. The variable is set to 1 and the previous value is returned as the result.

The operation is atomic, which means that it cannot be interrupted by another task. This also applies to multicore platforms.

For example, the call in the program is `dwOldValue := TEST_AND_SET(dw);`, in which the variables `dwOldValue` and `dw` must be of data type DWORD.
The following example shows a typical usage. Exclusive access to a type STRING variable, which is addressed via the pstrOutput pointer, should be implemented. The access to a string is not atomic. If multiple tasks write to the same string at the same time, then the contents may be inconsistent. With the TEST_AND_SET function, it is now possible to write the same STRING variable in different tasks.

```plaintext
FUNCTION ExclusiveStringWrite : BOOL
VAR_INPUT
    strToWrite : STRING;
    pstrOutput : POINTER TO STRING;
END_VAR
VAR_STAT
    dwSynch : DWORD;
END_VAR
VAR
    dwOldValue : DWORD;
END_VAR

    dwOldValue := TEST_AND_SET(dwSynch);  // Write the 1 and read the old value at the same time
    IF dwOldValue = 0 THEN  // 0 means: no other task is currently writing
        pstrOutput^ := strToWrite;  // Now you can write safely on the string
        dwSynch := 0;  // The DWORD must be reset
        ExclusiveStringWrite := TRUE;  // Writing was successful
    ELSE
        ExclusiveStringWrite := FALSE;  // Writing was not successful
```

See also
- § Chapter 1.3.1.24.3.64 “Operator ‘__COMPARE_AND_SWAP’” on page 552
- § “Multicore operators” on page 509

**Operator - Global namespace**

This operator is an extension of the IEC 61131-3 standard.

An instance path that begins with a dot (.) always opens a global namespace. If there is a local variable that has the same name <varname> as a global variable, then you refer to the global variable as .<varname>.

**Operator - Namespace for global variables lists**

This operator is an extension of the IEC 61131-3 standard.

You can use the name of a global variables list (GVL) as a namespace identifier for the variables that are defined in the list. This makes it possible to use variables with the same name in different global variables lists and still access specific variables uniquely. You use a dot (.) to prepend the name of the global variables list to the variable name.

<global variable list name>..<variable>

**Example**

```plaintext
globlist1.varx := globlist2.varx;
```

The globlist1 and globlist2 global variables lists each contain a varx variable. CODESYS copies the varx global variable from the globlist2 list to varx in the globlist1 list.
If you reference a variable that is declared in several global variables lists without referencing the prepended list name, then an error message is printed.

Operator - Library namespace

This operator is an extension of the IEC 61131-3 standard.

Syntax: `<library namespace>.<library identifier>

Example: `LIB_A.FB_A`

A library module identifier is appended with the library namespace (as a prefix separated by a dot) for unique and qualified access to the library module. The namespace usually coincides with the library name.

Example

A library is included in a project and contains the module `FB_A`. However, the function block with the same name is already available locally in the project. Identify the library module as `LIB_A.FB_A` in order to access the library module, not the local function block.

```plaintext
var1 := FB_A(in := 12); // Call of the project function FB_A  
var2 := LIB_A.FB_A(in := 22); // Call of the library function FB_A
```

You can define another identifier for the namespace. To do this, specify a namespace in the project information (library developers: when creating a library project). As an alternative, you can specify a specific namespace for a library in the library manager in the "Properties" dialog box (application developers: when creating an application).

See also

- `Chapter 1.3.1.19 “Using libraries” on page 371`
- `Chapter 1.3.1.25.2.14.4 “Command ‘Placeholders’” on page 1008`
- `Chapter 1.3.1.25.1.25 “Object ‘Library Manager’” on page 771`

Operator - Enumeration namespace

This operator is an extension of the IEC 61131-3 standard.

You can use the TYPE name of an enumeration for unique access to an enumeration constant. In this way, you can use the same constant names in different enumerations.

The enumeration name is prepended to the constant name with a dot (\`).

```plaintext
<enumeration name>.<constant name>
```

Example

The constant `Blue` is a component of both the enumeration `Colors` and the enumeration `Feelings`.

```plaintext
color := Colors.Blue; // Access to component blue in enumeration Colors

feeling := Feelings.Blue; // Access to component blue in enumeration Feelings
```

Operator 'INI'

The INI operator is a CoDeSys V2.3 operator. In CODESYS V3, the `FB_init` method replaces the INI operator. You can still use this operator in projects that are imported from CoDeSys V2.3.
The **INI** operator is used for initializing retain variables of a function block instance used in a POU.

Assign the operator to a Boolean variable.

**Syntax:**

```
<Boolean variable name> := INI <FB instance name>, <Boolean value>;
<Boolean value> : TRUE | FALSE
```

If the second parameter of the operator yields **TRUE**, then CODESYS initializes all retain variables that are defined in the function block `<FB instance name>`.

**Examples**

`fbinst` is the instance of the function block `fb1`, where the retain variable `retvar` is defined.

**ST:**

Declaration in the block:

```
VAR
  fbinst : fb1;
  b : BOOL;
END_VAR
```

Program part:

```
b := INI(fbinst, TRUE);
ivar := fbinst.retvar; (* => retvar is initialized *)
```

**FBD**

![FBD Diagram]

See also

- Chapter 1.3.1.24.10 “Methods ‘FB_Init’, ‘FB_Reinit’, and ‘FB.Exit’” on page 653
- Chapter 1.3.1.9.20 “Data persistence” on page 249

**1.3.1.24.4 Operands**

You declare constants with the keyword **CONSTANT**. Constants can be either local or global.

**Syntax:**

```
VAR CONSTANT <identifier>:<Type> := <initialization>; END_VAR
```

See also

- Chapter 1.3.1.24.4.1 “BOOL constants” on page 558
- Chapter 1.3.1.24.4.2 “TIME/LTIME constant” on page 558
- Chapter 1.3.1.24.4.3 “DATE constant” on page 560
- Chapter 1.3.1.24.4.4 “DATE_AND_TIME constant” on page 560
- Chapter 1.3.1.24.4.5 “TIME_OF_DAY constant” on page 561
- Chapter 1.3.1.24.4.6 “Numeric constants” on page 562
- Chapter 1.3.1.24.4.7 “REAL/LREAL constants” on page 562
- Chapter 1.3.1.24.4.8 “String constants” on page 563
- Chapter 1.3.1.24.4.9 “Typed literals” on page 564

See also

- Chapter 1.3.1.9.6 “Using input assistance” on page 208
variables

You declare variables as local or global in the declaration section of a POU or in a global variable list.

The position where you can use a variable depends on its data type.

See also

- Chapter 1.3.1.24.4.10 “Access to variables in arrays, structures, and blocks” on page 564
- Chapter 1.3.1.24.4.11 “Bit access to variables” on page 565
- Chapter 1.3.1.9.6 “Using input assistance” on page 208
- Chapter 1.3.1.9.2 “Declaration of variables” on page 172

other

- Chapter 1.3.1.24.4.12 “Addresses” on page 566
- Chapter 1.3.1.24.4.13 “Functions” on page 568

bool constants

BOOL constants are the truth values TRUE (1) and FALSE (0).

See also

- Chapter 1.3.1.24.5.1 “Data type ‘BOOL’” on page 570

Time/LTIME constant

You can use TIME constants to operate the standard timer modules. The constant has a size of 32 bits and therefore a resolution in milliseconds.

In addition, the time constant LTIME is available as a time basis for high-resolution timers. The LTIME constant has a dimension of 64 bits and therefore a resolution in nanoseconds.

Date constant

Syntax

<time keyword> # <length of time>

<time keyword> : TIME | time | T | t

<length of time> : ( <number of days>d )? ( <number of hours>h )? ( <number of minutes>m )? ( <number of seconds>s )? ( <number of milliseconds>ms )? // ( ... )? Optional

The order of time units must not be changed. However, it is not required to specify all units. It is permitted to write the units in uppercase.

Time units

- D | d: Days
- H | h: Hours
- M | m: Minutes
- s | s: Seconds
- MS | ms: Milliseconds
Examples

Correct time constants of an ST assignment

```plaintext
VAR
    timLength : TIME := T#14ms;
    timLength1 : TIME := T#100s12ms; // Overflow in the highest unit is allowed.
    timLength2 : TIME := T#12h34m15s;
    timCompare : TIME;
    xIsOK: BOOL;

    timLongest := T#49D17H2M47S295MS; // 4294967295
END_VAR

IF timLength < T#15MS THEN
    IF timCompare < timLength1 THEN
        xIsOK := TRUE;
    END_IF;
END_IF
```

Table 19: Incorrect usage:

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>timIncorrect := t#5m68s;</td>
<td>Overflow at a lower position</td>
</tr>
<tr>
<td>timIncorrect1 := 15ms;</td>
<td>Time marker T# missing</td>
</tr>
<tr>
<td>timIncorrect2 := t#4ms13d;</td>
<td>Incorrect order of time units</td>
</tr>
</tbody>
</table>

LTIME constant

Syntax

```plaintext
<long time keyword> # <length of high resolution time>

<long time keyword> : LTIME | ltime
<length of high resolution time> : <length of time> ( <number of microseconds>us )? ( <number of nanoseconds>ns )? // ( ...)? Optional
```

You can use the same units for LTIME constants as for TIME constants. Moreover, you can specify microseconds and nanoseconds because the specified time is calculated in higher time resolution. LTIME literals are treated internally as data type LWORD and therefore the value resolved in nanoseconds.

Additional time units

- US | us: Microseconds
- NS | ns: Nanoseconds

Examples of correct usage of an ST assignment:

```plaintext
PROGRAM PLC_PRG
VAR
    ltimLength := LTIME#1000d15h23m12s34ms2us44ns;
    ltimLength1 := LTIME#3445343m3424732874823ns;
END_VAR
```
DATE constant

Use DATE constants for specifying dates.

Syntax

```
$date keyword#$<year>-<month>-<day>
```

```
$date keyword$ : DATE | date | D | d
$year$ : 1970-2106
$month$ : 1-12
$day$ : 1-31
```

DATE literals are treated internally as data type DWORD, which corresponds to an upper limit of DATE#2106-2-7.

Example

```
PROGRAM PLC_PRG
VAR
    dateStart : DATE := D#2018-8-8;
    dateEnd : DATE := d#2018-8-31;
    dateCompare : DATE := DATE#1996-05-06;
    xIsDuringTheTime : BOOL;
    dateEarliest : DATE : D#1970-1-1; // = 0
    dateLatest : DATE : DATE#2106-2-7; // = 4294967295
END_VAR

IF dateStart < dateCompare THEN
    IF dateCompare < dateEnd THEN
        xIsDuringTheTime := TRUE;
    END_IF;
END_IF
```

See also

- % Chapter 1.3.1.24.5.5 “Time data types” on page 572
- % Chapter 1.3.1.24.4.4 “DATE_AND_TIME constant” on page 560
- % Chapter 1.3.1.24.4.5 “TIME_OF_DAY constant” on page 561

DATE_AND_TIME constant

DATE_AND_TIME constants are a combination of date and time.

Syntax

```
$date and time keyword#$<date and time value>
```

```
$date and time keyword$ : DATE_AND_TIME | date_and_time | DT | dt
$date and time value$ : <year>-<month>-<day>-<hour>:<minute>:<second>
$year$ : 1970-2106
```
DATE_AND_TIME literals are treated internally as data type DWORD. The time is therefore processed in seconds and can take values from January 1, 1970 00:00 and until February 7, 2106 06:28:15.

**Example**

```plaintext
PROGRAM PLC_PRG
VAR
   dtDate : DATE_AND_TIME := DATE_AND_TIME#1996-05-06-15:36:30;
   dtDate1: DATE_AND_TIME := DT#1972-03-29-00:00:00;
   dtDate2: DATE_AND_TIME := DT#2018-08-08-13:33:20.5;
   dtEarliest : DATE_AND_TIME := DATE_AND_TIME#1979-1-1-00:00:00; // 0
   dtLatest : DATE_AND_TIME := DATE_AND_TIME#2106-2-7-6:28:15; // 4294967295
END_VAR
```

See also

- § Chapter 1.3.1.24.5.5 “Time data types” on page 572
- § Chapter 1.3.1.24.4.3 “DATE constant” on page 560
- § Chapter 1.3.1.24.4.5 “TIME_OF_DAY constant” on page 561

**TIME_OF_DAY constant**

You can use this constant to define the time of day.

**Syntax**

```plaintext
<time keyword>#<time value>
```

<time keyword> : TIME_OF_DAY | time_of_day | TOD | tod
<time value> : <hour>:<minute>:<second>
<hour> : 0-23
<minute> : 0-59
<second> : 0.000-59.999

You can also specify fractions of a second. TIME_OF_DAY literals are nevertheless treated internally as DWORD and the value is resolved in milliseconds.

**Examples**

```plaintext
PROGRAM POU
VAR
   todClockTime : TIME_OF_DAY := TIME_OF_DAY#15:36:30.123;
   todEarliest : TIME_OF_DAY := TIME_OF_DAY#0:0:0.000;
   todLatest : TOD := TOD#23:59:59.999;
END_VAR
```
See also

- % Chapter 1.3.1.24.5.5 “Time data types” on page 572
- % Chapter 1.3.1.24.4.3 “DATE constant” on page 560
- % Chapter 1.3.1.24.4.4 “DATE_AND_TIME constant” on page 560

Numeric constants

Numeric values can be binary, octal, decimal, and hexadecimal numbers. If an integer value is not a decimal number, then you must write its base followed by the number sign (#) before the integer constant. You enter the hexadecimal digit values for the numbers 10 to 15 as usual with the letters A-F.

You can use an underscore within a numeric value.

<table>
<thead>
<tr>
<th>Examples:</th>
</tr>
</thead>
<tbody>
<tr>
<td>14</td>
</tr>
<tr>
<td>2#1001_0011</td>
</tr>
<tr>
<td>8#67</td>
</tr>
<tr>
<td>16#A</td>
</tr>
<tr>
<td>DINT#16#A1</td>
</tr>
</tbody>
</table>

This type of numeric value can be BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT, UDINT, REAL, or LREAL.

Implicit conversions from "larger" to "smaller" types are not permitted. You cannot simply use a DINT variable as an INT variable. For this, you have to use a type conversion function.

See also

- % Chapter 1.3.1.24.3 “Operators” on page 505
- % Chapter 1.3.1.24.4.9 “Typed literals” on page 564

As number constants basically are treated as integers, in divisions you must enter a constant in the format of a floating-point number in order not to loose the remainder. For example: Division 1/10 results in 0, division 1.0/10 results in 0.1.

REAL/LREAL constants

You can specify floating-point numbers as REAL and LREAL constants either in decimal notation or exponential notation with mantissa and exponent. The decimal point serves as the decimal separator according to the International System of Units (English).

Syntax of exponential notation

<significand> e | E <exponent>

exponent: -44..38 // REAL
exponent: -324..308 // LREAL
Table 20: REAL literal

<table>
<thead>
<tr>
<th></th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>7.4</td>
<td>Decimal number. 7.4 with a comma returns a compiler error</td>
</tr>
<tr>
<td>1/3.0</td>
<td>Decimal fraction for 0.333333343</td>
</tr>
<tr>
<td></td>
<td>Note: In the case of division of integer types, the result remains an integer type. In this case, the value is rounded. For example, 1/3 yields 0 as the result.</td>
</tr>
<tr>
<td>1.64e+009</td>
<td>Exponential notation</td>
</tr>
<tr>
<td>-3.402823e+38</td>
<td>Smallest number</td>
</tr>
<tr>
<td>-1E-44</td>
<td>Largest negative number</td>
</tr>
<tr>
<td>1.0E-44</td>
<td>Smallest positive number</td>
</tr>
<tr>
<td>3.402823e+38</td>
<td>Largest number</td>
</tr>
</tbody>
</table>

Table 21: LREAL literal

<table>
<thead>
<tr>
<th></th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-1.7976931348623157E+308</td>
<td>Smallest number</td>
</tr>
<tr>
<td>-4.94065645841247E-324</td>
<td>Largest negative number</td>
</tr>
<tr>
<td>4.94065645841247E-324</td>
<td>Smallest positive number</td>
</tr>
<tr>
<td>1.7976931348623157E+308</td>
<td>Largest number</td>
</tr>
</tbody>
</table>

See also

- Chapter 1.3.1.24.5.3 “Data type ‘REAL’ / ‘LREAL’” on page 570

String constants

A string constant is a character string enclosed in single straight quotation marks. The characters are coded according to the character set specified in ISO/IEC 8859-1. Therefore, a string constant can include spaces and accented characters, as these belong to this character set. This is also referred to as a string literal, or simply a string.

Example: 'Hello world!'

When a dollar sign ($) is in a string constant, the following two characters are interpreted as a hexadecimal code according to the coding in ISO/IEC 8859-1. The code also corresponds to ASCII code. In addition, please note the special cases.

Table 22: Hexadecimal code

<table>
<thead>
<tr>
<th>String with $ code</th>
<th>Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>'$&lt;8-bit code&gt;'</td>
<td>8-bit code: Two-digit hexadecimal number that is interpreted according to ISO/IEC 8859-1.</td>
</tr>
<tr>
<td>'$41'</td>
<td>A</td>
</tr>
<tr>
<td>'$9A'</td>
<td>©</td>
</tr>
<tr>
<td>'$40'</td>
<td>@</td>
</tr>
<tr>
<td>'$0D'</td>
<td>Control character: Line break (corresponds to '$R')</td>
</tr>
<tr>
<td>'$0A'</td>
<td>Control character: New line (corresponds to '$L' and '$N')</td>
</tr>
</tbody>
</table>
Table 23: Special cases

<table>
<thead>
<tr>
<th>String with $ code</th>
<th>Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>'$L',  '$l'</td>
<td>Control character: Line feed (corresponds to '$0A')</td>
</tr>
<tr>
<td>'$N',  '$n'</td>
<td>Control character: New line (corresponds to '$0A')</td>
</tr>
<tr>
<td>'$P',  '$p'</td>
<td>Control character: Form feed</td>
</tr>
<tr>
<td>'$R',  '$r'</td>
<td>Control character: Line break (corresponds to '$0D')</td>
</tr>
<tr>
<td>'$T',  '$t'</td>
<td>Control character: Tab</td>
</tr>
<tr>
<td>'$&quot;'</td>
<td>Dollar sign: $</td>
</tr>
<tr>
<td>'$'</td>
<td>Single straight quotation mark: '</td>
</tr>
</tbody>
</table>

Example

Constant declaration

```python
VAR CONSTANT
  constA : STRING := 'Hello world';
  constB : STRING := 'Hello world $21'; // Hello world!
END_VAR
```

Typed literals

With the exception of REAL/LREAL constants (LREAL is always used here), CODESYS uses the smallest possible data type when calculating with IEC constants. If you want to use another data type, then you can use typed literals without having to declare the constants explicitly. When doing this, provide the constants with a prefix that indicates the type.

**Syntax:**

```plaintext```
<type>#<literal>
```

- `<type>` defines the desired data type; possible values: BOOL, SINT, USINT, BYTE, INT, UINT, WORD, DINT, UDINT, DWORD, REAL, LREAL. You must capitalize the entire type name.

- `<literal>` defines the constants. The entry must match the data type defined in `<Type>`.

**Example:**

```plaintext```
var1 := DINT#34;
```

If CODESYS cannot convert the constant into the target type without data loss, then an error message is issued.

You can use typed constants wherever you can use normal constants.

Access to variables in arrays, structures, and blocks

**Syntax for access to**

- **Two-dimensional array components:** `<array name>[index1, index2]`
- **Structural variables:** `<structure name>.<variable name>`
- **Function block and program variables:** `<function block name>.<variable name>`

**See also**

- % Chapter 1.3.1.24.5.14 “Data type ’ARRAY’” on page 581
- % Chapter 1.3.1.24.5.16 “Structure” on page 594
- % Chapter 1.3.1.25.1.28.2 “Object ’Function Block’” on page 778
- % Chapter 1.3.1.25.1.28.1 “Object ’Program’” on page 777
Bit access to variables

**NOTICE!**
Implement concurrent bit access by two tasks only if the processor can execute bit access directly on the memory. All x86 and x64 systems have commands for bit access in memory. Systems such as ARM and PPC cannot access bits directly in the memory.

If two tasks execute bit access simultaneously, even though the processor cannot perform bit access directly in the memory, then proceed as follows. Use a semaphore (SysSemEnter) or a similar technique to prevent competing bit access. However, it is best to execute the bit access within a task.

**Syntax for declarations:**
```plaintext
<variable name> : <data type> ( := <initialization> )? ;
```
```
<data type> = BYTE | WORD | DWORD | LWORD | SINT | USINT | INT | DINT | UDINT | LINT | ULINT
```

**Syntax for calls:**
With a bit access, a single bit is accessed with read or write permissions within an integer variable. To do this, add a period and an index to the variable identifier that addresses the bit within the variable. Indexing is 0-based. The index is a positive integer starting at 0. You can also specify the index as a constant variable.

**Example**
```
PROGRAM PLC_PRG
VAR
  wA : WORD := 16#FF;
  xB : BOOL := 0;
END_VAR
wA.2 := xB;
```

In the program, the third bit of the variable `wA` is set to the value of variable `xB`.

```
wA = 2#1111_1111_1111_1011 = 16#FFFB
```

**Example: Bit access to a variable with a global constant variable**
You can use a global constant as an index to access a bit within an integer variable.

**Declaration in a global variable list:**
```
VAR_GLOBAL CONSTANT
c_usiENABLE : USINT := 2;
END_VAR
```

The constant `c_usiENABLE` addresses the bit in the integer variable.

**Declaration of a POU:**
```
VAR
  iX : INT;
END_VAR
```

**Call with bit access:**
```
iX.c_usiENABLE := TRUE; // The third bit in the variable iX is set TRUE
```
Bit access with BIT data types

With the BIT data type, you can combine single bits into a structure and then access them individually. The bit is then addressed with the component name.

Example:

Access to BIT data types

Declaration of the structure:

```plaintext
TYPE ControllerData :
  STRUCT
    Status_OperationEnabled : BIT;
    Status_SwitchOnActive : BIT;
    Status_EnableOperation : BIT;
    Status_Error : BIT;
    Status_VoltageEnabled : BIT;
    Status_QuickStop : BIT;
    Status_SwitchOnLocked : BIT;
    Status_Warning : BIT;
  END_STRUCT
END_TYPE
```

Declaration of the POU:

```plaintext
VAR
  ControllerDrive_1 : ControllerData;
END_VAR
```

Bit access:

```plaintext
ControllerDrive1.OperationEnabled := TRUE;
```

See also

- § Chapter 1.3.1.24.5.2 “Integer data types” on page 570
- § “Bit access in structures” on page 595
- § Chapter 1.3.1.24.5.10 “Data type ‘BIT’” on page 577

Addresses

CAUTION!
If you use pointers to addresses, then the contents of addresses can be moved during an online change. If you use absolute addresses, then the contents of addresses does not change during an online change.

Syntax:

```plaintext
%<memory area prefix> ( <size prefix> )? <memory position>
```

- **<memory area prefix>** : I | Q | M
- **<size prefix>** : X | B | W | D
- **<memory position>** : <number> ( .<number> )* // Depends on the target system

When defining an address, you use specific character strings to express memory position and size. An address is marked with the percent sign (%), followed by the memory range prefix, the optional size prefix, and the memory range position. The numbering that you use for addressing the memory position depends on the target system.
### Memory Range Prefix

<table>
<thead>
<tr>
<th>Prefix</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>Input memory range for &quot;Inputs&quot;&lt;br&gt;For physical inputs via input drivers, &quot;Sensors&quot;</td>
</tr>
<tr>
<td>Q</td>
<td>Output memory range for &quot;Outputs&quot;&lt;br&gt;For physical outputs via output drivers, &quot;Actuators&quot;</td>
</tr>
<tr>
<td>M</td>
<td>Flag memory range</td>
</tr>
</tbody>
</table>

### Size Prefix

<table>
<thead>
<tr>
<th>Size Prefix</th>
<th>Data Type</th>
<th>Data Width</th>
</tr>
</thead>
<tbody>
<tr>
<td>No size prefix</td>
<td>Single bit</td>
<td></td>
</tr>
<tr>
<td>X</td>
<td>Single bit</td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>BYTE</td>
<td>8 bits</td>
</tr>
<tr>
<td>W</td>
<td>WORD</td>
<td>16 bits</td>
</tr>
<tr>
<td>D</td>
<td>DWORD</td>
<td>32 bits</td>
</tr>
</tbody>
</table>

### Examples

- `%QX7.5`<br>Single bit address of the output bit 7.5
- `%Q7.5`<br>Word address of the input word 215
- `%IW215`<br>Byte address of the output byte 7
- `%MD48`<br>Address of a double word at memory position 48 in flag memory
- `%IW2.5.7.1`<br>Word address of an input word; interpretation dependent on the current controller configuration
- `VAR wVar AT %IW0 : WORD; END_VAR`<br>Variable declaration with address information of an input word
- `VAR xActuator AT %QW0 : BOOL; END_VAR`<br>Boolean variable declaration<br>Note: For Boolean variables, one byte is allocated internally if a single bit address is not specified. A change in the value of `xActuator` affects the range from QX0.0 to QX0.7.
- `VAR xSensor AT IX7.5 : BOOL; END_VAR`<br>Boolean variable declaration with explicit specification of a single bit address. On access, only the input bit 7.5 is read.

### Memory position

Make sure that the address is valid as follows:

To map a valid address in an application, you must know the required position (applicable memory range) in the process image: input memory range (I), output memory range (Q), and flag memory range (M) — see above. Furthermore, you have to specify the required size prefix: bit, BYTE, WORD, DWord (see above: X, B, W, D)
The current device configuration and device settings (hardware structure, device description, I/O settings) play a decisive part. Note specifically the differences in the interpretation of bit addresses for devices with "byte addressing mode" and devices with "word-oriented IEC addressing mode". For example, in a byte addressing device, the number before the point of bit address %IX5.5 addresses byte 5. On the other hand, in a word-addressed device, it addresses word 5. In contrast, addressing with a word or byte address is independent of the device type: with %IW5 always word 5 is addressed and with byte address %IB5 always byte 5. Regardless of size and addressing mode, you can address different memory cells therefore with the same address information.

The following table shows the comparison of byte addressing and word-oriented IEC addressing for bits, bytes, words, and double words. It also shows the overlapping memory ranges that are present in the case of byte addressing (see also the example below the table).

Regardless of size and addressing mode, you can address different memory cells therefore with the same address information.

<table>
<thead>
<tr>
<th>DWords / Words</th>
<th>Bytes</th>
<th>X (bits)</th>
</tr>
</thead>
<tbody>
<tr>
<td>byte addressing</td>
<td>word oriented IEC addressing</td>
<td>byte addressing</td>
</tr>
<tr>
<td>D0</td>
<td>W0</td>
<td>W0</td>
</tr>
<tr>
<td>D1</td>
<td>W1</td>
<td>W1</td>
</tr>
<tr>
<td>...</td>
<td>W2</td>
<td>W2</td>
</tr>
<tr>
<td>...</td>
<td>W3</td>
<td>W3</td>
</tr>
<tr>
<td>...</td>
<td>D1</td>
<td>W2</td>
</tr>
<tr>
<td>...</td>
<td>W3</td>
<td>W3</td>
</tr>
<tr>
<td>...</td>
<td>D2</td>
<td>W4</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>...</td>
<td>D(n-3)</td>
<td>D(n/4)</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>...</td>
<td>D(n)</td>
<td>D(n/4)</td>
</tr>
</tbody>
</table>

n = byte number

Example of memory range overlapping in the case of the byte addressing mode

D0 contains B0 - B3, W0 contains B0 and B1, W1 contains B1 and B2, and W2 contains B2 and B3. Consequently, in order to avoid overlap, you must not use W1 (also D1, D2, and D3) for addressing.

See also

- § Chapter 1.3.1.9.12.2 “AT declaration” on page 228

Functions

In ST, you can use a function call as an operand.

Example:

```
Result := Fct(7) + 3;
```

TIME() function

This function yields the time (in milliseconds) that has elapsed since system boot.
The data type is TIME.

Example in ST:

```plaintext
systime := TIME();
```

See also

- Chapter 1.3.1.25.1.28.3 “Object ‘Function’” on page 781

1.3.1.24.5 Data types

When programming in CODESYS, you can use different data types or instances of function blocks. Moreover, you assign a data type to every identifier. The data type defines how much memory CODESYS reserves and how these values are interpreted.

The following group of data types are provided:

**Standard data types**

CODESYS supports all IEC 61131-3 data types.

- Chapter 1.3.1.24.5.1 “Data type ‘BOOL’” on page 570
- Chapter 1.3.1.24.5.2 “Integer data types” on page 570
- Chapter 1.3.1.24.5.3 “Data type ‘REAL’ / ‘LREAL’” on page 570
- Chapter 1.3.1.24.5.4 “Data type ‘STRING’” on page 571
- Chapter 1.3.1.24.5.9 “Data type ‘WSTRING’” on page 577
- Chapter 1.3.1.24.5.5 “Time data types” on page 572
- Chapter 1.3.1.24.5.8 “Data type ‘LTIME’” on page 576

**Extensions to the IEC 61131-3 standard**

- Chapter 1.3.1.24.5.6 “Data type ‘UNION’” on page 572
- Chapter 1.3.1.24.5.10 “Data type ‘BIT’” on page 577
- Chapter 1.3.1.24.5.11 “Special data types ‘__UXINT’ and ‘__XWORD’” on page 577
- Chapter 1.3.1.24.5.13 “Reference” on page 579
- Chapter 1.3.1.24.5.12 “Pointers” on page 577

**User-defined data types**

You can create user-defined data types in two ways:

- DUT object in the POU window
- data type in a block in the declaration

**NOTICE!**

Please note the recommendations for naming and object.

- Chapter 1.3.1.24.5.14 “Data type ‘ARRAY’” on page 581
- Chapter 1.3.1.24.5.16 “Structure” on page 594
- Chapter 1.3.1.24.5.17 “Enumerations” on page 595
- Chapter 1.3.1.24.5.13 “Reference” on page 579
- Chapter 1.3.1.24.5.12 “Pointers” on page 577
- Chapter 1.3.1.24.5.18 “Subrange types” on page 599
Data type 'BOOL'

<table>
<thead>
<tr>
<th>Data Type</th>
<th>Values</th>
<th>Memory</th>
</tr>
</thead>
<tbody>
<tr>
<td>BOOL</td>
<td>TRUE (1), FALSE (0)</td>
<td>8 bit</td>
</tr>
</tbody>
</table>

See also
- Chapter 1.3.1.24.4.1 “BOOL constants” on page 558

Integer data types

CODESYS provides the following integer data types.

<table>
<thead>
<tr>
<th>Data Type</th>
<th>Lower Limit</th>
<th>Upper Limit</th>
<th>Memory</th>
</tr>
</thead>
<tbody>
<tr>
<td>BYTE</td>
<td>0</td>
<td>255</td>
<td>8 bit</td>
</tr>
<tr>
<td>WORD</td>
<td>0</td>
<td>65535</td>
<td>16 bit</td>
</tr>
<tr>
<td>DWORD</td>
<td>0</td>
<td>4294967295</td>
<td>32 bit</td>
</tr>
<tr>
<td>LWORD</td>
<td>0</td>
<td>264-1</td>
<td>64 bit</td>
</tr>
<tr>
<td>SINT</td>
<td>-128</td>
<td>127</td>
<td>8 bit</td>
</tr>
<tr>
<td>USINT</td>
<td>0</td>
<td>255</td>
<td>8 bit</td>
</tr>
<tr>
<td>INT</td>
<td>-32768</td>
<td>32767</td>
<td>16 bit</td>
</tr>
<tr>
<td>UINT</td>
<td>0</td>
<td>65535</td>
<td>16 bit</td>
</tr>
<tr>
<td>DINT</td>
<td>-2147483648</td>
<td>2147483647</td>
<td>32 bit</td>
</tr>
<tr>
<td>UDINT</td>
<td>0</td>
<td>4294967295</td>
<td>32 bit</td>
</tr>
<tr>
<td>LINT</td>
<td>-263</td>
<td>263-1</td>
<td>64 bit</td>
</tr>
<tr>
<td>UUINT</td>
<td>0</td>
<td>264-1</td>
<td>64 bit</td>
</tr>
</tbody>
</table>

NOTICE!
Information can be lost when converting from larger to smaller types.

See also
- Chapter 1.3.1.24.4.6 “Numeric constants” on page 562

Data type 'REAL' / 'LREAL'

The data types REAL and LREAL are floating-point types according to IEEE 754. They are necessary when using decimal numbers and floating-point numbers in decimal notation or exponential notation.
## Table 24: Target system: CODESYS Control Win V3

<table>
<thead>
<tr>
<th>Data type</th>
<th>Smallest value number</th>
<th>Largest value number</th>
<th>Storage space</th>
</tr>
</thead>
<tbody>
<tr>
<td>REAL</td>
<td>1.0E-44</td>
<td>3.402823E+38</td>
<td>32 bit</td>
</tr>
<tr>
<td>LREAL</td>
<td>4.94065645841247E-32</td>
<td>1.7976931348623157E+308</td>
<td>64 bit</td>
</tr>
</tbody>
</table>

### Example

```plaintext
PROGRAM PLC_PRG
    VAR
        rMax: REAL := 3.402823E+38; // Largest number
        rPosMin : REAL := 1.0E-44; // Smallest positive number
        rNegMax: REAL := -1.0E-44; // Largest negative number
        rMin: REAL := -3.402823E+38; // Smallest number
        lrMax: LREAL := 1.7976931348623157E+308; // Largest number
        lrPosMin : LREAL := 4.94065645841247E-32; // Smallest positive number
        lNegMax: LREAL := -4.94065645841247E-32; // Largest negative number
        lrMin: LREAL := -1.7976931348623157E+308; // Smallest number
    END_VAR
```

### NOTICE!

Support for the LREAL data type depends on the target device in use. Refer to the respective documentation as to whether or not the 64-bit type LREAL is converted to REAL or remains as LREAL when compiling the application. Conversion may result in the loss of information.

### NOTICE!

If the value of the REAL/LREAL number is outside of the value range of the integer, then an undefined result is yielded from a data type conversion from REAL or LREAL to SINT, USINT, INT, UINT, DINT, UDINT, LINT, or ULINT. The result depends on the target system. An exception error is also possible. To get code that is independent of the target system, the application must catch value range violations.

If the REAL/LREAL number is within the value range of the integer data type, then the conversion operates the same way on all systems.

See also
- “REAL/LREAL constants” on page 562

### Data type ‘STRING’

A variable of the STRING data type can include any character string. The amount of memory that is allocated during a declaration refers to characters and is shown in parentheses or brackets. If a size is not defined, then CODESYS allocates 80 characters by default.

As a rule, CODESYS does not limit the string length; however, the string function processes lengths from 1 to 255 only. If a variable is initialized with a string that is too long for the data type, then CODESYS truncates the string accordingly from the right.
NOTICE!
The memory required for a STRING variable is always one byte per character plus one additional byte (for example, 81 bytes for a STRING[80] declaration).

Example of a string declaration with 35 characters:

```
str : STRING(35):= 'This is a String';
```

See also

- § Chapter 1.3.1.24.4.8 “String constants” on page 563
- § Chapter 1.3.1.24.5.9 “Data type 'WSTRING'” on page 577

Time data types

Time data types are treated internally as DWORD. TIME and TIME_OF_DAY are resolved in milliseconds, and DATE_AND_TIME is resolved in seconds.

<table>
<thead>
<tr>
<th>Data type</th>
<th>Lower limit</th>
<th>Upper limit</th>
<th>Storage space</th>
<th>Resolution</th>
</tr>
</thead>
<tbody>
<tr>
<td>TIME</td>
<td>T#0d0h0m0s0ms</td>
<td>T#49d17h2m47s295ms</td>
<td>32 bit</td>
<td>Milliseconds</td>
</tr>
<tr>
<td>TIME_OF_DAY</td>
<td>TOD#0:0:0.000 00:00:00.000</td>
<td>TOD#23:59:59.99 23:59:59.999</td>
<td>32 bit</td>
<td>Milliseconds</td>
</tr>
<tr>
<td>DATE</td>
<td>D#1970-1-1 01/01/70</td>
<td>DATE#2106-2-7 06:28:15</td>
<td>32 bit</td>
<td>Day</td>
</tr>
<tr>
<td>DATE_AND_TIME</td>
<td>DT#1979-1-1-00:00 01/01/1970</td>
<td>DT#2106-2-7-6:28:15</td>
<td>32 bit</td>
<td>Seconds</td>
</tr>
</tbody>
</table>

See also

- § Chapter 1.3.1.24.4.5 “Data type 'LTIME'' on page 576
- § Chapter 1.3.1.24.4.2 “TIME/LTIME constant” on page 558
- § Chapter 1.3.1.24.4.3 “DATE constant” on page 560
- § Chapter 1.3.1.24.4.4 “DATE_AND_TIME constant” on page 560
- § Chapter 1.3.1.24.4.5 “TIME_OF_DAY constant” on page 561

Data type 'UNION'

A UNION is a data structure that usually contains different data types.

In a union, all components have the same offset and therefore the same amount of memory. In the following declaration example of a union, an assignment to `name.a` will also affect `name.b`.
Example

```plaintext
TYPE name:
UNION
    a : LREAL;
    b : LINT;
END_UNION
END_TYPE
```

Data type ‘ANY’ and ‘ANY_<type>’

When implementing a function or a method, you can declare inputs (VAR_INPUT) as variables with a generic IEC data type (ANY or ANY_<type>). As a result, you can implement calls that have different call parameters for the data type.

At runtime, you can retrieve the transferred value and its type by means of a predefined structure within the programming block for the input variable.

With compiler versions > 3.5.1.0, the generic IEC data types listed below are supported. The table shows the generic data type that permits elementary data types.

<table>
<thead>
<tr>
<th>Hierarchy of Generic Data Types</th>
<th>Elementary Data Types</th>
</tr>
</thead>
<tbody>
<tr>
<td>ANY</td>
<td>ANY_BIT</td>
</tr>
<tr>
<td>ANY_DATE</td>
<td>ANY_REAL</td>
</tr>
<tr>
<td>ANY_NUM</td>
<td>ANY_REAL</td>
</tr>
<tr>
<td>ANY_INT</td>
<td>ANY_INT</td>
</tr>
<tr>
<td>ANY_STRING</td>
<td></td>
</tr>
<tr>
<td>STRING</td>
<td></td>
</tr>
</tbody>
</table>
The transfer parameters of the function calls have different data types.

```plaintext
FUNCTION AnyBitFunc : BOOL
  VAR_INPUT
    value : ANY_BIT;
  END_VAR
FUNCTION AnyDateFunc : BOOL
  VAR_INPUT
    value : ANY_DATE;
  END_VAR
FUNCTION AnyFunc : BOOL
  VAR_INPUT
    value : ANY;
  END_VAR
FUNCTION AnyIntFunc : BOOL
  VAR_INPUT
    value : ANY_INT;
  END_VAR
FUNCTION AnyNumFunc : BOOL
  VAR_INPUT
    value : ANY_NUM;
  END_VAR
FUNCTION AnyRealFunc : BOOL
  VAR_INPUT
    value : ANY_REAL;
  END_VAR
FUNCTION AnyStringFunc : BOOL
  VAR_INPUT
    value : ANY_STRING;
  END_VAR

PROGRAM PLC_PRG
  VAR
    xValue : BOOL := TRUE;
    byValue : BYTE := 16#AB;
    wValue : WORD := 16#1234;
    dwValue : DWORD := 16#6789ABCD;
    lwValue : LWORD := 16#0123456789ABCDEF;
    sValue : STRING := 'xyz';
    wsValue : WSTRING := "abc";
    dtValue : DATE_AND_TIME := DT#2017-02-20-11:07:00;
    dValue : DATE := D#2017-02-20;
    todValue : TIME_OF_DAY := TOD#11:07:00;
    rValue : REAL := 42.24;
    lrValue : LREAL := 24.42;
    usiValue: USINT := 12;
    uiValue : UINT := 1234;
    udiValue : UDINT := 12345;
    uliValue : ULINT := 123456;
    siValue: SINT := -12;
    iValue : INT := -1234;
    diValue : DINT := -12345;
    liValue : LINT := -123456;
  END_VAR

  AnyFunc(xValue);
  AnyFunc(byValue);
  AnyFunc(wValue);
  AnyFunc(dwValue);
  AnyFunc(lwValue);
  AnyFunc(sValue);
  AnyFunc(wsValue);
  AnyFunc(dtValue);
  AnyFunc(todValue);
```
When compiling the code, an ANY data type is replaced internally with the following structure:

```plaintext
TYPE AnyType :
  STRUCT
    // the type of the actual parameter
typeclass : __SYSTEM.TYPE_CLASS ;
    // the pointer to the actual parameter
pvalue : POINTER TO BYTE;
    // the size of the data, to which the pointer points
diSize : DINT;
  END_STRUCT
  END_TYPE
```

The actual call parameter will assign the structure elements at runtime.
Example

This compares whether or not the two passed variables have the same type and the same value.

```
FUNCTION Generic_Compare : BOOL
VAR_INPUT
  any1 : ANY;
  any2 : ANY;
END_VAR
VAR
  pTest : POINTER TO ARRAY [0..100] OF POINTER TO DWORD;
  icount: DINT;
END_VAR

pTest := ADR(any1);
Generic_Compare := FALSE;
IF any1.typeclass <> any2.typeclass THEN
  RETURN;
END_IF
IF any1.diSize <> any2.diSize THEN
  RETURN;
END_IF
// Byte comparison
FOR icount := 0 TO any1.diSize-1 DO
  IF any1.pvalue[iCount] <> any2.pvalue[iCount] THEN
    RETURN;
  END_IF
END_FOR
Generic_Compare := TRUE;
RETURN;
// END_FUNCTION
```

Data type 'LTIME'

You can use the data type LTIME as a time base for high-resolution timer. A high-resolution timer has a resolution in nanoseconds.

<table>
<thead>
<tr>
<th>Data type</th>
<th>Lower limit</th>
<th>Upper limit</th>
<th>Storage space</th>
</tr>
</thead>
<tbody>
<tr>
<td>LTIME</td>
<td>0</td>
<td>213503d23h34m33s709ms551us615ns</td>
<td>64 bit</td>
</tr>
</tbody>
</table>

Syntax:

```
LTIME#<time declaration>
```

The time declaration can include units of time that apply for the TIME constant as well as:

- "us": microseconds
- "ns": nanoseconds

Example:

```
LTIME1 := LTIME#1000d15h23ms2us44ns
```

See also

- Chapter 1.3.1.24.5.5 “Time data types” on page 572
Data type 'WSTRING'

The data type WSTRING is interpreted in Unicode format as opposed to the data type STRING (ASCII). As a result of this coding, the number of displayed characters for WSTRING depends on the characters. A length of 10 for WSTRING means that the length of the WSTRING can take a maximum of 10 WORDs. However, for some characters in Unicode, multiple WORDs are required for coding a character so that the number of characters do not have to correspond to the length of the WSTRING (10 in this case). The data type requires 1 WORD of memory per character plus 1 WORD of extra memory. Each STRING requires only 1 byte. The data type WSTRING is terminated with a 0.

Example:  

```plaintext
wstr : WSTRING := "This is a WString";
```

See also

- § Chapter 1.3.1.24.5.4 “Data type 'STRING'” on page 571
- § Chapter 1.3.1.24.4.8 “String constants” on page 563

Data type 'BIT'

You can use the BIT data type only for single variables in structures or in a function block. The possible values are TRUE (1) and FALSE (0).

A BIT element requires 1 bit of memory. Therefore, you can reference single bits of a structure by name. BIT elements that are declared in succession are bundled in bytes. In this way, you can optimize memory use as opposed to BOOL types, which reserve 8 bits each. On the other hand, bit access is significantly more time-consuming. Therefore, you should use the BIT data type only when you need to define data in a predefined format.

See also

- § Chapter 1.3.1.24.5.16 “Structure” on page 594

Special data types '__UXINT' and '__XWORD'

CODESYS supports systems with address registers of 32-bit and 64-bit widths. For making the IEC code as independent from the target system as possible, you use the pseudo data types __UXINT and __XWORD. The compiler checks which target system types are current and then converts these data types into the appropriate standard data types.

- __UXINT is converted into ULINT on 64-bit platforms and UDINT on 32-bit platforms.
- __XWORD is converted into LWORD on 64-bit platforms and DWORD on 32-bit platforms.

Pointers

Pointers store the addresses of variables, programs, function blocks, methods, and functions while an application program is running. A pointer points to one of the objects mentioned or to a variable with any data type.

**Syntax of a pointer declaration:**

```plaintext
<identifier>: POINTER TO <data type | function block | program | method | function>;
```

When dereferencing a pointer, the value of the address to which the pointer points is determined. In order to dereference a pointer, append the content operator to the pointer identifier (see `pt^` in the example below).

Using the address operator ADR, you can assign the address of a variable to a pointer.
**Example**

```plaintext
VAR
    pt:POINTER TO INT;  (* declaration of pointer pt *)
    var_int1:INT := 5;  (* declaration of variables var_int1 and var_int2 *)
    var_int2:INT;
END_VAR

pt := ADR(var_int1); (* pointer pt is assigned to address of var_int1 *)
var_int2:= pt^;      (* value 5 of var_int1 is assigned to variable var_int2 by dereferencing of pointer pt *)
```

**NOTICE!**

If a pointer to a device input is used, then the access (for example `pTest := ADR(input);`) is considered to be a write access. This leads to a compiler warning when code is generated: "...invalid assignment target".

If you require a construct of this kind, you must first copy the input value (`input`) to a variable with write access.

In online mode, you can jump from a pointer to the declaration location of the referenced variable by clicking the "Go To Reference" command.

See also

- § Chapter 1.3.1.24.3.31 “Operator ‘Content Operator’” on page 524
- § Chapter 1.3.1.24.3.30 “Operator ‘ADR’” on page 524
- § Chapter 1.3.1.25.2.2.37 “Command ‘Go To Reference’” on page 876

**Function pointers to external functions**

CODESYS supports function pointers that replace the `INDEXOF` operator. You can pass these pointers to external libraries. However, CODESYS does not provide any means for calling a function pointer from within an application in the development system. The function of the runtime system for the registration of callback functions (system library function) expects the function pointer. Depending on which callback was registered, the runtime system implicitly calls the function concerned (in the case of STOP for example). So that such a system call is possible (runtime system), you must set the corresponding object property in the “Build” tab.

You can use the `ADR` operator for functions, programs, function blocks, and methods. CODESYS outputs the address of a pointer to the function, not the address of the function, because the values of functions can change after an online change. This address is valid as long as the function exists on the target system.

See also

- § Chapter 1.3.1.25.3.10.4 “Dialog ‘Properties’ - ‘Build’” on page 1048
- § Chapter 1.3.1.24.3.30 “Operator ‘ADR’” on page 524

**Index access to pointers**

In CODESYS, index access `"[]"` to type `POINTER`, `STRING`, and `WSTRING` variables is permitted.

- `pint[i]` returns the basic data type
- Index access to pointers is done arithmetically: If you use index access for a `POINTER TO` variable, then CODESYS computes the offset by `pint[i] = (pint + i * SIZEOF(base type))^`. The index access also causes an implicit dereferencing of the pointer. The resulting data type is the basic data type of the pointer. Note that `pint[7] != (pint + 7)^`. 
● When you use the index access with a variable of the type STRING, you get the character at the offset of the index expression. The result is a BYTE type. str[i] returns the i-th character of the string as SINT (ASCII).

● When you use the index access with a variable of the type WSTRING, you get the character at the offset of the index expression. The result is a WORD type. wstr[i] returns the i-th character of the string as INT (Unicode).

The result of the difference between two pointers is of type DWORD, even on 64-bit platforms, when the pointers are 64-bit pointers.

Note that it is possible to use references which control a value directly, in contrast to pointers.

Note that it is possible to monitor the memory access of pointers at runtime by the implicit monitoring function CheckPointer

See also
● Chapter 1.3.1.25.1.29.10 “POU 'CheckPointer'” on page 809

---

Reference

A REFERENCE is also a pointer, but it has some advantages over a POINTER:

● Ease of use: The reference does not have to be dereferenced explicitly (with ^) in order to access the contents of the referenced object.

● Nicer syntax for passing of values: If an input is a REFERENCE TO, then you do not have to write ADR(value) explicitly for a(refInput := value).

● Type safety: Unlike the pointer, for references the compiler checks that the base type is the same when assigning two references

You declare a reference in accordance with the following syntax:

Syntax

<identifier> : REFERENCE TO <data type>

Example

A : REFERENCE TO DUT;
B : DUT;
C : DUT;

A REF= B; // corresponds to A := ADR(B);
A := C; // corresponds to A^ := C;
You cannot declare references in the following ways:

- REFERENCE TO REFERENCE
- or ARRAY OF REFERENCE
- or POINTER TO REFERENCE.

Furthermore, you cannot declare a reference to a bit variable.

NOTICE!
CODESYS initializes references (at 0) with compiler version >= V3.3.0.0.

NOTICE!
When a reference is used for a device entry, write access is enabled (for example, ref REF= input;). This leads to a compiler warning when code is generated: "...invalid assignment target".

If you require this kind of construct, then first you must copy the input value (input) to a variable with write access.

Checking for valid references

You can use the operator __ISVALIDREF in order to check whether a reference points to a valid value, i.e. to a value not equal to 0.

Syntax:

<Boolean variable> := __ISVALIDREF(<with REFERENCE TO <data type> declared identifier>);

<Boolean variable> is TRUE when the reference points to a valid value; otherwise FALSE.

Example

Declaration:

ivar : INT;
ref_int : REFERENCE TO INT;
ref_int0 : REFERENCE TO INT;
testref : BOOL := FALSE;

Implementation:

ivar := ivar + 1;
ref_int REF= ivar;
ref_int0 REF= 0;
testref := __ISVALIDREF(ref_int); (* becomes TRUE, because ref_int points to ivar, which is non-zero *)
testref := __ISVALIDREF(ref_int0); (* becomes FALSE, because ref_int0 is set to 0 *)

In compiler version 3.5.7.40 and later, the implicit monitoring function "CheckPointer" also affects REFERENCE variables the same as pointer variables.

See also

- § Chapter 1.3.1.24.1.3.4.6 “Assignment operator ‘REF’” on page 435
- § Chapter 1.3.1.25.1.29.10 “POU ‘CheckPointer’” on page 809
Data type 'ARRAY'

An array is a collection of data elements of the same data type. CODESYS supports one- and multi-dimensional arrays of fixed or variable length.

Array of fixed length

You can define arrays in the declaration part of a POU or in global variable lists.

Syntax of the declaration of a one-dimensional array

```plaintext
<variable name> : ARRAY[<dimension>] OF <data type> ( := <initialization> )? ;
```

- `<variable name>`: Name of the variable.
- `<dimension>`: Lower index bound..<upper index bound>.
- `<data type>`: Elementary data types | User defined data types | Function block types.
- `( := <initialization> )?`: Optional.

Syntax of the declaration of a multi-dimensional array

```plaintext
<variable name> : ARRAY[<1st dimension>, <next dimension>+ ] OF <data type> ( := <initialization> )? ;
```

- `<variable name>`: Name of the variable.
- `<1st dimension>`: <1st lower index bound>..<1st upper index bound>.
- `<next dimension>`: <next lower index bound>..<next upper index bound>.
- `<data type>`: Elementary data types | User defined data types | Function block types.
- `( := <initialization> )?`: Optional.

The index limits are integers; maximum of the data type DINT.

Syntax for data access

```plaintext
<variable name>[<index of 1st dimension>, <index of next dimension>]*
```

- `( <index of 1st dimension>, <index of next dimension> )*`: 0, one or more further dimensions.

Note the capability of using the implicit monitoring function `CheckBounds()` to monitor the index limits at runtime.

Example

One-dimensional array of 10 integer elements

```plaintext
VAR
  aiCounter : ARRAY[0..9] OF INT;
END_VAR
```

Lower index limit: 0

Upper index limit: 9

Initialization

```plaintext
aiCounter : ARRAY[0..9] OF INT := [0, 10, 20, 30, 40, 50, 60, 70, 80, 90];
```

Data access

```plaintext
iLocalVariable := aiCounter[2];
```

The value 20 is assigned to the local variable.
Example

2-dimensional array

VAR
    aiCardGame : ARRAY[1..2, 3..4] OF INT;
END_VAR

1st dimension: 1 to 2
2nd dimension: 3 to 4

Initialization


Data access

iLocal_1 := aiCardGame[1, 3]; // Assignment of 10
iLocal_2 := aiCardGame[2, 4]; // Assignment of 20

Example

3-dimensional array

VAR
    aiCardGame : ARRAY[1..2, 3..4, 5..6] OF INT;
END_VAR

1st dimension: 1 to 2
2nd dimension: 3 to 4
3rd dimension: 5 to 6

Initialization

aiCardGame : ARRAY[1..2, 3..4, 5..6] OF INT := [10, 20, 30, 40, 50, 60, 70, 80];

Data access

iLocal_1 := aiCardGame[1, 3, 5]; // Assignment of 10
iLocal_2 := aiCardGame[2, 3, 5]; // Assignment of 20
iLocal_3 := aiCardGame[1, 4, 5]; // Assignment of 30
iLocal_4 := aiCardGame[2, 4, 5]; // Assignment of 40
iLocal_5 := aiCardGame[1, 3, 6]; // Assignment of 50
iLocal_6 := aiCardGame[2, 3, 6]; // Assignment of 60
iLocal_7 := aiCardGame[1, 4, 6]; // Assignment of 70
iLocal_8 := aiCardGame[2, 4, 6]; // Assignment of 80

Initialization

aiCardGame : ARRAY[1..2, 3..4, 5..6] OF INT := [2(10), 2(20), 2(30), 2(40)]; // Short notation for [10, 10, 20, 20, 30, 30, 40, 40]

Data access

iLocal_1 := aiCardGame[1, 3, 5]; // Assignment of 10
iLocal_2 := aiCardGame[2, 3, 5]; // Assignment of 10
iLocal_3 := aiCardGame[1, 4, 5]; // Assignment of 20
iLocal_4 := aiCardGame[2, 4, 5]; // Assignment of 20
iLocal_5 := aiCardGame[1, 3, 6]; // Assignment of 30
iLocal_6 := aiCardGame[2, 3, 6]; // Assignment of 30
iLocal_7 := aiCardGame[1, 4, 6]; // Assignment of 40
iLocal_8 := aiCardGame[2, 4, 6]; // Assignment of 40
Example

3-dimensional arrays of a user-defined structure

```plaintext
TYPE DATA_A
STRUCT
  iA_1 : INT;
  iA_2 : INT;
  dwA_3 : DWORD;
END_STRUCT
END_TYPE

PROGRAM PLC_PRG
VAR
  aData_A : ARRAY[1..3, 1..3, 1..10] OF DATA_A;
END_VAR

The array aData_A consists of a total of 3 * 3 * 10 = 90 array elements of data type DATA_A.

Initialize partially

```
aData_A : ARRAY[1..3, 1..3, 1..10] OF DATA_A := [(iA_1 := 1, iA_2 := 10, dwA_3 := 16#00FF), (iA_1 := 2, iA_2 := 20, dwA_3 := 16#FF00), (iA_1 := 3, iA_2 := 30, dwA_3 := 16#FFFF)];
```

In the example, only the first 3 elements are initialized explicitly. Elements to which no initialization value is assigned explicitly are initialized internally with the default value of the basic data type. This initializes the structure components at 0 starting with the element aData_A[2,1,1].

Data access

```
iLocal_1 := aData_A[1,1,1].iA_1; // Assignment of 1
dwLocal_2 := aData_A[3,1,1].dwA_3; // Assignment of 16#FFFF
```

Example

Array of a function block

```plaintext
FUNCTION BLOCK FBObject_A
VAR
  iCounter : INT;
END_VAR
...

PROGRAM PLC_PRG
VAR
  aObject_A : ARRAY[1..4] OF FBObject_A;
END_VAR

The array aObject_A consists of 4 elements. Each element instantiates a FBObject_A function block.

Function Call

```
aObject_A[2]();
```
### Example

**Implementation of FB_Something with method FB_Init**

```plaintext
FUNCTION_BLOCK FB_Something
  VAR
    _nId : INT;
    _lrIn : LREAL;
  END_VAR
  ...
  METHOD FB_Init : BOOL
    VAR_INPUT
      bInitRetains : BOOL;
      bInCopyCode : BOOL;
      nId : INT;
      lrIn : LREAL;
    END_VAR

    _nId := nId;
    _lrIn := lrIn;
  END_VAR

The function block FB_Something has a method FB_Init that requires 2 parameters.
```

**Instantiation of the array with initialization**

```plaintext
PROGRAM PLC_PRG
  VAR
    fb_Something_1 : FB_Something(nId := 11, lrIn := 33.44);
    a_Something : ARRAY[0..1, 0..1] OF FB_Something[(nId := 12, lrIn := 11.22), (nId := 13, lrIn := 22.33), (nId := 14, lrIn := 33.55), (nId := 15, lrIn := 11.22)];
  END_VAR
```

### Array of arrays

The declaration of an "array of arrays" is an alternative syntax for multidimensional arrays. A collection of elements is nested instead of dimensioning the elements. The nesting depth is unlimited.

**Syntax for declaration**

```plaintext
<variable name> : ARRAY[<first>] ( OF ARRAY[<next>] )+ OF <data type> ( := <initialization> )? ;

<first> : <first lower index bound>..<first upper index bound>

<next> : <lower index bound>..<upper index bound> // one or more arrays

<data type> : elementary data types | user defined data types | function block types

// (...)+ : One or more further arrays
// (...)? : Optional

**Syntax for data access**

```plaintext
<variable name>[<index of first array>] ( [<index of next array>] )+ ;
// (...)* : 0, one or more further arrays
```
Example

PROGRAM PLC_PRG
VAR
   aiPoints : ARRAY[1..2,1..3] OF INT := [1,2,3,4,5,6];
   ai2Boxes : ARRAY[1..2] OF ARRAY[1..3] OF INT := [ [1, 2, 3],
   [4, 5, 6]]; 
   [ [ [1, 2, 3, 4], [5, 6, 7, 8], [9, 10, 11, 12] ], [ [13, 14, 15,
   16], [17, 18, 19, 20], [21, 22, 23, 24] ] ];
   ARRAY[1..5] OF INT;
END_VAR

aiPoints[1, 2] := 1200;
ai2Boxes[1][2] := 1200;

The variables `aiPoints` and `ai2Boxes` collect the same data elements, however the syntax
for the declaration differs from that of the data access.

Array of variable length

In function blocks, functions, or methods, you can declare arrays of variable length in the
VAR_IN_OUT declaration section. The LOWER_BOUND and UPPER_BOUND operators are pro-
vided for determining the index limits of the actual used array at runtime.

Syntax of the declaration of a one-dimen-
sional array of variable length

<variable name> : ARRAY[*] OF <data type> ( := <initialization> )? ;

<data type> : elementary data types | user defined data types | function block types
// (...)? : Optional
**Syntax of the declaration of a multi-dimensional array of variable length**

<variable name> : ARRAY[* (, *)+] OF <data type> ( := <initialization> )?;

<data type> : elementary data types | user defined data types | function block types

// (...)+ : One or more further dimensions
// (...)? : Optional

**Syntax of the operators for calculating the limit index**

LOWER_BOUND( <variable name> , <dimension number> )
UPPER_BOUND( <variable name> , <dimension number> )

**Example**

The `SUM` function adds the integer values of the array elements and returns the calculated sum as a result. The sum is calculated across all array elements available at runtime. As the actual number of array elements will only be known at runtime, the local variable is declared as a one-dimensional array of variable length.

```plaintext
FUNCTION SUM: INT;
VAR_IN_OUT
    aiData : ARRAY[*] OF INT;
END_VAR

VAR
    diCounter, diResult : DINT;
END_VAR

diResult := 0;
FOR diCounter := LOWER_BOUND(aiData, 1) TO UPPER_BOUND(aiData, 1)
    diResult := diResult + A[i];
END_FOR;
SUM := diResult;
```

**See also**

- `Chapter 1.3.1.9.2.3 “Declaring arrays” on page 175`
- `Chapter 1.3.1.25.1.29.1 “POU 'CheckBounds’” on page 797`

**Data Type ‘__VECTOR’**

Vector operations are supported natively only on 64-bit processors and offer a performance advantage only on these processors. The data sheet of the controller provides information about the processor used on the controller.

Currently, vector operations on the x86/64-bit platforms with SSE2 and ARM64 with NEON are supported natively. On all other platforms, vector operations are translated into individual statements. For example, vector addition is then executed with multiple single addition operations.

The command set extensions of the processors are SIMD extensions. SIMD (Single Instruction, Multiple Data) describes a computer architecture in which multiple data sets of the same type are processed simultaneously in parallel and therefore faster with one command call. In vector operations, for example, 4 pairs of numbers can then be added at the same time.
A vector data type is an array of floating-point numbers with a maximum of 8 elements. The operators \_vc<operator name> are available for this data type. You can use these to implement vector operations without additional function calls.

When indexing a vector variable, you can access a single element of the vector. The index starts at 0 and goes until <vector size> - 1.

### Example

```plaintext
PROGRAM PLC_PRG
VAR
  vcA : __VECTOR[3] OF REAL;
END_VAR

vcA[0] := 1.1;
vcA[1] := 2.2;
vcA[2] := 3.3;
```

### Syntax

<variable name> : __VECTOR[ <vector size> ] OF <element type> ( := <initialization> )?

<vector size> : 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8
<element type> : REAL | LREAL

A vector data type is an array of floating-point numbers with a maximum of 8 elements. The operators __vc<operator name> are available for this data type. You can use these to implement vector operations without additional function calls.

### Syntax for index access

<variable name>[ <index> ]

<index> : 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7

When indexing a vector variable, you can access a single element of the vector. The index starts at 0 and goes until <vector size> - 1.

### Determining the optimal vector size

Use the optimal vector size depending on your target system as the vector size in order to program the most efficient code possible.

For target systems whose computer architecture is generally suitable for vector processing, we do not recommend using vectors of arbitrary size. There is an optimal vector size depending on the type of data processing of the processor. Vectors that are declared with this array size are processed as quickly as possible. Vectors that are declared as a larger array do not have an advantage in speed. Vectors that are declared as smaller arrays do not take full advantage of the processor's capabilities.

You can query the optimal size at runtime. You can find the information in the constants Constants.vcOptimalREAL (for vectors with REAL elements) and Constants.vcOptimalLREAL (for vectors with LREAL elements). The constants have the LREAL data type. If a constant returns the value 1 as the optimal value, then this means that accelerated vector processing is not available for the target system.
An application that is loaded on the CODESYS Control Win V3 x64 target system returns the following values at runtime:

![Device Application PLC_PRG screenshot](image1)

**Example**

```plc
PROGRAM PLC_PRG
VAR
  iOVS_REAL : INT; // Optimal vector size for REAL elements
  iOVS_LREAL : INT; // Optimal vector size for LREAL elements
END_VAR

iOVS_REAL := Constants.vcOptimalREAL;
iOVS_LREAL := Constants.vcOptimalLREAL;
```

The operator calculates the sum of two vectors.

**Operator **__VCADD**

**Syntax**

```plc
<vector variable> := <1st vector operand> __VCADD <2nd vector operand>;
```

**Example of addition**

```plc
FUNCTION_BLOCK FB_ADD
VAR
  vcA : __VECTOR[3] OF REAL := __VCSET_REAL(3, 3, 3);
  vcB : __VECTOR[3] OF REAL := __VCSET_REAL(1, 2, 3);
  vcResult : __VECTOR[3] OF REAL;
END_VAR

vcResult := vcA __VCADD vcB;
```

The operator calculates the difference between two vectors.

**Operator **__VCSUB**

**Syntax**

```plc
<vector variable> := <vector minuend> __VCSUB <vector subtrahend>;
```
### Example of subtraction

```plaintext
FUNCTION_BLOCK FB_SUB
VAR
  vcA : __VECTOR[3] OF REAL := __VCSET_REAL(3, 3, 3);
  vcB : __VECTOR[3] OF REAL := __VCSET_REAL(1, 2, 3);
  vcResult0 : __VECTOR[3] OF REAL;
  vcResult1 : __VECTOR[3] OF REAL;
END_VAR

vcResult0 := vcA __VCSUB vcB;
vcResult1 := vcB __VCSUB vcA;
```

### Example of multiplication

```plaintext
FUNCTION_BLOCK FB_MUL
VAR
  rScalar : REAL := 1.1;
  vcA : __VECTOR[3] OF REAL;
  vcB : __VECTOR[3] OF REAL;
  vcResult0 : __VECTOR[3] OF REAL;
  vcResult1 : __VECTOR[3] OF REAL;
END_VAR

vcResult0 := vcA __VCMUL vcB;
vcResult1 := rScalar __VCMUL vcB;
vcResult2 := vcA __VCMUL 3.3;
```

### Operator __VCMUL

The operator calculates the product of two vectors or a scalar (floating-point number) and a vector.

**Syntax**

```plaintext
<vector variable> := <1st vector operand> __VCMUL <2nd vector operand> | <scalar operand> __VCMUL <vector operand> | <vector operand> __VCMUL <scalar operand> ;
```

### Operator __VCDIV

The operator calculates the quotient of two vectors or a vector and a scalar.

**Syntax**

```plaintext
<vector variable> := <vector dividend> __VCDIV <vector divisor> | <vector dividend> __VCMUL <scalar divisor> ;
```
**Example of division**

```plaintext
FUNCTION_BLOCK FB_DIV
VAR
  iScalar : INT := 3;
  rScalar : REAL := 1.5;
  vcA : __VECTOR[3] OF REAL := __VCSET_REAL(3, 3, 3);
  vcB : __VECTOR[3] OF REAL := __VCSET_REAL(1, 2, 3);
END_VAR

vcResult0 := vcA __VCDIV vcB;
// ERROR CODE vcResult1 := rScalar __VCDIV vcB;
// ERROR CODE vcResult1 := iScalar __VCDIV vcB;
// ERROR CODE vcResult1 := 3.3 __VCDIV vcB;
vcResult2 := vcA __VCDIV 1.5;
vcResult2 := vcA __VCDIV iScalar;
vcResult2 := vcA __VCDIV rScalar;
```

**Operator _VCDOT**

The operator calculates the dot product (scalar product) of two vectors.

**Syntax**

```plaintext
<scalar variable> := <1st vector operand> __VCDOT <2nd vector operand> ;
```

**Example of a dot product**

```plaintext
FUNCTION_BLOCK FB_DOT
VAR
  rResult : REAL;
  vcA : __VECTOR[3] OF REAL := __VCSET_REAL(3, 3, 3);
  vcB : __VECTOR[3] OF REAL := __VCSET_REAL(1, 2, 3);
END_VAR

rResult := vcA __VCDOT vcB; // = 18
```

**Operator _VCSQRT**

The operator calculates the square root of each element in the vector.

**Syntax**

```plaintext
<vector variable> := __VCSQRT( <vector operand> ) ;
```

**Example of a square root**

```plaintext
FUNCTION_BLOCK FB_SQRT
VAR
  vcA : __VECTOR[3] OF REAL := __VCSET_REAL(4, 9, 16);
END_VAR

vcResult0 := __VCSQRT(vcA);
```
The operator calculates the maximum vector of two vectors. The maximum is determined element by element.

<vector variable> := __VCMAX( <1st vector operand>, <2nd vector operand>);

**Example of a maximum vector**

```plaintext
FUNCTION_BLOCK FB_MAX
VAR
  vcA : __VECTOR[3] OF REAL := __VCSET_REAL(3, 3, 3);
  vcB : __VECTOR[3] OF REAL := __VCSET_REAL(1, 2, 6);
  vcResult0 : __VECTOR[3] OF REAL;
END_VAR
vcResult0 := __VCMAX(vcA, vcB);
```

The operator calculates the minimum vector of two vectors. The minimum is determined element by element.

<vector variable> := __VCMIN( <1st vector operand>, <2nd vector operand>);

**Example of a minimum vector**

```plaintext
FUNCTION_BLOCK FB_MIN
VAR
  vcA : __VECTOR[3] OF REAL := __VCSET_REAL(3, 3, 3);
  vcB : __VECTOR[3] OF REAL := __VCSET_REAL(1, 2, 6);
  vcResult0 : __VECTOR[3] OF REAL;
END_VAR
vcResult0 := __VCMIN(vcA, vcB);
```

The operator sets all elements of a vector in a statement. The elements have the REAL data type.

<vector variable> := __VCSET_REAL( <first literal>, ( < next literal> )+ ) ;
( ... )+ // number of elements have to match

**Example**

```plaintext
FUNCTION_BLOCK FB_SET
VAR
  vcA : __VECTOR[3] OF REAL := __VCSET_REAL(3, 3, 3);
  vcB : __VECTOR[3] OF REAL := __VCSET_REAL(1, 2, 3);
END_VAR
vcA := __VCSET_REAL(4, 4, 4);
vcB := __VCSET_REAL(1.1, 2.2, 3.3);
```
The operator sets all elements of a vector at once in a statement. The elements have the LREAL data type. They can be used wherever variables are valid, such as in assignments in implementations or as parameters in function calls.

Syntax

```
<vector variable> := __VCSET_LREAL( <first literal>, ( <next literal> )+ )
```

Example

```
FUNCTION_BLOCK FB_SET
VAR
    vc1A : __VECTOR[3] OF LREAL := __VCSET_LREAL(3, 3, 3);
    vc1B : __VECTOR[3] OF LREAL := __VCSET_LREAL(1, 2, 3);
END_VAR

vc1A := __VCSET_LREAL(-1.7976931348623158E+308, 0.0, 1.7976931348623158E+308);
vc1B := __VCSET_LREAL(-1.7976931348623158E+308, 0.0, 1.7976931348623158E+308);
```

Operator __VCLOAD_REAL

The operator interprets any arbitrary memory area as a vector. This is helpful for connecting vector variables to existing code. The operator requires 2 parameters. The first parameter indicates the number of vector elements. The second parameter is a pointer to the REAL data.

Syntax

```
<vector variable> := __VCLOAD_REAL( <vector size>, <pointer to data of type REAL> )
```

Example of vectorization

```
FUNCTION_BLOCK FB_LOAD
VAR_INPUT
END_VAR
VAR_OUTPUT
END_VAR
VAR
    rData0 : REAL := 1.234;
    rData1 : REAL := 5.678;
    rData2 : REAL := 9.123;
    pData : POINTER TO REAL := ADR(rData0);

    vcA : __VECTOR[3] OF REAL := __VCSET_REAL(3, 3, 3);
END_VAR

vcA := __VCLOAD_REAL(3, pData);
```

Operator __VCLOAD_LREAL

The operator interprets any arbitrary memory area as a vector. This is helpful for connecting vector variables to existing code. The operator requires 2 parameters. The first parameter indicates the number of vector elements. The second parameter is a pointer to the LREAL data.

Syntax

```
<vector variable> := __VCLOAD_LREAL( <vector size>, <pointer to data of type LREAL> )
```

Example of vectorization

```
FUNCTION_BLOCK FB_LOAD
VAR_INPUT
END_VAR
VAR_OUTPUT
END_VAR
VAR
    rData0 : REAL := 1.234;
    rData1 : REAL := 5.678;
    rData2 : REAL := 9.123;
    pData : POINTER TO REAL := ADR(rData0);

    vcA : __VECTOR[3] OF REAL := __VCSET_REAL(3, 3, 3);
END_VAR

vcA := __VCLOAD_REAL(3, pData);
```
Example of vectorization

```plaintext
FUNCTION_BLOCK FB_LOAD
VAR_INPUT
END_VAR
VAR_OUTPUT
END_VAR
VAR
    lrData0 : LREAL := -1.7976931348623158E+308;
    lrData1: LREAL := 1.6E+308;
    lrData2 : LREAL := 1.7E+308;
    lrData3 : LREAL := -1.6E+308;
    plData: POINTER TO LREAL := ADR(lrData0);

    vclA : __VECTOR[4] OF LREAL := __VCSET_LREAL(4, 4, 4, 4);
END_VAR
vclA := __VCLOAD_LREAL(4, plData);
```

Example of storage

```plaintext
FUNCTION_BLOCK FB_STORE
VAR_INPUT
END_VAR
VAR_OUTPUT
END_VAR
VAR
    rData0 : REAL := 3;
    rData1: REAL := 3;
    rData2 : REAL := 3;
    pData: POINTER TO REAL := ADR(rData0);

    lrData0 : LREAL := 4;
    lrData1: LREAL := 4;
    lrData2 : LREAL := 4;
    lrData3 : LREAL := 4;
    plData: POINTER TO LREAL := ADR(lrData0);

    vcA : __VECTOR[3] OF REAL := __VCSET_REAL(1.234, 5.678, 9.123);
    vclA : __VECTOR[4] OF LREAL := __VCSET_LREAL(-1.7976931348623158E+308, 1.6E+308, 1.7E+308, -1.6E+308);
END_VAR
__VCSTORE(pData, vcA);
__VCSTORE(plData, vclA);
```

See also

- § Chapter 1.3.1.9.2.3 “Declaring arrays” on page 175
- § Chapter 1.3.1.25.1.29.1 “POU ‘CheckBounds’” on page 797
Structure

You create a structure in a project with a “DUT” object by clicking “Add Object”.

The structure declaration begins with the keywords TYPE and STRUCT, and ends with the keywords END_STRUCT and END_TYPE.

Syntax for the structure declaration:

```
TYPE <structure name>:
STRUCT
  <variable declaration 1>
  ...
  <variable declaration n>
END_STRUCT
END_TYPE
```

<structure name> is a type that CODESYS recognizes through the entire project and you can use as a standard data type.

You can also use nested structures. The only restriction is that you are not permitted to assign addresses to variables (because the AT declaration is not permitted).

**Example of a structure definition named Polygonline**

```
TYPE polygonline:
STRUCT
  start:ARRAY [1..2] OF INT;
  point1:ARRAY [1..2] OF INT;
  point2:ARRAY [1..2] OF INT;
  point3:ARRAY [1..2] OF INT;
  point4:ARRAY [1..2] OF INT;
  end:ARRAY [1..2] OF INT;
END_STRUCT
END_TYPE
```

See also

- ≡ Chapter 1.3.1.25.1.14 “Object 'DUT’” on page 735
- ≡ Chapter 1.3.1.24.3.28 “Operator 'EQ’” on page 523

**Initializing structures**

**Example:**

```
pPoly_1 : polygonline := ( start:=[3,3], point1:=[5,2], point2 := [7,3], point3 := [8,5], point4 := [5,7], end := [3,5]);
```

You are not permitted to use initializations with variables. For an example of initializing an array of a structure, please see the help page for the ARRAY data type.

See also

- ≡ Chapter 1.3.1.24.5.14 “Data type 'ARRAY’” on page 581

**Accessing structure members**

You access structure members with the following syntax:

```
<structure name>.<component name>
```

Therefore, you can use `poly_1.start` to access the `start` component of the `polygonline` structure in the above example.
See also

- § Chapter 1.3.1.24.5.14 “Data type 'ARRAY’” on page 581

**Bit access in structures**

Bit is a special data type that you define in structures only. It reserves one bit of memory and permits single bits of a structure to be addressed by a name.

```
TYPE <structure name>:
  STRUCT
    <bit name bit1> : BIT;
    <bit name bit2> : BIT;
    <bit name bit3> : BIT;
    ...
    <bit name bitn> : BIT;
  END_STRUCT
END_TYPE
```

You can access the BIT structure member with the following syntax:

```
<structure name>.<bit name>
```

You cannot use any references or pointers to BIT variables. Furthermore, you are not permitted to use BIT variables in arrays.

See also

- § Chapter 1.3.1.24.5.10 “Data type 'BIT’” on page 577

**Enumerations**

An enumeration is a user-defined data type composed of a series of comma-separated components (enumeration values) for declaring user-defined variables. Moreover, you can use the enumeration components like constants whose identifier `<enumeration name>.<component name>` is recognized globally in the project.

You declare an enumeration in a DUT object, which you have already created in the project by clicking “Add Object”.

**Declaration**

**Syntax**

```
( {attribute 'strict'} )? // Pragma optional but recommended
TYPE <enumeration name> :
  ( <first component declaration>,
    ( <component declaration> ,)+
    <last component declaration>
  ) ( <basic data type> )? ( := <default variable initialization> )? ;
END_TYPE
```

```
( ... )? : Optional
<component declaration> : <component name> ( := <component initialization> )?
<basic data type> : INT | UINT | SINT | USINT | DINT | UDINT | LINT | ULINT | BYTE | WORD | DWORD | LWORD
<variable initialization> : <one of the component names>
```
In an enumeration declaration, at least 2 components are usually declared. However, you can declare as many as you want. Every single component can be assigned its own initialization. Enumerations automatically have the basic data type INT, but you can specify another basic data type. Moreover, you can specify a component in the declaration with which an enumeration variable is then initialized.

The pragma {attribute 'strict'} causes a strict type test to be performed as described below.

```
{attribute 'qualified_only'}
{attribute 'strict'}
TYPE COLOR_BASIC :
{
  yellow,
  green,
  blue,
  black
}
; // Basic data type is INT, default initialization for all COLOR_BASIC variables is yellow
END_TYPE
```

**Enumeration with explicit basic data type**

Extensions to the IEC 61131-3 standard

The basic data type for an enumeration declaration is INT by default. However, you can also declare enumerations that are based explicitly on another integer data type.

```
<basic data type> : INT | UINT | SINT | USINT | DINT | UDINT | LINT | ULINT | BYTE | WORD | DWORD | LWORD
```

```
TYPE COLOR :
{
  white := 16#FFFFFF00,
  yellow := 16#FFFFFF00,
  green := 16#FF00FF00,
  blue := 16#FF0000FF,
  black := 16#88000000
} DWORD := black; // Basic data type is DWORD, default initialization for all COLOR variables is black
END_TYPE
```

**Strict programming rules**

**NOTICE!**

In CODESYS V3.5 SP7 and later, the pragma {attribute 'strict'} is added automatically in the first line when declaring an enumeration.

The strict programming rules are activated when adding the pragma {attribute 'strict'}. 

The following code is considered a compiler error:

- Arithmetic operations with enumeration components
  - For example, an enumeration variable cannot be used as a counter variable in a `FOR` loop.
- Assignment of a constant value, which does not correspond to an enumeration value, to an enumeration component
- Assignment of a non-constant variable, which has another data type as the enumeration, to an enumeration component

Arithmetic operations can lead to undeclared values being assigned to enumeration components. A better programming style is to use `SWITCH/CASE` statements for processing component values.

### Declaration and initialization of enumeration variables

#### Syntax

```
<variable name> : <enumeration name> ( := <initialization> )? ;
```

For a declaration of an enumeration variable with user-defined data type `<enumeration name>`, this can be initialized with an enumeration component.

#### Example

```plc
PROGRAM PLC_PRG
VAR
  colorCar: COLOR;
  colorTaxi : COLOR := COLOR.yellow;
END_VAR
```

The variable `colorCar` is initialized with `COLOR.black`. That is the default initialization for all enumeration variables of type `COLOR` and defined this way in the type declaration. The variable `colorTaxi` has its own initialization.

If no initializations are specified, then the initialization value is 0.

#### Example

```plc
PROGRAM PLC_PRG
VAR
  cbFlower : COLOR_BASIC;
  cbTree: COLOR_BASIC := COLOR_BASIC.green;
END_VAR
```

The variable `cbFlower` is initialized with `COLOR_BASIC.yellow`. That is the default initialization for all enumeration variables of type `COLOR_BASIC`. Because the enumeration declaration does not specify a component for initialization, the system automatically initializes with the component that has the value 0. This is usually the first of the enumeration components. However, it can also be another component that is not in the first position but explicitly initialized with 0.

The variable `cbTree` has an explicit initialization.

If no value is specified for both the type and the variable, then the following rule applies: If an enumeration contains a value for 0, then this value is the default initialization, and if not, then the first component in the list.
Example

**Initialization with the 0 component**

```plaintext
TYPE ENUM :
(
    e1 := 2,
    e2 := 0,
    e3
)
;
END_TYPE

PROGRAM PLC_PRG
VAR
    e : ENUM;
END_VAR

The variable `e` is initialized with `ENUM.e2`.
```

**Initialization with the first component**

```plaintext
TYPE ENUM2 :
(
    e1 := 3,
    e2 := 1,
    e3
)
;
END_TYPE

PROGRAM PLC_PRG
VAR
    e2 : ENUM2;
END_VAR

The variable `e2` is initialized with `ENUM.e1`.
```

**Unique access to enumeration components**

Extensions to the IEC 61131-3 standard

The enumeration components can also be used as constant variables with the identifier `<enumeration name>.<component name>`. Enumeration components are recognized globally in the project and access to them is unique. Therefore, a component name can be used in different enumerations.

Example

**Component blue**

```plaintext
PROGRAM PLC_PRG
VAR
    cbFlower : COLOR_BASIC;
    colorCar : COLOR;
END_VAR

(* unambiguous identifiers although the component names are identical *)
    cbFlower := COLOR_BASIC.blue;
    colorCar := COLOR.blue;

(* invalid code *)
    cbFlower := blue;
    colorCar := blue;
```
Subrange types

A subrange type is a data type whose value range is a subset of a base type.

Syntax for the declaration:

```
<name> : <int type> (<lower limit>..<upper limit>);
```

<table>
<thead>
<tr>
<th>&lt;name&gt;</th>
<th>valid IEC identifier</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;int type&gt;</td>
<td>data type of the subrange</td>
</tr>
<tr>
<td></td>
<td>(SINT, USINT, INT, UINT, DINT, UDINT, BYTE, WORD, DWORD, LINT, ULINT, LWORD).</td>
</tr>
<tr>
<td>&lt;lower limit&gt;</td>
<td>Lower limit of the range: constants that have to be compatible with the basic data type. The lower limit is also included in this range.</td>
</tr>
<tr>
<td>&lt;upper limit&gt;</td>
<td>Upper limit of the range: constants that have to be compatible with the base data type. The upper limit is also included in this range.</td>
</tr>
</tbody>
</table>

Examples:

```
VAR
  i : INT (-4095..4095);
  ui : UINT (0..10000);
END_VAR
```

If you assign a value to a subrange type in the declaration or implementation section that is not within this range (example: i:=5000), then CODESYS issues an error message.

Please note: In runtime mode, it is possible to monitor the range limits of a subrange type by using the implicit monitoring functions CheckRangeSigned and CheckRangeUnsigned.

See also

- Chapter 1.3.1.24.3.70 “Operator - Enumeration namespace” on page 556

1.3.1.24.6 Pragmas

1.3.1.24.6.1 Message pragmas................................................................. 600
1.3.1.24.6.2 Attribute pragmas............................................................. 601
1.3.1.24.6.3 Conditional pragmas......................................................... 637
1.3.1.24.6.4 Region pragma............................................................... 644

Pragma instructions affect the properties of one or more variables with regard to the compilation or pre-compilation process. Various categories of pragmas are available to you for this.
Message pragmas

Message pragmas serve to force the display of messages in the Message window during the compilation process.

Insertion position: separate or already existing line in the text editor of a POU.

Table 25: 4 types of message pragmas

<table>
<thead>
<tr>
<th>Pragma</th>
<th>Message type</th>
</tr>
</thead>
<tbody>
<tr>
<td>{text (&lt;'textstring'&gt;)}</td>
<td>“Text”: display of &lt;textstring&gt;.</td>
</tr>
<tr>
<td>{info (&lt;'textstring'&gt;)}</td>
<td>“Information”: display of &lt;textstring&gt;.</td>
</tr>
<tr>
<td>{warning (&lt;'textstring'&gt;)}</td>
<td>“Warning”: display of &lt;textstring&gt;.</td>
</tr>
<tr>
<td>{error (&lt;'textstring'&gt;)}</td>
<td>“Error”: display of &lt;textstring&gt;.</td>
</tr>
</tbody>
</table>

Unlike the attribute pragma 'obsolete', you define the warning locally for the current position.

In the CODESYS Message window you can jump with the help of the commands “Next Message” and “Previous Message” from a message of the category “Information”, “Warning” and “Error” to the source position of the message. This means you jump to the position where the pragma is added in the source code.

Example

```plaintext
VAR
  var : INT; {info 'TODO: should get another name'}
  bvar : BOOL;
  arrTest : ARRAY [0..10] OF INT;
  i:INT;
END_VAR
  arrTest[i] := arrTest[i]+1;
  ivar:=ivar+1;
{warning 'This is a warning'}
{text 'Part xy has been compiled completely'}
```

Display in the Message window:

<table>
<thead>
<tr>
<th>Messages</th>
</tr>
</thead>
<tbody>
<tr>
<td>Build</td>
</tr>
<tr>
<td>Description</td>
</tr>
<tr>
<td>---------</td>
</tr>
<tr>
<td>build started: Application: Res.App2</td>
</tr>
<tr>
<td>typify code</td>
</tr>
<tr>
<td>Compile time before typification: 6 ms</td>
</tr>
<tr>
<td>Compile time after typification: 15 ms</td>
</tr>
<tr>
<td>TODO: should get another name!</td>
</tr>
<tr>
<td>This is a warning</td>
</tr>
<tr>
<td>Part xy has been compiled completely</td>
</tr>
<tr>
<td>Compile complete -- 1 errors, 2 warnings</td>
</tr>
</tbody>
</table>
Attribute pragmas affect the compilation and the pre-compilation.

CODESYS supports a series of pre-defined attribute pragmas. In addition you can use user-defined pragmas, which you can query with the help of conditional pragmas before the compilation of the project.
Attributes are defined within the declaration part. Exception: For the objects Action and Transition, which have no own declaration part, you can define the attributes at the beginning of the implementation part.

When you define own attributes, please make them unambiguous. Uniqueness can be reached for example by adding a prefix to the attribute name. OEMs can use the vendor prefix for this purpose.

User-defined attributes

User-defined attributes are any application-defined or user-defined attributes that you can apply to POUs, actions, data type definitions and variables. You can query a user-defined attribute with the help of conditional pragmas before the compilation of the application.

You can query user-defined attributes with conditional pragmas with the operator hasattribute.

More detailed information and examples can be found in the chapter 'Conditional pragmas'.

Syntax:

{attribute 'attribute'}

Example for POUs and actions

Attribute 'vision' for function “fun1”

\[
\begin{align*}
\text{FUNCTION fun1 : INT} \\
\text{VAR_INPUT i : INT;} \\
\text{END_VAR}
\end{align*}
\]

Example for variables

Attribute 'DoCount' for variable ivar:

\[
\begin{align*}
\text{PROGRAM PLC_PRG} \\
\text{VAR} \\
\text{(attribute 'DoCount');} \\
\text{ivar:INT;} \\
\text{bvar:BOOL;} \\
\text{END_VAR}
\end{align*}
\]
Example for data types

Attribute 'aType' for data type DUT_1:

```plaintext
{attribute 'aType'}
TYPE DUT_1 :
  STRUCT
    a:INT;
    b:BOOL;
  END_STRUCT
END_TYPE
```

See also
-  Chapter 1.3.1.24.6.3 “Conditional pragmas” on page 637

Attribute 'call_after_global_init_slot'

```plaintext
NOTICE!
VAR_INPUT declarations in functions or methods that use the attribute lead to compile errors. Reason: Input variables are unknown in this case at the time of the call, which occurs implicitly during the online change.
```

The effect of this pragma is that all functions and programs containing this attribute are called after the global initialization. You define the order of calling by means of the attribute value.

**Syntax:**

```plaintext
{attribute 'call_after_global_init_slot' := '<slot>'}
```

*<slot>*: Integer value that defines the ranking in the order of the calls; the lower the value, the earlier the call takes place. If several function blocks have the same ranking for the attribute, then the order of their calls remains indefinite.

Insert location: First line above the declaration part of functions and programs

If a method possesses the attribute, CODESYS determines all instances of the corresponding function block and calls all instances in the specified slot. In this case you have no influence on the order of the instances among themselves.

See also
-  Chapter 1.3.1.24.10 “Methods 'FB_Init', 'FB_Reinit', and 'FB_Exit'” on page 653

Attribute 'call_after_init'

```plaintext
NOTICE!
VAR_INPUT declarations in functions or methods that use the attribute lead to compile errors. Reason: Input variables are unknown in this case at the time of the call, which occurs implicitly during the online change.
```

The effect of this pragma is that a method is called implicitly after the initialization of a function block instance. For reasons of performance you must add the attribute both to the function block and to the method in its own first line above the declaration part.
Syntax:

{attribute 'call_after_init'}

Call: First line above the declaration part of the method and the function block.
CODESYS calls the method after the method FB_init and after the variable values of an initialization expression in the instance declaration have become valid.

This functionality is supported from compiler version 3.4.1.0.

Example

Definition:

{attribute 'call_after_init'}
FUNCTION_BLOCK FB
... <function block definition>

{attribute 'call_after_init'}
METHOD FB_AfterInit
... <method definition>

The definition implements, for example, the following declaration in the subsequent code processing:

inst : FB := (in1 := 99);

Code processing:

inst.FB_Init();
inst.in1 := 99;
inst.FB_AfterInit();

This allows a reaction to the user-defined initialization in FB_AfterInit.

See also

- Chapter 1.3.1.24.10 “Methods ‘FB_Init’, ‘FB_Reinit’, and ‘FB_Exit’” on page 653

Attribute 'call_after_online_change_slot'

NOTICE!

VAR_INPUT declarations in functions or methods that use the attribute lead to compile errors. Reason: Input variables are unknown in this case at the time of the call, which occurs implicitly during the online change.

The effect of this pragma is that all functions and programs containing this attribute are called after an online change. You define the order of calling by means of the attribute <slot>.

Syntax:

{attribute 'call_after_online_change_slot' := '<slot>'}

<slot>: Integer value that defines the ranking in the order of the calls; the lower the value, the earlier the call takes place. If several function blocks have the same ranking for the attribute, then the order of their calls remains indefinite.

Call: First line above the declaration part of functions and programs.

If a method possesses the attribute, then CODESYS determines all instances of the function block concerned. CODESYS calls all instances in the specified slot. In this case you have no influence on the order of the instances among themselves.
NOTICE!
Since the application cannot run during the online change, each code executed in this situation can lead to a jitter. Therefore, keep the extent of the executive code as small as possible.

See also
- § Chapter 1.3.1.25.2.6.6 “Command ‘Online Change’” on page 928

Attribute 'call_before_global_exit_slot'

NOTICE!

VAR_INPUT declarations in functions or methods that use the attribute lead to compile errors. Reason: Input variables are unknown in this case at the time of the call, which occurs implicitly during the online change.

The effect of this pragma is that all functions and programs containing this attribute in a dedicated first line of their declaration are called before the GlobalExit. The GlobalExit takes place before a new download or a reset. Function blocks provided with an FB_Exit method are affected. The order of calling is defined by means of the attribute value.

Syntax:

{attribute 'call_before_global_exit_slot' := '<slot>'}

Insert location: First line above the declaration part of functions and programs.

<slot>: Integer value that defines the ranking in the order of the calls; the lower the value, the earlier the call takes place. If several function blocks have the same ranking for the attribute, then the order of their calls remains indefinite.

If a method possesses the attribute, then the method is called for all instances of the function block concerned. CODESYS calls all instances in the specified slot. In this case you have no influence on the order of the instances among themselves.

See also
- § Chapter 1.3.1.24.10 “Methods ‘FB_Init’, ‘FB_Reinit’, and ‘FB_Exit’” on page 653

Attribute 'call_on_type_change'

With this pragma, you can mark a method of a function block A that should be called when the data type changes for one or more function blocks B, C, etc. that are referenced by A. The referencing can be defined by a pointer variable or a REFERENCE variable.

Syntax:

{attribute 'call_on_type_change' := '<name of the first referenced function block>|<name of the second referenced function block>|<name of the ... referenced function block>'}

Insert location: Line above the first line in the method declaration.
Example

Function blocks with references

FUNCTION_BLOCK FB_A
  ...
  VAR
    var_pt: POINTER TO FB_B;
    var_ref: REFERENCE TO FB_C;
  END_VAR
  ...

{attribute 'call_on_type_change' := 'FB_B, FB_C'}

METHOD METH_react_on_type_change : INT
  VAR_INPUT
  ...

Method for reaction to a type change in the references FB_B and FB_C

Attribute 'const_replaced', attribute 'const_no_replaced'

The effect of the pragma \{attribute 'const_replaced'} is that the compiler option "Replace constants" is explicitly activated for all global constants containing this attribute in the constant declaration.

Accordingly, you insert the pragma \{attribute 'const_non_replaced'} in order to explicitly deactivate the compiler option "Replace constants". This can be desirable, for example, so that a constant is available in the symbol configuration.

The option "Replace constants" in the dialog box "Project Settings", category "Compile Options" is preset for the entire project.

Syntax:

\{attribute 'const_replaced'}

\{attribute 'const_non_replaced'}

Insertion position: line above the declaration line of the global variables.

Example

The constants iTestCon and bTestCon are available in the symbol configuration, because the option "Replace constants" is deactivated.

VAR_GLOBAL CONSTANT
  \{attribute 'const_non_replaced'}
  iTestCon    :   INT  := 12;
  (attribute 'const_non_replaced')
  bTestCon    :   BOOL := TRUE;
  rTestCon    :   REAL := 1.5;
END_VAR

VAR_GLOBAL
  iTestVar    :   INT  := 12;
  bTestVar    :   BOOL := TRUE;
END_VAR

See also

- Chapter 1.3.1.25.3.11.3 “Dialog Box ‘Project Settings’ - ‘Compileoptions’” on page 1062
- Chapter 1.3.1.10.2 “Symbol configuration” on page 275
Attribute 'dataflow'

With this pragma you control the data flow in the processing of function blocks in the FBD/LD/IL editor. The attribute defines the input or output of a function block to which the continuing connection to the next or previous function block is connected.

You may provide only 1 input and 1 output with the attribute in the declaration of a function block.

**Syntax:**

```
{attribute 'dataflow'}
```

Insertion position: line above the line with the declaration of the corresponding variables.

In the case of function blocks without the attribute 'dataflow', CODESYS determines the data flow as follows: first of all the connection is placed between an output and an input of same data type. The highest input or output variable of the function blocks is always taken. If there are no variables of a corresponding data type, CODESYS connects the highest output with the highest input of the neighboring function blocks.

**Example**

The connection between FB and the preceding function block is established via the input variable i1. The connection between FB and the following function block is established via the output variable outRes1.

```plaintext
FUNCTION_BLOCK FB
VAR_INPUT
  r1 : REAL;
  {attribute 'dataflow'}
  i1 : INT;
  i2 : INT;
  r2 : REAL;
END_VAR

VAR_OUTPUT
  {attribute 'dataflow'}
  outRes1 : REAL;
  out1 : INT;
  g1 : INT;
  g2 : REAL;
END_VAR
```

**See also**

- Chapter 1.3.1.9.3.1.1 “Programming function block diagrams (FBD)” on page 184

Attribute 'displaymode'

With this pragma you define the display mode of an individual variable. This definition overwrites the global setting for the display of the monitoring variable, which takes place via the commands in the menu “Debug ➔ Display Mode”.

**Syntax:**

```
{attribute 'displaymode':=<displaymode>}
```

The following definitions are possible
- Binary format
  - {attribute 'displaymode':='bin'}
  - {attribute 'displaymode':='binary'}
- Decimal format
  - attribute 'displaymode':='dec'}
  - {attribute 'displaymode':='decimal'}
- Hexadecimal format
  - {attribute 'displaymode':='hex'}
  - attribute 'displaymode':='hexadecimal'}

Insertion position: line above the line with the declaration of the corresponding variables.

Example

```plaintext
VAR
  {attribute 'displaymode':='hex'}
  dwVar1: DWORD;
END_VAR
```

See also
- % Chapter 1.3.1.25.2.7.24 “Command 'Display Mode' - 'Binary', 'Decimal', 'Hexadecimal’” on page 948

Attribute 'estimated-stack-usage'

The pragma provides an estimated value for the stack size requirement. Methods with recursive calls cannot pass a stack check because stack usage cannot be determined. As a result, a warning is issued. To prevent this warning, you can give the method an estimated value (in bytes) for the stack size requirement. Then the method passes the stack check successfully.

Syntax

```plaintext
{attribute 'estimated-stack-usage' := '<estimated stack size in bytes>'}
```

Example

```plaintext
{attribute 'estimated-stack-usage' := '127'} // 127 bytes
METHOD PUBLIC DoIt : BOOL
  VAR_INPUT
END_VAR
```

Insert location: First line above the declaration part of the method.
The section "Method call" includes an example that uses this pragma.

Recursive method call

Within its implementation, a method can call itself, either directly by means of the THIS pointer, or by means of a local variable for the assigned function block.

Use recursions mainly for processing recursive data types such as linked lists. In general, we recommend to be careful when using recursion, as unexpectedly deep recursions can cause stack overflow and machine downtime.
The following program PLC PRG calculates the factorial of a number in the FB_Factorial function block in a different way, each in its own method.

- Method m_Iterative: Iterative
- Method m_Pragmahed: Recursive with warning suppression
- Method m_Recursive: Recursive
- Method m_Temp: Temporary with warning suppression

A warning is issued for the m_Recursive method only.

// Contains the data of the factorial calculation of uiNumber
TYPE FACTORIAL_RESULT:
  STRUCT
    uiNumber : UINT;
    udiIterative : UDINT;
    udiRecursive : UDINT;
    udiPragmaed : UDINT;
    udiTemp : UDINT;
  END_STRUCT
END_TYPE

PROGRAM PLC_PRG
VAR
  fb_Factorial_A : FB_Factorial;
  factorial_A : FACTORIAL_RESULT := (uiNumber := 9,
    udiIterative := 0, udiRecursive := 0, udiPragmaed := 0);
END_VAR

fb_Factorial_A.p_Number := factorial_A.uiNumber;
factorial_A.udiIterative := fb_Factorial_A.m_Iterative();
factorial_A.udiRecursive := fb_Factorial_A.m_Recursive(uiN := factorial_A.uiNumber);
factorial_A.udiPragmaed := fb_Factorial_A.m_Pragmaed(uiN := factorial_A.uiNumber);
factorial_A.udiTemp := fb_Factorial_A.m_Temp(uiN := factorial_A.uiNumber);
// Factorial calculation in different ways
FUNCTION_BLOCK FB_Factorial
VAR_INPUT
END_VAR
VAR_OUTPUT
END_VAR
VAR
uiN : UINT;
udiIterative : UDINT;
udiPragmaed : UDINT;
udiRecursive : UDINT;
END_VAR

// Iterative calculation
METHOD PUBLIC m_Iterative : UDINT
VAR
uiCnt : UINT;
END_VAR
m_Iterative := 1;
IF    uiN > 1 THEN
    FOR uiCnt := 1 TO uiN DO
        m_Iterative := m_Iterative * uiCnt;
    END_FOR;
RETURN;
ELSE
    RETURN;
END_IF;

// Recursive calculation with suppressed warning
METHOD PUBLIC mPragmaed : UDINT
VAR_INPUT
uiN : UINT;
END_VAR
VAR
mPragmaed := 1;
IF    uiN > 1 THEN
    mPragmaed := uiN * THIS^.mPragmaed(uiN := (uiN - 1));
RETURN;
ELSE
    RETURN;
END_IF;

// Recursive calculation
METHOD PUBLIC m_Recursive : UDINT
VAR_INPUT
uiN : UINT;
END_VAR
VAR
m_Recursive := 1;
IF    uiN > 1 THEN
    m_Recursive := uiN * THIS^.m_Recursive(uiN := (uiN - 1));
RETURN;
ELSE
    RETURN;
END_IF;

// Called by temporary FB instance
METHOD PUBLIC m_Temp : UDINT
VAR_INPUT
uiN : UINT;
END_VAR
VAR
    fb_Temp : FB_Factorial;
END_VAR
m_Temp := 1;
IF    uiN > 1 THEN
    m_Temp := uiN * fb_Temp.m_Temp{uiN := (uiN - 1)};
    RETURN;
ELSE
    RETURN;
END_IF;

PROPERTY p_Number : UINT
    uiN := p_Number; //Setter method

Only the m_Recursive issues a warning when the program is executed.

See also
- Chapter 1.3.1.9.23.4 “Calling methods” on page 267
- Chapter 1.3.1.9.23 “Object-oriented programming” on page 263
- Chapter 1.3.1.25.1.28.5 “Object ‘Method’” on page 784
- Chapter 1.3.1.25.1.28.8 “Object ‘Property’” on page 791

Attribute 'ExpandFully'
The effect of this pragma is that the components of an array used as an input variable for referenced visualizations are made visible in the Properties dialog box of the visualization.

Syntax:
(attribute 'ExpandFully')

Insertion position: the line above the line with the declaration of the array.
The visualization "visu" is to be inserted into a frame inside the visualization "visu_main". arr is defined as an input variable in the interface editor of "visu" and will thus be available later for assignments in the Properties dialog box of the frames in "visu_main". In order to also make the individual components of arr available in this Properties dialog box, you must insert the attribute 'ExpandFully' directly before arr in the interface editor of visu. Declaration in the interface editor of "visu":

```
VAR_INPUT  
{attribute 'ExpandFully')
arr : ARRAY[0..5] OF INT;
END_VAR
```

Example

```
Example
```

Attribute 'global_init_slot'

This pragma defines the initialization order of function blocks or global variable lists. Variables in a list (GVL or POU) are initialized from top to bottom.

If there are several global variable lists, then the initialization order is not defined.

The initialization does not apply for the initialization of literal values, for example 1, 'hello', 3.6, or constants of base data types. However, you must define the initialization order yourself if there are dependencies between the lists. You can assign a defined initialization slot to a GVL or POU with the 'global_init_slot' attribute.

Syntax:
```
{attribute 'global_init_slot' := '<slot>'}
```

(slot): Integer value that defines the position in the call order. The default value for a POU (program, function block) is 50000. The default value for a GVL is 49990. A lower value means an earlier initialization. Caution: If several blocks or GVLs receive the same value for the 'global_init_slot' attribute, then the initialization order remains undefined.

Insert location: The pragma always affects the entire GVL or POU and therefore it must be located above the VAR_GLOBAL or POU declaration.
The program includes two global variable lists “GVL_1” and “GVL_2”, as well as a “PLC_PRG” program that uses variables from both lists. “GVL_1” uses the variable B for initializing a variable A, which is initialized in “GVL_2” with a value of 1000.

```
VAR_GLOBAL   //49990
A : INT := GVL_2.B*100;
END_VAR

VAR_GLOBAL   //49990
B : INT := 1000;
C : INT := 10;
END_VAR

PROGRAM PLC_PRG  //50000
VAR
ivar: INT := GVL_1.A;
ivar2: INT;
END_VAR
ivar:=ivar+1;
ivar2:=GVL_2.C;
```

In this case, the compiler prints an error because GVL_2.B is used for initializing GVL_1.A before “GVL_2” has been initialized. You can prevent this by using the global_init_slot attribute to position “GVL_2” before “GVL_1” in the initialization sequence.

In this example, “GVL_1” must have at least a slot value of 49989 in order to achieve the earliest initialization within the program. Every lower value has the same effect:

```
{attribute 'global_init_slot' := '100'}
VAR_GLOBAL
B : INT := 1000;
END_VAR
```

Note: Using GVL_2.C in the implementation part of “PLC_PRG” is also not critical even without using a pragma because both GVLs are initialized before the program in either case.

## Attribute 'hide'

The pragma prevents variables or POUs from being shown in the CODESYS user interface. In this way, you can intentionally hide variables or POUs from the user. As a result, the variables or POUs are neither visible in the Library Manager nor are they suggested in the input assistant or in the "List components" function. In online mode, they are not visible in the declaration part and debug functionalities cannot be applied to them.

If the instance path to a variable is known, then it can be addressed in the code even when it has the hide attribute.

**Syntax:**

```
{attribute 'hide'}
```

*Insert location:* For variables, above the line with the declaration of the variables. For POUs, in the first line.
The function block “myPOU” uses the attribute {attribute 'hide'}:

FUNCTION_BLOCK myPOU

VAR_INPUT
  a:INT;
  {attribute 'hide'}
  a_invisible: BOOL;
  a_visible: BOOL;
END_VAR

VAR_OUTPUT
  b:INT;
END_VAR

Two instances of the function block “myPOU” are defined in the main program.

PROGRAM PLC_PRG

VAR
  POU1, POU2: myPOU;
END_VAR

When the input value for POU1 is implemented, the "List components" function, which opens when you type POU1. (in the implementation part of “PLC_PRG”), displays the variables a, a_visible, and b, but not the hidden variable a_invisible.

Example

With the pragma hide_all_locals you can hide all local variables of a declaration.

See also

- § Chapter 1.3.1.24.6.2.14 “Attribute 'hide_all_locals'” on page 614

Attribute 'hide_all_locals'

This pragma prevents all local variables of a signature from being visible in the display of the 'List components' function, in the input assistant or in the declaration part in online mode.

Syntax:

{attribute 'hide_all_locals'}

Insertion position: first line above the declaration part of the POU.
Example  The function block “myPOU” uses the attribute:

```plaintext
{attribute 'hide_all_locals'}
FUNCTION_BLOCK myPOU
VAR_INPUT
  a: INT;
END_VAR

VAR_OUTPUT
  b: BOOL;
END_VAR

VAR
  c,d: INT;
END_VAR
```

Two instances of the function block “myPOU” are defined in the main program.

```plaintext
PROGRAM PLC_PRG
VAR
  POU1, POU2: myPOU;
END_VAR
```

While an input value for “POU1” is now implemented, the ‘List components’ function, which opens when typing POU1, in the implementation part of PLC_PRG, displays the variables a and b, but not the hidden local variables c or d.

See also

- § Chapter 1.3.1.24.6.2.13 “Attribute 'hide'” on page 613

**Attribute 'initialize_on_call'**

This pragma can be applied to input variables. It causes the input variables of a function block to be initialized on each call of the function block. If an input expects a pointer and this pointer has been removed in the course of an online change, this input will be set to zero.

**Syntax:**

```plaintext`
{attribute 'initialize_on_call'}
```

Insertion position: line above the line with the declaration of the corresponding input variables.

**Attribute 'init_namespace'**

The effect of this pragma is that a variable of the type `STRING` or `WSTRING`, which is declared in a library function block with this pragma, is initialized when used in the project with the current namespace of the library.

**Syntax**

```plaintext`
{attribute 'init_namespace'}
```

Insertion position: the line above the line with the declaration of the variables in a library function block.
The function block “POU” is provided with the necessary attributes:

```
FUNCTION_BLOCK POU
VAR_OUTPUT
  {attribute 'init_namespace'}
  myStr: STRING;
END_VAR
```

An instance fb of the function block POU is defined within the main program PLC_PRG:

```
PROGRAM PLC_PRG
VAR
  fb:POU;
  newString: STRING;
END_VAR
newString := fb.myStr;
```

The variable myStr is initialized with the current namespace, for example MyLib. This value is assigned to newString in the main program.

See also

- Chapter 1.3.1.25.1.25 “Object ‘Library Manager” on page 771

### Attribute 'init_on_onlchange'

The effect of this pragma is that the variable to which the pragma is applied is initialized with each online change.

**NOTICE!**

For compiler version 3.5.0.0 and later, a fast online change is performed for minor changes. In this case, only the modified blocks are compiled and downloaded. In particular, no initialization code is generated. This means that also no code is generated when variables with the `init_on_onlchange` attribute are initialized. As a rule, this has no effect because the attribute is used primarily for initializing variables with addresses. However, it cannot happen that a variable changes its address during an online change.

To secure the effect of the `init_on_onlchange` attribute in the entire application code, you must deactivate the fast online change in general for the application by using the compiler definition `no_fast_online_change`. To do this, insert the definition in the application “Properties” (“Build” tab).

**Syntax:**

```
{attribute 'init_on_onlchange'}
```

**Insert location:** The line above the line with the declaration of the variables.

See also

- Chapter 1.3.1.25.3.10.4 “Dialog ‘Properties’ - ‘Build’” on page 1048

### Attribute 'instance-path'

This pragma can be applied to a local STRING variable and causes this local STRING variable to be initialized in sequence with the device tree path of the POU to which it belongs. This can be useful for error messages. The application of the pragma requires the application of the attribute 'reflection' to the associated POU, as well as the application of the additional attribute 'noinit' to the STRING variable.

**Syntax:**

```
{attribute 'instance-path'}
```

The following function block contains the attributes 'reflection', 'instance-path' and 'noinit'.

```plaintext
{attribute 'reflection'}
FUNCTION_BLOCK POU
{attribute 'instance-path'}
{attribute 'noinit'}
  str: STRING;
END_VAR
```

An instance "myPOU" of the function block “POU” is defined within the main program "PLC_PRG":

```plaintext
PROGRAM PLC_PRG
VAR
  myPOU:POU;
  myString: STRING;
END_VAR
myPOU();
myString:=myPOU.str;
```

Following the initialization of the instance myPOU, the path of the instance myPOU is assigned to the string variable str, in the example PLCWinNT.Application.PLL_PB.myPOU. This path is assigned in the main program to the variable myString.

### Example

#### NOTICE!

You can define the length of a string to be whatever you like (even >255), but you must consider that the string will be truncated at the end if it is assigned to a variable whose data type is too small for it.

See also

- ¶ Chapter 1.3.1.24.6.2.34 “Attribute 'reflection'” on page 634
- ¶ Chapter 1.3.1.24.6.2.26 “Attribute 'noinit'” on page 623

### Attribute 'io_function_block', 'io_function_block_mapping'

With the 'io_function_block' attribute, you mark a function block in order to prepare it for the assignment to a channel in the I/O mapping of the device configuration. Then it is shown in the “Select function block” dialog.

With the 'io_function_block_mapping' attribute, you mark a parameter that should be used when mapping the FB to a device channel in this kind of function block. You can provide the attribute to multiple parameters of the function block. For I/O mapping, the first one is used automatically whose type matches the channel (input, output, data type).

**Syntax:**

```plaintext
{attribute 'io_function_block'}
{attribute 'io_function_block_mapping'}
```

**Insert location:** The line above the first line in the declaration of the function block, or the line above the parameter declaration.
Example

FUNCTION_BLOCK Scale_Output_Int
VAR_INPUT
  iInput : INT;
  iNumerator : INT;
  iDenominator : INT := 1;
  iOffset : INT := 0;
END_VAR
VAR_OUTPUT
  iOutput : INT;
END_VAR

See also

- Chapter 1.3.1.25.3.3 “Dialog ‘Select Function Block’” on page 1040
- “Linking a device with a function block instance” on page 167

Attribute 'linkalways'

The effect of this pragma is that the associated object is marked with the compiler and is thus always contained in the compiler information. This means that the object is always compiled and loaded to the PLC. The pragma works for POUs and GVLs that are located below an application or in libraries below an application or are contained in the “POUs” view. The “Always link” option in the “Compile” category of the object properties has the same effect.

Syntax:

{attribute 'linkalways'}

Insertion position: the line above the line with the keyword VAR_GLOBAL in the declaration part of POUs and GVLs.

If you use the symbol configuration editor, CODESYS uses the marked POU as the basis for the selectable variables of the symbol configuration used.

Example

The implementation of the global variable list “GVLMoreSymbols” uses the attribute 'linkalways':

{attribute 'linkalways'}
VAR_GLOBALS
  g_iVar1: INT;
  g_iVar2: INT;
END_VAR

Through this code the variables of “GVLMoreSymbols” are made available as selectable symbols.
Attribute 'monitoring'

The effect of this pragma is that you can monitor values of properties or function calls in the online view of the IEC editor or in a watch list. There are two possible attribute values for this: 'variable' and 'call'.

Syntax

```
{attribute 'monitoring' := 'variable'}
{attribute 'monitoring' := 'call'}
```

Monitoring of programming objects and their properties

In the online view of a function block or program, you can monitor the subordinate properties in addition to the local variables. This allows you to monitor the values of the Get and Set methods.

Insert either the pragma `{attribute 'monitoring' := 'variable'}` or `{attribute 'monitoring' := 'variable'}` in the declaration of the property block. The current values of the property are then displayed automatically in the IEC editor or in a watch list.
In online mode, the PLC_PRG object shows the value of the Minutes property at the call location inline in the ST editor. This is because the pragma `{attribute 'monitoring':='variable'}` is located in the declaration of the Minutes property.

Example

Check carefully for each application which attribute pragma is suitable for displaying the desired value. This depends on whether further operations with the variables are implemented within the property.

1. **Pragma `{attribute 'monitoring':='variable'}`:**

An implicit variable is created for the property, which is then always given the current property value when the application calls the `Set` or `Get` method. The value stored last in this variable is displayed in the monitoring.

2. **Pragma `{attribute 'monitoring':='call'}`:**

You can use this attribute only for properties that return simple data types or pointers, but not for structured types. The value to be monitored is read or written by calling the property directly. This means that the monitoring service of the runtime executes the `Get` or `Set` method of the property.

**NOTICE!**

When you insert the pragma `{attribute 'monitoring':='call'}` for monitoring, you have to pay attention to possible side effects. These kinds of side effects can occur if additional operations are implemented in the property.

**NOTICE!**

The pragma `{attribute 'monitoring'}` is also evaluated for the symbol configuration. Only read access is possible for the value `'variable'`.

With the context menu command “Add Watch”, a variable on which the cursor is currently positioned is applied directly into the monitoring list in online mode.
The forcing or writing of functions is not supported. However, you can implicitly implement forcing by adding an additional input parameter for the respective function, which serves as an internal force flag.

Function monitoring is not possible in the compact runtime system.

See also

- Chapter 1.3.1.25.1.28.8 “Object ‘Property’” on page 791

**Attribute 'no_assign', Attribute 'no_assign_warning'**

The pragma 'no_assign' results in compiler errors being displayed if an instance of the function block is assigned to another instance of the same function block. Such assignments are often to be avoided if the function block contains pointers and pointers lead to problems, because they are copied as well during the value assignment.

The pragma 'no_assign_warning' results in the same as for the pragma 'no_assign' with compiler warnings instead of compiler errors.

**Syntax:**

```plaintext
{attribute 'no_assign'}
```

Insert location: First line in the declaration part of a function block.
Assignment of function block instances containing pointers.

In this example the value assignment of the function block instances will lead to problems during the execution of `fb_exit`:

```plaintext
VAR_GLOBAL
  inst1 : TestFB;
  awsBufferLogFile : ARRAY [0..9] OF WSTRING(66);(* Area: 0,
  Offset: 0x1304 (4868)*)
  LogFile : SEDL.LogRecord := (sFileName := 'LogFile.log',
    pBuffer := ADR(awsBufferLogFile), udiMaxEntriesFile := UDINT#10000,
    udiMaxBuffered := UDINT#10, uiLineSize := UINT#64, wsSep := " ",
    xCircular := TRUE, siDateFormat := SINT#0, siTimeFormat := SINT#0);
END_VAR

PROGRAM PLC_PRG
VAR
  inst2 : TestFB := inst1;
  LogFileNew
END_VAR

In this case LogRecord manages a list of pointers, for which various actions are executed in the case of `fb_exit`. Problems result due to the assignment, because `fb_exit` will be executed twice. You should prevent this by adding the attribute 'no_assign' in the declaration of the function block "TestFB":

```plaintext
{attribute 'no_assign'}
FUNCTION_BLOCK TestFB
VAR_INPUT
...
```

The following compiler errors are then displayed:

```plaintext
C0328: Assignment not allowed for type TestFB
C0328: Assignment not allowed for type LogRecord
```

If the pragma `no_assign_warning` is used instead of the pragma `no_assign` for the function block "TestFB", then the C0328 message is issued as compiler warning, not as a compiler error.

Attribut 'no_check'

This pragma prevents the check function being called for the POU (POUs for implicit checks). Since the check functions can affect the processing speed of the program, it can be useful to apply the attribute to function blocks that have already been checked or are frequently called.

You add the pragma to the declaration of a POU.

Syntax:

```plaintext
{attribute 'no_check'}
```

Insertion position: first line in the declaration part of the POU.

**NOTICE!**

The attribute also automatically affects the child objects of a POU!

Example: If the attribute is entered in a program, check functions will also not be carried out for actions that are assigned to this program.
Attribute 'no_copy'

In general an online change requires a re-allocation of instances, for example of a POU. In the process, the value of the variable contained in the instance is copied.

The pragma prevents the value of the variable contained in the instance from being copied in the course of an online change; instead, the variable is re-initialized in the course of an online change. This can be useful for a local pointer variable that points to a variable that has just been shifted by the online change and thus has a changed address.

You insert the attribute in the declaration part above the line of the declaration of the variables concerned.

Syntax:

{attribute 'no_copy'}

Attribute 'no-exit'

This attribute suppresses the call of the FB_exit method of a function block for a certain one of its instances. To do this you insert the attribute in the line before the declaration of the function block instance.

Syntax:

{attribute 'no-exit'}

Example

The method "FB_exit" is added to the function block "POU_ex". Two instances of the function block "POU_ex" are created in the main program "PLC_PRG":

PROGRAM PLC_PRG
VAR
  POU1 : POU_ex;
  {attribute 'no-exit'}
  POU2 : POU_ex;
END_VAR

POU1 is called, POU2 is not called.

See also

● "Chapter 1.3.1.24.10 ‘Methods ‘FB_Init’, ‘FB_Reinit’, and ‘FB_Exit’’ on page 653"

Attribute 'noinit'

This pragma is applied to variables that should not be implicitly initialized.

Syntax:

{attribute 'no_init'}

{attribute 'no-init'}

{attribute 'noinit'}

Insertion position: line above the declaration line of the variables concerned in the declaration part.
Example

PROGRAM PLC_PRG
VAR
  A : INT;
  [attribute 'no_init']
  B : INT;
END_VAR

When the associated application is reset, the integer variable A is implicitly re-initialized with 0, whereas the variable B retains its current value.

Attribute 'no_instance_in_retain'

With this attribute you can prevent the instance of a certain function block from being stored in the retain area.

Syntax:

{attribute 'no_instance_in_retain'}

Insertion position:

Line above the FUNCTION_BLOCK declaration in the declaration part of the function block. An error message is then displayed if an instance of the function block is declared as a RETAIN variable.

See also

● Chapter 1.3.1.9.20 “Data persistence” on page 249

Attribute 'no_virtual_actions'

This pragma is used for function blocks that are derived from a function block implemented in SFC and use the fundamental SFC sequence of this base class. The actions called from it exhibit the same virtual behavior as methods. This means that the implementations of the actions in the base class can be replaced by the derived class with its own specific implementations.

If you apply the pragma to the base class, then its actions are protected against overloading.

Syntax:

{attribute 'no_virtual_actions'}

Insertion position: top line in the declaration part of the function block
The function block “POU_SFC” is the base class for the derived function block “POU_child”. The derived class “POU_child” calls the sequence of the base class written in SFC with the special variable SUPER.

The exemplary implementation of this sequence is limited to the initial step, followed by a single step with a linked step action ActiveAction. This step with a linked step action takes care of the configuration of the output variables.

```plaintext
an_int:=an_int+1;    // Counting the action calls
test_act:='father_action';
METH();              // Call of the method METH in order to set the string variable test_meth
```

In the case of the derived class POU_child the step action is replaced by a special implementation of ActiveAction. Active Action differs from the original only by the assignment of the string 'child_action' in place of 'father_action' at the variable test_act.

Likewise, the method “METH”, which assigns the string 'father_method' in the base class of the variable test_meth, is overwritten so that “test_meth” is now given the value 'child_method'. The main program “PLC_PRG” calls an instance of the function block POU_child named ‘Child’. As expected the value of the string reflects the call of action and method of the derived class:
Now, however, you place the pragma `{attribute 'no_virtual_actions'}` in front of the base class:

```plaintext
{attribute 'no_virtual_actions'}
FUNCTION_BLOCK POU_SFC...
```

As a result the behavior changes: while the implementation of the derived class is still used for method METH, the call of the step action now results in a call of the action `ActiveAction` of the base class. Therefore `test_act` is now given the value `'father_action'`:

```plaintext
<table>
<thead>
<tr>
<th>Expression</th>
<th>Type</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Child</td>
<td>POU_child</td>
<td></td>
</tr>
<tr>
<td>test_meth</td>
<td>STRING</td>
<td>'child_method'</td>
</tr>
<tr>
<td>test_act</td>
<td>STRING</td>
<td>'child_action'</td>
</tr>
<tr>
<td>an_int</td>
<td>INT</td>
<td>33</td>
</tr>
</tbody>
</table>
```

### Attribute 'pingroup'

The effect of this pragma is that the input pins or output pins (parameters) are grouped in the declaration of a function block. In the FBD/LD editor a pin group defined in this way can be displayed as an enlarged or reduced unit on the inserted function block. Several groups are possible and are distinguished by their names. CODESYS saves the respective state (reduced) per function block box with the project options.

**Syntax:**

```plaintext
{attribute 'pingroup' := '<group name>'}
```

**Insertion position:** line above the declaration of the input or output variables concerned in the declaration part of a function block.
Two groups are defined: general \((i_1, \text{out}_1)\) and group1 \((i_2, g_1)\). \(r_1, r_2, \text{outRes}_1\) and \(g_2\) are always displayed.

```plaintext
FUNCTION_BLOCK FB
VAR_INPUT
  r1 : REAL;
  \{attribute 'pingroup' := 'general'\}
  i1 : INT;
  \{attribute 'pingroup' := 'group1'\}
  i2 : INT;
  r2 : REAL;
END_VAR

VAR_OUTPUT
  \text{outRes}_1 : REAL;
  \{attribute 'pingroup' := 'general'\}
  \text{out}_1 : INT;
  \{attribute 'pingroup' := 'group1'\}
  g1 : INT;
  g2 : REAL;
END_VAR
```

**Example**

Attribute 'pin_presentation_order_inputs/outputs'

The pragmas are evaluated in the CFC, FBD, and LD graphical editors, causing the order of inputs/outputs of the affected function block to be displayed as specified. You program the order by assigning the names of the inputs/outputs to the attribute in the desired order.

**Syntax**

```plaintext
{attribute 'pin_presentation_order_inputs' := '<First_Input_Name>,
(<Next_Input_Name>,)* ( *, )? (<Next_Input_Name>,)*
<Last_Input_Name>'}
{attribute 'pin_presentation_order_outputs' := '<First_Output_Name>,
(<Next_Output_Name>,)* ( *, )? (<Next_Output_Name>,)*
<Last_Output_Name>'}
```

- **●** The terminal character serves as a wildcard for all inputs/outputs that are not specified in the display order. If the terminal character is missing, then the missing inputs/outputs are appended at the end.
- **●** The contents of the parentheses are optional.
- **●** The contents of the parentheses are optional again and can therefore occur not at all, one time, or several times.
- **●** Insert location: First line in the declaration part of a function block.

**NOTICE!**

This pragma is not evaluated when pragma \{attribute 'pingroup' := '<Group_Name>'\} is used.
Example

{attribute 'pin_presentation_order_inputs' := 'input_2,*,input_1'}
{attribute 'pin_presentation_order_outputs' := 'output_2, output_1'}
FUNCTION_BLOCK POU_BASE
VAR_INPUT
  input_1 : BOOL;
  input_2 : INT;
  input_3 : INT;
  input_4 : INT;
END_VAR

VAR_OUTPUT
  output_1 : BOOL;
  output_2 : INT;
  output_3 : INT;
  output_4 : BOOL;
END_VAR

FUNCTION_BLOCK PLC_PRG
VAR_INPUT
END_VAR
VAR_OUTPUT
END_VAR
VAR
  pouBase_A: POU_BASE;
END_VAR

In the representation of function module instance pouBase_A, the pragmas result in the following arrangement of input and output pins:

![Pin arrangement example]

See also
- ¶ Chapter 1.3.1.24.6.2.29 “Attribute 'pingroup'” on page 626

Attribute 'obsolete'

The effect of this pragma is that a defined warning is displayed for a data type definition during compilation if the data type (structure, function block, etc.) is used in the project. This enables you, for example, to draw attention to the fact that a data type is no longer valid because, for example, an interface has changed and this should also be implemented in the project.

In contrast to a message pragma this warning is defined centrally for all instances of a data type.

Syntax:

{attribute 'obsolete' := 'user defined text'}

Insertion position: line of the data type definition or in a line above it.
**Example**

The pragma is inserted in the definition function block `fb1`:

```plaintext
{attribute 'obsolete' := 'datatype fb1 not valid!'}
FUNCTION_BLOCK fb1
VAR_INPUT
i:INT;
END_VAR
```

If you use `fb1` as a data type, for example in `fbinst:fb1`, the following warning will be displayed when compiling the project: "datatype fb1 not valid".

See also
- Chapter 1.3.1.24.6.1 “Message pragmas” on page 600

**Attribute 'pack_mode'**

This pragma defines how a data structure is packed during the allocation. The attribute must be inserted above the data structure and affects the packing of the entire structure.

**Syntax:**

```plaintext
{attribute 'pack_mode' := '<Value>'}
```

Insert location: above the declaration of the data structure

**Table 26: Possible values for <Value>:**

<table>
<thead>
<tr>
<th>pack_mode</th>
<th>Associated packing method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>aligned,</td>
<td>All variables are allocated to byte addresses. There are no memory gaps.</td>
</tr>
</tbody>
</table>
| 1         | 1-byte-aligned            | There are
- 1-byte variables at byte addresses
- 2 byte variables at addresses divisible by 2. A maximum gap of 1 byte results.
- 4 byte variables at addresses divisible by 2. A maximum gap of 1 byte results.
- 8 byte variables at addresses divisible by 2. A maximum gap of 1 byte results.
- Strings are always at byte addresses. No gaps result. |
| 2         | 2-byte-aligned            | |

PLC Automation with V3 CPUs
Programming with IEC 61131-3 editor > CODESYS Development System
2020/12/10
3ADR010583, 1, en_US
629
<table>
<thead>
<tr>
<th>pack_mode</th>
<th>Associated packing method</th>
<th>Description</th>
</tr>
</thead>
</table>
| 4         | 4-byte-aligned            | There are  
  - 1-byte variables at byte addresses  
  - 2 byte variables at even addresses. A maximum gap of 1 byte results.  
  - 4 byte variables at addresses divisible by 4. A maximum gap of 3 byte results.  
  - 8 byte variables at addresses divisible by 4. A maximum gap of 3 byte results.  
  - Strings are always at byte addresses. No gaps result. |
| 8         | 8-byte-aligned            | There are  
  - 1-byte variables at byte addresses  
  - 2 byte variables at addresses divisible by 2. A maximum gap of 1 byte results.  
  - 4 byte variables at addresses divisible by 4. A maximum gap of 3 byte results.  
  - 8 byte variables at addresses divisible by 8. A maximum gap of 7 byte results.  
  - Strings are always at byte addresses. No gaps result. |

Depending on the structure, there may be no difference in the memory mapping of the individual modes. Therefore, the memory allocation of a structure with pack_mode = 4 can correspond to that of pack_mode = 8.

Arrays of structures: If the structures are combined in arrays, then bytes are added at the end of the structure so that the next structure is aligned.

Example 1

```plaintext
Example

{attribute 'pack_mode' := '1'}

TYPE myStruct:
  STRUCT
    Enable: BOOL;
    Counter: INT;
    MaxSize: BOOL;
    MaxSizeReached: BOOL;
  END_STRUCT
END_TYPE
```

The memory range for a variable of the data type myStruct is allocated 'aligned': If the storage address of its component Enable is 0x0100, for example, then the component Counter follows at the address 0x0101, MaxSize at address 0x0103 and MaxSizeReached at address 0x0104. In the case of 'pack_mode' := 2, Counter would be at 0x0102, MaxSize at 0x0104 and MaxSizeReached at 0x0106.
Example 2

```plaintext
STRUCT
  Var1 : BOOL := 16#01;
  Var2 : BYTE  := 16#11;
  Var3 : WORD  := 16#22;
  Var4 : BYTE  := 16#44;
  Var5 : DWORD := 16#88776655;
  Var6 : BYTE  := 16#99;
  Var7 : BYTE  := 16#AA;
  Var8 : DWORD := 16#AA;
END_TYPE
```

<table>
<thead>
<tr>
<th>pack_mode = 0</th>
<th>pack_mode = 1</th>
<th>pack_mode = 2</th>
<th>pack_mode = 4</th>
<th>pack_mode = 8</th>
</tr>
</thead>
<tbody>
<tr>
<td>Variable</td>
<td>Value</td>
<td>Variable</td>
<td>Value</td>
<td>Variable</td>
</tr>
<tr>
<td>0</td>
<td>Var1</td>
<td>01</td>
<td>Var1</td>
<td>01</td>
</tr>
<tr>
<td>1</td>
<td>Var2</td>
<td>11</td>
<td>Var2</td>
<td>11</td>
</tr>
<tr>
<td>2</td>
<td>Var3</td>
<td>22</td>
<td>Var3</td>
<td>22</td>
</tr>
<tr>
<td>3</td>
<td>...</td>
<td>00</td>
<td>...</td>
<td>00</td>
</tr>
<tr>
<td>4</td>
<td>Var4</td>
<td>44</td>
<td>Var4</td>
<td>44</td>
</tr>
<tr>
<td>5</td>
<td>Var5</td>
<td>55</td>
<td>Var5</td>
<td>55</td>
</tr>
<tr>
<td>6</td>
<td>...</td>
<td>66</td>
<td>...</td>
<td>66</td>
</tr>
<tr>
<td>7</td>
<td>...</td>
<td>77</td>
<td>...</td>
<td>77</td>
</tr>
<tr>
<td>8</td>
<td>...</td>
<td>88</td>
<td>...</td>
<td>88</td>
</tr>
<tr>
<td>9</td>
<td>Var6</td>
<td>99</td>
<td>Var6</td>
<td>99</td>
</tr>
<tr>
<td>10</td>
<td>Var7</td>
<td>AA</td>
<td>Var7</td>
<td>AA</td>
</tr>
<tr>
<td>11</td>
<td>Var8</td>
<td>AA</td>
<td>Var8</td>
<td>AA</td>
</tr>
<tr>
<td>12</td>
<td>...</td>
<td>00</td>
<td>...</td>
<td>00</td>
</tr>
<tr>
<td>13</td>
<td>...</td>
<td>00</td>
<td>...</td>
<td>00</td>
</tr>
<tr>
<td>14</td>
<td>...</td>
<td>00</td>
<td>...</td>
<td>00</td>
</tr>
<tr>
<td>15</td>
<td>...</td>
<td>00</td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td></td>
<td></td>
<td>Var8</td>
<td>AA</td>
</tr>
<tr>
<td>17</td>
<td>...</td>
<td>00</td>
<td>...</td>
<td>00</td>
</tr>
<tr>
<td>18</td>
<td>...</td>
<td>00</td>
<td>...</td>
<td>00</td>
</tr>
<tr>
<td>19</td>
<td>...</td>
<td>00</td>
<td>...</td>
<td>00</td>
</tr>
<tr>
<td>20</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>21</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>22</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>23</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>24</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>25</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>26</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>27</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```
<table>
<thead>
<tr>
<th>pack_mode = 0</th>
<th>pack_mode = 1</th>
<th>pack_mode = 2</th>
<th>pack_mode = 4</th>
<th>pack_mode = 8</th>
</tr>
</thead>
<tbody>
<tr>
<td>Variable</td>
<td>Value</td>
<td>Variable</td>
<td>Value</td>
<td>Variable</td>
</tr>
<tr>
<td>28</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>29</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>30</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>31</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Example 3

```plaintext
STRUCT
Var1 : BYTE := 16#01;
Var2 : LWORD := 16#11;
Var3 : BYTE := 16#22;
Var4 : BYTE := 16#44;
Var5 : DWORD := 16#88776655;
Var6 : BYTE := 16#99;
Var7 : BYTE := 16#AA;
Var8 : WORD := 16#AA;
END_TYPE
```

<table>
<thead>
<tr>
<th>Variable</th>
<th>Value</th>
<th>pack_mode = 0</th>
<th>pack_mode = 1</th>
<th>pack_mode = 2</th>
<th>pack_mode = 4</th>
<th>pack_mode = 8</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Var1</td>
<td>01</td>
<td>01</td>
<td>01</td>
<td>Var1</td>
<td>Var1</td>
</tr>
<tr>
<td>1</td>
<td>Var2</td>
<td>11</td>
<td>11</td>
<td>00</td>
<td>00</td>
<td>00</td>
</tr>
<tr>
<td>2</td>
<td>...</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>3</td>
<td>...</td>
<td>00</td>
<td>00</td>
<td>...</td>
<td>00</td>
<td>00</td>
</tr>
<tr>
<td>4</td>
<td>...</td>
<td>00</td>
<td>00</td>
<td>...</td>
<td>00</td>
<td>00</td>
</tr>
<tr>
<td>5</td>
<td>...</td>
<td>00</td>
<td>00</td>
<td>...</td>
<td>00</td>
<td>00</td>
</tr>
<tr>
<td>6</td>
<td>...</td>
<td>00</td>
<td>00</td>
<td>...</td>
<td>00</td>
<td>00</td>
</tr>
<tr>
<td>7</td>
<td>...</td>
<td>00</td>
<td>00</td>
<td>...</td>
<td>00</td>
<td>00</td>
</tr>
<tr>
<td>8</td>
<td>...</td>
<td>00</td>
<td>00</td>
<td>...</td>
<td>00</td>
<td>00</td>
</tr>
<tr>
<td>9</td>
<td>Var3</td>
<td>22</td>
<td>22</td>
<td>22</td>
<td>00</td>
<td>00</td>
</tr>
<tr>
<td>10</td>
<td>Var4</td>
<td>44</td>
<td>44</td>
<td>44</td>
<td>Var3</td>
<td>Var3</td>
</tr>
<tr>
<td>11</td>
<td>Var5</td>
<td>55</td>
<td>55</td>
<td>55</td>
<td>44</td>
<td>44</td>
</tr>
<tr>
<td>12</td>
<td>...</td>
<td>66</td>
<td>66</td>
<td>66</td>
<td>Var5</td>
<td>Var5</td>
</tr>
<tr>
<td>13</td>
<td>...</td>
<td>77</td>
<td>77</td>
<td>77</td>
<td>66</td>
<td>66</td>
</tr>
<tr>
<td>14</td>
<td>...</td>
<td>88</td>
<td>88</td>
<td>88</td>
<td>77</td>
<td>77</td>
</tr>
<tr>
<td>15</td>
<td>Var6</td>
<td>99</td>
<td>99</td>
<td>99</td>
<td>88</td>
<td>88</td>
</tr>
<tr>
<td>16</td>
<td>Var7</td>
<td>AA</td>
<td>AA</td>
<td>Var6</td>
<td>99</td>
<td>Var5</td>
</tr>
<tr>
<td>17</td>
<td>Var8</td>
<td>AA</td>
<td>AA</td>
<td>Var7</td>
<td>AA</td>
<td>Var4</td>
</tr>
<tr>
<td>18</td>
<td>...</td>
<td>00</td>
<td>00</td>
<td>Var8</td>
<td>AA</td>
<td>Var8</td>
</tr>
<tr>
<td>19</td>
<td>...</td>
<td>00</td>
<td>00</td>
<td>...</td>
<td>88</td>
<td>...</td>
</tr>
<tr>
<td>20</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Var6</td>
<td>Var5</td>
</tr>
<tr>
<td>21</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Var7</td>
<td>AA</td>
</tr>
<tr>
<td>22</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Var8</td>
<td>AA</td>
</tr>
<tr>
<td>23</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>...</td>
<td>88</td>
</tr>
<tr>
<td>24</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Var6</td>
<td>99</td>
</tr>
<tr>
<td>25</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Var7</td>
<td>AA</td>
</tr>
<tr>
<td>26</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Var8</td>
<td>AA</td>
</tr>
<tr>
<td>27</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>...</td>
<td>00</td>
</tr>
</tbody>
</table>
```
Attribute 'qualified_only'

The effect of this pragma is that variables of a global variable list are only addressed by specifying the global variable name, for example `gvl.g_var`. This also applies to variables of the type Enumeration and can be helpful in avoiding being mistaken for local variables.

**Syntax:**

```
{attribute 'qualified_only'}
```

Insertion position: line above `VAR_GLOBAL` in a GVL

**Example**

Global Variable List "GVL":

```
{attribute 'qualified_only'}
VAR_GLOBAL
 iVar: INT;
END_VAR
```

Within a POU, for example "PLC_PRG", the global variable `iVar` can only be addressed using the prefix `GVL`:

```
GVL.iVar:=5;
```

Conversely, the following incomplete call of the variable will create an error:

```
iVar:=5;
```

Attribute 'reflection'

This purpose of this pragma is to identify function blocks in which CODESYS is to search for local STRING variables to which the attribute 'instance-path' is applied.

**Syntax:**

```
{attribute 'reflection'}
```

An example can be found in the description of the attribute 'instance-path'

See also

- § Chapter 1.3.1.24.6.2.18 “Attribute ‘instance-path’” on page 616

Attribute 'to_string'

The pragma affects how the result of converting an enumeration component with the `TO_STRING` operator is output. If the enumeration declaration has the pragma, then the name of the enumeration component appears as a string instead of the numeric value.

**Syntax:**
(attribute 'to_string')

Insert location: First line above the declaration part of the enumeration.

Example

Declaration of the enumeration color:

```
{attribute 'to_string'}
TYPE color :
(
    red := 0,
    blue := 1,
    green := 2
);
END_TYPE
```

Conversion with TO_STRING:

```
PROGRAM PLC_PRG
VAR
    i_color: Color;
    s_show_color: STRING;
END_VAR
i_color := 1;
s_show_color := TO_STRING(i_color);
```

In this case, \texttt{s\textunderscore show\textunderscore color} gets the value 'blue' instead of '1' as the conversion result.

See also

- \(\Phi\) Chapter 1.3.1.24.5.17 “Enumerations” on page 595

Attribute 'subsequent'

This pragma is used to allocate variables to a memory space directly behind one another. If the list changes, the entire variable list is allocated to a new memory space. This pragma is used in programs and global variable lists.

Syntax:

```
{attribute 'subsequent'}
```

\textbf{NOTICE!}

\begin{itemize}
  \item VAR\_TEMP in a program with the attribute 'subsequent' leads to a compiler error.
\end{itemize}

If a variable appears in the list 'RETAIN', the total list is saved as 'RETAIN'.

Attribute 'symbol'

The pragma \{attribute 'symbol'} defines which variables of a program or a global variable list are to be adopted into the symbol configuration. This means that the variables are exported as symbols to a symbol list. This symbol list is then available for external access both as an XML file in the project directory and as a file that is invisible to the user on the target system. For example, the symbol list is then available for access by an OPC server. The variables thus equipped with a symbol are loaded by CODESYS to the controller, even if they are not explicitly configured or visible in the editor of the symbol configuration.
In any case, however, an object “Symbol configuration” must be created below the application concerned in the device tree.

**Syntax:**
```
{attribute 'symbol' := '<access possibilities>'}
```

<access possibilities>: none, read, write, readwrite. The default value readwrite applies if no parameter is specified.

**Insertion position:**
- in order to affect only an individual variable, you must place the pragma in the line before the variable declaration.
- In order to be effective for all variables in the declaration part of a program, you must place the pragma in the first line of the declaration editor. In this case, too, you can still set instructions for individual variables explicitly in the respective line.

### Example
With the following configuration the variables A and B are exported with read and write permission. Variable D is exported with read permission.
```
{attribute 'symbol' := 'readwrite'}
PROGRAM PLC_PRG
VAR
  A : INT;
  B : INT;
{attribute 'symbol' := 'none'}
  C : INT;
{attribute 'symbol' := 'read'}
  D : INT;
END_VAR
```

### See also
- Chapter 1.3.1.9.7 “Using pragmas” on page 211
- Chapter 1.3.1.10.2 “Symbol configuration” on page 275

**Attribute 'warning disable', attribute 'warning restore'**

This pragma causes certain warnings to be suppressed. The `warning restore` pragma causes a suppressed message to be reactivated.

**Syntax:**
```
{warning disable <compiler ID>}
{warning restore <compiler ID>}
```

<compiler ID>: ID located at the beginning of an error or a warning message.

### Example
**Compiler message:**
```
typify code ...
C0196: Implicit conversion from unsigned Type 'UINT' to signed Type 'INT': possible change of sign
Compile complete -- 0 errors
```

Applying the pragma to a variable declaration:
```
VAR
  {warning disable C0195}
  test1 : UINT := -1;
{warning restore C0195}
  test2 : UINT := -1;
END_VAR
```

test1 does not generate an error message, test2 generates an error message.
Attribute 'enable_dynamic_creation'

The pragma `enable_dynamic_creation` is needed for using the `__NEW` operator for function blocks.

**Syntax:**

```
{attribute 'enable_dynamic_creation'}
```

Insert location: First line in the declaration of the function block.

See also

- ¶ Chapter 1.3.1.24.3.58 “Operator '__NEW'” on page 541

Attribute 'ProcessValue'

With the 'ProcessValue' attribute, you mark a component of a structure. In the CFC editor, you can then use the command “Use attributed member as input” in order to connect this structure to an input of scalar type.

**Syntax:**

```
{attribute 'ProcessValue'}
```

Insert location: Line above the affected structure variable.

**Example**

```plaintext
TYPE QINT :
  STRUCT
    Status : STRING;
    {attribute 'ProcessValue'}
    Value1 : INT;
    Value2 : INT;
  END_STRUCT
END_TYPE
```

See also

- ¶ Chapter 1.3.1.25.2.12.36 “Command 'Use Attributed Member as Input'” on page 990

Conditional pragmas

The purpose of conditional pragmas is to influence the generation of code in the pre-compilation process or the compilation process. The ST implementation language supports these pragmas.

**NOTICE!**

They use conditional pragmas in the implementations of POUs. CODESYS does not evaluate these conditional pragmas if you use them in the declaration part.

With conditional pragmas you affect whether implementation code is taken into account for the compilation. For example, you can make this dependent on whether a certain variable is declared, whether a certain function block exists, etc.
<table>
<thead>
<tr>
<th>Pragma</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>{define &lt;identifier&gt;}</td>
<td>The value can be queried and compared later with hasvalue.</td>
</tr>
<tr>
<td>{undefine &lt;identifier&gt;}</td>
<td>The {define} statement of the identifier &lt;identifier&gt; is canceled, and the identifier is 'undefined' again from now on. The pragma is ignored if the specified identifier is not defined at all.</td>
</tr>
<tr>
<td>{IF &lt;expr&gt;}... {ELSIF &lt;expr&gt;}... {ELSE}... {END_IF}</td>
<td>These are pragmas for the conditional compilation. The specified expressions &lt;expr&gt; must be constant at the time of compilation; they are evaluated in the order in which they appear here until one of the expressions indicates a non-zero value. The text linked to the instruction is compiled; the other lines are ignored. The order of the sections is fixed. The ELSIF and ELSE sections are optional. The ELSIF-segments may occur any number of times. You can use several conditional compilation operators within the constants &lt;expr&gt;.</td>
</tr>
<tr>
<td>&lt;expr&gt;</td>
<td>You can use one or more operators within the constant expression &lt;expr&gt; within the conditional compilation pragma {IF} or {ELSIF}.</td>
</tr>
</tbody>
</table>

See also

- § Chapter 1.3.1.9.3.3 “Structured text (ST), extended structured ext (ExST)” on page 200

### Operator defined (<identifier>)

This operator causes the expression to be given the value TRUE. The requirement is that the identifier <identifier> was defined with the help of a {define} instruction and not undefined again afterwards with an {undefine} instruction; otherwise FALSE is returned.

#### Example

Requirement: The applications App1 and App2 exist. The variable pdef1 is defined by a {define} statement in App1, but not in App2.

```plaintext
{IF defined (pdef1)}
(* This code is processed in App1 *)
{info 'pdef1 defined'}
  hugo := hugo + SINT#1;
{ELSE}
(* the following code is only processed in App2 *)
{info 'pdef1 not defined'}
  hugo := hugo - SINT#1;
{END_IF}
```

This also contains an example of a message pragma: Only the message pdef1 defined is displayed in the message view when the application is compiled, because pdef1 is actually defined. The message pdef1 not defined is displayed if pdef1 is not defined.

### Operator defined (variable: <variable>)

This operator causes the expression to be given the value TRUE if the variable <variable> is declared within the current scope; otherwise FALSE is returned.
<table>
<thead>
<tr>
<th>Requirement</th>
<th>Description</th>
</tr>
</thead>
</table>
| The two applications App1 and App2 exist. The variable `g_bTest` is declared in App1, but not in App2. | `{IF defined (variable: g_bTest))
(* the following code is only processed in App2*)
g_bTest := x > 300;
(END_IF)` |
| The operator causes the expression to be given the value TRUE if a data type is declared with the identifier `<identifier>`; otherwise FALSE is returned. | |
| The two applications App1 and App2 exist. The data type `DUT` is declared in App1, but not in App2. | `{IF defined (type: DUT))
(* the following code is only processed in App1*)
bDutDefined := TRUE;
(END_IF)` |
| The operator causes the expression to be given the value TRUE if a function block or an action with name `<pou-name>` exists; otherwise FALSE is returned. | |
| The two applications App1 and App2 exist. The function block `CheckBounds` exists in App1, but not in App2. | `{IF defined (pou: CheckBounds))
(* the following code is only processed in App1 *)
arrTest[CheckBounds(0,i,10)] := arrTest[CheckBounds(0,i,10)] + 1;
ELSE
(* the following code is only processed in App2 *)
arrTest[i] := arrTest[i]+1;
(END_IF)` |
| Not yet implemented! | The operator causes the expression to be given the value TRUE if a task is defined with the name `<identifier>`; otherwise FALSE is returned. |
| The two applications App1 and App2 exist. The task `PLC_PRG_Task` is defined in App1, but not in App2. | `{IF defined (task: PLC_PRG_Task))
(* the following code is only processed in App1 *)
   erg := plc_prg.x;
ELSE
(* the following code is only processed in App2 *)
   erg := prog.x;
(END_IF)` |
Not yet implemented!

The operator causes the expression to be given the value **TRUE** if a resource object with the name `<identifier>` exists for the application; otherwise **FALSE** is returned.

**Example**

Requirement: The two applications App1 and App2 exist. A resource object `glob_var1` of the global variable list exists for App1, but not for App2.

```plaintext
{IF defined (resource:glob_var1)}
(* the following code is only processed in App1 *)
gvar_x := gvar_x + ivar;
{ELSE}
(* the following code is only processed in App2 *)
x := x + ivar;
{END_IF}
```

The operator causes the expression to be given the value **TRUE** if the application runs on a simulated device, i.e. in simulation mode.

The operator causes the expression to be given the value **FALSE**, if the CPU memory is organized in Big Endian (Motorola byte order).

If the expression returns the value **TRUE**, then the code generator produces an FPU code (for the floating-point unit processor) when calculating with **REAL** values. Otherwise CODESYS emulates FPU operations, which is much slower.

**<register size>: Size of a CPU register in bits**

This operator causes the expression to return the value **TRUE** if the size of a CPU register is equal to `<register size>`.

Possible values for `<register size>`

- **16** for 186 and C16x,
- **64** for X86-64 bit
- **32** for X86-32 Bit

The checked pack mode depends on the device description, not on the pragma that can be specified for individual DUTs.

**This operator causes the expression to be given the value **TRUE** if the attribute `<attribute>` is specified in the first line of the declaration part of the function block `<pou name>`; otherwise **FALSE** is returned.**
Requirement: The two applications App1 and App2 exist. The function fun1 is declared in App1 and App2. However, in App1 it is also provided with the pragma {attribute 'vision'}.

**In App1:**

{attribute 'vision'}
FUNCTION fun1 : INT
VAR_INPUT
    i : INT;
END_VAR
VAR
END_VAR

**Function fun1 in App2:**

FUNCTION fun1 : INT
VAR_INPUT
    i : INT;
END_VAR
VAR
END_VAR

**Pragma instruction:**

{IF hasattribute (pou: fun1, 'vision')}
(* the following code is only processed in App1 *)
gervar := fun1(ivar);
{END_IF}

See also

- Chapter 1.3.1.24.6.2.1 “User-defined attributes” on page 602

**Operator: hasattribute**

The operator hasattribute(variable: <variable>, '<attribute>') causes the expression to be given the value TRUE if the pragma {attribute '<attribute>'} is assigned to the variable in the line before the variable declaration; otherwise FALSE is returned.

**Example**

Requirement: The two applications App1 and App2 exist. The variable g_globalInt is used in App1 and App2, but in App1 the attribute 'DoCount' is assigned to it in addition.

**Declaration of g_globalInt in App1**

VAR_GLOBAL
    {attribute 'DoCount'}
    g_globalInt : INT;
    g_multiType : STRING;
END_VAR

**Declaration of g_globalInt in App2**

VAR_GLOBAL
    g_globalInt : INT;
    g_multiType : STRING;
END_VAR

**Pragma instruction:**

{IF hasattribute (variable: g_globalInt, 'DoCount')}
(* the following code is only processed in App1 *)
g_globalInt := g_globalInt + 1;
{END_IF}
This operator causes the expression to be given the value TRUE if the variable <variable> is of the data type <type-spec>; otherwise FALSE is returned.

Possible data types for <type-spec>:

- LREAL
- REAL
- LINT
- DINT
- INT
- SINT
- ULINT
- UDINT
- UINT
- USINT
- TIME
- LWORD
- DWORD
- WORD
- BYTE
- BOOL
- STRING
- WSTRING
- DATE_AND_TIME
- DATE
- TIME_OF_DAY

Example

Requirement: The two applications App1 and App2 exist. The variable g_multitype is declared in App1 with data type LREAL, in App2 with data type STRING.

```plaintext
{IF (hastype (variable: g_multitype, LREAL))
 (* the following code is only processed in App1 *)
  g_multitype := (0.9 + g_multitype) * 1.1;
{ELSiF (hastype (variable: g_multitype, STRING))
 (* the following code is only processed in App2 *)
  g_multitype := 'this is a multitalent';
{END_IF}
```

Operator hasvalue

This operator causes the expression to be given the value TRUE if a variable is defined with the identifier <define-ident> and has the value <char-string>; otherwise FALSE is returned.

```plaintext
Operator hasvalue (define-ident, '<char-string>')
```
### Example

Requirement: The two applications App1 and App2 exist. The variable test is used in the applications App1 and App2; in App1 it is given the value 1, in App2 the value 2.

```plaintext
{IF hasvalue(test,'1')}
(* the following code is only processed in App1 *)
x := x + 1;
{ELSIF hasvalue(test,'2')}
(* the following code is only processed in App2 *)
x := x + 2;
{END_IF}
```

### Operator `hasconstantvalue(<variable>, <literal expression>)`

You can use this operator to query the declared value of a constant.

```plaintext
{IF hasconstantvalue(test,'1')}
(* the following code is only processed in App1 *)
x := x + 1;
{ELSIF hasconstantvalue(test,'2')}
(* the following code is only processed in App2 *)
x := x + 2;
{END_IF}
```

### Example

**Requirement:**

```plaintext
{IF hasconstantvalue(test,'1')}
(* the following code is only processed in App1 *)
x := x + 1;
{ELSIF hasconstantvalue(test,'2')}
(* the following code is only processed in App2 *)
x := x + 2;
{END_IF}
```

### Operator `NOT <operator>`

The expression is given the value TRUE if the reverse value of `<operator>` returns the value TRUE. `<operator>` can be one of the operators described in this chapter.

```plaintext
{IF defined (pou: PLC_PRG1) AND NOT (defined (pou: CheckBounds))}
(* the following code is only processed in App2 *)
bANDNotTest := TRUE;
{END_IF}
```

### Example

**Requirement:**

```plaintext
{IF defined (pou: PLC_PRG1) AND NOT (defined (pou: CheckBounds))}
(* the following code is only processed in App2 *)
bANDNotTest := TRUE;
{END_IF}
```

### Operator `<operator> AND <operator>`

The expression is given the value TRUE if the two specified operators return TRUE. `<operator>` can be one of the operators described in this chapter.

```plaintext
{IF defined (pou: PLC_PRG1) AND (defined (pou: CheckBounds))}
(* the following code is only processed in App1 *)
bANDTest := TRUE;
{END_IF}
```

### Example

**Requirement:**

```plaintext
{IF defined (pou: PLC_PRG1) AND (defined (pou: CheckBounds))}
(* the following code is only processed in App1 *)
bANDTest := TRUE;
{END_IF}
```
The expression returns TRUE if one of the two specified operators returns TRUE. `<operator>` can be one of the operators described in this chapter.

Example

Requirement: The two applications App1 and App2 exist. The POU PLC_PRG1 exists in App1 and App2, and the POU CheckBounds exists only in App1.

```plaintext
{IF defined (pou: PLC_PRG1) OR (defined (pou: CheckBounds))} (* the following code is only processed in App1 and in App2 *)
bORTest := TRUE;
{END_IF}
```

Operator

() parenthesizes the operators.

See also

- § Chapter 1.3.1.9.7 “Using pragmas” on page 211
- § Chapter 1.3.1.24.2.1 “User-defined attributes” on page 602

Region pragma

This pragma is used for grouping several lines into one block in a text editor. The block can be named. Region pragmas can also be nested.

Code with region pragma: Expanded and collapsed views

1.3.1.24.7 Identifiers

Rules for identifier designation

- An identifier must not contain spaces or special characters.
- Capitalization is ignored. For example, VAR1 and var1 refer to the same variable.
- The underscore is recognized. For example, A_BCD and AB_CD are treated as two different identifiers. Multiple consecutive underscores are not permitted.
- The length of an identifier is unlimited.

Rules for multiple use of identifiers (namespaces)

- An identifier must not be declared two times locally.
- An identifier can be used more than one time globally. If a local variable has the same name as a global variable, then the local variable has priority within the POU.
- An identifier must not be identical to a keyword, such as the VAR_global scope.
● A variable that is declared in a global variable list can have the same name as a variable defined in another GVL. CODESYS provides features that extend the standard for the namespace or scope of variables:
  – Global namespace operator:
    An instance path that begins with a dot always opens a global namespace. If there is a local variable (for example, `ivar`) that has the same name as a global variable, then you refer to the global variable as `.ivar`.
  – The name of a global variable list can define the namespace uniquely for the include variables. Therefore, you can declare variables with the same name in different global variables list and still uniquely reference by prepending the list name.
    For example, `globlist1.ivar := globlist2.ivar; (* ivar from GVL globlist2 is copied to ivar in GVL globlist1 *)`.
  – Variables that are defined in the global variable list of a library included in the project can be addressed uniquely according to the following syntax:
    `<name scope library>.< GVL name>.<variable name>`
    For example, `globlist1.ivar := lib1.globlist1.ivar (* ivar from GVL globlist1 in library lib1 is copied to ivar in GVL globlist1 *)`.

● When inserting a library, you also use the library manager to define a namespace. In this way, you can make unique references to library blocks and library via `<namespace library>.<block name|variable name>`. Please note that when libraries are nested, you must reference the namespaces of all libraries are in succession
    Example: If `Lib1` is referenced by `Lib0`, then the block `func` in `Lib1` is addressed via `Lib0.Lib1.fun:ivar := Lib0.Lib1.fun(4, 5); (* return value from func is copied to variable ivar in the project *)`

We recommend that you apply the following rules in addition to the items that you have to consider specifically for variables declaration. By doing this, you will achieve the best possible harmonization when assigning names.

**Recommendations for variable names**
Whenever possible, you should name variables in Hungarian notation in applications and libraries. Find a meaningful, short, English name for each variable as a base name, which can consist of several words. Write the first letter of each word in uppercase, the remaining letters in lowercase. In front of the base name, append a prefix in lowercase to indicate the data type of the variable.

**Example:** `iFileSize : INT;`

<table>
<thead>
<tr>
<th>Data type</th>
<th>Prefix</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>BOOL</td>
<td>x</td>
<td>We intentionally recommend <code>x</code> as the prefix for Boolean variables in order to distinguish them from identifiers of the data type <code>BYTE</code>. The prefix indicates the view of an IEC programmer.</td>
</tr>
<tr>
<td>BYTE</td>
<td>by</td>
<td>Reserved</td>
</tr>
<tr>
<td>WORD</td>
<td>w</td>
<td>Bit string; not for arithmetical operations</td>
</tr>
<tr>
<td>DWORD</td>
<td>dw</td>
<td>Bit string; not for arithmetical operations</td>
</tr>
<tr>
<td>LWORD</td>
<td>lw</td>
<td>Bit string; not for arithmetical operations</td>
</tr>
<tr>
<td>SINT</td>
<td>si</td>
<td></td>
</tr>
</tbody>
</table>
### Data type

<table>
<thead>
<tr>
<th>Data type</th>
<th>Prefix</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>USINT</td>
<td>usi</td>
<td></td>
</tr>
<tr>
<td>INT</td>
<td>i</td>
<td></td>
</tr>
<tr>
<td>UINT</td>
<td>ui</td>
<td></td>
</tr>
<tr>
<td>DINT</td>
<td>di</td>
<td></td>
</tr>
<tr>
<td>UDINT</td>
<td>udi</td>
<td></td>
</tr>
<tr>
<td>LINT</td>
<td>li</td>
<td></td>
</tr>
<tr>
<td>UUINT</td>
<td>uli</td>
<td></td>
</tr>
<tr>
<td>REAL</td>
<td>r</td>
<td></td>
</tr>
<tr>
<td>LREAL</td>
<td>lr</td>
<td></td>
</tr>
<tr>
<td>STRING</td>
<td>s</td>
<td></td>
</tr>
<tr>
<td>WSTRING</td>
<td>ws</td>
<td></td>
</tr>
<tr>
<td>TIME</td>
<td>tim</td>
<td></td>
</tr>
<tr>
<td>LTIME</td>
<td>ltim</td>
<td></td>
</tr>
<tr>
<td>TIME_OF_DAY</td>
<td>tod</td>
<td></td>
</tr>
<tr>
<td>DATE_AND_TIME</td>
<td>dt</td>
<td></td>
</tr>
<tr>
<td>DATE</td>
<td>date</td>
<td></td>
</tr>
<tr>
<td>POINTER</td>
<td>p</td>
<td></td>
</tr>
<tr>
<td>ARRAY</td>
<td>a</td>
<td></td>
</tr>
<tr>
<td>Enumeration</td>
<td>e</td>
<td></td>
</tr>
</tbody>
</table>

### Example

```plaintext
VAR
    bySubIndex: BYTE;
    xFlag: BOOL;
    udiCounter: UDINT;
END_VAR
```

### Identifier

<table>
<thead>
<tr>
<th>Identifier</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nested declaration</td>
<td>The prefixes are appended in the order of declaration.</td>
<td>pabyTelegramData: POINTER TO ARRAY [0..7] OF BYTE;</td>
</tr>
</tbody>
</table>
| Function block instance         | Prefix: Abbreviation for the name of the function block or data type. | cansdoReceivedTelegram: CAN_SDOTelegram; TYPE CAN_SDOTelegram : (* prefix: sdo *) STRUCT
  □ wIndex: WORD;
  □ bySubIndex: BYTE;
  □ byLen: BYTE;
  □ aby: ARRAY [0..3] OF BYTE;
END_STRUCT
END_TYPE                          |
<table>
<thead>
<tr>
<th>Identifier</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Local constant</td>
<td>Prefix: c_ followed by the type prefix and variable name</td>
<td>VAR CONSTANT</td>
</tr>
<tr>
<td>Local constant variable</td>
<td></td>
<td>c_uiSyncID: UINT := 16#80; END_VAR</td>
</tr>
<tr>
<td>Global variable</td>
<td>An additional prefix is appended to the library prefix. g_</td>
<td>VAR_GLOBAL</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CAN_g_iTest: INT; END_VAR</td>
</tr>
<tr>
<td>Global constants</td>
<td>An additional prefix is appended to the library prefix. gc_</td>
<td>VAR_GLOBAL CONSTANT</td>
</tr>
<tr>
<td>Global constant variable</td>
<td></td>
<td>CAN_gc_dwExample: DWORD; END_VAR</td>
</tr>
</tbody>
</table>

Recommendations for variable names
CODESYS V3.x libraries

<table>
<thead>
<tr>
<th>Identifier for</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Variable names</td>
<td>Corresponds to the description for variable names, with the exception that global variables and constants do not require library prefixes because the namespace replaces the function.</td>
<td>g_iTest: INT; (* declaration *) CAN.g_iTest (usage, call in the program)</td>
</tr>
<tr>
<td>Identifier for</td>
<td>Description</td>
<td>Example</td>
</tr>
<tr>
<td>---------------</td>
<td>-------------</td>
<td>---------</td>
</tr>
</tbody>
</table>
| Structures    | Library prefix followed by an underscore and a short, informative description of the structure. The associated prefix for created variables of this structure should follow the colon as a comment. | TYPE CAN_SDOTelegram : (* prefix: sdo *) STRUCT
    wIndex : WORD;
    bySubIndex : BYTE;
    byLen : BYTE;
    abyData: ARRAY [0..3] OF BYTE; END_STRUCT END_TYPE |
| Enumerations  | Library prefix followed by an underscore and the identifier in uppercase. Please note: In past CODESYS versions, enumeration values > 16#7FFF caused errors because they were not automatically converted to INT. Therefore, enumerations should always be defined with correct INT values. | TYPE CAL_Day :
    (CAL_MONDAY,
     CAL_TUESDAY,
     CAL_WEDNESDAY,
     CAL_THURSDAY,
     CAL_FRIDAY,
     CAL_SATURDAY,
     CAL_SUNDAY
    );
Declaration:
eToday: CAL_Day; |
### Identifier for

**DUT names in CODESYS V3 libraries**

The namespace replaces the need for the library prefix. Therefore, it is omitted. Enumeration values are also defined without a library prefix.

<table>
<thead>
<tr>
<th>Identification</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Library with namespace CAL</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TYPE DAY :</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>MONDAY</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>TUESDAY,</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>WEDNESDAY,</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>THURSDAY,</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>FRIDAY,</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>SATURDAY,</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>SUNDAY</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Declaration:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>eToday: CAL.Day;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Use in the application</td>
<td></td>
<td></td>
</tr>
<tr>
<td>IF eToday = CAL.Day.MONDAY THEN</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Recommendations for identifiers for POUs, functions, function blocks, programs

<table>
<thead>
<tr>
<th>Identifier for</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>POUs: Functions, function blocks, programs</strong></td>
<td>Library prefix followed by an underscore and a short, informative POU name. Like for variables, the first letter of each word is uppercase and all other letters are lowercase. We recommend that you compose the POU name from a verb and a noun. For function blocks, the associated prefix for created instances should follow the name as a comment.</td>
<td>FUNCTION_BLOCK CAN_SendTelegram (* prefix: canst *)</td>
</tr>
<tr>
<td><strong>Actions</strong></td>
<td>Only actions that the block itself calls, beginning with prv_. Otherwise, actions do not contain prefixes.</td>
<td></td>
</tr>
</tbody>
</table>
### Identifier for

<table>
<thead>
<tr>
<th>Identifier for</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>POU</td>
<td>The namespace replaces the need for the library prefix. Therefore, it is omitted.</td>
<td>FUNCTION_BLOCK SendTelegram (* prefix: canst *)</td>
</tr>
<tr>
<td>Method</td>
<td>Only methods that the block itself calls, beginning with <code>prv_</code>. Otherwise, methods do not contain prefixes.</td>
<td></td>
</tr>
<tr>
<td>Interface</td>
<td>I</td>
<td>ICANDevice</td>
</tr>
</tbody>
</table>

### Recommendations for identifiers for visualizations

**NOTICE!**

Please note that a visualization is not named the same as another block in the project because this may cause problems when changing visualizations.

See also
- *Chapter 1.3.1.9.2 “Declaration of variables” on page 172*
- *Chapter 1.3.1.24.5 “Data types” on page 569*
- *Chapter 1.3.1.24.2 “Variables” on page 489*

### 1.3.1.24.8 Shadowing rules

In CODESYS, you are generally allowed to use the same identifier for different elements. For example, a POU and a variable can be named the same. However, you should avoid this practice in order to prevent confusion.

Negative example: In the following code snippet, a local function block instance has the same name as a function:

**Example**

```plaintext
FUNCTION YYY : INT
;
END_FUNCTION

FUNCTION_BLOCK XXX
;
END_FUNCTION_BLOCK

PROGRAM PLC_PRG
VAR
    YYY : XXX;
END_VAR
YYY();
END_PROGRAM
```

In such a case as this, it is unclear whether the instance or the function is called in the program.

To make sure that names are always unique, you should follow naming conventions, such as certain prefixes for variables. Rules for assigning identifiers can be found in the "Identifiers" section of the help.
Naming conventions can be checked automatically using the static code analysis of CODESYS. Static code analysis could also detect the duplicate use of the name `YYY` and report it as an error.

The consistent use of the attribute `qualified_only` for enumerations and global variable lists and the use of qualified libraries can also prevent ambiguous situations.

**Shadowing:** The compiler does not report any errors or warnings if the same identifier is used for different elements. Instead, the compiler searches the code in a specific order for the declaration of the identifier. If a declaration is found, then the compiler does not search for any other declarations elsewhere. If other declarations do exist, then they are "shadowed" for the compiler. The following section describes the shadowing rules (that is, the search order that the compiler uses when searching for the declaration for identifiers). The section "Ambiguous access and qualified access" provides ways to prevent ambiguous access and bypass shadowing rules.

### Search order in the application

When the compiler encounters a single identifier in the code of an application, it searches for the corresponding declaration in the following order:

1. Local variables of a method
2. Local variables in the function block, program, or function, and in any base function blocks
3. Local methods of the POU
4. Global variables in the application, if the `qualified_only` attribute is not set in the variable list where the global variables are declared
5. Global variables in a parent application, if the `qualified_only` attribute is not set in the variable list where the global variables are declared
6. Global variables in referred libraries when neither the library nor the variable list requires qualified access
7. POU or type names from the application (that is, names of global variable lists, function blocks, and so on)
8. POU or type names from a parent application
9. POU or type names from a library
10. Namespaces of locally referred libraries and libraries that are published by libraries
11. Global variables in the “POUs” view, unless the `qualified_only` attribute is set in the variable list where they are declared
12. POU or type names from the “POUs” view (that is, names of global variable lists, function blocks, and so on)

Libraries that are inserted in the Library Manager of the “POUs” view are mirrored in the Library Manager in all applications in the project with the appropriate placeholder resolution. These libraries then form a common namespace with the libraries in the application. Therefore, there is no shadowing of libraries in the pool by libraries in the application.

### Search order in the library

When the compiler encounters a single identifier in the code of a library, it searches for the corresponding declaration in the following order:

1. Local variables of a method
2. Local variables in the function block, program, or function, and in any base function blocks
3. Local methods of the POU
4. Global variables in the local library, if the `qualified_only` attribute is not set in the variable list where the global variables are declared
5. Global variables in referred libraries when neither the library nor the variable list requires qualified access
6. POU or type names from the local library (that is, names of global variable lists, function blocks, and so on)
7. POU or type names from a referred library
8. Namespaces of locally referred libraries and libraries that are published by locally refereed libraries

**Ambiguous access and qualified access**

Despite these search orders, ambiguous access can still occur. For example, this is the case when a variable with the same name exists in two global variable lists that do not require qualified access. Such a case is reported by the compiler as an error (for example: ambiguous use of the name XXX).

This kind of ambiguous use can be made unique by means of qualified access, for example by accessing via the name of the global variable list (example: GVL.XXX).

Qualified access can also always be used to avoid shadowing rules.
- The name of the global variable list can be used to uniquely access a variable in the list.
- The name of a library can be used to uniquely access elements in the library.
- The THIS pointer be used to uniquely access variables in a function block, even if a local variable with the same name exists in a method of the function block.

To find the declaration location of an identifier at any time, use the command "Edit ➔ Browse ➔ Go to Definition". This can be especially helpful if the compiler produces an apparently obscure error message.

**Searching in instance paths**

The search orders described above do not apply to identifiers that exist as components in an instance path or to identifiers that are used as inputs in calls.

For access of the following type yy.component, it depends on the entity described by yy where the declaration of component is searched for.

If yy denotes a variable with a structured data type (that is, type STRUCT or UNION), then component is searched for in the following order:
- Local variables of the function block
- Local variables of the basic function block
- Methods of the function block
- Methods of the basic function block

If yy denotes a global variable list or a program, then component is searched for in this list only.

If yy denotes a namespace of a library, then component is searched for in this library exactly as described in the section above "Search order in the library".

Only in the second instance does the compiler decide whether access to the found element is granted (that is, whether the variable is only locally accessible, or whether a method is private). If access is not allowed, an error is issued.

See also
- § Chapter 1.3.1.24.7 “Identifiers” on page 644
- § Chapter 1.3.1.24.6.2.33 “Attribute qualified_only” on page 634
- § Chapter 1.3.1.24.2.15 “THIS” on page 502

**1.3.1.24.9 Keywords**

In all editors, you must capitalize keywords that for example denote scopes, data types, or operators.

Keywords cannot be used as variable names.
When CODESYS creates implicit code, variables and functions are generally given a name that is prepended with two underscores (__). The use of double underscores in the implementation code is prevented automatically. This eliminates conflicts between internal system identifiers and identifiers assigned by the programmer.

The following keywords are used in the CODESYS export format. Therefore, you may not use them as identifiers:

- ACTION
- END_ACTION
- END_FUNCTION
- END_FUNCTION_BLOCK
- END_PROGRAM

Other valid keywords:

- VAR_ACCESS
- READ_ONLY
- READ_WRITE
- PARAMS

**1.3.1.24.10 Methods 'FB_Init', 'FB_Reinit', and 'FB.Exit'**

You can declare the methods explicitly in order to influence the initialization of function block variables, as well as the behavior when exiting function blocks.

The type of the return value for the implicit methods is **BOOL**. The value is not evaluated by the system, but the type should not be changed.

**FB_Init** is always available implicitly and it is used primarily for initialization. For a specific influence, you can also declare the methods explicitly and provide additional code there with the standard initialization code.

**FB_Reinit** must be implemented explicitly. If this method exists, then it is called after the instance of the affected function block is copied. That happens during an online change after changes to the function block declaration (signature change) in order to reinitialize the new instance module. To reinitialize the basic implementation of the function block, you must call **FB_Reinit** explicitly.
FB Exit must be implemented explicitly. If there is an implementation, then the method is called before the controller removes the code of the function block instance (implicit call).

The following shows some use cases of these methods for different operating conditions.

Operating condition "First download"

When downloading an application to a PLC with factory settings, the memory of all variables must be offset to the required initial state. In this way, the data areas of function block instances are assigned the required values. By the explicit implementation of FB_Init for function blocks, you can react specifically to this situation in the application code. By evaluating the method parameters bInCopyCode (FALSE) and bInitRetains (TRUE), you can detect this operating condition clearly. (See "Operating condition "Online Change"" and "Operating condition "Re-download"".)

Within the scope of the online change, you can influence the initialization of function block instances by means of the methods FB Exit, FB_Init, and FB_Reinit. During the online change, the changes to the application that were made in offline mode are applied in the running PLC. This is the reason that the old instances of the function blocks are replaced by new instances as much as possible without incident. If no changes were made to the declaration part of a function block in the application before login, but in the implementation only, then the data areas are not replaced. Only code blocks are replaced. Then the methods FB Exit, FB_Init, and FB_Reinit are not called.

In the code of the FB_Init method, the parameter bInCopyCode (TRUE) can be evaluated to detect whether or not an online change is being executed.

The following calls occur in succession during an online change:

1. FB Exit

   old_inst.FB Exit(bInCopyCode := TRUE);

   You can call FB Exit when exiting the old instance in order to trigger specific cleanup tasks before the copy operation. In this way, you can prepare the data for the following copy operation and influence the state of the new instance. You can notify other parts of the application about the pending change in location in the memory. Pay special attention to the variables of type POINTER and REFERENCE. These may no longer refer to the required memory locations after the online change. Interface variables (INTERFACE) are handled separately by the compiler and they are adapted accordingly during the online change. External resources such as sockets, files, or other handles can be applied by the new instance, in some case unchanged. Often they do not have to be treated specially during an online change. (See "Operating condition "Re-download""

2. FB_Init

   new_inst.FB_Init(bInitRetains := FALSE, bInCopyCode := TRUE);

   FB Init is called before the copy operation and can be used in order to execute specific operations for the online change. For example, you can initialize variables accordingly at the new location in the memory, or notify other parts of the application about the new location of specific variables in the memory.

3. Copy operation: copy

   copy(&old_inst, &new_inst);

   Existing values remain unchanged. For this purpose, they are copied from the old instance into the new instance.
4. `FB_Reinit`
   `new_inst.FB_Reinit();`

   This method is called after the copy operation and should set defined values for the variables of the instance. For example, you can initialize variables accordingly at the new location in the memory, or notify other parts of the application about the new location of specific variables in the memory. Design the implementation independent of the online change. The method can also be called from the application at any time in order to reset a function block instance to its original state.

   **With the `{attribute 'no_copy'}` attribute, you can prevent that this is copied during the online change for a single variable of the function block. It always retains the initial value.**

See also

- § Chapter 1.3.1.25.2.3.19 “Command ‘Settings of Memory Reserve for Online Change’ ” on page 894

### Operating condition "New download"

When downloading an application, an existing application may be replaced on the PLC. Therefore, the provision of memory for the present function blocks must be regulated. You can use the `FB_Exit` method for implementing the required steps for this. For example, you can offset external resources (with socket and file handles) in a defined state.

You can detect this operating condition by checking whether or not the parameter `bInCopyCode = FALSE` for the `FB_Exit` method.

### Operating condition "Start of application"

The initial assignments are processed before the first cycle of the application tasks.

**Example**

```
T1 : TON := (PT:=t#500ms);
```

These kinds of assignments are executed only after calling `FB_Init`. In order to control the effects of these assignments, you can provide a function block or a method of a function block with the `{attribute 'call_after_init'}` attribute. You must add the attribute above the declaration part of the function block body and above the declaration part of the corresponding method. A POU that extends another POU which uses the `{attribute 'call_after_init'}` attribute must also have the attribute. For the benefit of clarity, we recommend that the corresponding methods are overwritten with the same name, the same signature, and the same attribute. This requires calling `SUPER^.MyInit`. The name of the method can be chosen without restriction. (Exceptions: `FB_Init`, `FB_Reinit`, and `FB_Exit`). The method is called after processing the initial assignments and before starting the application tasks. Therefore, the method can react to user input.

When using `FB_Init` or `{attribute 'call_after_init'}`, remember that detecting errors in the `FB_Init` method or in methods decorated with the `{attribute 'call_after_init'}` attribute is tedious, because the setting of breakpoints may not have the expected effect.

**NOTICE!**

If the explicitly defined initialization code is reached during execution, then the function block instance is already completely initialized via the implicit initialization code. Therefore, there must not be a `SUPER^.FB_Init` call.
**NOTICE!**

FB_Init replaces the INI operator used in CoDeSys V2.3. The methods cannot be compared to the design of a constructor, such as in C#, C++, or Java. This has consequences for function blocks that extend other function blocks. (See below: "Derived function blocks")

### Interface of method FB_Init

METHOD FB_Init : BOOL
VAR_INPUT
  bInitRetains : BOOL; // TRUE: the retain variables are initialized (reset warm / reset cold)
  bInCopyCode : BOOL; // TRUE: the instance will be copied to the copy code afterward (online change)
END_VAR

You can declare additional function block inputs in an FB_init method. Then you have to set these inputs in the declaration of the function block instance.

### Example Method FB_Init for the serialdevice function block

METHOD PUBLIC FB_Init : BOOL
VAR_INPUT
  bInitRetains : BOOL; // initializing of retain variable
  bInCopyCode : BOOL; // instance is copied to copy code
  iCOMnum : INT; // additional input: number of the COM interface, that is to be observed
END_VAR

Instantiation of the serialdevice function block:

```plaintext
com1: serialdevice(iCOMnum:=1);
com0: serialdevice(iCOMnum:=0);
```

### Interface of method FB_Reinit

METHOD FB_Reinit : BOOL

There is the mandatory parameter bInCopyCode.

### Interface of method FB_Exit

METHOD FB_Exit : BOOL
VAR_INPUT
  bInCopyCode : BOOL; // TRUE: the exit method is called in order to leave the instance which will be copied afterwards (online change).
END_VAR
If a function block is derived from another function block, then the `FB_Init` method of the derived function block must define the same parameters as the `FB_Init` method of the basic function block. However, you can add further parameters in order to set up a special initialization for the instance.

The function blocks `MainFB`, `SubFB`, and `SubSubFB` are derived from each other. Therefore, `SubFB EXTENDS MainFB` and `SubSubFB EXTENDS SubFB` apply.

### Example

<table>
<thead>
<tr>
<th>Calling order of methods</th>
<th>FB Exit and FB_Init:</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. fbSubSubFb.FB_Exit(...)</td>
<td></td>
</tr>
<tr>
<td>2. fbSubFb.FB_Exit(...)</td>
<td></td>
</tr>
<tr>
<td>3. fbMainFb.FB_Exit(...)</td>
<td></td>
</tr>
<tr>
<td>4. fbMainFb.FB_Init(...)</td>
<td></td>
</tr>
<tr>
<td>5. fbSubFb.FB_Init(...)</td>
<td></td>
</tr>
<tr>
<td>6. fbSubSubFb.FB_Init(...)</td>
<td></td>
</tr>
</tbody>
</table>

See also
- `Chapter 1.3.1.25.1.28.5 “Object ’Method’” on page 784`
- `Chapter 1.3.1.24.6.2.3 “Attribute ’call_after_init’” on page 603`
- `Chapter 1.3.1.24.6.2.24 “Attribute ’no_copy’” on page 623`
- `Chapter 1.3.1.24.2.14 “SUPER” on page 501`
1.3.1.24.11 Error messages

1.3.1.24.11.1 Compiler error C0001................................................................. 660
1.3.1.24.11.2 Compiler error C0002................................................................. 661
1.3.1.24.11.3 Compiler error C0003................................................................. 661
1.3.1.24.11.4 Compiler Error C0004................................................................. 661
1.3.1.24.11.5 Compiler error C0005................................................................. 662
1.3.1.24.11.6 Compiler error C0006................................................................. 662
1.3.1.24.11.7 Compiler error C0007................................................................. 662
1.3.1.24.11.8 Compiler error C0008................................................................. 663
1.3.1.24.11.9 Compiler error C0009................................................................. 663
1.3.1.24.11.10 Compiler error C0010................................................................. 663
1.3.1.24.11.11 Compiler error C0011................................................................. 664
1.3.1.24.11.12 Compiler error C0013................................................................. 664
1.3.1.24.11.13 Compiler error C0016................................................................. 664
1.3.1.24.11.14 Compiler error C0018................................................................. 665
1.3.1.24.11.15 Compiler error C0020................................................................. 665
1.3.1.24.11.16 Compiler error C0022................................................................. 666
1.3.1.24.11.17 Compiler error C0023................................................................. 666
1.3.1.24.11.18 Compiler error C0026................................................................. 666
1.3.1.24.11.19 Compiler error C0027................................................................. 667
1.3.1.24.11.20 Compiler error C0030................................................................. 667
1.3.1.24.11.21 Compiler error C0031................................................................. 667
1.3.1.24.11.22 Compiler error C0032................................................................. 668
1.3.1.24.11.23 Compiler error C0035................................................................. 668
1.3.1.24.11.24 Compiler error C0037................................................................. 669
1.3.1.24.11.25 Compiler error C0038................................................................. 669
1.3.1.24.11.26 Compiler error C0039................................................................. 669
1.3.1.24.11.27 Compiler error C0040................................................................. 670
1.3.1.24.11.28 Compiler error C0041................................................................. 670
1.3.1.24.11.29 Compiler error C0043................................................................. 671
1.3.1.24.11.30 Compiler error C0044................................................................. 671
1.3.1.24.11.31 Compiler error C0045................................................................. 672
1.3.1.24.11.32 Compiler error C0046................................................................. 672
1.3.1.24.11.33 Compiler error C0047................................................................. 672
1.3.1.24.11.34 Compiler error C0048................................................................. 673
1.3.1.24.11.35 Compiler error C0049................................................................. 673
1.3.1.24.11.36 Compiler error C0050................................................................. 673
1.3.1.24.11.37 Compiler error C0061................................................................. 674
1.3.1.24.11.38 Compiler error C0062................................................................. 674
1.3.1.24.11.39 Compiler error C0064................................................................. 675
1.3.1.24.11.40 Compiler error C0066................................................................. 675
1.3.1.24.11.41 Compiler error C0068................................................................. 676
1.3.1.24.11.42 Compiler error C0069................................................................. 676
1.3.1.24.11.43 Compiler error C0070................................................................. 676
1.3.1.24.11.44 Compiler error C0072................................................................. 677
1.3.1.24.11.45 Compiler error C0074................................................................. 677
1.3.1.24.11.46 Compiler error C0075................................................................. 677
1.3.1.24.11.47 Compiler error C0076................................................................. 678
1.3.1.24.11.48 Compiler error C0077................................................................. 678
1.3.1.24.11.49 Compiler error C0080................................................................. 678
1.3.1.24.11.50 Compiler error C0081................................................................. 679
<table>
<thead>
<tr>
<th>Line</th>
<th>Error Code</th>
<th>Source File</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.3.1.24.11.51</td>
<td>Compiler error C0082.</td>
<td>...</td>
</tr>
<tr>
<td>1.3.1.24.11.52</td>
<td>Compiler error C0084.</td>
<td>...</td>
</tr>
<tr>
<td>1.3.1.24.11.53</td>
<td>Compiler error C0086.</td>
<td>...</td>
</tr>
<tr>
<td>1.3.1.24.11.54</td>
<td>Compiler error C0087.</td>
<td>...</td>
</tr>
<tr>
<td>1.3.1.24.11.55</td>
<td>Compiler error C0089.</td>
<td>...</td>
</tr>
<tr>
<td>1.3.1.24.11.56</td>
<td>Compiler error C0090.</td>
<td>...</td>
</tr>
<tr>
<td>1.3.1.24.11.57</td>
<td>Compiler error C0091.</td>
<td>...</td>
</tr>
<tr>
<td>1.3.1.24.11.58</td>
<td>Compiler error C0094.</td>
<td>...</td>
</tr>
<tr>
<td>1.3.1.24.11.59</td>
<td>Compiler error C0096.</td>
<td>...</td>
</tr>
<tr>
<td>1.3.1.24.11.60</td>
<td>Compiler error C0097.</td>
<td>...</td>
</tr>
<tr>
<td>1.3.1.24.11.61</td>
<td>Compiler error C0098.</td>
<td>...</td>
</tr>
<tr>
<td>1.3.1.24.11.62</td>
<td>Compiler error C0101.</td>
<td>...</td>
</tr>
<tr>
<td>1.3.1.24.11.63</td>
<td>Compiler error C0102.</td>
<td>...</td>
</tr>
<tr>
<td>1.3.1.24.11.64</td>
<td>Compiler error C0104.</td>
<td>...</td>
</tr>
<tr>
<td>1.3.1.24.11.65</td>
<td>Compiler error C0114.</td>
<td>...</td>
</tr>
<tr>
<td>1.3.1.24.11.66</td>
<td>Compiler error C0116.</td>
<td>...</td>
</tr>
<tr>
<td>1.3.1.24.11.67</td>
<td>Compiler error C0117.</td>
<td>...</td>
</tr>
<tr>
<td>1.3.1.24.11.68</td>
<td>Compiler error C0118.</td>
<td>...</td>
</tr>
<tr>
<td>1.3.1.24.11.69</td>
<td>Compiler error C0119.</td>
<td>...</td>
</tr>
<tr>
<td>1.3.1.24.11.70</td>
<td>Compiler error C0120.</td>
<td>...</td>
</tr>
<tr>
<td>1.3.1.24.11.71</td>
<td>Compiler error C0122.</td>
<td>...</td>
</tr>
<tr>
<td>1.3.1.24.11.72</td>
<td>Compiler error C0124.</td>
<td>...</td>
</tr>
<tr>
<td>1.3.1.24.11.73</td>
<td>Compiler error C0125.</td>
<td>...</td>
</tr>
<tr>
<td>1.3.1.24.11.74</td>
<td>Compiler error C0126.</td>
<td>...</td>
</tr>
<tr>
<td>1.3.1.24.11.75</td>
<td>Compiler error C0130.</td>
<td>...</td>
</tr>
<tr>
<td>1.3.1.24.11.76</td>
<td>Compiler error C0131.</td>
<td>...</td>
</tr>
<tr>
<td>1.3.1.24.11.77</td>
<td>Compiler error C0132.</td>
<td>...</td>
</tr>
<tr>
<td>1.3.1.24.11.78</td>
<td>Compiler error C0136.</td>
<td>...</td>
</tr>
<tr>
<td>1.3.1.24.11.79</td>
<td>Compiler error C0139.</td>
<td>...</td>
</tr>
<tr>
<td>1.3.1.24.11.80</td>
<td>Compiler error C0140.</td>
<td>...</td>
</tr>
<tr>
<td>1.3.1.24.11.81</td>
<td>Compiler error C0141.</td>
<td>...</td>
</tr>
<tr>
<td>1.3.1.24.11.82</td>
<td>Compiler error C0142.</td>
<td>...</td>
</tr>
<tr>
<td>1.3.1.24.11.83</td>
<td>Compiler error C0143.</td>
<td>...</td>
</tr>
<tr>
<td>1.3.1.24.11.84</td>
<td>Compiler error C0144.</td>
<td>...</td>
</tr>
<tr>
<td>1.3.1.24.11.85</td>
<td>Compiler error C0145.</td>
<td>...</td>
</tr>
<tr>
<td>1.3.1.24.11.86</td>
<td>Compiler error C0149.</td>
<td>...</td>
</tr>
<tr>
<td>1.3.1.24.11.87</td>
<td>Compiler error C0161.</td>
<td>...</td>
</tr>
<tr>
<td>1.3.1.24.11.88</td>
<td>Compiler error C0162.</td>
<td>...</td>
</tr>
<tr>
<td>1.3.1.24.11.89</td>
<td>Compiler error C0168.</td>
<td>...</td>
</tr>
<tr>
<td>1.3.1.24.11.90</td>
<td>Compiler error C0169.</td>
<td>...</td>
</tr>
<tr>
<td>1.3.1.24.11.91</td>
<td>Compiler error C0174.</td>
<td>...</td>
</tr>
<tr>
<td>1.3.1.24.11.92</td>
<td>Compiler error C0175.</td>
<td>...</td>
</tr>
<tr>
<td>1.3.1.24.11.93</td>
<td>Compiler error C0177.</td>
<td>...</td>
</tr>
<tr>
<td>1.3.1.24.11.94</td>
<td>Compiler error C0178.</td>
<td>...</td>
</tr>
<tr>
<td>1.3.1.24.11.95</td>
<td>Compiler error C0182.</td>
<td>...</td>
</tr>
<tr>
<td>1.3.1.24.11.96</td>
<td>Compiler error C0185.</td>
<td>...</td>
</tr>
<tr>
<td>1.3.1.24.11.97</td>
<td>Compiler error C0189.</td>
<td>...</td>
</tr>
<tr>
<td>1.3.1.24.11.98</td>
<td>Compiler error C0190.</td>
<td>...</td>
</tr>
<tr>
<td>1.3.1.24.11.99</td>
<td>Compiler error C0191.</td>
<td>...</td>
</tr>
<tr>
<td>1.3.1.24.11.100</td>
<td>Compiler error C0195.</td>
<td>...</td>
</tr>
<tr>
<td>1.3.1.24.11.101</td>
<td>Compiler error C0196.</td>
<td>...</td>
</tr>
</tbody>
</table>
Compiler error C0001

**Message:** Constant '<constant value>' too large for type '<data type>'

**Possible error cause:** A typed constant is too large for the given data type or a constant is too large for each possible data type.

**Error correction:** Use smaller constants or an appropriate data type for a typed constant.

**Example of the error:**

```plaintext
PROGRAM PLC_PRG
VAR
  test1: INT;
  test2: INT;
  test3: LREAL;
END_VAR

test1 := 12345678912345566991923939292939911;
test2 := INT#123456;
test3 := 10E500;
```

---

**Message:** Constant '<constant value>' too large for type '<data type>'

**Possible error cause:** A typed constant is too large for the given data type or a constant is too large for each possible data type.

**Error correction:** Use smaller constants or an appropriate data type for a typed constant.

**Example of the error:**

```plaintext
PROGRAM PLC_PRG
VAR
  test1: INT;
  test2: INT;
  test3: LREAL;
END_VAR

test1 := 12345678912345566991923939292939911;
test2 := INT#123456;
test3 := 10E500;
```
Compiler error C0002

Message: '<operator 1>' or '<operator 2>' expected instead of '<tag>'
Possible error cause: Syntax error
Error correction: Use the correct syntax.

Example of the error:

```
PROGRAM PLC_PRG
  Fun(1;

  --> C0002: ',' or ')' expected instead of ';'
```
END_VAR

test2 := test1.z;

--> C0032: Type 'Unknown type: 'test1.z'' cannot be converted to type 'INT'
--> C0004: 'z' is to a component of 'DUT'

Compiler error C0005

Message: Constant overflow in address '<address>'
Possible error cause: At least one component in the address does not fit into a 32-bit integer value.
Error correction: Use a valid address expression.

Example of the error:

PROGRAM PLC_PRG
VAR
  X: BYTE;
END_VAR

X := %QB5555555555;

--> C0005: Constant overflow in address '%??'

Compiler error C0006

Message: '<operator>' expected instead of '<token>'
Possible error cause: Syntax error
Error correction: Use the correct syntax.

Example of the error:

PROGRAM PLC_PRG
VAR
  x: INT;
  bTest : BOOL;
END_VAR

IF bTest
  x := 9;
END_IF

--> C0006: 'THEN' expected instead of 'x'

Compiler error C0007

Message: Expression expected instead of '<token>'
Possible error cause: Syntax error
Error correction: Use the correct syntax.
Example of the error:

PROGRAM PLC_PRG
VAR
  x: INT;
  bTest : BOOL;
END_VAR

IF THEN
  x := 9;
END_IF

--> C0007: Expression expected instead of 'THEN'

Compiler error C0008

Message: Unexpected end-of-file found: '<operator 1>', '<operator 2>', or '<operator 3>' expected
Possible error cause: Syntax error
Error correction: Use the correct syntax.

Example of the error:

PROGRAM PLC_PRG
VAR
  x: INT;
  bTest : BOOL;
END_VAR

IF bTest THEN
  x := 9;

--> C0008: Unexpected end-of-file found: 'ELSIF', 'ELSE' or 'END_IF' expected

Compiler error C0009

Message: Unexpected token '<token>' found
Possible error cause: Syntax error
Error correction: Use the correct syntax.

Example of the error:

PROGRAM PLC_PRG
VAR
END_VAR

END_FOR;

--> C0009: Unexpected token 'END_FOR' found

Compiler error C0010

Message: Unexpected end-of-file found: '<token>' expected
Possible error cause: Syntax error
Error correction: Use the correct syntax.
Example of the error:

```plaintext
PROGRAM PLC_PRG
VAR
  i: INT;
END_VAR

FOR i := 0 TO 2 DO
  ;
--> C0010: Unexpected end-of-file 'END_FOR' found
```

**Compiler error C0011**

Message: No 'CASE' label found

Possible error cause: Syntax error in a CASE statement. A statement in a CASE statement is not assigned to a CASE label.

Error correction: Add a CASE label.

Example of the error:

```plaintext
PROGRAM PLC_PRG
VAR
  i: INT;
  x: INT;
END_VAR

CASE i OF
  x := 9;
END_CASE
--> C0011: No 'CASE' label found

Error correction:

CASE i OF
  0:
    x := 9;
END_CASE
```

**Compiler error C0013**

Message: At least one statement is expected.

Possible error cause: At least one statement is expected at some positions in the code. For example, in the THEN and ELSE part of an IF statement, or in the body of a FOR loop.

Error correction: Add at least one statement at the selected position. It is enough to write a blank statement ";".

Example of the error:

```plaintext
PROGRAM PLC_PRG
VAR
  bTest: BOOL;
END_VAR

IF bTest THEN
  END_IF
--> C0013: At least one statement is expected
```

**Compiler error C0016**

Message: Counter initialization expected
Possible error cause: Syntax error in a FOR loop. The counter variable is not initialized correctly.

Error correction: Pay attention to the correct syntax of the FOR loop.

```
FOR i := 0 TO 10 DO
  ;
END_FOR
```

Example of the error:

```
PROGRAM PLC_PRG
VAR
  i: INT;
END_VAR

FOR i TO 10 DO
  ;
END_FOR

--> C0015: Counter initialization expected
```

Error correction:

```
PROGRAM PLC_PRG
VAR
  i: INT;
END_VAR

FOR i := 0 TO 10 DO
  ;
END_FOR
```

Compiler error C0018

Message: <expression> is not a valid assignment target

Possible error cause: An expression with no write permission is on the left side of an assignment. Example: a constant.

Error correction: Assign only to variables that have write access.

Example of the error:

```
PROGRAM PLC_PRG
VAR
  i: INT;
END_VAR

VAR CONSTANT
  j: INT := 0;
END_VAR

j := i;

--> C0018: 'j' is not a valid assignment target
```

Compiler error C0020

Message: '<statement>' is no valid statement

Possible error cause: Syntax error (for example, too few or too many characters)

Error correction: Make sure that the syntax is correct.

Example of the error:

```
PROGRAM PLC_PRG
VAR
  x : INT;
END_VAR

x = 2;

--> C0020: '(x = 2); ' is no valid statement
```
Example:
```
x := 2;
```

**Compiler error C0022**

**Message:** '<operator>' needs exactly '<number of operands>' operands  
**Possible error cause:** Too many or too few operands are assigned to an operator.  
**Error correction:** Assign the required number of operands to the operator.

**Example of the error:**
```
PROGRAM PLC_PRG
VAR
  i : INT;
  pt: POINTER TO INT;
END_VAR

pt := ADR(i,1);
```

**Error correction:**
```
Example:
pt := ADR(i);
```

**Compiler error C0023**

**Message:** '<operator>' needs at least '<number of operands>' operands  
**Possible error cause:** Too few operands are assigned to an operator.  
**Error correction:** Assign the required number of operands to the operator.

**Example of the error:**
```
PROGRAM PLC_PRG
VAR
  i : INT;
END_VAR

i := MUX(30,40);
```

**Error correction:**
```
Example:
i := MUX(30,40,50);
```

**Compiler error C0026**

**Message:** Identifier expected instead of '<invalid identifier>'  
**Possible error cause:** An invalid identifier is passed to a method.  
**Error correction:** Use valid identifiers.
Example of the error:

METHOD 123
VAR_INPUT
END_VAR

--> C0243: The name used in the signature is not identical to the object name
--> C0026: Identifier expected instead of '123'

Error correction:

Example:
METHOD METH123

Compiler error C0027

Message: size of string expected after '
Possible error cause: The length of the string is not specified.
Error correction: Specify a string length between the parentheses.

Example of the error:

PROGRAM PLC_PRG
VAR
str : STRING();
END_VAR

--> C0027: size of string expected after '('
--> C0006: ';, :=, REF=, ( or ' expected instead of ')''

Error correction:

Example:
str : STRING(100);

Compiler error C0030

Message: Direct Address expected after 'AT' instead of '<identifier>'
Possible error cause: Either an invalid address or no address is assigned after 'AT'.
Error correction: Specify a valid address.

Example of the error:

PROGRAM PLC_PRG
VAR
i AT ABC : INT;
END_VAR

--> C0030: Direct Address expected after 'AT' instead of 'ABC'

Error correction:

Example:
i AT %IW0 : INT;

Compiler error C0031

Message: Type definition expected instead of '<no data type>'
Possible error cause: An invalid type definition is assigned to the identifier.
Error correction: Specify a valid type definition.
Example of the error:

```plaintext
PROGRAM PLC_PRG
  VAR
    i : 0;
  END_VAR

--> C0031: Type definition expected instead of '0'
```

Error correction:

```plaintext
Example:
  i : INT;
```

Compiler error C0032

**Message:** Type ‘<type 1>’ can not be converted to type ‘<type 2>’

**Possible error cause:** A variable is assigned to another variable with an incompatible type.

**Error correction:** Use a type conversion.

Example of the error:

```plaintext
PROGRAM PLC_PRG
  VAR
    test1: INT;
    test2: STRING;
  END_VAR

  test1 := test2;

-->C0032: Type 'STRING' cannot be converted to type 'INT'
```

Error correction:

```plaintext
Example:
  test1 := TO_INT(test2);
```

Compiler error C0035

**Message:** Program name, function or function block instance expected instead of '<invalid function>'

**Possible error cause:** A function is called that does not exist.

**Error correction:** Make sure that only program names, functions, and function Block Instances that exist are called.

Example of the error:

```plaintext
PROGRAM PLC_PRG
  VAR
  END_VAR

  PLC_PRG.METH1();

  METHOD METH
    VAR_INPUT
    END_VAR

--> C0004: 'METH1' is no component of 'PLC_PRG'
--> C0035: Program name, function or function block instance expected instead of 'PLC_PRG.METH1'
```

Error correction:

```plaintext
Example:
  PLC_PRG.METH();
```
Compiler error C0037
Message: '<invalid input>' is no input of '<function name>'
Possible error cause: A local variable is defined in a function call.
Error correction: Declare the variable as an input parameter.

Example of the error:
PROGRAM PLC_PRG
VAR
  i : INT;
END_VAR
i := TEST(iVar := 1);

FUNCTION TEST : INT
VAR
  iVar : INT;
END_VAR

--> C0037: 'iVar' is no input of 'TEST'

Error correction:
Example:
VAR_INPUT
  iVar : INT;
END_VAR

Compiler error C0038
Message: '<invalid output>' is no output of '<function name>'
Possible error cause: A local variable is handled as an output in a function call.
Error correction: Declare the variable as an output parameter.

Example of the error:
PROGRAM PLC_PRG
VAR
  i : INT;
  x : INT;
END_VAR
i := TEST(iVar => x);

FUNCTION TEST : INT
VAR
  iVar : INT;
END_VAR

--> C0038: 'iVar' is no output of 'TEST'

Error correction:
Example:
VAR_OUTPUT
  iVar : INT;
END_VAR

Compiler error C0039
Message: VAR_IN_OUT '<invalid variable>' must be assigned in call of '<function block name>'
Possible error cause: An IN_OUT variable is not passed to a function block that requires an IN_OUT variable.
Error correction: Assign the IN_OUT variable.
Example of the error:

```plaintext
PROGRAM PLC_PRG
VAR
   inst : FB;
END_VAR

FUNCTION_BLOCK FB
VAR_IN_OUT
   inout : INT;
END_VAR

--> C0039: VAR_IN_OUT 'inout' must be assigned in call of 'FB'

Error correction:

Example:
inst(inout := i);
```

Compiler error C0040

Message: Function '<function name>' requires exactly '<number of inputs>' input

Possible error cause: Too many or too few parameters are passed to the called function.

Error correction: Pass exactly as many parameters to the function as are expected.

Example of the error:

```plaintext
PROGRAM PLC_PRG
VAR
   i : INT;
END_VAR

i := TEST(1,2);

FUNCTION TEST : INT
VAR_INPUT
   IN: INT;
END_VAR

--> C0040: Function 'TEST' requires exactly '1' inputs

Error correction:

Example:
i := Test(1);
```

Compiler error C0041

Message: VAR_IN_OUT parameter '<parameter name>' of '<function name>' needs variable with write access as input

Possible error cause: The passed parameter is not a variable with write access (but a constant for example).

Error correction: Pass a VAR_IN_OUT parameter with write access to the function.

Example of the error:

```plaintext
PROGRAM PLC_PRG
VAR
   i : INT;
   x : INT;
END_VAR

i := Test(31415);

FUNCTION TEST : INT
```
VAR_IN_OUT
  in_out: INT;
END_VAR

--> C0041: VAR_IN_OUT' parameter 'in_out' of 'TEST' needs variable with write access as input

Error correction:
Example:
i := Test(x);

Compiler error C0043

Message: Wrong formal parameter: '<parameter name>' expected in this place
Possible error cause: The parameters are assigned to the function explicitly in the wrong order.
Error correction: Specify the parameters in the correct order.

Example of the error:
PROGRAM PLC_PRG
VAR
  i : INT;
END_VAR
i := Test(iPar2 := 2, 5);

FUNCTION Test : INT
VAR_INPUT
  iPar1 : INT;
  iPar2 : INT;
END_VAR

--> C0043: Wrong formal parameter: 'iPar1' expected in this place
--> C0412: Multiple input assignments for parameter ''

Error correction:
i := Test(5, iPar2 := 2);

Compiler error C0044

Message: Assignment to input missing for parameter '<input variable name>' in call of '<function block name>'
Possible error cause: A parameter is passed although an input variable is not declared.
Error correction: Declare an input variable.

Example of the error:
PROGRAM PLC_PRG
VAR
  inst : FB;
END_VAR
inst(1);

FUNCTION_BLOCK FB
VAR_INPUT
--- C0044: Assignment to input missing for parameter '1' in call of 'FB'

Error correction:
Example:
VAR_INPUT
  _in : INT;
END_VAR

Compiler error C0045

Message: Use of 'THIS' is not allowed in this context
Possible error cause: In order to be assigned to the current instance, THIS can be used only in a method, action, transition, or in the body of a function block. This error message appears for all other positions.
Error correction: Use THIS in an allowed context only.

Example of the error:
PROGRAM PLC_PRG
VAR
  test1: INT;
END_VAR

THIS^.test1 := 19;

--- C0018: 'THIS^.test1' is not a valid assignment target
--- C0062: 'THIS^' is not a structure variable
--- C0045: Use of 'THIS' is not allowed in this context

Compiler error C0046

Message: Identifier '<identifier name>' not defined
Possible error cause: An identifier is used that is not declared.
Error correction: Declare the variables that you want to use.

Example of the error:
PROGRAM PLC_PRG
VAR
  END_VAR
  i := 1;

--- C0018: 'i' is no valid assignment target
--- C0046: Identifier 'i' not defined

Error correction:
Example:
VAR
  i : INT;
END_VAR

Compiler error C0047

Message: Cannot apply indexing with '[' to an expression of type '<data type>'
Possible error cause: A data type that is not an array is indexed with '[]'.

Error correction: Index data types with '[]' only if they are declared as arrays.

Example of the error:

```plaintext
PROGRAM PLC_PRG
VAR
  i : INT;
END_VAR
i[1];
```

--> C0047: Cannot apply indexing with '[]' to an expression of type 'INT'

Compiler error C0048

Message: Array requires exactly '<number>' indexes

Possible error cause: Too many or too few indexes are specified when using an array.

Error correction: Specify as many indexes as there are dimensions assigned to the array.

Example of the error:

```plaintext
PROGRAM PLC_PRG
VAR
  arr1 : ARRAY[1..2,1..3] OF INT;
END_VAR
arr1[1] := 5;
```

--> C0048: Array requires exactly 2 indexes

Error correction:
Example:
```
arr1[1,2] := 5;
```

Compiler error C0049

Message: The constant index '<index value>' is not within the range from '<start index>' to '<end index>'

Possible error cause: An index is specified that is outside the size of the array.

Error correction: Use only indexes that are within the size of the array.

Example of the error:

```plaintext
PROGRAM PLC_PRG
VAR
  arr1 : ARRAY[1..2] OF INT;
END_VAR
arr1[3] := 1;
```

--> C0049: The constant index '3' is not within the range from '1' to '2'

Error correction:
Example:
```
arr1[2] := 1;
```

Compiler error C0050

Message: Bitaccess requires literal or symbolic integer constant
Possible error cause: No literal or an integer constant is specified in a bit access.

Error correction: Use a literal or an integer constant.

Example of the error:

```plaintext
PROGRAM PLC_PRG
VAR
  i : INT;
  x : INT;
END_VAR

i.x := FALSE;
```

--> C0018: 'i.x' is no valid assignment target
--> C0050: Bitaccess requires literal or symbolic integer constant

Error correction:

```plaintext
Example:
i := Test(x);
```

Compiler error C0061

Message: Bitaccess on function call is not allowed

Possible error cause: Bit access is performed on a function.

Error correction: Use bit access only for supported data types.

Example of the error:

```plaintext
PROGRAM PLC_PRG
VAR
END_VAR

Test().2;

FUNCTION Test : INT
VAR_INPUT
END_VAR

--> C0061: Bitaccess on function call is not allowed
```

Compiler error C0062

Message: '<variable name>' is no structured variable

Possible error cause: A variable that is not a structure variable is treated like a structure variable.

Error correction: Make sure that the variable is a structure variable.

Example of the error:

```plaintext
PROGRAM PLC_PRG
VAR
  pt : PUNKT;
  i : INT;
END_VAR

i.x := 1024;

TYPE Punkt :
```
STRUCT
  x : REAL;
  y : REAL;
END_STRUCT
END_TYPE

--> C0018: 'i.x' is no valid assignment target
--> C0062: 'Variable' is no structured variable

Example:
pt.x := 1024;

Compiler error C0064
Message: Dereferencing requires a pointer
Possible error cause: A variable that is not a pointer variable is dereferenced.
Error correction: Dereference only variables that are pointer variables.

Example of the error:
PROGRAM PLC_PRG
VAR
  i : INT;
  pi : POINTER TO INT;
END_VAR
i^:=1;

--> C0018: 'i^' not a valid assignment target
--> C0064: Dereferencing requires a pointer

Error correction: Example:
pi := ADR(i);
pi^ := 1;

Compiler error C0066
Message: Cannot compare type '<data type>' with type '<data type>'
Possible error cause: Two data types are compared which cannot be compared with each other.
Error correction: Compare only data types that can be compared with each other.

Example of the error:
PROGRAM PLC_PRG
VAR
  i : INT;
  re : REAL;
  str: STRING;
  b : BOOL;
END_VAR
b := i > str;

--> C0066: Cannot compare type 'INT' with type 'STRING'

Error correction: Example:
b := i > re;
Compiler error C0068

Message: Compare not possible on objects of type ‘<data type>’

Possible error cause: Objects are being compared in which a comparison is not possible.

Error correction: Compare only data types in which a comparison is possible (INT, REAL, etc.).

Example of the error:

```plaintext
PROGRAM PLC_PRG
VAR
  b : BOOL;
  arr1 : ARRAY [1..2] OF INT;
  arr2 : ARRAY [1..2] OF INT;
END_VAR

b := arr1 > arr2;

--> C0068: Compare not possible on objects of type 'ARRAY [1..2]'
```

Compiler error C0069

Message: Compare not possible on objects of type ‘<data type>’ or ‘<data type>’

Possible error cause: Two different objects are being compared in which a comparison is not possible.

Error correction: Compare only data types in which a comparison is possible (INT, REAL, etc.).

Example of the error:

```plaintext
PROGRAM PLC_PRG
VAR
  b : BOOL;
  arr1 : ARRAY [1..2] OF INT;
  arr2 : ARRAY [1..3] OF INT;
END_VAR

b := arr1 > arr2;

--> C0069: Compare not possible on objects of type 'ARRAY [1..2]' or 'ARRAY [1..3]'
```

Compiler error C0070

Message: 'INI' operator needs function block instance or data unit type instance

Possible error cause: Neither a function block instance nor a DUT instance is applied to the INI operator.

Error correction: Pass only function block instances or DUT instances to the INI operator.

Example of the error:

```plaintext
PROGRAM PLC_PRG
VAR
  b : BOOL;
  inst : FB;
END_VAR

b := INI(b, TRUE);

FUNCTION_BLOCK FB
VAR

```
INI operator needs function block instance or data unit type instance

Example:
\[ b := \text{INI}(\text{inst}, \text{TRUE}); \]

Compiler error C0072

Message: Operator <operator name>' is not possible on type '<data type>'

Possible error cause: An operator is applied to an incompatible type.

Error correction: Apply operators only on compatible types.

Example of the error:

```plaintext
PROGRAM PLC_PRG
VAR
  i : INT;
  str : STRING;
END_VAR

str := \text{ABS}(str);
```

Error correction:

Example:
\[ i := \text{ABS}(i); \]

Compiler error C0074

Message: Unexpected array initialisation

Possible error cause: Syntax error in the array initialization

Error correction: Correct the syntax

Example of the error:

```plaintext
PROGRAM PLC_PRG
VAR
  arr1 : INT := [1,2,3,4,5,6];
END_VAR
```

Error correction:

Example:
\[ \text{arr1} : \text{ARRAY }[1..6] \text{ OF INT := } [1,2,3,4,5,6]; \]

Compiler error C0075

Message: Too many initializers for array

Possible error cause: Too many values are specified for the size of the array.

Error correction: The number of assigned values must correspond to the size of the array.
Example of the error:

```plaintext
PROGRAM PLC_PRG
VAR
  arr1 : ARRAY [1..5] OF INT := [1,2,3,4,5,6];
END_VAR
```

--> C0075: Unexpected array initialisation

Error correction:
```
Example:
arr1 : ARRAY [1..6] OF INT := [1,2,3,4,5,6];
```

Compiler error C0076

Message: Unexpected structure initialisation
Possible error cause: Syntax error in the structure initialization
Error correction: Make sure that the syntax is correct.

Example of the error:

```plaintext
PROGRAM PLC_PRG
VAR
  st1 : INT := (p1 := 1);
END_VAR
```

--> C0076: Unexpected structure initialisation
--> C0032: Cannot convert type 'STRUCT(p1:=1)' to type 'INT'
--> C0046: Identifier 'p1' not defined
--> C0018: 'p1' is no valid assignment target

Error correction:
```
Example:
st1 : STRUCT1 := (p1:=1,p2:=10);
```

Compiler error C0077

Message: Unknown type: '<data type>'
Possible error cause: Invalid data type in the declaration (maybe a syntax error)
Error correction: Specify valid data types only.

Example of the error:

```plaintext
PROGRAM PLC_PRG
VAR
  i : INTEGER;
END_VAR
```

--> C0077: Unknown type: 'INTEGER'

Error correction:
```
Example:
arr1 : ARRAY[1..2] OF STRUCT1 := (p1:=1,p2:=10);
```

Compiler error C0080

Message: Functionblock '<function block name>' must be instantiated to be accessed
Possible error cause: Missing function Block Instantiation
Error correction: Instantiate the function block.
Example of the error:

```plaintext
PROGRAM PLC_PRG
VAR
END_VAR

FB();

FUNCTION_BLOCK FB
VAR
END_VAR

--> C0080: Functionblock 'FB' must be instantiated to be accessed

Error correction:

Example:
VAR
  inst : FB;
END_VAR
  inst();
```

Compiler error C0081

Message: Unexpected Pragma: '<pragma name>' found without matching 'if'
Possible error cause: The IF condition is missing when using the pragma.
Error correction: Complete the IF condition of the pragma.

Example of the error:

```plaintext
PROGRAM PLC_PRG
VAR
  i : INT;
END_VAR
  i := 5;
  {END_IF}

--> C0081: Unexpected Pragma: 'END_IF' found without matching 'if'

Error correction:

Example:
  {IF <expression>}
  i := 5;
  {END_IF}
```

Compiler error C0082

Message: '<invalid pragma>' is no valid condition for pragma
Possible error cause: When using a pragma, an invalid expression is used in the IF condition.
Error correction: Use valid pragma conditions.

Example of the error:

```plaintext
PROGRAM PLC_PRG
VAR
  i : INT;
END_VAR
```
{IF abc}
i := 5;
{END_IF}

--> C0082: '!!!ERROR!!!' is no valid condition for pragma

Error correction:
Example:
{IF defined (abc)}

Compiler error C0084

**Message:** '<pragma operand>' is no valid operand for pragma

**Possible error cause:** Syntax error

**Error correction:** Use valid pragma operands.

Example of the error:
PROGRAM PLC_PRG
VAR
END_VAR

{IF defined(0)}
{END_IF}

--> C0084: 'defined(null)' is no valid operand for pragma

Error correction:
Example:
{IF defined (abc)}

Compiler error C0086

**Message:** No definition found for interface '<interface name>'

**Possible error cause:** An undefined interface is used.

**Error correction:** Define the interface.

Example of the error:
PROGRAM PLC_PRG
VAR
    inst : FB;
END_VAR

FUNCTION_BLOCK FB IMPLEMENTS XY
VAR
END_VAR

--> C0086: No definition found for interface 'XY'

Error correction:
Example:
INTERFACE XY

Compiler error C0087

**Message:** There is no implementation for method '<method name>' defined in interface '<interface name>'.

**Possible error cause:** One of the methods specified by the interface has not be provided by the implemented function block.
**Error correction:** Implement all methods that are specified by the interface.

**Example of the error:**

```plaintext
PROGRAM PLC_PRG
VAR
    inst : FB;
END_VAR

INTERFACE XY
METHOD METH1
VAR_INPUT
END_VAR

FUNCTION_BLOCK FB IMPLEMENTS XY
VAR
END_VAR
METHOD METH2
VAR_INPUT
END_VAR

--> C0087: There is no implementation for method 'METH1' defined in interface 'XY'
```

**Compiler error C0089**

**Message:** Interface of overridden method '{0}' of interface '{1}' does not match declaration

**Possible error cause:** The signature of the implemented method does not match the signature of the method in the interface.

**Error correction:** Make sure that the same return types and parameters are declared.

**Example of the error:**

```plaintext
PROGRAM PLC_PRG
VAR
    inst : FB;
END_VAR

INTERFACE XY
METHOD METH1
VAR_INPUT
    iPar : INT;
END_VAR

FUNCTION_BLOCK FB IMPLEMENTS XY
VAR
END_VAR
METHOD METH1
VAR_INPUT
END_VAR

--> C0089: Interface of overridden method 'METH1' of interface 'XY' does not match declaration
```
Compiler error C0090

**Message:** No definition found for base class '<function name>'

**Possible error cause:** The function block specified as the base does not exist or is not a function block.

**Error correction:** Use a function block as the base.

**Example of the error:**

```plaintext
PROGRAM PLC_PRG
VAR
  inst : FB;
END_VAR

FUNCTION_BLOCK FB EXTENDS POU
VAR
END_VAR

FUNCTION POU
VAR
END_VAR

--> C00090: No definition found for base class 'POU'
```

Compiler error C0091

**Message:** Recursion in base function block list: <function name>

**Possible error cause:** A base function block is extended by itself.

**Error correction:** Recursion in base function block lists is not possible.

**Example of the error:**

```plaintext
PROGRAM PLC_PRG
VAR
  inst : FB;
END_VAR

FUNCTION_BLOCK FB EXTENDS FB
VAR
END_VAR

--> C00091: Recursion in base function block list: FB -> FB
```

Compiler error C0094

**Message:** Interface of overridden method '<method name>' of interface '<function block name>' doesn't match declaration

**Possible error cause:** The signature of the method of the first interface does not match the signature of the method in the second interface, which is extended by the first.

**Error correction:** Align the signatures.

**Example of the error:**

```plaintext
PROGRAM PLC_PRG
VAR
  xyz : XY2;
END_VAR
```
FUNCTION_BLOCK XY
METHOD METH1
VAR_INPUT
END_VAR

FUNCTION_BLOCK XY2 EXTENDS XY
METHOD METH1
VAR_INPUT
  iPar : BOOL;
END_VAR

--> C00094: Interface of the overridden method METH1 of interface XY doesn't match declaration

Compiler error C0096

Message: Only one base function block may be defined in EXTENDS-list.
Possible error cause: Two or more base function blocks are defined in the EXTENDS list.
Error correction: Define only one base function block in the EXTENDS list.

Example of the error:
PROGRAM PLC_PRG
VAR
  fb : FB;
END_VAR

FUNCTION_BLOCK FB EXTENDS FB2, FB3
VAR
END_VAR

FUNCTION_BLOCK FB2
VAR
END_VAR

FUNCTION_BLOCK FB3
VAR
END_VAR

--> C00096: Only one base function block may be defined in EXTENDS-list

Compiler error C0097

Message: Duplicate definition of variable '<variable name>' in function block '<function block name>' and in base '<base function block name>'
Possible error cause: A variable is declared with the same name in a function block and its base.
Error correction: Use different variable names.

Example of the error:
PROGRAM PLC_PRG
VAR
  fb : FB;
END_VAR

FUNCTION_BLOCK FB EXTENDS FB2
VAR
  i : INT;
END_VAR

FUNCTION_BLOCK FB2
VAR
  i : INT;
END_VAR

--> C00097: Duplicate definition of variable 'i' in function block 'FB' and in base 'FB2'

Compiler error C0098

Message: The keyword "FUNCTIONBLOCK" is no longer supported. Use "FUNCTION_BLOCK" instead.
Possible cause of error: Syntax error
Error correction: Use the keyword "FUNCTION_BLOCK".

Example of the error:

PROGRAM PLC_PRG
VAR
  inst : FB;
END_VAR

FUNCTIONBLOCK FB
VAR
END_VAR

--> C0098: The keyword "FUNCTIONBLOCK" is no longer supported. Use "FUNCTION_BLOCK" instead. Use "FUNCTION_BLOCK" instead.

Compiler error C0101

Message: Data Recursion: '<recursion>'
Possible error cause: Recursive data initialization over two function blocks
Error correction: Avoid recursions for data initialization.

Example of the error:

PROGRAM PLC_PRG
VAR
  inst0 : FB1;
END_VAR

FUNCTION_BLOCK FB1
VAR
  inst1 : FB2;
END_VAR

FUNCTION_BLOCK FB2
VAR
  inst2 : FB1;
END_VAR

--> C0101: Data Recursion: FB1->FB2->FB1
Compiler error C0102

**Message:** Out of retain memory: Variable '<variable name>', <byte size> bytes.

**Possible error cause:** More retain memory is used than is available on the PLC. It is also possible that the retain memory is too fragmented due to incremental builds.

**Error correction:** Use the "Clean" for fragmenting the memory. This will force the reallocation of all data at the next build.

Compiler error C0104

**Message:** 'Out of global data memory: Variable '<variable name>', <byte size> bytes.

**Possible error cause:** More memory for data is used than is available on the PLC. It is also possible that the memory is too fragmented due to incremental builds.

**Error correction:** Use the "Clean" for fragmenting the memory. This will force the reallocation of all data at the next build.

Compiler error C0114

**Message:** Invalid destination <jump label> for 'JMP'

**Possible error cause:** Syntax error or typographical error in the JMP destination

**Error correction:** Correct the typographical or syntax error.

**Example of the error:**

```
PROGRAM PLC_PRG
VAR
END_VAR
JMP 0;
```

--> C0114: Invalid destination 0 for 'JMP'

Compiler error C0116

**Message:** The label '<jump label>' is a duplicate

**Possible error cause:** A label is defined multiple times.

**Error correction:** Define each label one time only.

**Example of the error:**

```
PROGRAM PLC_PRG
VAR
END_VAR
JMP label;
label:
label:
```

--> C0116: The label 'LABEL' is a duplicate

Compiler error C0117

**Message:** No such label '<jump label>' within the scope of the 'JMP' statement

**Possible error cause:** A jump is made to a label that does not exist.
**Error correction:** Define the label that you specify as the destination.

**Example of the error:**

```plaintext
PROGRAM PLC_PRG
VAR
END_VAR
JMP A;
```

--> C0117: No such label 'A' within the scope of the 'JMP' statement

**Compiler error C0118**

**Message:** The label '<jump label>' has not been referenced.

**Possible error cause:** A jump label is defined that is not referenced.

**Error correction:** Remove the unused jump labels.

**Example of the error:**

```plaintext
PROGRAM PLC_PRG
VAR
END_VAR
LABEL:
```

--> C0118: The label ' LABEL' has not been referenced

**Compiler error C0119**

**Message:** An 'FB_init'-Method of a functionblock or struct needs two inputs 'bInitRetains' and 'bInCopyCode' of type BOOL.

**Possible error cause:** One or both of the inputs 'bInitRetains' and 'bInCopyCode' of type BOOL is missing.

**Error correction:** Define the missing inputs.

**Example of the error:**

```plaintext
PROGRAM PLC_PRG
VAR
  inst : FB;
END_VAR

FUNCTION_BLOCK FB
METHOD FB_init
  VAR_INPUT
END_VAR
```

--> C0119: An 'FB_init'-Method of a functionblock or struct needs two inputs 'bInitRetains' and 'bInCopyCode' of type BOOL

**Error correction:**

```plaintext
Example:
METHOD FB_init
  VAR_INPUT
    bInitRetains : BOOL;
    bInCopyCode : BOOL;
END_VAR
```
Compiler error C0120

Message: An 'FB Exit'-Method of a functionblock or struct needs an input 'bInCopyCode' of type BOOL.

Possible error cause: The input 'bInCopyCode' of type BOOL is missing.

Error correction: Define the input.

Example of the error:

```plaintext
PROGRAM PLC_PRG
  VAR
    inst : FB;
  END_VAR

  FUNCTION_BLOCK FB
    METHOD FB_exit
      VAR_INPUT
      _bInCopyCode : BOOL;
    END_VAR
  END_FUNCTION_BLOCK

--> C0120: An 'FB Exit'-Method of a functionblock or struct needs an input 'bInCopyCode' of type BOOL.

Error correction:

Example:

METHOD FB_exit
  VAR_INPUT
    _bInCopyCode : BOOL;
  END_VAR
```

Compiler error C0122

Message: Expression 'SUPER' is not allowed in this context

Possible error cause: "SUPER^" is used outside of derived function blocks.

Error correction: Use "SUPER^" in function blocks only.

Example of the error:

```plaintext
PROGRAM PLC_PRG
  VAR
  END_VAR

  SUPER^.METH(TRUE, TRUE);

--> C0122: Expression 'SUPER' is not allowed in this context
```

Compiler error C0124

Message: 'Initialization' is no valid initialization for an enumeration

Possible error cause: A data type that is not ANY_INT is used for the enum initialization.

Error correction: Use only ANY_INT for enum initializations.

Example of the error:

```plaintext
PROGRAM PLC_PRG
  VAR
    inst : DUT;
  END_VAR

  TYPE DUT :
```

2020/12/10
(  
    enum_member := 1.5  
) DWORD;  
END_TYPE

--> C0032: Cannot convert type 'LREAL' to type 'DUT'  
--> C0124: 'Initialization' is no valid initialization for an  
enumeration

Compiler error C0125

Message: The constant <constant value> is assigned to more than one enumeration.
Possible error cause: The same value is assigned to two or more enumerations.
Error correction: Assign different values to the enumerations.

Example of the error:

PROGRAM PLC_PRG  
VAR  
  inst : DUT;  
END_VAR

TYPE DUT :  
{  
    enum_member := 0,  
    enum_member2 := 0  
};  
END_TYPE

--> C0125: The constant 0 is assigned to more than one enumeration

Compiler error C0126

Message: Variable of type '<data type>' requires exactly 1 Index
Possible error cause: Multiple indexes are assigned to a variable with one index.
Error correction: Assign only one index.

Example of the error:

PROGRAM PLC_PRG  
VAR  
  pi : POINTER TO INT;  
END_VAR  
pi[0,1] := 0;

--> C0126: Variable of type 'POINTER TO INT' requires exactly 1 Index

Error correction:

Example:

pi[0] := 0;

Compiler error C0130

Message: <object> '<object name>' referenced without parentheses '(())'
Possible error cause: A method is referenced without parentheses.
Error correction: Always reference methods by means of parentheses.
Example of the error:

```plaintext
PROGRAM PLC_PRG
VAR
  inst : FB;
END_VAR
inst.METH1

--> C0130: METHOD 'METH1' referenced without parentheses '()'
```

Error correction:

```plaintext
Example:
inst.METH1();
```

Compiler error C0131

**Message:** '<value>' is not allowed as operand for 'ADR'

**Possible error cause:** A constant is passed as an operand to the operator ADR.

**Error correction:** Use only valid operands for ADR.

Example of the error:

```plaintext
PROGRAM PLC_PRG
VAR
  i : INT;
  pt : POINTER TO INT;
END_VAR
pt := ADR(1);

--> C0131: '1' is not allowed as operand for 'ADR'
```

Error correction:

```plaintext
Example:
pt := ADR(i);
```

Compiler error C0132

**Message:** No enclosing loop of which to EXIT

**Possible error cause:** EXIT is used outside of a loop.

**Error correction:** Use EXIT inside of a loop only.

Example of the error:

```plaintext
PROGRAM PLC_PRG
VAR
END_VAR
EXIT;

--> C0132: No enclosing loop of which to EXIT
```

Compiler error C0136

**Message:** ambiguous use of name '<variable name>'

**Possible error cause:** A variable is declared in multiple GVLs.

**Error correction:** Qualify the variable with the desired GVL.
Example of the error:

```plaintext
PROGRAM PLC_PRG
VAR
  j : INT := g_i;
END_VAR

GVL1:
VAR_GLOBAL
  g_i : INT;
END_VAR

GVL2:
VAR_GLOBAL
  g_i : INT;
END_VAR

--> C0136: ambiguous use of name 'g_i'
```

Error correction:

Example:

```plaintext
j : INT := GVL1.g_i;
```

Compiler error C0139

**Message:** The code `<code>` has no effect. Is this the intent?

**Possible error cause:** The written code is syntactically correct but does not do anything.

**Error correction:** Write code that has a purpose.

Example of the error:

```plaintext
PROGRAM PLC_PRG
VAR
  i : INT;
END_VAR

i;

--> C0139: The code 'i;' has no effect. Is this the intent?
```

Compiler error C0140

**Message:** Reference assign is only allowed to variables of Reference type

**Possible error cause:** An attempt is made to assign a reference value to a variable not defined as a reference type.

**Error correction:** Define the variable as a reference type.

Example of the error:

```plaintext
PROGRAM PLC_PRG
VAR
  i : INT;
  I_r : INT;
END_VAR

I_r REF= i;

--> C0140: Reference assign is only allowed to variables of Reference type
```

Error correction:

Example:

```plaintext
I_r : REFERENCE TO INT;
```
Compiler error C0141

**Message:** Reference assign needs variable with write access

**Possible error cause:** A constant is assigned to the reference assignment.

**Error correction:** Assign a writable variable.

**Example of the error:**

```plaintext
PROGRAM PLC_PRG
VAR
  i : INT;
  I_r : REFERENCE TO INT;
END_VAR

I_r REF= 314;

--> C0141: Reference assign needs variable with write access
```

**Error correction:**

```
Example:
I_r REF= i;
```

Compiler error C0142

**Message:** A local variable named '<variable name>' is already defined in '<pou name>'

**Possible error cause:** The same variable name is used two times.

**Error correction:** Use different variable names.

**Example of the error:**

```plaintext
PROGRAM PLC_PRG
VAR
  i : INT;
  i : INT;
END_VAR

--> C0142: A local variable named 'i' is already defined in 'PLC_PRG'
```

Compiler error C0143

**Message:** The property '<property name>' cannot be used in this context because it lacks the get accessor

**Possible error cause:** The property does not have Get access.

**Error correction:** Make sure that the property has a Get access definition.

**Example of the error:**

```plaintext
PROGRAM PLC_PRG
VAR
  i : INT;
  inst: FB;
END_VAR

i := inst.Prop;

FUNCTION_BLOCK FB
VAR
END_VAR
```
PROPERTY Prop : INT
Set;

--> C0143: The property 'Prop' cannot be used in this context because it lacks the get accessor

Compiler error C0144

Message: Inheritance only allowed in Functionblocks, Interfaces and Structures
Possible error cause: An attempt is made to use inheritance in an object that does not permit inheritance.
Error correction: Use EXTENDS in function blocks, interfaces, and structures only.

Example of the error:
PROGRAM PLC_PRG
VAR
    inst : DUT_1;
END_VAR

TYPE DUT:
    (enum_member := 0);
END_TYPE

TYPE DUT_1 EXTENDS DUT:
    (enum_memberX := 0);
END_TYPE

--> C0144: Inheritance only allowed in Functionblocks, Interfaces and Structures

Compiler error C0145

Message: Interfaces can only be implemented by Functionblocks
Possible error cause: An attempt is made to implement an interface outside of a function block.
Error correction: Implement interfaces only in function blocks.

Example of the error:
PROGRAM PLC_PRG
VAR
END_VAR

POU();

FUNCTION POU IMPLEMENTS ITF
VAR
END_VAR

--> C0145: Interfaces can only be implemented by Functionblocks

Compiler error C0149

Message: Variable declarations are not allowed in interfaces
Possible error cause: An attempt is made to define a variable in an interface.

Error correction: Do not define variables in interfaces.

Example of the error:

```
PROGRAM PLC_PRG
VAR
  inst : ITF;
END_VAR

INTERFACE ITF
VAR_INPUT
  i : INT;
END_VAR

--> C0149: Variable declarations are not allowed in interfaces
```

Compiler error C0161

Message: Border <array bound> of array is no constant value

Possible error cause: A variable is specified as an array bound.

Error correction: Use constants for the array bounds.

Example of the error:

```
PROGRAM PLC_PRG
VAR
  i : INT := 3;
  arr1 : ARRAY[1..i] OF INT;
END_VAR

--> C0161: Border 'i' of array is no constant value
```

Error correction:

```
Example:
arr1 : ARRAY[1..3] OF INT;
```

Compiler error C0162

Message: Number <number of array values> of array initialisations is no constant value

Possible error cause: The initialization \([\text{Wert1}, \text{AnzahlWert2} (\text{Wert2})]\) works only with a constant for \(\text{AnzahlWert2}\).

Error correction: Use constants only.

Example of the error:

```
PROGRAM PLC_PRG
VAR
  i : INT := 3;
  arr1 : ARRAY[1..4] OF INT := [1,i(7)];
END_VAR

--> C0162: Number 'i' of array initialisations is no constant value
```

Error correction:

```
Example:
arr1 : ARRAY[1..4] OF INT := [1,3(7)];
```

Compiler error C0168

Message: ‘VAR_CONFIG’ declaration only allowed in VAR_CONFIG - list
Possible error cause: 'VAR_CONFIG' is used outside of a VAR_CONFIG list.
Error correction: Use 'VAR_CONFIG' only in VAR_CONFIG lists.

Example of the error:

PROGRAM PLC_PRG
VAR_CONFIG
  i : INT;
END_VAR

--> C0168: 'VAR_CONFIG' declaration only allowed in VAR_CONFIG - list

Compiler error C0169

Message: 'VAR_GLOBAL' declaration only allowed in Global variable list
Possible error cause: 'VAR_GLOBAL' is used outside of global variable lists.
Error correction: Use 'VAR_GLOBAL' in global variable lists only.

Example of the error:

PROGRAM PLC_PRG
VAR_GLOBAL
  i : INT;
END_VAR

--> C0169: 'VAR_GLOBAL' declaration only allowed in Global variable list

Compiler error C0174

Message: 'VAR_TEMP' declaration not allowed in this place
Possible error cause: 'VAR_TEMP' is used outside of a program or function block.
Error correction: Use 'VAR_TEMP' inside of programs and function blocks only.

Example of the error:

PROGRAM PLC_PRG
VAR
END_VAR

FUN();

FUNCTION FUN
VAR_TEMP
END_VAR

--> C0174: 'VAR_TEMP' declaration not allowed in this place

Compiler error C0175

Message: 'RETAIN' or 'PERSISTENT' not allowed in this place
Possible error cause: 'RETAIN' or 'PERSISTENT' is used in a function.
Error correction: Use 'RETAIN' or 'PERSISTENT' at the intended locations.
Example of the error:

PROGRAM PLC_PRG
VAR
  inst : POU;
END_VAR

FUNCTION POU
VAR
END_VAR

--> C0175: 'RETAIN' or 'PERSISTENT' not allowed in this place

See also
- Chapter 1.3.1.24.2.13 “Retain variable - RETAIN” on page 500
- Chapter 1.3.1.24.2.13 “Retain variable - RETAIN” on page 500

Compiler error C0177

Message: `<object>` is of type 'type' and cannot be instantiated
Possible error cause: An attempt is made to instantiate a function.
Error correction: Instantiate only objects that can be instantiated.

Example of the error:

PROGRAM PLC_PRG
VAR
  inst : POU;
END_VAR

FUNCTION POU
VAR
END_VAR

--> C0177: 'POU' is of type 'FUNCTION' and cannot be instantiated

Compiler error C0178

Message: No external access to 'VAR_IN_OUT' parameter '<parameter name>' of '<object name>'
Possible error cause: An attempt is made to remotely access a 'VAR_IN_OUT' parameter.
Error correction: Do not remotely access 'VAR_IN_OUT' parameters.

Example of the error:

PROGRAM PLC_PRG
VAR
  inst : FB;
  i : INT;
END_VAR
  i := inst.in_out;

FUNCTION_BLOCK FB
VAR_IN_OUT
  _in_out : INT;
END_VAR

--> C0178: No external access to 'VAR_IN_OUT' parameter 'in_out' of 'FB'
Compiler error C0182

**Message:** Return type is only possible for POUs of Type FUNCTION and METHOD

**Possible error cause:** An attempt is made to define a return value in a program.

**Error correction:** Define a return value only in methods and functions.

**Example of the error:**

```plc
PROGRAM PLC_PRG : BOOL
VAR
END_VAR

--> C0182: Return type is only possible for POUs of Type FUNCTION and METHOD
```

Compiler error C0185

**Message:** It is not possible to perform component access '.', index access '[' or call '(' on result of function call. Assign result to help variable first.

**Possible error cause:** Component or index access to the result of a function call is performed.

**Error correction:** Assign the result to a variable in order to access.

**Example of the error:**

```plc
PROGRAM PLC_PRG
VAR
    it : ITF;
END_VAR

POU_1()[0].METH1();

FUNCTION POU_1 : ARRAY[0..0] OF ITF

INTERFACE ITF

METHOD METH1

--> C0185: It is not possible to perform component access '.', index access '[' or call '(' on result of function call. Assign result to help variable first.
```

Compiler error C0189

**Message:** ';' expected instead of '<token>'

**Possible error cause:** Syntax error

**Error correction:** Make sure that the syntax is correct.

**Example of the error:**

```plc
PROGRAM PLC_PRG
VAR
    INT
END_VAR

--> C0009: Unexpected Token '<Token>' found
--> C0189: ';' expected instead of 'INT'
```
Compiler error C0190

Message: ';' expected instead of end of POU
Possible error cause: Syntax error in the POU
Error correction: Make sure that the syntax is correct.

Example of the error:
```
PROGRAM PLC_PRG
VAR
  i : INT;
END_VAR
i := 5
```

Compiler error C0191

Message: The operator 'INDEXOF' is no longer supported. Use ADR instead. ADR on a POU-Name returns a Pointer to a Pointer to the function code.
Possible error cause: The outdated operator 'INDEXOF' is used.
Error correction: Use the operator 'ADR'.

Compiler error C0195

Message: Implicit conversion from signed Type '<data type 1>' to unsigned Type '<data type 2>' : possible change of sign
Possible error cause: A sign conflict may have been missed in the implicit conversion.
Error correction: Convert only data types with the same sign implicitly.

Example of the error:
```
PROGRAM PLC_PRG
VAR
  i : INT;
  b : UINT;
END_VAR
b := i;
```

Compiler error C0196

Message: Implicit conversion from unsigned Type '<data type 1>' to signed type '<data type 2>' : possible change of sign
Possible error cause: A sign conflict may have been missed in the implicit conversion.
Error correction: Use explicit conversions.

Example of the error:
```
PROGRAM PLC_PRG
VAR
  i : INT;
  b : UINT;
END_VAR
```
i := b;

--> C0196: Implicit conversion from unsigned Type 'UINT' to signed type 'INT': possible change of sign

Compiler error C0197

**Message:** Implicit conversion from '<data type 1>' to '<data type 2>': possible loss of information

**Possible error cause:** An attempt is made to convert a variable from data type DINT or LINT to data type REAL.

**Error correction:** For DINT, use the data type LREAL, and when converting from LINT to LREAL make sure that the value of LINT does not exceed the capacity of LREAL.

**Example of the error:**

```plaintext
PROGRAM PLC_PRG
VAR
    i : DINT;
    b : REAL;
END_VAR
b := i;

--> C0197: Implicit conversion from 'DINT' to 'REAL': possible loss of information
```

Compiler error C0198

**Message:** String constant '<string value>' too long for destination type '<data type>'

**Possible error cause:** The string constant has too many characters.

**Error correction:** Use shorter string constants or declare larger strings.

**Example of the error:**

```plaintext
PROGRAM PLC_PRG
VAR
    str : STRING(4) := '12345';
END_VAR

--> C0198: String constant '12345' too long for destination type 'STRING(4)'
```

Compiler error C0199

**Message:** Interface '<interface name>' must be instantiated to be accessed

**Possible error cause:** An attempt is made to access an interface method without the interface being instantiated.

**Error correction:** Instantiate the interface.

**Example of the error:**

```plaintext
PROGRAM PLC_PRG
VAR
    ITF.METH();
```
INTERFACE ITF
METHOD METH
VAR_INPUT
END_VAR

--> C0199: Interface 'ITF' must be instantiated to be accessed

Example correction:
itest: ITF;

Compiler error C0201

Message: Type '<data type 1>' is not equal to type '<data type 2>' of VAR_IN_OUT 'Variable'

Possible error cause: The data type that is passed to the function as a VAR_IN_OUT parameter does not match the data type defined in it.

Error correction: Pass a variable with the correct data type.

Example of the error:
PROGRAM PLC_PRG
VAR
  Inst: POU;
  b : BOOL;
END_VAR

inst(in_out := b);

FUNCTION_BLOCK POU
VAR_IN_OUT
  _in_out : INT;
END_VAR

--> C0201: Type 'BOOL' is not equal to type 'INT' of VAR_IN_OUT 'Variable'

Compiler error C0203

Message: Only Structures and Function Blocks can contain variables of type BIT.

Possible error cause: An attempt is made to declare a variable of type BIT outside of structures and function blocks.

Error correction: Declare variables of type BIT only in structures and function blocks.

Example of the error:
PROGRAM PLC_PRG
VAR
  b : BIT;
END_VAR

--> C0203: Only Structures and Function Blocks can contain variables of type BIT.

Compiler error C0204

Message: Variables of type BIT must be declared within a VAR_INPUT-, VAR_OUTPUT or VAR-block.

Possible error cause: An attempt is made to define a variable of type BIT or as a VAR_IN_OUT parameter.
**Error correction:** Define variables of type BIT only within a VAR_INPUT, VAR_OUTPUT or VAR block.

**Example of the error:**

```plaintext
PROGRAM PLC_PRG
VAR
    inst : FB;
END_VAR

FUNCTION_BLOCK FB
VAR_IN_OUT
    b : BIT;
END_VAR

--> C0204: Variables of type BIT must be declared within a VAR_INPUT-, VAR_OUTPUT or VAR-block
```

**Compiler error C0205**

**Message:** POINTER TO BIT is not allowed

**Possible error cause:** An attempt is made to declare a POINTER TO BIT.

**Error correction:** Do not declare POINTER TO BIT.

**Example of the error:**

```plaintext
PROGRAM PLC_PRG
VAR
    pt : POINTER TO BIT;
END_VAR

--> C0205: POINTER TO BIT is not allowed
```

**Compiler error C0206**

**Message:** BIT is not allowed as base type of an array

**Possible error cause:** An attempt is made to declare a BIT array.

**Error correction:** Do not declare BIT arrays.

**Example of the error:**

```plaintext
PROGRAM PLC_PRG
VAR
    arr : ARRAY[1..2] OF BIT;
END_VAR

--> C0206: BIT is not allowed as base type of an array
```

**Compiler error C0208**

**Message:** 'MOD' is not defined for 'REAL'

**Possible error cause:** An attempt is made to perform a modulo operation with a variable of type REAL.

**Error correction:** Modulo operations are only possible with variables of type ANY_INT.
Example of the error:

```
PROGRAM PLC_PRG
VAR
    r1 : REAL;
END_VAR
r1 := r1 MOD 2;
```

--> C0208: 'MOD' is not defined for 'REAL'

Compiler error C0211

**Message:** Variable declaration expected instead of <expression>

**Possible error cause:** Syntax error

**Error correction:** Make sure that the syntax is correct.

Example of the error:

```
PROGRAM PLC_PRG
VAR
    VAR
    END_VAR
END_VAR
```

--> C0211: Variable declaration expected instead of VAR END_VAR

Compiler error C0212

**Message:** VAR, VAR_INPUT, VAR_OUTPUT or VAR_INOUT expected instead of <expression>

**Possible error cause:** Syntax error

**Error correction:** Make sure that the syntax is correct.

Example of the error:

```
PROGRAM PLC_PRG
i : INT;
```

--> C0212: VAR, VAR_INPUT, VAR_OUTPUT or VAR_INOUT expected instead of i : INT;

Compiler error C0216

**Message:** Case label duplicate

**Possible error cause:** A CASE label is used multiple times.

**Error correction:** Use each CASE label only one time.

Example of the error:

```
PROGRAM PLC_PRG
VAR
    i : INT;
END_VAR

CASE i OF
    1: i := i+1;
    1: i := i+2;
```
ELSE
  i := i+10;
END_CASE;

--> C0216: Case label duplicate

Compiler error C0217

Message: Case label <case label> also contained in range <case range begin> .. <case range end>

Possible error cause: A CASE label is part of the range of another CASE label.

Error correction: Make sure that there is no intersecting.

Example of the error:

PROGRAM PLC_PRG
VAR
  i : INT;
END_VAR
CASE i OF
  3..5: i := i+2;
  4: i := i+2;
ELSE
  i := i+10;
END_CASE;

--> C0217: Case label 4 also contained in range 3 .. 5

Compiler error C0218

Message: Case label requires literal or symbolic integer constant

Possible error cause: An attempt is made to use a variable as a CASE label.

Error correction: Use only literals and symbolic integer constants.

Example of the error:

PROGRAM PLC_PRG
VAR
  i : INT;
  a : INT := 2;
END_VAR
CASE i OF
  1: i := i+1;
  a: i := i+2;
ELSE
  i := i+10;
END_CASE;

--> C0218: Case label requires literal or symbolic integer constant

Compiler error C0219

Message: Case contains overlapping range <case range 1 begin> .. <case range 1 end> and <case range 2 begin> .. <case range 2 end>

Possible error cause: Two branches of CASE markers have the same elements or subsets.

Error correction: Make sure that there is no intersecting.
Example of the error:
```
PROGRAM PLC_PRG
VAR
  i : INT;
END_VAR
CASE i OF
  3..5: i := i+2;
  1..4: i := i+2;
ELSE
  i := i+10;
END_CASE;

--> C0219: Case contains overlapping range 1 .. 4 and 3 .. 5
```

Compiler error C0221

**Message:** Direct Address '<address>' malformed

**Possible error cause:** An address is not displayed completely.

**Error correction:** Make sure that the address is displayed correctly.

Example of the error:
```
PROGRAM PLC_PRG
VAR
  xVar : BOOL;
END_VAR;
xVar := %IX0;

--> C0221: Direct Address '%IX0' malformed
```

Error correction:
```
Example:
xVar := %IX0.2;
```

Compiler error C0222

**Message:** Outputs can't be of type 'REFERENCE TO'

**Possible error cause:** An attempt is made to define `REFERENCE TO` as an output parameter.

**Error correction:** Do not use `REFERENCE TO` as an output parameter.

Example of the error:
```
PROGRAM PLC_PRG
VAR
  inst : FB;
END_VAR
FUNCTION_BLOCK FB
VAR_OUTPUT
  re : REFERENCE TO INT;
END_VAR

--> C0222: Outputs can't be of type 'REFERENCE TO'
```

Compiler error C0224

**Message:** Call Recursion: <recursion>

**Possible error cause:** A function calls itself.

**Error correction:** Make sure that functions are not recursive.
Example of the error:

```
PROGRAM PLC_PRG
VAR
END_VAR

POU();

FUNCTION POU
VAR
END_VAR

POU();

--> C0224: Call Recursion: POU -> POU
```

Compiler error C0227

**Message:** Initialisation of constant variable `<constant name>` not constant

**Possible error cause:** A constant is initialized with a variable.

**Error correction:** Initialize constants only with constant values.

Example of the error:

```
PROGRAM PLC_PRG
VAR
 i : INT;
END_VAR
VAR CONSTANT
 k : INT := i;
END_VAR

--> C0227: Initialisation of constant variable 'k' not constant
```

Compiler error C0228

**Message:** No initial value for constant variable '<constant name>'

**Possible error cause:** A constant is not initialized.

**Error correction:** Initialize the constants.

Example of the error:

```
PROGRAM PLC_PRG
VAR
END_VAR
VAR CONSTANT
 k : INT;
END_VAR

--> C0228: No initial value for constant variable 'k'
```

Error correction:

Example:

```
k : INT := i;
```

Compiler error C0234

**Message:** First Operand of `__QueryInterface` must be an interface reference or the instance of a function block

**Possible error cause:** Incorrect operands are passed to the operator `__QueryInterface`.

**Error correction:** Pass an interface reference or the instance of a function block.
Example of the error:

```plaintext
PROGRAM PLC_PRG
VAR
    a : INT;
    ITFref, ITFref2 : ITF;
    ITFref2 : ITF2;
END_VAR

__QueryInterface(a , ITFref);

INTERFACE ITF EXTENDS __SYSTEM.IQueryInterface
INTERFACE ITF2 EXTENDS ITF

--> C0234: First Operand of __QueryInterface must be an interface reference or the instance of a function block
```

Error correction:

```plaintext
Example:
__QueryInterface(ITFref2, ITFref);
```

Compiler error C0235

**Message:** Second Operand of __QueryInterface must be an interface reference

**Possible error cause:** Incorrect operands are passed to the operator __QueryInterface.

**Error correction:** Pass an interface reference.

Example of the error:

```plaintext
PROGRAM PLC_PRG
VAR
    a : INT;
    ITFref, ITFref2 : ITF;
    ITFref2 : ITF2;
END_VAR

__QueryInterface(ITFref2, a);

INTERFACE ITF EXTENDS __SYSTEM.IQueryInterface
INTERFACE ITF2 EXTENDS ITF

--> C0235: Second Operand of __QueryInterface must be an interface reference
```

Error correction:

```plaintext
Example:
__QueryInterface(ITFref2, ITFref);
```

Compiler error C0236

**Message:** Wrong type definition for VAR_EXTERNAL <variable name>

**Possible error cause:** The variable is declared in VAR_GLOBAL / VAR_EXTERNAL as different types.

**Error correction:** Use the same type definition in VAR_GLOBAL and VAR_EXTERNAL.

Example of the error:

```plaintext
PROGRAM PLC_PRG
VAR_EXTERNAL
    ig : STRING;
END_VAR
```
Compiler error C0236

Message: Wrong type definition for VAR_EXTERNAL ig
Possible error cause: An attempt is made to declare a variable in VAR_EXTERNAL which does not exist in VAR_GLOBAL.
Error correction: Make sure that the identifiers match.

Example of the error:

```plaintext
PROGRAM PLC_PRG
VAR_EXTERNAL
    ig : INT;
END_VAR

--> C0236: Wrong type definition for VAR_EXTERNAL ig
```

Compiler error C0237

Message: No global definition found for VAR_EXTERNAL '<variable name>'
Possible error cause: An attempt is made to declare a variable in VAR_EXTERNAL which does not exist in VAR_GLOBAL.
Error correction: Make sure that the identifiers match.

Example of the error:

```plaintext
PROGRAM PLC_PRG
VAR_EXTERNAL
    i : INT;
END_VAR

VAR_GLOBAL
    ig : INT;
END_VAR

--> C0237: No global definition found for VAR_EXTERNAL 'i'
```

Compiler error C0238

Message: No initial value allowed for VAR_EXTERNAL <variable name>
Possible error cause: An attempt is made to initialize a variable in VAR_EXTERNAL.
Error correction: Do not initialize variables in VAR_EXTERNAL.

Example of the error:

```plaintext
PROGRAM PLC_PRG
VAR_EXTERNAL
    ig : INT := 2;
END_VAR

VAR_GLOBAL
    ig : INT;
END_VAR

--> C0238: No initial value allowed for VAR_EXTERNAL ig
```

Compiler error C0239

Message: Interface <interface name 1> does not extend <interface name 2>
Possible error cause: The used interface does not extend another interface.
Error correction: Extend the interface.

Example of the error:

```plaintext
PROGRAM PLC_PRG
VAR
    ITFref : ITF;
    ITFref2 : ITF2;
END_VAR
```

Example of the error:
__QueryInterface(ITRef2,ITRef);

INTERFACE ITF
INTERFACE ITF2 EXTENDS ITF

--> C0239: Interface ITF__Union does not extend __System.IQueryInterface

**Error correction:**

Example:
INTERFACE ITF EXTENDS __System.IQueryInterface

---

**Compiler error C0240**

**Message:** First Operand of __QueryPointer must be an interface reference or the instance of a function block

**Possible error cause:** Incorrect operands are passed to the operator __QueryPointer.

**Error correction:** Pass an interface reference or the instance of a function block.

**Example of the error:**

PROGRAM PLC_PRG
VAR
  a : INT;
  ITRef : ITF;
  pt : POINTER TO FB;
END_VAR

__QueryPointer(a,pt);

--> C0240: First Operand of __QueryPointer must be an interface reference or the instance of a function block

**Error correction:** __QueryPointer (ITRef, pt);

---

**Compiler error C0241**

**Message:** Second Operand of __QueryPointer must be pointer

**Possible error cause:** Incorrect operands are passed to the operator __QueryPointer.

**Error correction:** Pass a pointer.

**Example of the error:**

PROGRAM PLC_PRG
VAR
  b : INT;
  ITRef : ITF;
  pt : POINTER TO FB;
END_VAR

__QueryPointer(ITRef,b);

INTERFACE ITF EXTENDS __System.IQueryInterface

--> C0241: Second Operand of __QueryPointer must be pointer

**Error correction:** __QueryPointer (ITRef, pt);
Compiler error C0242

Message: Operand of __DELETE must be pointer
Possible error cause: An incorrect operand is passed to the operator __DELETE.
Error correction: Pass a pointer.

Example of the error:

```plaintext
PROGRAM PLC_PRG
VAR
  a : INT;
  pt : POINTER TO INT;
END_VAR
__DELETE(a);

--> C0242: Operand of __DELETE must be pointer
```

Compiler error C0243

Message: The name used in the signature is not identical to the object name
Possible error cause: The object name differs from the name used in the code.
Error correction: make sure that the names are the same.

Example of the error:

```plaintext
FUNCTION_BLOCK POU
VAR_IN_OUT
  arrin : ARRAY [*] OF INT;
END_VAR
VAR
  arretest : ARRAY [0..5] OF INT;
```

Compiler error C0380

Message: The Operators LOWER_BOUND and UPPER_BOUND are only supported for arrays of variable length.
Possible error cause: One of the two operators LOWER_BOUND or UPPER_BOUND is not used for an array of variable length.
Error correction: Use the operators LOWER_BOUND and UPPER_BOUND only for an array of variable length.
test1: DINT;
test2: DINT;
END_VAR

test1 := UPPER_BOUND(arrin, 0);
test2 := UPPER_BOUND(arrtest, 0);

--> C0380: The operators LOWER_BOUND and UPPER_BOUND are supported only for arrays with variable length.

Compiler error C0509

Message: Multiple assignments for operator '__New' not allowed

Possible error cause: In one line of code, the assignment operator "=" is called a multiple number of times with the __New operator.

Error correction: Program the memory allocation with the __New operator in a separate line of code for each pointer that points to dynamically allocated memory.

Example of the error:

PROGRAM PLC_PRG
VAR
   pbAlpha : POINTER TO BYTE; // Typed pointer to Alpha
   pbBeta: POINTER TO BYTE; // Typed pointer to Beta
   xInit : BOOL := TRUE;
   xDelete : BOOL;
END_VAR

IF (xInit) THEN
   pbBeta := pbAlpha := __NEW(BYTE); // Incorrect code for memory allocation
END_IF

pbBeta := pbAlpha := 16#01;

IF (xDelete) THEN
   __DELETE(pbAlpha); // Frees memory of pointer
END_IF

--> C0509: Multiple assignments for operator '__New' not allowed

Error correction:

PROGRAM PLC_PRG
VAR
   pbAlpha : POINTER TO BYTE; // Pointer to Alpha
   pbBeta: POINTER TO BYTE; // Pointer to Beta
   xInit : BOOL := TRUE;
   xDelete : BOOL;
END_VAR

IF (xInit) THEN
   pbAlpha := __NEW(BYTE); // Allocates memory for Alpha
   pbBeta := __NEW(BYTE); // Allocates memory for Beta
END_IF

pbBeta := pbAlpha := 16#01; // Multiple assignment

IF (xDelete) THEN
   __DELETE(pbAlpha); // Frees memory of pointer
END_IF
 Compiler error C0511

**Message:** The function block '<function block name>' is ABSTRACT and cannot be used as a target for an assignment.

**Possible error cause:** A value was assigned to an abstract function block. The concrete function blocks may have different types and therefore cannot be copied.

**Error correction:** In order to copy the data of the function block, concrete function blocks have to be used.

**Example of the error:**

```plaintext
PROGRAM PLC_PRG
VAR
refAbstract1 : REFERENCE TO AbstractPOU;
refAbstract2 : REFERENCE TO AbstractPOU;
END_VAR

refAbstract1 := refAbstract2;
--> C0511: The function block 'refAbstract1' is ABSTRACT and cannot be used as a target for an assignment.

**Error correction:**
Use the reference assignment REF= to assign the reference refAbstract1 to the same function block as refAbstract2.
```

1.3.1.25 Reference, User Interface

1.3.1.25.1 Objects

Objects in CODESYS provide special functionalities to create applications. Examples: Application, program, function, Library Manager, devices, image pool. Objects are managed in tree structures in the views "Devices", "POUs" and "Modules".

You can add an object to the belonging "tree" by use of the command “Project ➤ Add Object”. The possible insert positions depends on the position within the tree.

Each object provides properties, which can be viewed and accessed with the command from the context menu of the object.

See also

- Chapter 1.3.1.24.3.58 “Operator __NEW__” on page 541
- Chapter 1.3.1.24.6.2.39 “Attribute enable_dynamic_creation” on page 637

Object 'Application'

**Symbol:** 🐤

The object "Application" is presented as a node in the device tree. It comprises objects required for an executable control program.
You can insert an application object below a “PLC Logic” node, that is below a programmable device, or as a child application below an already existing application object (parent application).

Below each application there must be a “Task Configuration”, where you define which program of the application will be called by which task, using which settings.

Additionally, below an application you insert the modules of your control program, for example POU s, global variables lists, library manager. These modules will be only available for this application and its child applications.

Additionally the application may use instances of project-global modules. Project-global modules are managed in the “POUs” view. Using instances of such modules follows the thinking of object-oriented programming.

Below a “PLC Logic” device object you can insert multiple applications. They must have unique names.

![Application Structure Diagram]

**NOTICE!**
An Online Change after a modification of the parent application will remove the child application from the PLC.

If there are multiple applications directly below a device object, you must define which application shall be responsible for the I/O-handling of the device. CODESYS will use the variables of this application for the communication with the target system. Set the application in tab “PLC Settings” of the device editor.

The application to be currently used for online mode operations, must be set "active"; see “App2” in the figure shown above.

You can set special properties for an application in the “Application Build Options” tab of the “Properties” dialog. For example: Activation of dynamic memory allocation.

When downloading the application to the PLC, you can add information on the application content. Again this is to be activated in the “Application Build Options” tab. Then you will be able later to compare the application on the PLC with the application currently active in CODESYS.

If you want to add application-specific information on author, version and an application-specific short description, then modify the common Project Information entries, using the “Information” tab of the application “Properties” dialog.
When you are going to log in to the target system (PLC or simulation) with an application, CODESYS first will check, which applications are currently on the PLC and whether the application parameters on the target match those of the project configuration. Appropriate messages will tell you about mismatches and possible proceedings. During this step you also may remove applications from the PLC.

In the “Applications” tab of the device editor you can see, which applications are currently available on the device. In this tab you can remove applications from the target system. Possibly the list also shows applications, which are not represented by a separate object in the device tree, like for example <Application>_symbols.app. This application contains a symbol list created for the application (see on "Symbol Configuration").

See also
- Chapter 1.3.1.8.1 “Device tree and device editor” on page 155
- Chapter 1.3.1.25.1.37 “Object 'Task Configuration'” on page 834
- Chapter 1.3.1.25.10.9 “Dialog 'Properties' - 'Application Build Options'” on page 1051
- Chapter 1.3.1.11 “Transferring an application to the PLC” on page 295
- Chapter 1.3.1.19.9 “Tab "PLC Settings"” on page 751
- Chapter 1.3.1.10.2 “Symbol configuration” on page 275
- Chapter 1.3.14.1 “Executing the online change” on page 355
- Chapter 1.3.25.2.11 “Command 'Project information'” on page 902

Object 'Device Application'

Symbol: 📁

The “Device Application” object centrally manages all device-global data that can be used by all applications of the device. The applications can be restricted therefore to exclusively control programs.

It depends on the device whether or not the device application is supported.

If the “Device Application” object is selected, then you can insert the following other objects:

- “Application” (on or more applications for the logical mapping of your control program)
- and the following objects for global data and variables that are available for all applications of the device:
  - “Library Manager” (inserted automatically)
  - “Global Variable List”
  - “Network Variable List”
  - “Persistent Variable Configuration” (if supported by the device)
  - “Persistent Variables”
  - “Recipe Manager”
  - “Task Configuration”: In practice, the task configuration of the device application often contains the bus cycle task of the bus master. In order to harmonize the timing of the processing of the POUs with the bus cycle task, you can configure a task of type “External” or “Parent task” in the task configuration of an application. Refer to the help page for “Create device applications”.

Inserting into existing project with conventional application organization

To insert the “Device Application” object into an existing project with conventional application organization, proceed as follows:

- Create a new device entry in the tree.
- Delete automatically generated applications below the “PLC Logic” node.
- Insert a “Device Application” object for the selected “PLC Logic” node.
- Move the necessary objects from the old device structure to the new structure and delete the previous device entry.
In online mode, there are commands in the context menu of the “Device Application” object for simultaneously starting, stopping, and resetting of applications of the device.

See also
- § Chapter 1.3.1.16 “Using device applications” on page 358
- § Chapter 1.3.1.16.1 “Creating device applications” on page 359
- § Chapter 1.3.1.25.1.4 “Object ‘Persistent Variable Configuration’” on page 714
- § Chapter 1.3.1.25.2.22.6 “Command ‘Start All Applications’” on page 1037
- § Chapter 1.3.1.25.2.22.7 “Command ‘Stop All Applications’” on page 1037
- § Chapter 1.3.1.25.2.22.5 “Command ‘Reset Warm All Applications’” on page 1037
- § Chapter 1.3.1.25.2.22.4 “Command ‘Reset Cold All Applications’” on page 1036

Object ‘POU Locations’

This object is available only for specific controllers. It is displayed automatically in the device tree. The object cannot be added or removed manually. The object can be used for mapping the executable code of an application in different code areas on the controller. Specifically small controllers often have limited internal code areas (flash memory). If one or more additional code areas (for example, external flash memory) are available on the controller, then the location of the code POUs of an application can be changed specifically.

If there are no specific requirements, then the code POUs are stored sequentially in the code areas (“default”). This means that the next code POUs are stored in the next areas only when the first code area is filled. In the “POU Locations” editor, you will see the current location of the POUs in the memory areas and you can change them specifically.

Editor ‘POU Locations’

Double-clicking the “POU Locations” object in the device tree of the controller opens the editor. Then it receives the entry “<application>“. After a code generation, all program blocks of the application are displayed with the respective object type, current location in the memory, and code size.

In the “Configured Location” column, you can set one of the memory areas other than the “Current Location” for each POU or library.

In order to move the POUs to the recently configured memory locations, you first have to “Clean” and then “Generate Code” again.

Make sure to pay attention to the messages in the category “POU Locations”. This also shows when a code POU cannot be moved as expected.

<table>
<thead>
<tr>
<th>“Clean”</th>
<th>Deletion of the compile information for the application. Corresponds to the menu command “Build ➔ Clean”. Requirement for moving the POUs to the configured memory locations.</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Generate code”</td>
<td>Starting of the code generation for the application. Corresponds to the menu command “Build ➔ Generate Code”. Requirement for moving the POUs to the configured memory locations.</td>
</tr>
<tr>
<td>“Objects”</td>
<td>Objects of the application, including the objects from the referenced libraries</td>
</tr>
<tr>
<td>“Type”</td>
<td>Object type; examples: “Function block”, “Method”, “Library”</td>
</tr>
<tr>
<td>“Current location”</td>
<td>Current memory location of the POU: area_&lt;n&gt;.</td>
</tr>
</tbody>
</table>
### Object 'Persistent Variable Configuration'

Symbol: ![Persistent Variable Configuration](image)

The object can be inserted below a "Device Application" if the device supports static memory areas (static memory) for persistent and retain variables. It is used for segmenting this static memory. This allows for memory mapping for persistent and retain variables in a global memory in the device application and in memory areas for each application.

For devices that support static memory areas, the memory area for the persistent and retain variables is 100% for the device application by default.

**NOTICE!**
Changes to this configuration, as all changes to the device application, require a complete download of all applications.

### Table 27: Editor 'Persistent Variable Configuration'

<table>
<thead>
<tr>
<th>Static Memory Segmentation</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Application</strong></td>
</tr>
<tr>
<td><strong>Offset</strong></td>
</tr>
<tr>
<td><strong>Size</strong></td>
</tr>
</tbody>
</table>

**Graphical representation of memory mapping**

Memory mapping for applications is represented graphically. A horizontal bar represents the memory, beginning with offset 0 on the left. Each application appears as a block on the bar according to their values for offset and size. By clicking an application block, the associated values are displayed in the fields above.

**Example:**

![Persistent Variable Configuration](image)
Object 'Alarm Configuration'

The object “Alarm Configuration” is for the management of alarms. It consists of alarm classes, alarm groups and optionally an alarm storage. The alarm configuration does not have a configuration.

If you insert the object, CODESYS also automatically inserts the following predefined objects:

- Alarm class “Error”
- Alarm class “Info”
- Alarm class “Warning”
- Alarm storage “AlarmStorage”

You can use these objects, but you don't have to. You can delete them and replace them by objects that you add yourself.

See also

- Chapter 1.3.1.25.1.6 “Object 'Alarm Class’” on page 715
- Chapter 1.3.1.25.1.7 “Object 'Alarm Group’” on page 718
- Chapter 1.3.1.25.1.8 “Object 'Alarm Storage’” on page 721

Object 'Alarm Class'

An alarm class describes the general properties of an alarm. These include the type of acknowledgement of the alarm and the actions that are to take place automatically if a certain alarm status is attained. In the definition of the alarm class you can also define the font of the message in the visualization elements.

<table>
<thead>
<tr>
<th>“Priority”</th>
<th>Specification of the priority of the alarms in this class (0-255). The priority describes the priority of the alarm condition. 0 represents the highest priority, 255 the lowest.</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Archiving”</td>
<td>CODESYS records the alarms in this class. To do this you must create an object “Alarm Storage”.</td>
</tr>
</tbody>
</table>

Table 28: “Acknowledgment”

<table>
<thead>
<tr>
<th>“Acknowledgement method”: Definition of how an alarm is acknowledged.</th>
<th>“REP”: Alarm inactive after rectification of the cause</th>
</tr>
</thead>
<tbody>
<tr>
<td>Activating 3.</td>
<td>Deactivating 5.</td>
</tr>
</tbody>
</table>

| | “ACK”: Alarm inactive after acknowledgement; acknowledgement method for events |
| | Activating 5. | Confirming 7. |
"REP_ACK": Alarm inactive after rectification of the cause and acknowledgement

"ACK_REP": Alarm inactive after acknowledgement and rectification of the cause

"ACK_REP_ACK": Inactive after optional acknowledgement that the alarm was received, rectification of the cause and acknowledgement that the alarm situation was ended

Alarm status:
- ① Normal: No alarm situation
- ② Active: Alarm situation has occurred
- ③ Wait for confirmation: there is no longer an alarm; acknowledgement has not taken place yet
- ④ Active, Acknowledged: Alarm still exists; acknowledgement already completed

Status transitions:
- ⑤ Activate: Alarm situation occurs ("alarm coming")
- ⑥ Deactivate: Alarm situation ended ("alarm going")
- ⑦ Confirm: Acknowledgement of an alarm that is already ended
- ⑧ ACK: Acknowledgement of an alarm that still exists

Note: CODESYS displays this diagram in a tooltip if you position the mouse pointer over the selection box in the configuration editor.

You frequently have to ensure that the user notices the alarm during operation. Various actions are available in the alarm class for this. The user must confirm (acknowledge) the alarm in order for the alarm to be removed from the alarm list.

"Acknowledge separately"  ✔ During operation the user cannot acknowledge the alarm together with other alarms, but must acknowledge it separately. This prevents the user from inadvertently acknowledging an alarm together with other alarms.

Table 29: “Notification Actions”

You can assign a list of actions to each alarm class that are to be executed as soon as the alarm is subject to a status transition.

<table>
<thead>
<tr>
<th>“Action”</th>
<th>Select one of the following actions by double-clicking the action column:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>“Variable”: Value assignment of a variable</td>
</tr>
<tr>
<td></td>
<td>“Execute”: Execution of a program</td>
</tr>
<tr>
<td></td>
<td>“Call”: Call a function block instance</td>
</tr>
</tbody>
</table>
“Activate”  
“Deactivate”  
“Confirm”  
“ACK”  

The available status transitions depend on the selected “acknowledgement method”.

☑️: The action is executed with the status transition.

“Details”  
Displays a summary of the action. You define the action below the table or edit the action directly in this row.

“Deactivation”:  
Variable via which an application can deactivate the action.

<table>
<thead>
<tr>
<th>Table 30: “Variable/Execute/Call”</th>
</tr>
</thead>
<tbody>
<tr>
<td>You define the parameters of the action in the field below the table. The parameters depend on the selected action:</td>
</tr>
</tbody>
</table>

- **Variable:**
  - “Variable”: Variable to which you assign a value or an expression. In the field on the right: Assigned variable or expression. You can select the variables via the input assistant (....).
  - In the case of boolean variables you can use the placeholder ALARM, in the case of integer variables the placeholder STATE. In the case of string variables you can enter any IEC literal (e.g. "too high").
    - ALARM: Status (TRUE or FALSE) of the alarm of this alarm class that was last activated/deactivated
    - STATE: Status of the alarm of this alarm class that last changed its status. (0: Normal, 2: Active, 3: WaitingForConfirmation, 4: ActiveAcknowledged)

- **Execute:**
  - “Executable file”: Path and file name of the executable file on the controller.
  - “Parameter”: One or more parameters that you add to the call. You separate several parameters by a space.

- **Call:**
  - “Function block instance”: Function block that is called. The function block must implement the interface IAlarmNotifiable (AlarmManager.library). The desired action on a change of state is implemented in the method Execute.
  - “Additional parameter structure”: If the function block requires an additional parameter structure, the structure members are listed in this table and can be filled here. An additional parameter structure is defined in the function block with the attribute AlarmManagerAdditionalData.

  **Example:** The function block has the attribute {attribute 'AlarmManagerAdditionalData' := 'StructEmailParams'}.

  The structure employed must consist of scalar components or components of the type POINTER TO. CODESYS does not support the ARRAY data type or user-defined data types.

  **Note:** Using the attribute {attribute 'AlarmManagerMandatoryParameter'} you can identify structure members as mandatory parameters.
Table 31: “Presentation Options for Alarm Table/Alarm Banner”

<table>
<thead>
<tr>
<th>“Status”</th>
<th>Alarm status. The available statuses depend on the selected “acknowledgement method”.</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Font”</td>
<td>The standard dialog for the selection of a font or color is opened with the button.</td>
</tr>
<tr>
<td>“Background color”</td>
<td>Background color for the alarm message</td>
</tr>
<tr>
<td>“Bitmap”</td>
<td>You can add a bitmap by entering the bitmap ID. If no image file is assigned to this ID yet, the standard file search dialog opens. After having selected a file, the bitmap definition of the image pool “GlobalImagePool” is added.</td>
</tr>
<tr>
<td>“Transparent”</td>
<td>Activate this option if the bitmap has a transparent color.</td>
</tr>
<tr>
<td>“Transparent color”</td>
<td>Color that is to be displayed transparently. Relevant only if the “Transparent” option has been activated.</td>
</tr>
</tbody>
</table>

See also
- % Chapter 1.3.1.9.21.2 “Configuring the alarm management” on page 258
- % Chapter 1.3.1.25.1.7 “Object ‘Alarm Group’” on page 718
- % Chapter 1.3.1.25.1.8 “Object ‘Alarm Storage’” on page 721

Object ‘Alarm Group’

Use alarm groups for organizing individual alarms in the alarm management of an application. Each alarm is assigned to one alarm group only. To configure an alarm group, insert an “Alarm Group” object below an “Alarm Configuration” object in the device tree.

A common deactivation variable can be assigned to all alarms that are assigned to a group. You can create a hierarchical structure of alarm groups in the device tree with the help of folder elements.

The dialog of the alarm group editor includes a table listing all alarms of the group. The dialog displays the basic settings for the entire group, such as deactivation,archiving, and associated text list, as well as the properties of the individual alarms. You can edit the values in the list. Note the option of exporting and importing alarms to and from a CSV file (see below).

Dialog ‘Alarm Group’

<table>
<thead>
<tr>
<th>“Text list”</th>
<th>Name of the text list where the alarm message text is saved (see below: “Message”). CODESYS generates the list automatically when you create an alarm group. You can also select an existing text list here.</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Archiving”</td>
<td>Selection of the alarm storage object for the alarm group. When you specify an alarm storage object, CODESYS activates the alarm storage.</td>
</tr>
<tr>
<td>“Deactivation”</td>
<td>Boolean variable that deactivates all alarms in this group. CODESYS does not trigger these alarms any longer, even if the alarm condition is fulfilled. Alarms that are already active are not affected by this.</td>
</tr>
</tbody>
</table>
Table 32: Table of alarms

You enter the properties of the individual alarms in the corresponding table columns. Depending on the selected observation type, CODESYS sets additional parameters in input fields below the table.

<table>
<thead>
<tr>
<th>“ID”</th>
<th>This unique ID corresponds to the ID of the associated text list. You can change the ID in the “ID” field of the table. The ID must always be unique within the alarm group. A change of the ID in the alarm table also directly causes a change in the text list, and the other way around.</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Observation type”</td>
<td>Drop-down list of observation types. Depending on the type, different editing fields are available below the table that define the condition for the alarm. The fields define an expression that triggers the alarm. See the following table.</td>
</tr>
<tr>
<td>“Details”</td>
<td>This field shows the configuration settings (defined in the fields below the table) for the selected alarm type. You can also enter the condition directly into the field. If you input an expression that does not match the currently selected observation type, the type is automatically adapted.</td>
</tr>
<tr>
<td>“Deactivation”</td>
<td>Variable for deactivating the alarm (optional).</td>
</tr>
<tr>
<td>“Class”</td>
<td>Assigned alarm class.</td>
</tr>
<tr>
<td>“Message”</td>
<td>Required entry of a message text (message) that CODESYS will display in the “Alarm Table” visualization element when the alarm occurs. The text entered here is transferred automatically by CODESYS to the text list for the alarm group. You can insert line breaks by pressing [Ctrl]+[Enter]. CODESYS supports the following placeholders, which are replaced by the actual value at runtime:</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Placeholder</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;DATE&gt;</td>
<td>Date of the change to the present status.</td>
</tr>
<tr>
<td>&lt;TIME&gt;</td>
<td>Time of the last status change.</td>
</tr>
<tr>
<td>&lt;EXPRESSION&gt;</td>
<td>Expression (defined within the alarm) that triggered the alarm.</td>
</tr>
<tr>
<td>&lt;PRIORITY&gt;</td>
<td>Priority of the alarm (defined in the alarm class).</td>
</tr>
<tr>
<td>&lt;TRIGGERVALUE&gt; *</td>
<td>Value that caused the alarm condition to become true.</td>
</tr>
<tr>
<td>&lt;ALARMID&gt;</td>
<td>Alarm ID as displayed in the first column of the alarm table.</td>
</tr>
<tr>
<td>&lt;CLASS&gt;</td>
<td>Name of the alarm class (defined in the alarm).</td>
</tr>
<tr>
<td>&lt;ALLDEFAULT&gt;</td>
<td>Complete information about the alarm is output.</td>
</tr>
<tr>
<td>&lt;CURRENTVALUE&gt; *</td>
<td>Actual value of the monitored variable.</td>
</tr>
<tr>
<td>&lt;LATCH1&gt; *</td>
<td>Value of the first latch variable at the time when the alarm was triggered.</td>
</tr>
<tr>
<td>&lt;LATCH2&gt; *</td>
<td>Value of the second latch variable at the time when the alarm was triggered.</td>
</tr>
<tr>
<td>&lt;ALARM&gt;</td>
<td>TRUE in the case of the alarm status ‘Active’, FALSE with every other alarm status.</td>
</tr>
<tr>
<td>&lt;STATE&gt;</td>
<td>Alarm status: 0 = ‘normal’, 1 = ‘active’, 2 = ‘waiting for ACK’, 3 = ‘active, acknowledged’ (still active and already acknowledged).</td>
</tr>
</tbody>
</table>
* For `<TRIGGERVALUE>`, `<CURRENTVALUE>`, and `<LATCHn>`, you can also use formatting specifications, such as those that are valid for the C function "printf" (example: "The value is `<CURRENTVALUE> %d`").

By default, only one column is available for message text. You can add more columns from the context menu which will be available in the "Data type" property for column configuration of the "Alarm table" and "Alarm banner" visualization elements.

"Min. pending time": Defines the minimum duration for fulfilling the alarm condition so that CODESYS triggers the alarm. In this way, CODESYS can inhibit alarm conditions that occur only for a short time.

IEC 61131 compliant format (example: `t=#2ms`).

"Latch var 1", "Latch var 2" The latch variables are used to record additional values when the alarm becomes active. A latch variable has to be a scalar type (8-byte) or a STRING or WSTRING type. CODESYS records a latch variable or a latch expression when the alarm becomes active. In this way, it allows additional information about the alarm to be provided.

Note: With the respective configuration of the "Alarm table" element or "Alarm banner" element, you can filter the occurring alarms in the visualization by the contents of the latch variable.

Example: When monitoring a motor speed, the temperature and the current are additionally recorded.

By default, columns are available for two latch variables. You can add more columns from the context menu which will be available in the "Data type" property for column configuration of the "Alarm table" and "Alarm banner" visualization elements.

"Higher prio. alarm" When the alarm is triggered, it can be acknowledged by triggering another alarm.

Example: A temperature monitor should issue a warning at > 60 °C and an error at > 90 °C. If the temperature rises, then the warning is displayed starting at 60 °C. At 90 °C and higher, the alarm for the error is triggered (this automatically acknowledges the 60 °C warning).

Table 33: Monitoring types

<table>
<thead>
<tr>
<th>Monitoring type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Digital&quot;:</td>
<td>&quot;Expression&quot;: Expression to be monitored. On the right side of the expression for the comparison. In the center, you select the desired comparison operator (&quot;=&quot; or &quot;&gt;&quot;).</td>
</tr>
<tr>
<td>&quot;Upper limit&quot;:</td>
<td>&quot;Expression&quot;: As described for &quot;Digital&quot;, but with the compare options &quot;&gt;&quot; or &quot;&gt;=&quot;, and optionally a &quot;Hysteresis in %&quot;.</td>
</tr>
<tr>
<td>&quot;Lower limit&quot;:</td>
<td>&quot;Expression&quot;: As described for &quot;Digital&quot;, but with the compare options &quot;&lt;&quot; or &quot;&lt;=&quot; and optionally a &quot;Hysteresis in %&quot;.</td>
</tr>
<tr>
<td>&quot;Inside range&quot;:</td>
<td>&quot;Expression&quot;: Expression to be monitored. &quot;Range&quot;: CODESYS triggers the alarm when the monitored expression is within the range defined here. The left field defines an expression for the lower limit of this range. The right field defines an expression for the upper limit. The expression to be monitored is displayed in the field in-between. You have to set the comparison operators accordingly. You can define a &quot;Hysteresis in %&quot; optionally.</td>
</tr>
</tbody>
</table>
### “Outside range”:

“Expression”: Expression to be monitored.

“Range”: CODESYS triggers the alarm when the monitored expression is outside the range defined here. The left field defines an expression for the lower limit of this range. The right field defines an expression for the upper limit. The expression to be monitored is displayed in the field in-between. You have to set the comparison operators accordingly. You can define a “Hysteresis in %” optionally.

### “Change”:

“Expression”: Expression to be monitored. CODESYS triggers the alarm when its value changes.

### “Event”:

In this case, the application triggers the alarm (event alarm) by using functions from the library `AlarmManager.library`.

1) For this observation type, you can monitor the specified expression as absolute or relative. Absolute means that the limit value is defined by a fixed value or a variable that returns a fixed value. Relative means that the limit value depends on an expression; for example: upper limit: “Variable x >= 0.9 * y”.

“Hysteresis in %”: When you specify a hysteresis, the alarm condition is fulfilled until a certain deviation from the specified limit value is reached. The size of the deviation is specified as a percentage [%] of the limit value. Example: Upper limit: `i_temp >= 30` Hysteresis: 10%. The alarm is active when the variable `i_temp` reaches or exceeds the value 30. The alarm is no longer active if the value falls below 27.

The context menu includes the following commands for adding and removing additional columns for messages and latch variables:

- “Add message column”
- “Remove message column”
- “Add latch var column”
- “Remove latch var column”

See also

- **Alarm Table in the Visualization** § Chapter 1.3.5.18.1.22 “Visualization element ‘Alarm table’” on page 1395
- § Chapter 1.3.1.9.21.2 “Configuring the alarm management” on page 258
- § Chapter 1.3.1.9.21.3 “Calling event alarms in the program” on page 261
- § Chapter 1.3.25.1.6 “Object ‘Alarm Class’” on page 715
- § Chapter 1.3.25.1.8 “Object ‘Alarm Storage’” on page 721

### Exporting and importing of alarms

In the context menu of the alarm table you will find commands for the export and import of alarms to and from a CSV file:

- **“Export Alarms”**: The command opens the default dialog for saving a file in the file system. The default file type is `Alarm Group Export File (*.csv)`. This way you can export the current alarm list to a CSV file.
- **“Import Alarms”**: The command opens the default dialog for searching for a file in the file system. The default file type is `Alarm Group Export File (*.csv)` for describing alarm definitions.

### Object ‘Alarm Storage’

**NOTICE!**

In case of a reset (origin), CODESYS clears the alarm storage file.

In “Alarm Storage”, you define the settings for the database file where CODESYS saves the alarms that occur. You can insert the object only once below an alarm configuration.
There is exactly one memory file located in a database. As an option, CODESYS saves this file in a file directory on the controller. You cannot change the name it is derived from the application name according to the following convention: `<application name>.alarmstorage.sqlite`. The alarm classes and alarm groups define whether or not the memory file should be used.

<table>
<thead>
<tr>
<th>“Subdirectory:”</th>
<th>Name of a subdirectory on the PLC where CODESYS saves the memory file (optional).</th>
</tr>
</thead>
<tbody>
<tr>
<td>“No limit”</td>
<td>No limit for the file size for the alarm storage.</td>
</tr>
<tr>
<td>“Maximum number of records”</td>
<td>Number of recorded entries that CODESYS saves. When this number is reached and a new entry is queued in the ring buffer, CODESYS deletes the oldest entry.</td>
</tr>
<tr>
<td>“Maximum storage size”</td>
<td>Maximum size of the memory file with specification of the unit. CODESYS converts this size implicitly into an approximate maximum number of entries that can be saved in the file (ring buffer).</td>
</tr>
<tr>
<td>“Circular buffer”</td>
<td>For display purposes only (not editable).</td>
</tr>
</tbody>
</table>

- : Memory in a circular buffer
- : Memory in an unlimited buffer

The records that CODESYS saves in the database can be displayed in the “Alarm Table” visualization element.

See also
- § Chapter 1.3.1.9.21.2 “Configuring the alarm management” on page 258
- § Chapter 1.3.1.25.1.6 “Object ‘Alarm Class’” on page 715
- § Chapter 1.3.1.25.1.7 “Object ‘Alarm Group’” on page 718

Command 'CSV Export of Alarm Storage'

**Function:** This command opens the “Alarm Storage Export Configuration” dialog box.

**Call:** Context menu of the “Alarm Storage” object.

**Requirements:** The application is in online mode with an alarm configuration. An alarm storage is configured.

This command opens the “Alarm Storage Export Configuration” dialog box.

**Dialog box 'Alarm Storage Export Configuration' - 'Basic Settings'**

**“Export file”:** Path to the generated CSV file in the file system

**Table 34: “Alarms to Export”**

| “Export all” | All listed alarms are exported. |
| “Export alarms in timestamp range (timestamp activated)” | Defined time range for the export of the listed alarms. This refers to the timestamp when the alarm was activated. The min./max. time stamps that are available in the alarm storage or were the last ones to be used in an export are shown by means of “From” and “Until”. When selecting the time range, the smallest unit is one second. |
Table 35: “Language”

| “Language” | The drop-down list includes all languages that are available in text lists for alarm texts, as well as the default language (<Standard>). |

Dialog box
'Alarm Storage Export Configuration' - 'Advanced Settings'

Table 36: “Sort by Time Stamp Activated”

| “Ascending” | Sort order by the time stamp when the alarm was activated |
| “Descending” | |

Table 37: “File Format”

| “Formatted values and timestamps” | The time stamps are formatted with the date and time format as defined below. REAL and LREAL values are formatted in number format. |
| “Raw data from alarm storage file” | The time stamps and values are written unformatted to the CSV file, as they appear in the alarm storage. Alarms, alarm groups, and alarm classes are written in non-readable form to the CSV file as internal IDs. |

Table 38: “Formatting Options”

| Defines the formatting of the entries in the CSV file. Possible only if the “Formatted values and timestamps” data format is selected. |
| “REAL format (printf syntax)” | Setting: “%.1f” |
| “Date format” | The settings are applied from the system control. |
| “Time format” | The settings are applied from the system control. |

Table 39: “Separator”

| “Semicolon” | The entries in the CSV file are separated by a semicolon. |
| “Comma” | The entries in the CSV file are separated by a comma. |
| “Tab” | The entries in the CSV file are separated by a tab. |

| “Local time” | Available only if the “Formatted values and timestamps” data format is selected. |
| ☑: The time stamps are written to the CSV file in local time. |
| ☐: The time stamps are written to the CSV file in universal time coordinated (UTC). |

Object ‘Data Source Manager’

Symbol: 

This object is used as a node for managing the data sources below it. At least one data source must be available. An application with the data source manager communicates with remote devices.

See also

- Chapter 1.3.1.10.4 “Data source manager” on page 280
Command 'Add object' > 'Data source'  

**Function:** This command opens the “Add data source” dialog.

**Call**
- “Project” menu
- Context menu in the “Devices” view of the CODESYS perspective
- Context menu in the “Data sources” view of the “HMI” perspective

**Requirement:** The “Data source manager” object is selected that should have an additional data source.

---

**Dialog 'Add data source'**

<table>
<thead>
<tr>
<th>“Name”</th>
<th>Example: Data_Source_A</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Select data source type”</td>
<td>Data source type that fits the controller configurations to establish communication.</td>
</tr>
<tr>
<td></td>
<td>“CODESYS Symbolic”: The data is transferred by means of symbolic monitoring. This requires that symbols are configured in the symbol configuration of the remote control application.</td>
</tr>
<tr>
<td></td>
<td>Note: As long as the symbol configuration is not affected after an application change, you have the advantage that the application does not have to be updated in the local device.</td>
</tr>
<tr>
<td></td>
<td>Tip: Use this communication connection unless there are no resources are available for a symbol configuration in the remote PLC.</td>
</tr>
<tr>
<td></td>
<td>“CODESYS ApplicationV3”: Data is transferred by means of the CODESYS address protocol. This requires that the address information between the remote PLC and the local device match. Otherwise a connection cannot be established.</td>
</tr>
<tr>
<td></td>
<td>Advantage: A symbol configuration is not required in the remote application.</td>
</tr>
<tr>
<td></td>
<td>Note: For changes to the remote application, the local application must be updated (for example, the HMI application).</td>
</tr>
<tr>
<td></td>
<td>Tip: Use this communication for embedded or mini PLCs when there are no available resources for the symbol configuration.</td>
</tr>
<tr>
<td>“Add”</td>
<td>Opens the dialog “Initialize data source - Provider settings”. The contents of the dialog depend on the selected data source type.</td>
</tr>
</tbody>
</table>

---

**NOTICE!**

The remote controller should be running and the remote controller application loaded and started.

---

See also
- § Chapter 1.3.1.25.1.9 “Object ‘Data Source Manager’” on page 723
- § Chapter 1.3.1.25.1.10 “Object ‘Data Source’ - Tab ‘Variables’” on page 725

---

`Dialog 'Initialize data source wizard - Provider settings' (for 'CODESYS Symbolic')`

The settings of this dialog are described in the following section: Object ‘Data source’ - Tab ‘Communication’.

This dialog is used for the initial configuration of the connection when you select “CODESYS Symbolic” as the data source type. The communication takes place by means of symbolic monitoring. The configuration can be modified later in the editor of the data source in the “Communication” tab.
See also
- "Tab 'Communication' for CODESYS Symbolic" on page 728

The settings of this dialog are described in the following section: Object 'Data source' - Tab 'Communication'.

This dialog is used for the initial configuration of the connection when you select "CODESYS ApplicationV3" as the data source type. The communication takes place by means of address monitoring.

See also
- "Tab 'Communication' for CODESYS ApplicationV3" on page 732

The settings of this dialog are described in the section "Object 'Data source'", heading "Dialog 'Choose variables'".

Function: You can select the variables for the data transfer from the variables of the remote PLC. By clicking "Exit", the data source is initialized and the data types and variables (data interface) are declared below the folder "DataSources_Objects". You can modify the settings in the editor of the data source object.

Call: Automatic

See also
- "Dialog 'Choose Variables" on page 726

Object 'Data Source' - Tab 'Variables'

Symbol:

The object (type "Data source") manages the access to the data of a remote device in the tabs "Variables", "Type mappings", "Communication", and "General and diagnosis". The status bar which is always visible notifies you about the data source type and the most important communication settings.

See also
- Chapter 1.3.1.10 "Working with control networks" on page 270
- Chapter 1.3.1.25.1.11 "Object 'Data Source' - Tab 'Type Mappings'" on page 727
- Chapter 1.3.1.25.1.12 "Object 'Data Source' - Tab 'Communication'" on page 728
- Chapter 1.3.1.25.1.13 "Object 'Data Source' - Tab 'General and Diagnosis'" on page 734

Tab 'Variables'

The variables for the data originating from the remote source are declared in the global variable list <name of data source>. The GVL acts as a data interface for remote control and it is located below the application and folder "DataSources_Objects".

In this way, you can develop a local application offline based on the symbol information without a connection to a data source.

"Update variables" Establishes a connection to the remote device and opens the dialog "Choose variables".

2020/12/10 3ADR010583, 1, en_US 725
<table>
<thead>
<tr>
<th><strong>“Local variable”</strong></th>
<th>Variable in the local application. Contains the remote data.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>“Access rights”</strong></td>
<td>Access permission of the variables. The respective remote variable has the same access permission.</td>
</tr>
<tr>
<td></td>
<td>- ⚔️: Write permission. Every time the values changes, the variable is updated on the controller.</td>
</tr>
<tr>
<td></td>
<td>- ⚔️: Read access. Every time the values changes on the controller, the variable is updated in the application.</td>
</tr>
<tr>
<td></td>
<td>- ⚔️: Read and write access</td>
</tr>
<tr>
<td><strong>“Update always”</strong></td>
<td>☑: The controller data is updated automatically (by means of the data source). A variable is updated automatically if it is used in the visualization, trend, recipe, or as an alarm.</td>
</tr>
<tr>
<td></td>
<td>Note: That is the recommended setting type.</td>
</tr>
<tr>
<td></td>
<td>☑: The variable is updated in each cycle.</td>
</tr>
<tr>
<td></td>
<td>Note: Activate the option only if the variable is used in IEC code exclusively. If a variable is used in the visualization code, then it is updated automatically.</td>
</tr>
<tr>
<td></td>
<td>Note: When an instance of a function block or a data type is updated in this way, the instance is always transferred completely.</td>
</tr>
<tr>
<td><strong>“Create or map”</strong></td>
<td>Mapping type how the remote variable is mapped to the local variable.</td>
</tr>
<tr>
<td></td>
<td>- ⚔️: Mapping to a specific created variable with the data type of the remote variable. The control data is mapped 1:1. That is the recommended mapping type. The variable is declared in the GVL <code>&lt;name of data source&gt;</code>.</td>
</tr>
<tr>
<td></td>
<td>- ⚔️: Mapping to an existing variable. This requires that the existing variable has the same data type.</td>
</tr>
<tr>
<td></td>
<td>- ⚔️: Mapping to a specific created variable with type-conforming data type to the remote data type: remote and local data types are not the same, but compatible. For example, a type-conforming data type can be available in a library. The variable is declared in the GVL <code>&lt;name of data source&gt;</code>.</td>
</tr>
<tr>
<td><strong>“Type mappings”</strong></td>
<td>Data type of the remote variable. If the variable does not have a scalar type, the type is listed in the “Type mappings” tab.</td>
</tr>
<tr>
<td><strong>“Remote variable”</strong></td>
<td>Variable in the remote PLC</td>
</tr>
</tbody>
</table>

**Dialog 'Choose Variables'**

**Symbol:** 🔄

**Function:** This dialog lists the remote variables that are accessed by means of the configured connection.

**Call:** Click “Update variables” in the “Variables” tab.

**Requirement:** The remote PLC is running. The control application is downloaded.
### Variables

The remote variables are listed in the tree view. The top node is identified by the remote application name. Its variables are listed below that. Structured data is listed with all of its subordinate elements.

**Example:** `appControl_A`

- The variable is selected for transferring to the local device. When the variable is structured, it is accepted with all subelements. If the variables themselves are subelements, then only this subelement is accepted without accepting the structure completely.

- Red font: When a variable is displayed in a red font, the variable is not available (anymore) in the remote PLC.

- Note: You can remove the variable from the list by clicking “Uncheck unavailable variables”.

  - The variable is not selected for the transfer.

  - The variable has expandable elements. By clicking the symbol, the variable is extended by their elements.

  **“Insert elements structured”**

  - The selected variables are transferred with this structure if they are structured.

  - The variable is transferred unstructured with a scalar data type.

  **“Uncheck unavailable variables”**

  Requirement: The link is visible when previously are no longer available in the variable available on the remote PLC. These variables are marked in red in the window above. The symbol configuration or the application presumably changed in the remote PLC.

  By clicking the command, the red variables are removed from the drop-down list.

### Status bar

Information about the data source type and the most important communication settings

If the communication is established by means of the data source type CODESYS Symbolic, then the name of the data source type, the connection type, and the network name of the remote device are displayed.

If the communication is established by means of data source type CODESYS ApplicationV3, then the name of the data source type, the storage location of the remote project, and the instance name of the remote application.

**Example:** `CODESYS ApplicationV3 (D:\Projects\Project_A): Project_A.App_A`

### Object 'Data Source' - Tab 'Type Mappings'

The tab lists the non-scalar data types as they are currently available in the “DataSources_Objects” folder. You can edit or delete the data type declaration by selecting a data type and then the declared elements in the lower window. Moreover, you can modify the name, reset access rights, map another type, or select another remote variable.

See also

- Chapter 1.3.1.10.4.2 “Editing data source variables” on page 288

### Tab ‘Type Mappings’
"Local type" | Data type in the local application.
---|---
"Create or map" | ● Mapping to a new created data type. Declared in the "DataSources_Objects" folder.
| ● Mapping to an existing data type
| ● Mapping to a type-conforming data type. Declared in the "DataSources_Objects" folder.
"Mapping name" | Name of the data type
"Remote type" | Data type of the remote PLC

List with the subordinate elements of the selected data type.

| "Local variable" | Local variable name of the element of the selected data type |
| "Access rights" | Access rights to the element |
| "Type mapping" | Data type of the element |
| "Remote variable" | Remote variable name of the element of the selected data type |

[Del] Removes the selected element.

Object 'Data Source' - Tab 'Communication'

The tab includes the communication settings for the remote data source. When adding a data source, a data source type was selected, and the communication settings to the data source were configured depending on it. Then communication settings are managed and edited on this tab. The contents of the tab vary according to the data source type and connection.

See also
- Chapter 1.3.1.10.4.1 “Adding initial data sources” on page 282
- Chapter 1.3.1.25.1.9 “Object 'Data Source Manager'” on page 723

Tab 'Communication' for CODESYS Symbolic

This is the tab when CODESYS Symbolic has been selected as the data source type in the initial configuration. For an active connection, communication takes place by means of symbolic monitoring. Instead of establishing a connection, a reference can be made to a symbol file where the required variable information is stored. In this way, you can develop a local application offline based on the symbol information without a connection to a data source.

| "Variable information" | Source of the variable information |
| "From connection settings" | A connection is established actively according to the communication settings specified below ("Connection type" and "Connection settings"). The variable information is read from the remote control application. |
| "<device name>.<application name>.symbol configuration" | The variable information is read from the symbol configuration, which is part of the active project and located in the device tree at the object of the remote controller below the application. At this time, no active connection is established. |
“From symbol file” The variable information is read from a symbol configuration file that is stored on the development system. Specify this data in the input field “Choose symbol file”. At this time, no active connection is established to a controller.

“Choose symbol file” Path of the symbol file
Requirement: “From symbol file” is set in “Variable information”.
The symbol file is stored on the development system and contains the required variable information. By default, a symbol file path is created in the project directory in the following structure: <project folder>\<project name>.<device name>.<application name>.xml.
Example: D:\Projects\Project_A\VisualizeWithHMI.Device.Application.xml
Note: If the “Alarm table” element or “Trend” element is used in the visualization, then the symbol file required for symbolic access and the respective project must both be saved in the same folder. The project contains the configuration for the alarm table element or the trend recording for the trend element. This is the default case for automatically generated symbol files.
Example: D:\Projects\Project_A\VisualizeWithHMI.project

“Connection type” Connection type between the remote PLC and the local device. Depending on the selected connection type, the following settings below change.
Note: If possible, avoid a direct connection without a gateway.

“CODESYS V2” The devices are located in the same network. The V2 runtime system on the remote PLC provides a communication interface.

“CODESYS V2 (via Gateway)” The devices are not located in the same network. They are connected via a V2 gateway.

“CODESYS V3” The devices are located in the same network. The V3 runtime system on the remote PLC provides a communication interface. Default

“CODESYS V3 (via Gateway)” The devices are not located in the same network. They are connected via a V3 gateway.

<table>
<thead>
<tr>
<th>Table 40: Connection settings for CODESYS V2</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>“PLC”</strong></td>
</tr>
<tr>
<td><strong>“Driver type”</strong></td>
</tr>
<tr>
<td>● “Tcp/Ip (Level 2 Route)”</td>
</tr>
<tr>
<td>● “Tcp/Ip (Level 2)”</td>
</tr>
<tr>
<td>● “Tcp/Ip”</td>
</tr>
<tr>
<td><strong>“Address”</strong></td>
</tr>
<tr>
<td>Example: localhost for the currently used system</td>
</tr>
<tr>
<td><strong>“Port”</strong></td>
</tr>
<tr>
<td>Example: 1200</td>
</tr>
<tr>
<td><strong>“Blocksize”</strong></td>
</tr>
<tr>
<td>Example: 128</td>
</tr>
<tr>
<td>Requirement: The driver type is “Tcp/Ip (Level 2)”.</td>
</tr>
<tr>
<td><strong>“TargetId”</strong></td>
</tr>
<tr>
<td>Example: 0</td>
</tr>
<tr>
<td>Requirement: The driver type is “Tcp/Ip (Level 2 Route)”.</td>
</tr>
<tr>
<td><strong>“Motorola byteorder”</strong></td>
</tr>
<tr>
<td>☑: Byte order on the PLC in big endian (Motorola format)</td>
</tr>
<tr>
<td>☐: Byte order in little endian (Intel format)</td>
</tr>
</tbody>
</table>
Table 41: Connection settings for CODESYS V2 (via gateway)

<table>
<thead>
<tr>
<th><strong>“Gateway”</strong></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>“IP address”</strong></td>
<td>Example: localhost</td>
</tr>
<tr>
<td><strong>“Port”</strong></td>
<td>Example: 1217</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>“PLC”</strong></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>“Name or address of device”</strong></td>
<td>Example of device name: PLC_A</td>
</tr>
<tr>
<td></td>
<td>Example of device address: [ABCD]</td>
</tr>
<tr>
<td><strong>“Type of name or address”</strong></td>
<td>● “Node name (automatically derived)”</td>
</tr>
<tr>
<td></td>
<td>● “Node address (automatically derived)”</td>
</tr>
<tr>
<td></td>
<td>● “IP address (automatically derived)”</td>
</tr>
<tr>
<td></td>
<td>● “Node name”</td>
</tr>
<tr>
<td></td>
<td>● “Node address”</td>
</tr>
<tr>
<td></td>
<td>● “IP address”</td>
</tr>
</tbody>
</table>

Note: For this connection, a “CoDeSys V2.3 Gateway Server” (V2 Gateway) must be also installed on the development computer where CODESYS V3 is running.

Table 42: Connection settings for CODESYS V3

<table>
<thead>
<tr>
<th><strong>“PLC”</strong></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>“Name or address of device”</strong></td>
<td>The input depends on the selected “Type of name or address”.</td>
</tr>
<tr>
<td></td>
<td>Example of device name: PLC_A</td>
</tr>
<tr>
<td></td>
<td>Example of device address: [ABCD]</td>
</tr>
</tbody>
</table>
| | Opens the input for selecting the program variables for dynamic configuration. Requirement: “Dynamic from variable” is selected in “Type of name or address”.
| | Example IEC variable of the application: POU.dssCommVar. This variable must have the data type DatasourceSym.ConnectionSetup. |
| **“Type of name or address”** | ● “Node name (automatically derived)” |
| | ● “Node address (automatically derived)” |
| | ● “IP address (automatically derived)” |
| | ● “Node name” |
| | ● “Node address” |
| | ● “IP address” |
| | ● “Dynamic from variable” |
| | The device name or address is configured dynamically at runtime by means of an IEC variable of data type DatasourceSym.ConnectionSetup. The data type DatasourceSym.ConnectionSetup (struct) is defined in the Datasource Symbolic Access library. For the configuration, the structure member xDataValid first has to be set to FALSE. If the address data has been specified, then xDataValid has to be set back to TRUE. |
| | Use case: The device name or address is not available when a project is being created. |
| | The dynamic configuration can also be used to change the settings at runtime without restarting the HMI application. |
| | Note: For this connection type, the connection also does not take place dynamically via gateway. |
Table 43: Connection settings for CODESYS V3 (via gateway)

<table>
<thead>
<tr>
<th>“Gateway”</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>“IP address”</td>
<td>Example: localhost</td>
</tr>
<tr>
<td>“Port”</td>
<td>Example: 1217</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>“PLC”</th>
<th></th>
</tr>
</thead>
</table>
| “Name or address of device”| The input depends on the selected “Type of name or address”.
Example of device name: PLC_A
Example of device address: [ABCD] |

Opens the input for selecting the program variables for dynamic configuration. Requirement: “Dynamic from variable” is selected in “Type of name or address”.

Example of an IEC variable of the application: POU.dssCommVar. This variable must have the data type DatasourceSym.ConnectionSetup.

| “Type of name or address”  | ● “Node name (automatically derived)”
|                           | ● “Node address (automatically derived)”
|                           | ● “IP address (automatically derived)”
|                           | ● “Node name”
|                           | ● “Node address”
|                           | ● “IP address”
|                           | ● “Dynamic from variable”

The device name or address is configured dynamically at runtime by means of an IEC variable of data type DatasourceSym.ConnectionSetup. The data type DatasourceSym.ConnectionSetup (struct) is defined in the Datasource Symbolic Access library. For the configuration, the structure member xDataValid first has to be set to FALSE. If the address data has been specified, then xDataValid has to be set back to TRUE.

Use case: The device name or address is not available when a project is being created.

The dynamic configuration can also be used to change the settings at runtime without restarting the HMI application.

Note: For this connection type, the connection also takes place dynamically via gateway.
Table 44: “Advanced”

<table>
<thead>
<tr>
<th>“Used as”</th>
<th>“Don’t use”: The text is not used in “INI contents”.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>“Extend the configuration by the following content”: The text in “INI Content” is also used.</td>
</tr>
<tr>
<td></td>
<td>“Configure completely with the following content”: the text in “INI Content” is used for the configuration.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>“INI Content”</th>
<th>Example:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>[PLC:PLC_IdArti]</td>
</tr>
<tr>
<td></td>
<td>interfacetype=ARTI</td>
</tr>
<tr>
<td></td>
<td>active=1</td>
</tr>
<tr>
<td></td>
<td>logevents=1</td>
</tr>
<tr>
<td></td>
<td>motorola=0</td>
</tr>
<tr>
<td></td>
<td>nologin=0</td>
</tr>
<tr>
<td></td>
<td>timeout=10000</td>
</tr>
<tr>
<td></td>
<td>precheckidentity=0</td>
</tr>
<tr>
<td></td>
<td>tries=3</td>
</tr>
<tr>
<td></td>
<td>waittime=12</td>
</tr>
<tr>
<td></td>
<td>reconnecttime=10</td>
</tr>
<tr>
<td></td>
<td>buffersize=0</td>
</tr>
<tr>
<td></td>
<td>device=Tcp/Ip (Level 2 Route)</td>
</tr>
<tr>
<td></td>
<td>instance=PLCWinNT_TCPIP_L2Route</td>
</tr>
<tr>
<td></td>
<td>parameters=4</td>
</tr>
<tr>
<td></td>
<td>parameter0=Address</td>
</tr>
<tr>
<td></td>
<td>value0=localhost</td>
</tr>
<tr>
<td></td>
<td>parameter1=Port</td>
</tr>
<tr>
<td></td>
<td>value1=1200</td>
</tr>
<tr>
<td></td>
<td>parameter2=TargetId</td>
</tr>
<tr>
<td></td>
<td>value2=0</td>
</tr>
<tr>
<td></td>
<td>parameter3=Motorola byteorder</td>
</tr>
<tr>
<td></td>
<td>value3=No</td>
</tr>
</tbody>
</table>

Requirement: “Used as” is “Extend the configuration by the following content” or “Configure completely with the following content”.

Table 45: “Login Configuration”

If a user management is configured on the remote PLC, then valid access data is required at login.

<table>
<thead>
<tr>
<th>“Type”</th>
<th>“Login using the following credentials”: The access data is hard-coded in the “User name” and “Password” settings. They are used at each connection attempt.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>“Login using the credentials determined at runtime”: At runtime, a dialog prompts for the name and password. The “Username” and “Password” settings are disabled.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>“User name”</th>
<th>Example: max.smith</th>
</tr>
</thead>
<tbody>
<tr>
<td>Requirement: The setting “Type” is “Login using the following credentials”.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>“Password”</th>
<th>Example: ...</th>
</tr>
</thead>
</table>
| Requirement: The setting “Type” is “Login using the following credentials”.

Tab ‘Communication’ for CODESYS ApplicationV3

This is the tab when CODESYS ApplicationV3 has been selected as the data source type in the initial configuration. The communication takes place by means of address monitoring.
**“Select the project type”**
- “Current project”: The control application is part of the current project.
- “Other project”: The control application is stored as a separate project and not part of the current project.

**“Choose file”**
Name and path of the project that the application contains for the remote PLC.
Example: D:\plcs\plc_A.project
Requirement: The “Select the project type” is “Other Project”.

Window that lists the detected control variables in a tree view. When the control data is displayed here, a connection can be established.

---

**Table 46: “Target Device”**

| **“Automatic configuration”** | CODESYS attempts to read the configuration automatically from the source project. At this time, the communication settings of the device are accepted in the active network path. They are configured in the editor of the device in the “Communication” tab. |
| “[undetermined]”: No configuration can be read. |

**“Manual configuration”**

**“Dynamic from variable”**
The communication parameters are configured at runtime by means of an IEC variable of the application of data type `DatasourceAppV3.ConnectionSetup`.  
 Opens the input for selecting the IEC variables for dynamic configuration.  
 The data type `DatasourceAppV3.ConnectionSetup` (struct) is defined in the `Datasource ApplicationV3 Access` library. For the configuration, the structure member `xDataValid` first has to be set to `FALSE`. If the address data has been specified, then `xDataValid` has to be set back to `TRUE`.  
 Use case: The communication parameters are not available yet when a project is being created.

**“Use device address”**
The communication takes place through the address specified here.

**“From device”**
By clicking “From device”, CODESYS automatically enters the data of the connected data source device as it is configured in its “Communication Settings”.

**“Derive device address automatically”**:  
The communication takes place through the address that is defined in the “Communication Settings” of the target device.  
Note: Recommended setting

**“Use device address”**
The address of the connected data source device is entered as configured in its communication settings.  
Example: [057B]

The following cases lead to an incorrect address:
- The gateway for communication with the device is running on a computer with multiple addresses.
- The device where the application is running with the data source manager has multiple IP addresses or other interfaces (USB, serial, etc.) for communication with CODESYS.
- The CODESYS runtime system and the gateway are not located in the same Ethernet subnet.  
  Hint: Use the “Automatic configuration” setting.

**“Search for the target device using the network scan”**
Possible search criteria for the device:

**“Node name”**
Example: WST06

**“Target type”**
Example: 4096
"Target ID"  Example: 0000 0001

"Target version"  Example: 1.0.0.0

"Network location":
- "Direct child of the data sources plc": The scanned remote PLC has an address that is running with the address of the local controller (of the data source manager).
  Example: Data sources PLC: 0000.0001; remote source PLC: 0000.0001.0001
- "Direct child of node with address": Specify the address of the parent node
- "Direct child of the data source plc or of the node with address": Combination of both options above.

"Search type":
- "First found device": The first controller in the device tree is selected that fulfills the specified criteria.
- "Exactly found device": The controller is selected that fulfills the specified criteria.
  Note: The data source manager waits until the network scan is complete. This usually takes about 10 seconds.

Table 47: “Login Configuration”

If a user management is configured on the remote PLC, then valid access data is required at login.

"Type"

- “Login using the following credentials”: The access data is hard-coded in the “User name” and “Password” settings. They are used at each connection attempt.
- “Login using the credentials determined at runtime”: At runtime, a dialog prompts for the name and password.

"User name"  Example: max.smith
  Requirement: The setting “Type” is “Login using the following credentials”.

"Password"  Example: ...
  Requirement: The setting “Type” is “Login using the following credentials”.

"Advanced"  
  ☑: The following settings are visible.
  Requirement: “Automatic configuration” is activated.

"Default communication buffer"  Example: 50000

Object ‘Data Source’ - Tab ‘General and Diagnosis’
The “General and Diagnosis” tab provides information about the status of the data source communication.

See also
- ☰ Chapter 1.3.1.10.4.4 “Updating data interfaces” on page 291

Tab ‘General and Diagnosis’
Table 48: “Update Configuration”

"Update rate (ms)"  Example: 200
Object 'DUT'

Symbol: ⚫ for DUT without text list support; ⬇️ for DUT (type: enumeration) with text list support

A DUT (data unit type) describes a user-specific data type. You can add a “DUT” object below the application or in the “POUs” view.

When adding the object, you can specify definition (see below: Dialog “Add DUT”).

Syntax

```
TYPE <identifier> : <DUT component declaration>
END_TYPE
```

The component declaration depends on the selected type, structure, or enumeration.

Example:

```
The following two DUTs define the structures struct1 and struct2, in which struct2 extends struct1. This mean that struct2.a can be used to access the a variable.

    TYPE struct1 :
        STRUCT
            a:INT;
            b:BOOL;
        END_STRUCT
    END_TYPE

    TYPE struct2 EXTENDS struct1 :
        STRUCT
            c:DWORD;
            d:STRING;
        END_STRUCT
    END_TYPE
```

See also

- ☰ Chapter 1.3.1.24.5.16 “Structure” on page 594
- ☰ Chapter 1.3.1.24.5.17 “Enumerations” on page 595
- ☰ Chapter 1.3.1.24.5.6 “Data type ‘UNION’” on page 572

Dialog 'Add DUT'

Function: You use this dialog log for configuring a new DUT. The DUT (Data Unit Type) describes a user-specific data type.

Call: Menu bar “Project ➔ Add Object ➔ DUT”; context menu of the application object.

| “Name” | DUT name |
Table 50: “Type”

<table>
<thead>
<tr>
<th>“Structure”</th>
<th>The DUT consists of a structure of various data types. “Extends”: The structure extends an existing structure. In this way, the definitions of the extended DUT are automatically valid in the new DUT.</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Enumeration”</td>
<td>The DUT consists of a series of integer constants. For declaration syntax, please refer to the help page for &quot;Enumerations&quot;. “Add text list support”: The text list support allows the localization of enumeration components. In addition, both the symbolic and numeric enumeration values are shown in the text display of a visualization element. When an enumeration variable with text list support is specified in the “Text variable” property of a visualization element, the enumeration variable obtains the additional &lt;enumeration name&gt;. By clicking the buttons on the right edge of the editor, you can switch between “Textual view” and “Localized view” (text list). The DUT object is found as the symbol in the device tree and in the “POUs” view. Example: You use the PLC_PRG.enVar variable (type myEnum). myEnum is a DUT with text list support. Then the entry in the property editor looks like PLC_PRG.enVar &lt;myEnum&gt;. If the enumeration type is modified in the application, then you are prompted to allow CODESYS to update the affected visualizations accordingly. Note: Text list support can be added or removed at any time for an existing enumeration object. For this purpose, the commands “Add Text List Support” and “Remove Text List Support” are provided in the context menu of the object.</td>
</tr>
<tr>
<td>“Alias”</td>
<td>Definition of an alias for a “Base type”. You can type in the base type directly or select it using the input assistant or array assistant.</td>
</tr>
<tr>
<td>“Union”</td>
<td>All components reserve the same storage space for this data type. The memory requirement of a union is determined by the memory requirement of its &quot;largest&quot; component.</td>
</tr>
</tbody>
</table>

See also
- % Chapter 1.3.1.24.5.16 “Structure” on page 594
- % Chapter 1.3.1.24.5.17 “Enumerations” on page 595
- % Chapter 1.3.1.24.5.6 “Data type 'UNION'” on page 572
- % Chapter 1.3.1.25.2.20.12 “Command 'Add Text List Support'” on page 1024
- % Chapter 1.3.1.25.2.20.13 “Command 'Remove Text List Support'” on page 1024
- Help for CODESYS Visualization: Using texts

Object ‘External File’

An “External File” is any file that you add to the project in the “POUs” view or “Devices” view. Click “Project ➔ Add Object” to open the “Add External File” dialog and define how the file belongs to the project.

An external file which was inserted in the “POUs” view is never downloaded to the controller.

An external file which was added in the “Devices” view is always downloaded to the controller when an online change or a download is performed due to an IEC code change.

When an external file is downloaded to the controller, it is not updated in the project.

Dialog ‘Add External File’
“File path” | The button opens a dialog for selecting a file in the local file system.
---|---
“Name” | Object name for the file in CODESYS. If you do not type anything, the file will have its previous name.

Table 51: “File Handling”
| “Remember the link” | The file is available in the project only as long as it exists in the defined file path. |
| “Remember the link and embed into project” | CODESYS saves an internal copy of the file in the project, as well as the link to the defined file path. The update option selected below applies as long as the external file exists there. Otherwise CODESYS uses the version saved in the project. |
| “Embed into project” | CODESYS saves only one copy of the file in the project. There is no longer a link to the external file. |

Table 52: “Change Tracking”
| “Reload the file automatically” | If the external file changes, then CODESYS updates the file in the project. |
| “Prompt whether to reload the file” | If the external file changes, then a dialog prompt opens whether CODESYS should also update the file in the project. |
| “Do nothing” | The file remains unchanged in the project, even if the external file changes. |

| “Display File Properties” | Clicking this button opens the default “Properties of <file name>” dialog, which you can also open in the Windows file system by right-clicking the file. |
| “Open” | The file object is inserted into the device tree (“Devices” or “POUs” view) and opened in the editor for the matching file format. |

See also
- § Chapter 1.3.1.3.2 “Adding objects” on page 125
- § Chapter 1.3.1.25.2.4.1 “Command ‘Add Object’” on page 897
- § Chapter 1.3.1.25.3.10.7 “Dialog ‘Properties’ - ‘External file’” on page 1050

Object ‘C Code Module’

Symbol: 

This object is for integrating C-code modules into a CODESYS project. You define the necessary configuration parameters on the tabs of the object.

The object “C Code Module” is added to the application in the device tree. Together with the object, CODESYS automatically appends the folders “Extensions”, “IEC Interface” and “Source Files”.

(1) “C Code Module”
(2) “Extensions”
(3) “IEC Interface”

(4) “Source Files”

You can add the following objects to a “C Code Module”:

- “C Code File”
- “DUT”
- “POU”

**Tab 'Includes & Defines'**

<table>
<thead>
<tr>
<th>“Include Paths”</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>“Path”</td>
<td>Display of the include paths</td>
</tr>
<tr>
<td>“Add”</td>
<td>Opens the dialog box “Find folder” for adding an additional include path.</td>
</tr>
<tr>
<td>“Remove”</td>
<td>Removes the selected include path</td>
</tr>
<tr>
<td>“Defines”</td>
<td>List of the defines</td>
</tr>
</tbody>
</table>

**Tab 'Libraries'**

<table>
<thead>
<tr>
<th>“Library paths”</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>“Path”</td>
<td>List of the library paths</td>
</tr>
<tr>
<td>“Add”</td>
<td>Adds a new library path</td>
</tr>
<tr>
<td>“Remove”</td>
<td>Deletes the selected library path</td>
</tr>
<tr>
<td>“Libraries”</td>
<td>List of the libraries</td>
</tr>
</tbody>
</table>

**Tab 'Components'**

<table>
<thead>
<tr>
<th>“Component”</th>
<th>2 “Component” windows</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>The left-hand window displays all available runtime system components.</td>
</tr>
<tr>
<td></td>
<td>The right-hand window displays the runtime system components that you have added to this window using the “Add” button in order to be able to use their functions in C-source code files from externally implemented libraries.</td>
</tr>
<tr>
<td>“Add”</td>
<td>Moves the runtime system component selected in the left-hand “Component” window into the right-hand “Component” window.</td>
</tr>
<tr>
<td>“Remove”</td>
<td>Removes the runtime system component selected in the right-hand window and displays it in the left-hand window again.</td>
</tr>
<tr>
<td>“Function”</td>
<td>List of the functions that are provided by the selected runtime system component.</td>
</tr>
</tbody>
</table>

**Dialog box 'Add C Code Module'**

<table>
<thead>
<tr>
<th>“Name”</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>“Source folder”</td>
<td>Input field for folders containing the C-source code files</td>
</tr>
<tr>
<td></td>
<td>Opens the dialog box “Search Folder”</td>
</tr>
<tr>
<td>“Monitor folder for source changes”</td>
<td>CODESYS monitors whether the original directory of the project on the hard disk changes and, in such a case, it opens a dialog box in which you can consent to the updating of the files in CODESYS.</td>
</tr>
</tbody>
</table>
Object ‘C Code File’

Symbol: 🗄

This object contains the C-code for the C-integration in its editor.

The C-code files can be created in the following ways:

- You add an object “C Code Module” to the application and, in the dialog box “Add C Code Module”, you specify a source directory containing C-code files; you import this directory into CODESYS.
- You add an object “C Code File” to a “C Code Module” and in doing so you specify a C-file in the dialog box “Add C Code File” that you import into CODESYS.
- You add an object “C Code File” to a “C Code Module” and in doing so you specify the name for a new C-file in the dialog box “Add C Code File”. As opposed to the two other methods you create an empty file by doing this.

Possible file formats of a C-code file:

- *.c
- *.ccp
- *.h
- *.hpp

You can make simple changes to the C-code in the text editor.

Objects for C-stubs:
'iec_external.c' and 'iec_external.h'

If you create C-stubs with the command “Create Stub Implementation in C”, these are stored in the objects “iec_external.c” and “iec_external.h” of the “Extensions” folder. The object “iec_external.c” contains the C-stubs, “iec_external.h” contains the associated declarations.

NOTICE!

The object “iec_external.c” contains a marked range for header includes (range between /** INCLUDE SECTION BEGIN **/ and /** INCLUDE SECTION END **/). You may not change this range!

See also

- § Chapter 1.3.1.25.2.5.13 “Command ‘C Integration – Create Stub Implementation in C’” on page 921
- § Chapter 1.3.1.9.11 “Integrating C Modules” on page 222
- § Chapter 1.3.1.25.1.17 “Object ‘C Code File’” on page 739

Object ‘C Implemented Library’

Symbol: 🗄

You add the object to a library project in the “POUs” view. A “C Implemented Library” can contain several precompiled modules, each of which is assigned to a certain device. The inserted modules are not visible in the library manager.

If you double-click on the object in the “POUs” view, it opens in its editor.
Table 53: Compiled Components

<table>
<thead>
<tr>
<th>Object files for devices</th>
</tr>
</thead>
<tbody>
<tr>
<td>List of the devices with the assigned module with the following information:</td>
</tr>
<tr>
<td>Device, Version, File name, File size</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Add</th>
</tr>
</thead>
<tbody>
<tr>
<td>Opens the dialog box “Select Device”, where you can assign a precompiled module to a device and add it to the C-implemented library.</td>
</tr>
<tr>
<td>Any number of modules can be added to such an object. When downloading a concrete project with this library to a device, only the respectively matching file is loaded to the device.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Remove</th>
</tr>
</thead>
<tbody>
<tr>
<td>Removes the selected entry in “Object files for devices”.</td>
</tr>
</tbody>
</table>

Table 54: “Choose Device”

<table>
<thead>
<tr>
<th>Object File</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input of a file that matches the “Device” and can be loaded by the operating system as a dynamic object, for example for Windows: *.dll, for Linux: *.so</td>
</tr>
<tr>
<td>Please note: The file *.dll must contain the title of the library project in its name. Example: if the library project has name XYlib, then the “Object File” must be called: &lt;name&gt;_XYlib.dll</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Device</th>
</tr>
</thead>
<tbody>
<tr>
<td>● “Vendor”</td>
</tr>
<tr>
<td>● List of the devices</td>
</tr>
<tr>
<td>● Options</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>● Information about the device.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Chose device</th>
</tr>
</thead>
<tbody>
<tr>
<td>The runtime module precompiled in the “object file” is assigned to the selected “device”.</td>
</tr>
</tbody>
</table>

See also

● Chapter 1.3.1.9.11 “Integrating C Modules” on page 222

Object ‘Device’ and generic device editor

Symbol: 

A device object represents a type of hardware; examples: control device, fieldbus node, bus node, drive, I/O module, monitor. The arrangement of the device objects in the device tree, that is the view “Devices” in CODESYS, maps the hardware structure. In the device object configuration editors inter alia you connect the controller I/Os with project variables.

Use command “Add Device” or “Insert Device” to insert a device object in the device tree. Depending on the insert position CODESYS always offers the currently matching devices.

A double-click on a device object in the device tree opens the associated device editor. The editor provides generic and device-specific tabs for the device configuration.

See also
Generic device editor

The generic device editor contains tabs for the configuration of a PLC device in CODESYS. Additionally there are device-specific tabs, so that the configuration editor consists of many different dialogs, depending on the device.

The editor opens after a double-click the device object in the device tree ("Devices" view).

You can make general settings for a device editor in the CODESYS “Options” in the “Device Editor” category. For example, you can show and hide the tabs of the generic device editor.

A device editor is given the name of the device. The following tabs of the generic device editor can be included:

- “Communication”: Configuration of the connection between the development system and a programmable device (PLC). Not available in the case of pure I/O devices.
- “Applications”: List of the applications on the controller.
- “<device> Parameters”: Display and configuration of device parameters.
- “Files”: Configuration of the file transfers between a host file system and the device.
- “Log”: Display of the PLC log file.
- “PLC Settings”: Configuration of the handling of the I/Os: which application, behavior in the stop state, updating, bus cycle options, etc.
- “PLC Shell”: Text-based control monitor for interrogating certain information from the controller.
- “Users and Groups”: User management with regard to the device at runtime.
- “Access Rights”: Rights for access to objects and files on the device.
- “Symbol Rights”: Access rights of individual user groups to symbols (symbol sets) on the device.
- “Task List”: Overview of all inputs and outputs, which are assigned to tasks – useful for troubleshooting.
- “Status”: Device-specific status and diagnostic messages.
- “Information”: General information about the device (name, vendor, version etc.)

See also

- % Chapter 1.3.1.8.1 “Device tree and device editor” on page 155
- % Chapter 1.3.1.8 “Configuring I/O links” on page 155

Tab 'Communication Settings'

In this tab of the generic device editor, you define the connection between CODESYS and the device where your application should run.

If you prefer the classic mode of display for the dialog, then select it in the CODESYS “Options” (“Device Editor” category).

You select a gateway and a target device from the list boxes. The possible selections depend on the entries in the “Manage gateways” and “Manage favorite devices” dialogs (see the “Gateway” menu).

You can also type in select the target device directly with the IP address (example: “192.168.101.109”), device address (example: “[056D]”), or device name (example: “MyDevice”). After the device is entered successfully, CODESYS searches for the device in the network of the gateway.
The option of searching by device name requires unique device names in the network.

The filled circle on the lower right corner of the gateway symbol provides information about the connection status:

- Red: CODESYS cannot establish the connection.
- Green: The connection is established.
- Black: The connection status is unknown.

Some communication protocols allow regular checking of the gateway so that the status cannot be displayed.

Clicking the filled circle of the target device starts a network scan for the device. This works only if the network is not already being scanned.
“Scan network” This button opens the “Select Device” dialog. It lists all configured gateways with the associated devices. You can select one target device from this list. Refer to the description of the classic view below for details about this dialog.

“Gateway” This menu includes the following commands:
- “Add New Gateway”: Opens the “Gateway” dialog for defining a new gateway channel.
- “Manage Gateways”: Opens the “Manage Gateways” dialog with an overview of all gateways. You can add or delete entries here or change their order.
- “Configure the Local Gateway”: Opens the “Gateway Configuration” dialog. You can configure the block drivers for the local gateway.

“Device” This menu includes the following commands:
- “Add Current Device to Favorites”: Adds the currently set device to the list of favorite devices.
- “Manage Favorite Devices”: Opens the favorites dialog with a list of all preferred devices. In this dialog, you can add or delete entries or change their order. The top device is the default.
- “Rename Active Device”: Opens the “Change Device Name” dialog.
- “Wink Current Device”: Devices that support this function illuminate a flashing signal.
- “Send Echo Service”: CODESYS sends five echo services to the PLC. These are used to test the network connection, similar to the ping function. The services are sent first without data packets and then with data packets. The scope of the data packets depends on the communication buffer of the PLC. A message box opens with information about the average echo service delay and the scope of the sent data packets.
- “Store Communication Settings in Project”: CODESYS saves the communication settings in the project for reuse on the same computer.
  Note: If you use the project on another computer, then you have to reset the active path.
- “Confirmed Online Mode”: CODESYS requires you to confirm the following when calling the following online commands (for safety purposes): Force values, Write values, Multiple loading, Remove force list, Single cycle, Start, Stop.
- “Filter Network Scans by Target ID”: The display is limited on the devices that have the same target ID as the current device configured in the project.
- “Encrypted Communication”: The communication to this controller is encrypted. A certificate of the controller is required in order to log in to the controller. If the certificate is not available, then an error message opens prompting whether or not the certificate should be displayed and installed.
  If the “Enforce Encrypted Communication” option is selected as “Security level” in the “Security Screen” view, then the “Encrypted Communication” command is disabled here.
- “Change Communication Policy” Opens the “Change Communication Policy” dialog for changing the device setting for the encryption of communication.

Table 55: Dialog “Change Communication Policy”

| If a new communication policy is selected in this dialog, then the configuration on the controller is changed. | “Communication” |

2020/12/10  3ADR010583, 1, en_US  743
<table>
<thead>
<tr>
<th>“Current policy”</th>
<th>Shows the currently selected policy for the encryption of communication</th>
</tr>
</thead>
<tbody>
<tr>
<td>“New policy”</td>
<td>Drop-down list for the new policy for encryption</td>
</tr>
<tr>
<td></td>
<td>● “No encryption”: The controller does not support encrypted communication.</td>
</tr>
<tr>
<td></td>
<td>● “Optional encryption”: The controller supports encrypted and unencrypted communication.</td>
</tr>
<tr>
<td></td>
<td>● “Enforced encryption”: The controller supports encrypted communication only.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>“Device User Management”</th>
<th>Shows the currently selected policy for user management</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Current policy”</td>
<td></td>
</tr>
<tr>
<td>“New policy”</td>
<td>Drop-down list for the new policy for user management</td>
</tr>
<tr>
<td></td>
<td>● “Optional user management”: It is the responsibility of the user to enable user management on the device or leave the device unprotected.</td>
</tr>
<tr>
<td></td>
<td>● “Enforced user management”: The user management on the device is enabled and cannot be disabled by the user.</td>
</tr>
</tbody>
</table>

See also
- Chapter 1.3.1.11.2 “Encrypting communication, changing security settings” on page 296
- Chapter 1.3.1.25.3.13.9 “Dialog ‘Options’ - ‘Device Editor’” on page 1081
- Chapter 1.3.1.25.2.4.5 “Command ‘Scan for Devices’” on page 899
- Chapter 1.3.1.25.2.18.1 “Command ‘Add gateway’” on page 1012
- Chapter 1.3.1.25.2.4.3 “Command ‘Insert Device’” on page 898
- Chapter 1.3.1.25.2.18.2 “Command ‘Configure the local gateway’” on page 1013

**Communication Settings - Classic Mode**
In the CODESYS options, you can activate the classic mode of the dialog ("Tools ➔ Options", category "Device editor").

<table>
<thead>
<tr>
<th>“Select the network path to the controller”</th>
<th>Gateway channel for the connection.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Select the channel from the lower part of the view.</td>
</tr>
</tbody>
</table>
Table 56: “View displaying configured gateway channels and network devices”

<table>
<thead>
<tr>
<th>Left side of view</th>
<th>Tree structure of the configured gateway channels with the connected devices in the local network:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><img src="image" alt="Gateway-1" /> + <img src="image" alt="User1[0146]" /> <img src="image" alt="User2[0124]" /></td>
</tr>
</tbody>
</table>

Note: CODESYS saves these entries on the local system, not in the project.

The device entries are preceded by a device symbol (●). Entries with a target ID that are different from those currently configured in the project are displayed in gray.

Clicking “Scan Network” refreshes the list.

Note: If you created the first project on the local system, then the local gateway is listed as an entry in the tree by default. CODESYS starts this gateway automatically on system boot.

The filled circle on the lower right corner of the gateway symbol provides information about the connection status:
- Red: CODESYS Development System cannot establish the connection.
- Green: The connection is established.
- Black: The connection status is unknown.

Note: Some communication protocols allow regular checking of the gateway so that the status cannot be displayed.

Each of the device entries in the tree consist of a symbol followed by the “Device Name” > [“Device Address”]. On the right side of the view, you also see the “Target ID”, “Target Name”, “Target Type”, “Target Vendor”, and “Target Version”.

Right side of view | Information about the gateway channel of device selected on the left side of the view.

When a gateway channel is selected in the left view, the following information is displayed: “Device Name”, “IP-Address”, “Port”, “Driver”

When a device is selected in the left view, the following information is displayed (depending of the device): “Device name”, “Device address”, “Number of channels”, “Block driver”, “Serial number”, “Encrypted communication”, “Target vendor”, “Target ID”, “Target name”, “Target type”, “Target version”.

Table 57: “Filter and sorting functions on the right side of the dialog”

<table>
<thead>
<tr>
<th>Filter</th>
<th>You can reduce the displayed list of devices that have the same “Target ID” as the current device configured in the project.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sorting order</td>
<td>You can sort the list by “Name” or “Device Address” in alphabetical or ascending order.</td>
</tr>
</tbody>
</table>

Table 58: Command buttons on the right side of the dialog

<table>
<thead>
<tr>
<th>Set Active Path</th>
<th>The command sets the selected communications channel as active. Double-clicking the entry in the channel tree achieves the same result.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add Gateway</td>
<td>This command opens the “Gateway” dialog where you can define a gateway that CODESYS should add to the current configuration.</td>
</tr>
<tr>
<td>Add Device</td>
<td>This command opens the “Add Device” dialog. Here you can manually define a device that is to be inserted under the gateway entry currently selected in the tree. Please note the functionality of “Scan Network” as well.</td>
</tr>
<tr>
<td>Scan Network</td>
<td>The command starts a search for available devices in the local network. The configuration tree of the gateway is refreshed accordingly.</td>
</tr>
</tbody>
</table>
Table 59: “Commands in the context menu of the gateway tree and device tree in the dialog”

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Scan for Device by Address”</td>
<td>The command searches the network for devices with a unique address as given in the configuration tree. CODESYS displays the detected devices with the given address below the gateway. The search always applies to the devices below the selected gateway or below the selected entry.</td>
</tr>
<tr>
<td>“Scan for Device by Name”</td>
<td>The command searches the network for devices with the same name as given in the configuration tree. Capitalization is ignored here. CODESYS displays the detected devices below the gateway with the given name together with its unique device address. The search always applies to the devices below the selected gateway or below the selected entry.</td>
</tr>
<tr>
<td>“Scan for Device by IP Address”</td>
<td>The command searches the network for devices with a unique IP address as given in the configuration tree. CODESYS displays the detected devices with the given address below the gateway together with its name. The search always applies to the devices below the selected gateway or below the selected entry.</td>
</tr>
<tr>
<td>“Send Echo Service”</td>
<td>CODESYS sends five echo services to the PLC. These are used to test the network connection, similar to the ping function. The services are sent first without data packets and then with data packets. The scope of the data packets depends on the communication buffer of the PLC. A message box opens with information about the average echo service delay and the scope of the sent data packets.</td>
</tr>
<tr>
<td>“Delete Selected Device”</td>
<td>This command deletes the selected device from the channel tree.</td>
</tr>
<tr>
<td>“Edit Gateway”</td>
<td>This command opens the “Gateway” dialog for editing the settings for the selected gateway.</td>
</tr>
<tr>
<td>“Configure the Local Gateway”</td>
<td>This command opens a dialog for configuring a local gateway. This provides an alternative to manually editing the Gateway.cfg file.</td>
</tr>
</tbody>
</table>

Table 60: Options in the lower part of the dialog

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
</table>
| “Don’t store communication settings in project” | ☑️: The current connection settings are saved to the local CODESYS options on your computer, not in the project. This means that you have to reset them if you use the project on another system.  
☐: The settings are saved in the project and CODESYS Development System restores them automatically when you open the project on another computer. |
| “Confirmed online mode”                       | ☑️: CODESYS requires you to confirm the following when calling the following online commands (for safety purposes): “Force Values”, “Write Values”, “Multiple Loading”, “Release Forcelist”, “Single Cycle”, “Start”, “Stop”. |

See also

- ☰ Chapter 1.3.1.25.3.13.9 “Dialog ‘Options’ - ‘Device Editor’” on page 1081
- ☰ Chapter 1.3.1.25.2.4.5 “Command ‘Scan for Devices’” on page 899
- ☰ Chapter 1.3.1.25.2.18.1 “Command ‘Add gateway’” on page 1012
- ☰ Chapter 1.3.1.25.2.4.3 “Command ‘Insert Device’” on page 898
- ☰ Chapter 1.3.1.25.2.18.2 “Command ‘Configure the local gateway’” on page 1013

Tab 'Parameters'

This dialog is intended for test purposes. Its values should be changed only by experts.

The device-specific parameters are displayed in a table on this tab of the generic device editor. The device description defines which parameters you can edit in this dialog.
You can sort the entries in alphabetically ascending or descending order or in the default order by clicking the column header.

<table>
<thead>
<tr>
<th>“Parameter”</th>
<th>Parameter name, not editable</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Type”</td>
<td>Data type of the parameter, not editable</td>
</tr>
<tr>
<td>“Value”</td>
<td>Initially displays the default value of the parameter, directly or the corresponding symbol name. Non-editable parameters are displayed in light-gray. If the parameter is editable you can open an input field, a drop-down list or a file selection dialog with a double-click the table field and use it to change the value.</td>
</tr>
<tr>
<td>“Default value”</td>
<td>Default value of the parameter defined by the device description, not editable</td>
</tr>
<tr>
<td>“Unit”</td>
<td>Unit of measure for the value (example: &quot;ms&quot; for milliseconds; not editable)</td>
</tr>
<tr>
<td>“Description”</td>
<td>Short description of the parameter specified by the device description, not editable</td>
</tr>
</tbody>
</table>

See also

- Chapter 1.3.1.25.1.19.1 “Generic device editor” on page 741

**Tab ‘Applications’**

On this tab of the generic device editor you can see which applications exist on the device. Depending on the system you can delete the applications from the device or retrieve detailed information about the application.

<table>
<thead>
<tr>
<th>“Applications on the PLC”</th>
<th>List of the applications found via “Refresh list” during the last scan of the control device.</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Delete”</td>
<td>Deletes the application selected in the list or all listed applications on the controller</td>
</tr>
<tr>
<td>“Delete All”</td>
<td>Note: If a safety controller is inserted below a PLC, then this command can permanently interrupt the communication links of the safety controller to other safety controllers (via safety network variables), to field devices, and to the development system. The safe field devices and the other safety controller can enter the safe state as a reaction. The connection to the development system is affected only in the case of a safety controller that is connected to the main controller via a fieldbus. For more information, refer to the section “Subordinate Safety Controller”.</td>
</tr>
<tr>
<td>“Details”</td>
<td>Opens the dialog box “Details”. It displays information defined for the application on the “Information” tab of the dialog box “Properties”.</td>
</tr>
<tr>
<td>“Contents”</td>
<td>Requirement: The “Download the application info” option is activated in the “Properties” of the application object on the “Application generation options” tab. This causes information about the contents of the application to be additionally loaded to the PLC. The “Contents” button opens a dialog box with additional information about the differences between the latest generated code and the application code that exists on the controller. The different modules are displayed in a comparison view.</td>
</tr>
<tr>
<td>“Refresh List”</td>
<td>The controller is scanned for applications and the list is refreshed accordingly</td>
</tr>
</tbody>
</table>

You can configure the commands “Remove Application from Device” and “Remove Applications from Device” by means of the dialog box form “Tools ➔ Customize”. These commands correspond to the “Delete” and “Delete All” buttons.
Tab 'Backup and Restore'

In this tabbed page of the generic device editor, you can backup and restore the application-specific file on the PLC by saving and reading a zip archive.

Requirement: The communication settings are correct for connection to the device. The application for backup is available on the PLC.

<table>
<thead>
<tr>
<th>Table 61: Menu Bar</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>“Backup”</strong></td>
</tr>
<tr>
<td>This button opens a menu with the following commands:</td>
</tr>
<tr>
<td>- ✋ “Read Backup Information from Device”: This command searches for application-specific files from the $PlcLogic$ directory of the PLC and lists them in a table in the lower part of the tabbed page.</td>
</tr>
<tr>
<td>- 📁 “Create Backup File and Save to Disk”:</td>
</tr>
<tr>
<td>Requirement: The “Read Backup Information from Device” command was used for determining the backup-related files. These files are located in the table in the lower part of the tabbed page.</td>
</tr>
<tr>
<td>This command compresses the files in the table set as “Active” and the meta.info information file into a backup zip file. The file extension is tbf (=&quot;Target Backup File&quot;).</td>
</tr>
<tr>
<td>- ✋ “Save Backup File to Device”:</td>
</tr>
<tr>
<td>Requirement: The backup file has been saved to the disk. This command saves the backup file to the TBF directory of the PLC.</td>
</tr>
</tbody>
</table>

| **“Restore”**       |
| This button opens a menu with the following commands: |
|   - 📁 “Load Backup File from Disk”: This command opens the “Open” dialog box for navigating the file system for a saved backup file. The included files are listed in a table in the lower part of the tabbed page. |
|   - ✋ “Load Backup File from Device”: This command generates a list of all backup files found on the PLC. Select one of these files to view its contents in a table on the tabbed page. For the restore operation, you can deactivate optional components and edit the comments. |
|   - ✋ “Restore on Device”: This command is available if at least one component of the backup file that is currently loaded in the tabbed page is set to active. It prompts for restoring the application status on the device. The user interface is blocked during restore. You can cancel the operation. |

<table>
<thead>
<tr>
<th>Table 62: “Target Information”</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>“ID”</strong></td>
</tr>
<tr>
<td><strong>“Type”</strong></td>
</tr>
<tr>
<td><strong>“Version”</strong></td>
</tr>
</tbody>
</table>
**Table 63: “Backup Information”**

| “File name” | Storage path of the backup file. Clicking the symbol ( ) opens the file system dialog box. Example: PlcLogic$/Application/Application.crc |
| “Size of active files” | (in kilobytes) Total size of the files set as active in the table (example: 206 KB (210965 bytes)). |
| “Mode” | Defines the scope of the backup: “Application”. The application-related files are added to the archive. |
| “Comment” | Optional entry for comments to be saved in the meta.info file of the backup and reading when the files are restored. |

**Table 64: Table of files for backup**

| “Active” | ☑: Optional files can be deactivated here for exclusion in the backup file. Required components are shown here with a green check mark (no check box). |
| “Component” | Affected components (example: file system) |
| “File” | Name of the component file to back up (example: $PlcLogic$/ Application/Application.app) |
| “Size” | File size in bytes (example: 43280) |
| “Requires STOP” | ☑: For components, the application must be stopped before backup and restore. A dialog prompt will open to warn you of any backup or restore conflicts. |

See also

-  Chapter 1.3.1.13.10 “Backup and restore” on page 353

**Tab 'Synchronized Files’**

This tab of the generic device editor lists the files that are downloaded to the PLC when the application is downloaded. For example, these are external files that were added to an application.

Implicit files, such as the source code archive file, are displayed here only if their time of download is configured for this and the “Show implicit files for application download on the editor of a PLC” option is activated in the CODESYS options (category “Device Editor”).

| “File name” | Name of the file below the application, or direct name of the implicitly transferred file (example: archive.prj). Double-click the file name to open the file. |
| “Host path” | Storage location or original location of the file (example: D:\Proej\Files). Double-click the path to open the directory in the file explorer. |
| “Time interval” | Time interval of the file update on the PLC (example: After download / online change of the application). |
| “Information” | Object-dependent additional information (example: Object: external file). |
| “Origin” | General origin type of the file (example: External file object, Source code download provider). |

See also

-  Chapter 1.3.1.25.3.13.9 “Dialog ‘Options’ - ‘Device Editor’” on page 1081
-  Chapter 1.3.1.25.1.15 “Object ‘External File’” on page 736
Tab 'Files'

In this tab of the generic device editor, you can transfer files between CODESYS (host) and the PLC. If the communication settings are correct and the PLC is online, then CODESYS establishes the connection automatically to the PLC for the duration of the file transfer.

Table 65: "Left Side View, Host"

<table>
<thead>
<tr>
<th>“Location”</th>
<th>Path in the file system of the host; set like in the default file manager. Subdirectories and files are shown in the lower part of the view with name, size, and change date.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Opens the dialog box for creating a new directory in the set path</td>
</tr>
<tr>
<td></td>
<td>Deletes the selected files or directories</td>
</tr>
<tr>
<td></td>
<td>Updates the list of files and directories for the set path (location)</td>
</tr>
<tr>
<td>![&lt;&lt;] ![&gt;&gt;]</td>
<td>Copies the selected files and directories to the other file system immediately. If a file is not already available in the target directory, then it is created. If it is already available and not write-protected, then it is overwritten. Otherwise a message is shown.</td>
</tr>
<tr>
<td></td>
<td>corresponds to the “Write File to Controller” command</td>
</tr>
<tr>
<td></td>
<td>corresponds to the “Write File from Controller” command</td>
</tr>
</tbody>
</table>

By default, the commands “Write File to Controller” and “Write File from Controller” are not included in any menu. You can add it to a menu by using the dialog box from “Tools ➤ Customize” (command category “Online”).

See also
- Chapter 1.3.1.25.1.19.1 “Generic device editor” on page 741
- Chapter 1.3.1.15 “Copying files to/from PLC” on page 357

Tab 'Log'

You can view the PLC log in this tab of the generic device editor. It lists the events that were recorded on the target system. This concerns:
- Events during the startup and shutdown of the system (components loaded, with version)
- Application download and loading of the boot application
- Custom entries
- Log entries from I/O drivers
- Log entries from data sources

The “Log” tab also opens when you click “Open Log Page”. You can configure this as a menu command in the “Customize” dialog.

“Offline recording”
- Default settings
- The PLC also records actions that are not related to the connection with the controller.

“UTC time”
- Standard setting; the time stamp is converted to the local time on the computer as indicated by the time zone of the operating system.
- The time stamp of the runtime system is displayed.
### "Severity"

There are four categories for the severity of the event:

- 🟠: Message
- 🟡: Warning
- 🟥: Error
- 🟢: Debugging

You can show or hide each category with the help of the corresponding button in the bar above the list. Each button shows the number of log entries of the category concerned.

### "Time stamp"

Date and time (example: 12-01-2007 09:48)

### "Description"

Description of the event, for example Import function failed of

### "Component"

Name of the runtime system component concerned, e.g. CmpApp

<table>
<thead>
<tr>
<th>Drop-down list with component names</th>
<th>The log list displays only events that concern the selected component</th>
</tr>
</thead>
<tbody>
<tr>
<td>Logger</td>
<td>Drop-down list with all available recordings. The standard setting is the &lt;Default Logger&gt; specified by the target system, at present identical to 'PlcLog' for the CODESYS runtime system</td>
</tr>
<tr>
<td>Refreshes the log list</td>
<td>Refreshes the log list</td>
</tr>
<tr>
<td>Exports the list contents to an xml file. You can select the file name and storage directory.</td>
<td>Exports the list contents to an xml file. You can select the file name and storage directory.</td>
</tr>
<tr>
<td>Imports a log list from an xml file. The list is then displayed in a separate window.</td>
<td>Imports a log list from an xml file. The list is then displayed in a separate window.</td>
</tr>
<tr>
<td></td>
<td>The displayed log list is emptied, i.e. all entries are deleted.</td>
</tr>
</tbody>
</table>

**Note for error checking:** For exceptions with the description *SOURCEPOSITION*, the affected function opens in the editor by double-clicking it or from the “Display Source Code in Editor” command in the context menu. The cursor jumps to the line that is causing the error. You can also perform this diagnosis when you have the CODESYS project archive, including the download information files and the exported log file. If the affected function is protected, then the following message appears: "The source code is not available for <function name>".

See also

- Chapter 1.3.1.25.1.19.1 “Generic device editor” on page 741
- Chapter 1.3.1.25.1.19 “Object ‘Device’ and generic device editor” on page 740
- Chapter 1.3.1.13.6 “Reading the PLC log” on page 349
- Chapter 1.3.1.25.3.14.1 "Dialog ‘Customize’ - ‘Menu’” on page 1089

### Tab 'PLC Settings'

On this tab of the generic device editor you make the basic settings for the configuration of the PLC, for example the handling of inputs and outputs and the bus cycle task.

| "Application for I/O handling" | Application that is responsible for the I/O handling. |
Table 66: “PLC Settings”

<table>
<thead>
<tr>
<th>“Refresh I/Os in stop”</th>
<th>CODESYS refreshes the values of the input and output channels even if the PLC is in the stop state. If the watchdog detects a malfunction, the outputs are set to the predefined default values.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>CODESYS does not refresh the values of the input and output channels when the PLC is in the stop state.</td>
</tr>
</tbody>
</table>

“Behavior of the outputs at stop”

Handling of the output channels when the controller enters the stop state:

- “Retain values”: The current values are retained.
- “All outputs to default value”: The default values resulting from the I/O mapping are assigned.
- “Execute program”: You can control the handling of the output values via a program contained in the project, which CODESYS executes at “STOP”. Enter the name of the program in the field on the right.

“Always update variables”

Global setting that defines whether or not CODESYS updates the I/O variables in the bus cycle task. This setting is effective for I/O variables of the slaves and modules only if ’deactivated’ is defined in their update settings.

- “Deactivated (update only if used in a task)”: CODESYS updates the I/O variables only if they are used in a task.
- “Activates 1 (use bus cycle task if not used in another task)”: CODESYS updates the I/O variables in the bus cycle task if they are not used in any other task.
- “Activate 2 (always in bus cycle task)”: CODESYS updates all variables in each cycle of the bus cycle task, regardless of whether they are used and whether they are mapped to an input or output channel.

Table 67: “Bus Cycle Options”

| “Bus cycle task” | Task that controls the bus cycle. By default the task defined by the device description is entered.  
By default the bus cycle setting of the superordinate bus device (use cycle settings of the superordinate bus) applies, i.e. the device tree is scanned upwards for the next valid bus cycle task definition.  
Pay strict attention to the following notes! |

NOTICE!
Before you select the “<unspecified>” setting for the bus cycle task, you should be aware that “<unspecified>” means that the default setting given in the device description goes into effects. You should therefore check this description. Use of the task with the shortest cycle time may be defined as the default there, but use of the task with the longest cycle time could equally well be defined!

NOTICE!
For fieldbuses, a fixed cycle matrix is necessary to assure a determined behavior. Therefore, do not use the type ‘free-running’ for a bus cycle task.
### Table 68: “Additional Settings”

<table>
<thead>
<tr>
<th>Setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Force variables for the I/O mapping”</td>
<td>This setting is available only if it is supported by the device. When compiling the application CODESYS creates two global variables for each I/O channel that is mapped to a variable in the dialog “I/O Mapping”. You can use these variables for the forcing of the input or output value on this channel, for example via an HMI visualization.</td>
</tr>
<tr>
<td>“Activate diagnostics for devices”</td>
<td>CODESYS automatically integrates the library CAA Device Diagnosis in the project and creates an implicit function block for each device. If there is already a function block for the device, then either an extended FB is used (for example with EtherCAT) or a further FB instance is added. This then contains a general implementation of the device diagnostics. By means of the FB instances you can determine the status of all devices in the application and evaluate errors. In addition, the library contains functions for the programmatic editing of the device tree. Example: Scanning of all children of a bus system, jumping to the parent element.</td>
</tr>
<tr>
<td>“Display I/O warnings as errors”</td>
<td>Warnings concerning the I/O configuration are displayed as errors.</td>
</tr>
</tbody>
</table>

See also:
- § Chapter 1.3.1.25.1.19.1 “Generic device editor” on page 741
- § Chapter 1.3.1.25.1.19.11 “Tab ‘<device name> I/O Mapping’” on page 755
- § Chapter 1.3.1.25.2.5.1 “Command ‘Build’” on page 916
- PDF document ‘CAA Device Diagnosis’, which is a component of the library.

### Tab ‘PLC Shell’

This tab of the generic device editor includes a text-based control monitor for querying specific information from the controller. You can specify device-dependent commands for this and receive the response from the controller in a result window.

### Table 69: 3S-Smart Software Solutions GmbH standard commands

<table>
<thead>
<tr>
<th>Command with possible parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>?</td>
<td>List of available PLC shell commands with possible parameters and short description.</td>
</tr>
<tr>
<td>getcmdlist</td>
<td>List of names of available PLC shell commands.</td>
</tr>
<tr>
<td>mem &lt;memory address&gt; [&lt;size&gt;]</td>
<td>Shows a hex dump of the defined memory range. The size parameter is optional and describes the number of bytes that should be output. Default value: 16. Example: mem 16x0422139C 8</td>
</tr>
<tr>
<td>reflect</td>
<td>Repeats the given command (for testing the connection)</td>
</tr>
<tr>
<td>applist</td>
<td>Shows a list of all loaded applications that defines the order in the list, beginning with 0 (application index).</td>
</tr>
<tr>
<td>pid [&lt;application name&gt;</td>
<td>&lt;application index&gt;]'</td>
</tr>
<tr>
<td>pinf [&lt;application name&gt;</td>
<td>&lt;application index&gt;]'</td>
</tr>
<tr>
<td>startprg [&lt;application name&gt;</td>
<td>&lt;application index&gt;]'</td>
</tr>
<tr>
<td>Command with possible parameters</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------------------</td>
<td>-------------</td>
</tr>
<tr>
<td>stopprg [&lt;application name&gt;</td>
<td>&lt;application index&gt;]</td>
</tr>
<tr>
<td>resetprg [&lt;application name&gt;</td>
<td>&lt;application index&gt;]</td>
</tr>
<tr>
<td>resetprgcold [application name]</td>
<td>&lt;application index&gt;]</td>
</tr>
<tr>
<td>reload [&lt;application name&gt;</td>
<td>&lt;application index&gt;]</td>
</tr>
<tr>
<td>getprgstat [application name]</td>
<td>&lt;application index&gt;]</td>
</tr>
<tr>
<td>plcload</td>
<td>Shows the processor load of the controller (in percent).</td>
</tr>
<tr>
<td>rtsinfo</td>
<td>Shows information about the runtime system, for example the processor and version of the runtime system.</td>
</tr>
<tr>
<td>channelinfo</td>
<td>Provides information about the communication channel.</td>
</tr>
<tr>
<td>rtc-get</td>
<td>Provides the universal time (UTC) via the DataTime string</td>
</tr>
<tr>
<td>rtc-set</td>
<td>Sets the universal time (UTC) via the DataTime string (see ISO 8601)</td>
</tr>
<tr>
<td>listpcicards [&lt;VendorID&gt;]</td>
<td>Provides a list of PCI adapters (all or by &lt;VendorID&gt;).</td>
</tr>
<tr>
<td>gettaskgroups</td>
<td>Provides a list of all task groups, their tasks, and the CPU core binding.</td>
</tr>
<tr>
<td>cert-getapplist</td>
<td>Shows all registered and used certificates (ID of the component and usage).</td>
</tr>
<tr>
<td>cert-genselfsigned [number for search result by &quot;cert-getapplist&quot;&gt; &lt;expdays=]</td>
<td>Generates self-signed certificates. The validity period of the certificate can be specified by means of expdays=. The default is 365 days.</td>
</tr>
<tr>
<td>cert-gendhparams [length in bits]</td>
<td>Generates the parameters for the Diffie-Hellman key exchange. Caution: This operation can last several minutes.</td>
</tr>
<tr>
<td>cert-getcertlist [trust level]</td>
<td>Lists all certificates of the specified trust level. If a trust level is not given, then all certificates are listed. Possible trust levels</td>
</tr>
<tr>
<td>cert-createcsr [number for search result by &quot;cert-getapplist&quot;&gt;]</td>
<td>Generates CSR files for all applications.</td>
</tr>
<tr>
<td>cert-import &lt;trust level&gt; &lt;filename.cer&gt;</td>
<td>Imports the specified certificate.</td>
</tr>
<tr>
<td>cert-export &lt;trust level&gt; [number for search result by &quot;cert-getcertlist&quot;]</td>
<td>Exports the specified certificate.</td>
</tr>
<tr>
<td>cert-remove &lt;trust level&gt; &lt;number for search result by &quot;cert-getcertlist&quot; or &quot;all&quot;&gt;</td>
<td>Removes the specified certificate</td>
</tr>
</tbody>
</table>
### Command with possible parameters

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>cpuLoad</td>
<td>Shows the processor load of the CPU (for multicore, each processor core)</td>
</tr>
<tr>
<td>getTaskGroups</td>
<td>Shows a list of defined task groups. The assigned tasks are shown for each task group.</td>
</tr>
<tr>
<td>getMulticoreInfo</td>
<td>Shows whether or not multicore is supported and the number of available processor cores.</td>
</tr>
</tbody>
</table>

*: Application name: Name of the application in the device tree; application index: Results from the list of all applications on the controller that you can call with the “applist” command. Index 0 for the application in the list, and 1 for the second, and so on.

See also
- Chapter 1.3.1.13.8 “Using PLC shell for requesting information” on page 351
- Chapter 1.3.1.25.1.19.1 “Generic device editor” on page 741
- Chapter 1.3.1.25.2.4.11 “Command ‘Project information’” on page 902

### Tab `<device name> I/O Mapping`

**Devices with I/O channels**

This tab is displayed in device editors for devices with I/O channels. It shows the available channels and allows for the mapping of input, output, and memory addresses of the controller to variables or entire function blocks of the application. In this way you create the so-called 'I/O mapping'.

The application that is to take care of the I/O handling is defined on the “PLC Settings” tab.

> You can use the ‘online configuration mode’ if the device supports it. In this mode you can access the I/Os of the hardware without having to load a real application to the device beforehand.

> You can also create the I/O mapping in the dialog “Edit IO Mapping”. Here you get a mapping list with search and filter functions for an entire device tree.

**NOTICE!**

**Mapping ‘too large’ data types**

- If a variable of a data type that is larger than a byte is mapped to a byte address, the value of the variable will be truncated to byte size there! For monitoring the variable value in the ‘I/O Mapping’ dialog, this means the following: In the root element of the address, the value is displayed that the variable currently has in the project. The current individual bit values of the byte are displayed in succession in the bit elements below that, but this may not be sufficient for the entire variable value.

Example of the tab “<device name> I/O Mapping” for a CAN bus slave:
The tab contains a table for the editing of the I/O mapping. The information displayed for the inputs and outputs originates from the device description.

| “Find” (1)   | Input field for a search string for the mapping table. The search results are marked in yellow. |
| “Filter” (2) | Drop-down list for filtering I/O mapping displayed listed in the mapping table: |
|             | ● “Show all” |
|             | ● “Show only outputs” |
|             | ● “Show only inputs” |
|             | ● “Show only unmapped variables” |
|             | ● “Show only mapped variables” |
|             | ● “Show only mapping to existing variables” |
|             | ● “Show only mapping to new variables” |
| “Add FB for I/O channel” (11) | Depending on the device, available if the channel entry is selected in the mapping table. Opens the “Select Function Block” dialog for selecting the function block that should be linked directly to the channel. |
| “Go to instance” (12) | Available if the entry is selected in the mapping table. Jumps to the corresponding entry in the “<device name> IEC Objects” tab. |
Depending on the device, the inputs and outputs of the device are displayed as nodes and below them, indented, the associated channels or, depending on the device, only the implicitly created device instance.

The symbol indicates the type of channel:

- ✴️: Input
- ✩: Output

Double-clicking the cell opens an input field.

- Option 1: The variable already exists; specify complete path: <application name>,<module name>,<variable name>; example: appl.plc_prg.ivar; input assistance via [xxx].
- Option 2: The variable does not exist yet; enter a simple name; automatically created internally as a global variable.

Depending on the device, inputs or outputs can be linked directly to a function block. In this case, the button ✴️ "Add FB for I/O Channel" is activated. See above.

<table>
<thead>
<tr>
<th>“Mapping” (3)</th>
<th>Type of mapping:</th>
</tr>
</thead>
<tbody>
<tr>
<td>✴️</td>
<td>Variable already exists</td>
</tr>
<tr>
<td>✩</td>
<td>New variable</td>
</tr>
<tr>
<td>✭</td>
<td>Mapping to function block instance</td>
</tr>
</tbody>
</table>

| “Channel” (4) | Symbolic name of the channel. |

| “Address” (5) | Address of the channel, for example %IW0. |

Address strikethrough: indicates that you should not assign any further variables to this address. Reason: Although the variable specified here – as an already existing variable – is managed at a different memory location, ambiguities could result during the writing of the values, particularly with outputs.

- ✱: Indicates that this address has been edited and fixed. If the arrangement of the device objects in the device tree changes, CODESYS does not adapt this address automatically.

| “Type” (6) | Data type of the channel, for example BOOL. |

Structures or bit fields defined in the device description are displayed only if they are part of the IEC standard and are identified as IEC data types in the device description. Otherwise the table cell remains empty.

When mapping structured variables, the editor prevents you from entering both the structure variable (example: %QB0) and individual structure elements (example: %QB0.1 and QB0.2). Therefore, if there is a main entry with a subtree of bit channel entries in the mapping table, then the following applies: You can input a variable either into the line of the main entry, or into the lines of the sub-elements (bit channels), but not into both.

| “Default value” | Default value of the parameter that applies to the channel: Appears only if the option “Set all outputs to default” is activated in the “PLC Settings” for the behavior of the outputs at stop. |

Note: For compiler version V3.5 SP11 and later, the initialization value of the variables is used automatically as the default value when mapping to an existing variable. You can edit the “Default value” field only if you map to a new created variable or if no mapping is specified. In older versions, users had to specify explicitly that the default value and initialization value were identical.

| “Unit” (7) | Unit for the parameter value, for example ms for milliseconds. |

| “Description” (8) | Brief description of the parameter. |

| “Current value” | Actual value of the parameter applied to the channel; displayed in online mode only. |
"Reset mapping" (9) CODESYS resets the mapping settings to the default values defined by the device description file.

"Update variables" (10) Definition for the device object about updating I/O variables. The default value is defined in the device description:
- "Use parent device setting": Update according to the setting of the superordinate device.
- "Enabled 1 (use bus cycle task if not used in any task)": CODESYS updates the I/O variables in the bus cycle task if they are not used in any other task.
- "Enabled 2 (always in bus cycle task)": CODESYS updates all variables in each cycle of the bus cycle task, regardless of whether they are used and whether they are mapped to an input or output channel.

If a UNION is represented by I/O channels in the mapping dialog, it depends on the device whether mapping to the root element is also possible.

Devices with I/O drivers For devices with I/O drivers, you can set the bus cycle task here in the "I/O Mapping" tab if the general settings should not be used ("PLC Settings" tab).

Table 70: Bus cycle options

| "Bus cycle task" | The drop-down list provides all tasks that are defined in the task configuration of the active application (example: "MainTask"). In case of "Use parent bus cycle setting", the settings of the parent node will be used. |

General information about the bus cycle task As a rule, for each IEC task the used input data is read at the start of each task (1) and the written output data is transferred to the I/O driver at the end of the task (3). The implementation in the I/O driver is decisive for further transfer of the I/O data. It is responsible for the time frame and time point that the actual transfer to the corresponding bus system occurs.

The bus cycle task of the PLC can be defined globally for all fieldbuses in the PLC settings. For some fieldbuses, however, you can change this independent of the global setting. The task with the shortest cycle time is used as the bus cycle task (setting: "unspecified" in the PLC settings). The messages are normally sent on the bus in this task.

Other tasks copy only the I/O data from an internal buffer that is exchanged only with the physical hardware in the bus cycle task.

![Diagram of bus cycle task and data flow](image-url)
(1) Read inputs from input buffer          (2) IEC task
(3) Write outputs to output buffer      (4) Bus cycle
(5) Input buffer                                    (6) Output buffer
(7) Copy data to/from bus                         (8) Bus cycle task, priority 1, 1 ms
(9) Bus cycle task, priority 5                      (10) Bus cycle task, priority 5
(11) Bus cycle task, priority 10, interrupted by task 5

**Task usage**

The "Task Deployment" provides an overview of used I/O channels, the set bus cycle task, and the usage of channels.

```
WARNING!

If an output is written in various tasks, the status is undefined, as this can be overwritten in each case.

If the same inputs are used in various tasks, it is possible for the input to change during the processing of a task. This happens if the task is interrupted by a task with a higher priority and causes the process image to be read again.

Solution: At the beginning of the IEC task, copy the input variables to variables and then work only with the local variables in the rest of the code.

Conclusion: Using the same inputs and outputs in several tasks does not make any sense and can lead to unexpected reactions in some cases.
```

See also

- Chapter 1.3.1.8.3 “Configuring devices and I/O mapping” on page 162
- Chapter 1.3.1.25.1.19.1 “Generic device editor” on page 741
- Chapter 1.3.1.25.2.4.31 “Command ‘Edit I/O Mapping’” on page 914
- Chapter 1.3.1.25.1.19.12 “Tab <device name> IEC Objects” on page 759
- Chapter 1.3.1.25.3.3 “Dialog ‘Select Function Block’” on page 1040
- Chapter 1.3.1.25.2.4.35 “Command ‘Online Config Mode’” on page 915
- Chapter 1.3.1.25.1.19.9 “Tab ‘PLC Settings’” on page 751

**Tab <device name> IEC Objects'**

```
NOTICE!

Please note that manually creating another instance of the device object can lead to malfunctions.
```

In this tab of the generic device editor, "objects" are listed that allow for access to the device from the IEC application. In online mode, this is used as the monitoring view.

For devices for which a function block instance is created implicitly, at least this instance is listed as an object here in the table. This instance can be used, for example, in order to restart a bus or to query information from the application. The device type determines whether this kind of device instance is available and which access options it has. Please refer to the help for the special device configuration.

Instances of function blocks that are linked with inputs or outputs of the device are also displayed here. The mapping of a function block to a channel is defined in the “<device name> I/O Mapping” tab. The "Go to Instance" command takes you directly to the affected object from there.

In addition, you can create more objects in the table here that are not yet linked with a device channel.
In online mode, you can use the table of IEC objects as a monitoring view. It also shows the current value, the address, and the comment for the function block variable at the channel. Finally, it provides the capability of writing and forcing values.

| **“Add”** | Opens the “Select Function Block” dialog for creating a new instance or for editing the instance selected in the table. |
| **“Edit”** | Deletes the selected entry. |
| **“Delete”** | Jumps from the selected entry directly to the corresponding mapping in the “<device name> I/O Mapping” tab. |
| **“Variable”** | The object name comprises the device name and the function block name. Example: EL2004 Relay. Changing the device name has an immediate effect. The part of the name after the device name is editable here. |
| **“Mapping”** | Mapping type, as in the “<device name> I/O Mapping” tab |
| **“Type”** | Data type: Here it is the name of the function block. |
| **“Value”** | In online mode only: Display of the current value, the address, and the comment for the variable at the channel. Moreover, the option of specifying a value for writing or forcing the variable. |
| **“Prepared value”** |
| **“Address”** |
| **“Comment”** |

See also

- ☢ Chapter 1.3.1.25.1.19.11 “Tab '<device name> I/O Mapping’” on page 755

**Tab 'Users and Groups’**

**NOTICE!**

**Recommendations for Data Security**

In order to minimize the risk of data security violations, we recommend the following organizational and technical actions for the system where your applications are running. As far as possible, avoid exposing the PLC and control networks to open networks and the internet. Use additional data link layers for protection, such as a VPN for teleaccess and install firewall mechanisms. Limit access to authorized persons, change any existing standard passwords during the initial commissioning and continue to change them regularly.

On this tab of the generic device editor you edit the device user management of the controller. Depending on how it is supported by the device, you can define user accounts and user groups. In combination with the configuration on the “Access Rights” tab, you thus control access to control objects and files at runtime.

Requirements: The controller has a user management and allows it to be edited. You have login data in order to be able to log in to the controller.

**It is possible to apply user account definitions from the project user management into the device user management (see below: “Import” button).**
Table 71: Toolbar of the tab

<table>
<thead>
<tr>
<th>Button</th>
<th>Description</th>
</tr>
</thead>
</table>
| Synchronization | Switches on and off the synchronization between the editor and the user management on the device. If the button is not pressed, then the editor is blank or it contains a configuration that you loaded from the hard disk. If the button is pressed, then CODESYS synchronizes the display in the editor continuously with the current user management on the connected device. If you activate the synchronization while the editor contains a user configuration that is not synchronized with the device yet, then you are prompted what should happen to the editor contents. Options:  
  - “Upload from the device and overwrite the editor content”: The configuration on the device is loaded into the editor, overwriting the current contents.  
  - “Download the editor content to the device and overwrite the user management there”: The configuration in the editor is transferred to the device and applied there. |
| Import from disk | Opens the default dialog for selecting and importing a user management configuration from the hard disk. When you click the button in the “Users and Groups” tab, the file type is “Device user management files (*.dum)”. When you click the button in the “Access Rights” tab, the file type is “Device rights management files (*.dum)”. |
| Export to disk | Opens the default dialog for saving a file to the hard disk. This saves the user management configuration as an XML file. The data type is “Device rights management files (*.drm)”. |

“Device user” User name of the user currently logged in on the device

Table 72: “Users”

All currently defined users, and below them their memberships of user groups, are listed in a tree structure.

<table>
<thead>
<tr>
<th>Button</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Add”</td>
<td>Opens the “Add User” dialog for creating a new user account.</td>
</tr>
<tr>
<td>“Import”</td>
<td>Opens the dialog “Import User”. It displays all the user accounts defined in the project user management. Select the desired entries and click “OK” in order to import them into the device user management. CAUTION: The passwords are NOT applied.</td>
</tr>
<tr>
<td>“Edit”</td>
<td>Opens the dialog “Edit User &lt;user name&gt;”. It corresponds to the “Add User” dialog and you can change the settings of the user account.</td>
</tr>
<tr>
<td>“Delete”</td>
<td>Deletes the account of the currently selected user.</td>
</tr>
</tbody>
</table>

Table 73: “Groups”

All currently defined groups, and below them the users assigned to them, are listed in a tree structure.

<table>
<thead>
<tr>
<th>Button</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Add”</td>
<td>Opens the dialog “Add Group”. Define a new group name. From the list of defined users, select those that are to belong to the group. Confirm the selection with “OK”. The group appears in the tree.</td>
</tr>
<tr>
<td>“Import”</td>
<td>Opens the dialog “Import User”. It displays all the user groups defined in the project user management. Select the desired entries and click “OK” in order to import them into the device user management.</td>
</tr>
</tbody>
</table>
**Table 74: “Add Dialog ‘Add user’”**

<table>
<thead>
<tr>
<th><strong>“Name”</strong></th>
<th>Name of the new user</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>“Default group”</strong></td>
<td>Drop-down list with all configured user groups. Every user must belong to at least one group. You define this here as a default group.</td>
</tr>
<tr>
<td><strong>“Password”</strong></td>
<td></td>
</tr>
<tr>
<td><strong>“Confirm password”</strong></td>
<td></td>
</tr>
<tr>
<td><strong>“Password strength”</strong></td>
<td>Password security in stages from “Very weak” to “Very good”.</td>
</tr>
<tr>
<td><strong>“Hide password”</strong></td>
<td>☑: The password is shown only with asterisks “*” when it is typed in.</td>
</tr>
<tr>
<td><strong>“Password can be changed by the user”</strong></td>
<td></td>
</tr>
<tr>
<td><strong>“Password must be changed at first login”</strong></td>
<td></td>
</tr>
</tbody>
</table>

See also
- Chapter 1.3.1.25.1.19 “Object ‘Device’ and generic device editor” on page 740
- Chapter 1.3.1.25.1.19.1 “Generic device editor” on page 741
- Chapter 1.3.1.11.3 “Handling of device user management” on page 300
- Chapter 1.3.1.25.1.19.14 “Tab ‘Access Rights’” on page 762
- Chapter 1.3.1.25.2.6.14 “Command ‘Add Device User’” on page 934

**Tab ‘Access Rights’**

**NOTICE!**

**Recommendations for data security**

In order to minimize the risk of data security violations, we recommend the following organizational and technical actions for the system where your applications are running. Whenever possible, avoid exposing the PLC and control networks to open networks and the Internet. Use additional data link layers for protection, such as a VPN for teleaccess, and install firewall mechanisms. Restrict access for authorized persons only, and change any existing standard passwords during the initial commissioning. In addition, change them regularly.

**NOTICE!**

Detailed information on the concept and use of device user management is provided in "Handling of Device User Management".

There you will also find the following instructions on how to use the editor:
- Initial login to the controller for editing and viewing its user management
- Setting up a new user in the user management of the controller
- Changing access rights to controller objects in the user management of the controller
- Loading user management from a *.dum file, modifying it, and downloading it to the controller in offline mode
On this tab of the device editor, you define the device access rights of device users to objects on the controller. As in the project user management, users must be members of at least one user group and only user groups can be granted certain access rights.

Requirements for the “Access Rights” tab to be displayed:
- In the CODESYS options, in the “Device editor” category, the “Show access rights page” option must be selected.
  Note that this CODESYS option can be overwritten by the device description.

Requirements for the access rights to be granted to user groups
- A component for the user management has to be available on the controller. That is the primary requirement.
- Users and user groups have to be configured on the “Users and Groups” tab.

Table 75: Toolbar of the tab

<table>
<thead>
<tr>
<th>Icon</th>
<th>Description</th>
</tr>
</thead>
</table>
| ![Synchronization](image) | Switches on and off the synchronization between the editor and the user management on the device. If the button is not pressed, then the editor is blank or it contains a configuration that you loaded from the hard disk. If the button is pressed, then CODESYS synchronizes the display in the editor continuously with the current user management on the connected device. If you activate the synchronization while the editor contains a user configuration that is not synchronized with the device yet, then you are prompted what should happen to the editor contents. Options:  
  - “Upload from the device and overwrite the editor content”: The configuration on the device is loaded into the editor, overwriting the current contents.  
  - “Download the editor content to the device and overwrite the user management there”: The configuration in the editor is transferred to the device and applied there. |
| ![Import from disk](image) | Opens the default dialog for selecting and importing a user management configuration from the hard disk. When you click the button in the “Users and Groups” tab, the file type is “Device user management files (*.dum)”. When you click the button in the “Access Rights” tab, the file type is “Device rights management files (*.drm)”. |
| ![Export to disk](image) | Opens the default dialog for saving a file to the hard disk. This saves the user management configuration as an XML file. The data type is “Device rights management files (*.drm)”. |
| ![Device user](image) | User name of the user currently logged in on the device |

Table 76: “Objects”

In the tree structure, the objects are listed to which actions can be executed at runtime. The objects are each assigned by their object source and partially sorted in object groups. In the “Rights” view, you can configure the access options for a user group to a selected object.

Object source (root node)
- “File system objects ➔ Device”: In these objects, the rights can be granted to folders of the current execution directory of the controller.  
- “Runtime objects ➔ /”: In these objects, all objects are managed that have online access in the controller and therefore have to control the access rights.

A description of the objects is located in the table. ➔ “Overview of the objects” on page 765

Object groups and objects (indented)

---

2020/12/10

3ADR010583, 1, en_US

763
Table 77: “Rights”

In general, the access rights are inherited from the root object (also “Device” or “/”) to the subobjects. This means that if a permission of a user group is denied or explicitly granted to a parent object, then this first affects all child objects.

The table applies for the object that is currently selected in the tree. For every user group, it shows the rights currently configured for the possible actions on this object.

<table>
<thead>
<tr>
<th>Possible actions on the object:</th>
</tr>
</thead>
<tbody>
<tr>
<td>● “Add/Remove”</td>
</tr>
<tr>
<td>● “Modify”</td>
</tr>
<tr>
<td>● “View”</td>
</tr>
<tr>
<td>● “Execute”</td>
</tr>
</tbody>
</table>

When an object is clicked, a table on the right side shows the access rights of the available user groups for the selected object.

This allows you to quickly see:

● Which access rights are evaluated by an object
● Which user group has which effective rights to which object

Meanings of the symbols

● +: Access right granted explicitly
● -: Access right denied explicitly
● 🔴: Access right granted through inheritance
● 🔴: Access right denied through inheritance
● 🕰️: The access right was not granted or denied explicitly and also not inherited by the parent object. Access is not possible.
● No symbol: Multiple objects are selected that have different access rights.

Change the permission by clicking the symbol.
The “Logger” object on the “Access Rights” tab was created by the "Logger" component and controls its access rights. It is located directly below the "Device" runtime object.

The possible access rights for this object can be granted only for the “View” action.

Initially, each object has a read access. This means that every user can read the "Logger" of a controller. If this access right should be denied for a single user group ("Service" in the example), then the read access to the logger object has to be denied explicitly.

Overview of the objects

| “Runtime objects” ➔ “Device” |  |
| "Logger" | Online access to the logger is read only. Therefore, only the “View” access right can be granted or denied here. |
| "PlcLogic" | All IEC applications are inserted here automatically as child objects during download. When an application is deleted, it is removed automatically. This allows specific control of online access to the application. Access rights can be assigned centrally over all applications in the "PlcLogic". The "Administrator" and "Developer" user groups have full access to the IEC applications. The "Service" and "Watch" user groups only have read access (for example for read-only monitoring of values). |

The following table shows which action is affected in particular when a specific access right is granted for an IEC application.

- The right has to be set explicitly.
-: The right is not relevant.

<table>
<thead>
<tr>
<th>“Application”</th>
<th>Operation</th>
<th>Access rights</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>&quot;Add/ Remove&quot;</td>
<td>&quot;Execute&quot;</td>
</tr>
<tr>
<td>Login</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Create</td>
<td>x</td>
<td>-</td>
</tr>
<tr>
<td>Create child object</td>
<td>x</td>
<td>-</td>
</tr>
<tr>
<td>Delete</td>
<td>x</td>
<td>-</td>
</tr>
<tr>
<td>Download / online change</td>
<td>x</td>
<td>-</td>
</tr>
<tr>
<td>Create boot application</td>
<td>x</td>
<td>-</td>
</tr>
<tr>
<td>Operation</td>
<td>Read</td>
<td>Write</td>
</tr>
<tr>
<td>-----------------------------------------------</td>
<td>------</td>
<td>-------</td>
</tr>
<tr>
<td>Read variable</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Write variable</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Force variable</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Set and delete breakpoint</td>
<td>-</td>
<td>x</td>
</tr>
<tr>
<td>Set next statement</td>
<td>-</td>
<td>x</td>
</tr>
<tr>
<td>Read call stack</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Single cycle</td>
<td>-</td>
<td>x</td>
</tr>
<tr>
<td>Switch on flow control</td>
<td>-</td>
<td>x</td>
</tr>
<tr>
<td>Start / Stop</td>
<td>-</td>
<td>x</td>
</tr>
<tr>
<td>Reset</td>
<td>-</td>
<td>x</td>
</tr>
<tr>
<td>Restore retain variables</td>
<td>-</td>
<td>x</td>
</tr>
<tr>
<td>Save retain variables</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

**“PlcShell”**

Only the “Modify” permission is evaluated at this time. This means that only when the “Modify” permission has been granted to a user group can PLC shell commands also be evaluated.

**“RemoteConnections”**

Additional external connections to the controller can be configured below this node. Currently, access to the CODESYS OPC UA server can be configured here.

**“Settings”**

This is the online access to the configuration settings of a controller. By default, access to “Modify” is granted only to the administrator.

**“UserManagement”**

This is the online access to the user management of a controller. By default, read/write access is granted only to the administrator.

**“X509”**

This controls the online access to the X.509 certificates. Two types of access are distinguished here:

- Read (“View”)
- Write (“Modify”)

Every operation is assigned to one of these two access rights. Each operation is inserted as a child object below X509. Therefore, access per operation can now be fine-tuned even more.

**“File system objects ➔ /”**

All folders from the execution path of the controller are inserted below the “/” file system object. This allows you to grant specific rights to each folder of the file system.

See also

-  § Chapter 1.3.1.11.3 “Handling of device user management” on page 300
-  § Chapter 1.3.1.25.1.19.1 “Generic device editor” on page 741
-  § Chapter 1.3.1.25.1.19.13 “Tab ‘Users and Groups’” on page 760

**Tab ’Symbol Rights’**

In this tab of the generic device editor, you define the access rights of different user groups (clients) to the individual symbol sets available on the controller.
Requirement: User management must be set up on the PLC. An application was downloaded to the controller for which symbol sets were defined in the CODESYS project. They have access data for logging in to the controller.

In the “Symbol Sets” view, all symbol sets are listed below the “Application” node whose definition was downloaded with the application to the controller.

In the “Rights” view, the user groups defined in the user management of the controller are listed in a table. When a symbol set is selected, you see the access rights of the corresponding user group to the symbols of this set. Access granted; Access not granted. You can change the access rights by double-clicking the symbol.

Click the button to save the current access configuration to an XML file. The file type is “Device symbol management files (*.dsm)”. Click the button to read a file like this from the hard drive.

See also
● “Creating symbol sets with different access rights for different control clients” on page 276
● Chapter 1.3.1.25.1.19.1 “Generic device editor” on page 741
● Chapter 1.3.1.25.1.19.14 “Tab 'Access Rights’” on page 762
● Chapter 1.3.1.25.1.19.13 “Tab 'Users and Groups’” on page 760

Tab 'Task deployment’

This sub-dialog box of the device editor displays a table of inputs and outputs as well as their assignment to the defined tasks.

The information only becomes visible after code has been generated for the application. It is used for troubleshooting, because it shows where inputs or outputs are used in several tasks with different priorities. Multiple use can lead to undefined values through overwriting.
Table 78: “I/O deployment for tasks” (1)

<table>
<thead>
<tr>
<th>“I/O Channels” (2)</th>
<th>All inputs and outputs of the linked devices. The display corresponds to that in the dialog box “I/O Mapping” of the device editor. By double-clicking on an input or output you can open the associated I/O mapping editor.</th>
</tr>
</thead>
<tbody>
<tr>
<td>“&lt;task name&gt;” (3)</td>
<td>A column appears for every task defined in the task configuration. The title contains the task name and priority. The priority of the tasks decreases from first to the last column. A red cross appears in the box for inputs and outputs that are written or read by a task: 🔴. In addition, the task defined as a “Bus cycle task” in the “PLC Settings” of the device editor is marked at these points with a blue double arrow symbol 🔄. Following a mouse-click on the title cell, only the I/Os assigned to this task are displayed. Following a mouse-click on the “I/O channels” cell, all channels are shown again.</td>
</tr>
</tbody>
</table>

See also
- § Chapter 1.3.1.25.1.19.1 “Generic device editor” on page 741
- § Chapter 1.3.1.25.1.19.11 “Tab <device name> I/O Mapping” on page 755
- § Chapter 1.3.1.25.1.19.9 “Tab ‘PLC Settings’” on page 751
- § Chapter 1.3.1.9.17.1 “Creating a task configuration” on page 240

Tab 'Status'

This tab of the generic device editor displays status information, for example "Running" or "Stopped", and specific diagnostic messages from the respective device, also information about the card used and the internal bus system.

See also
- § Chapter 1.3.1.25.1.19.1 “Generic device editor” on page 741

Tab 'Information'

This tab of the generic device editor displays general information that originates from the device description file: name, vendor, categories, version, order number, description, if necessary an illustration.

See also
- § Chapter 1.3.1.25.1.19.1 “Generic device editor” on page 741

Object 'GlobalTextList'

Symbol: 📜

This object is for the management and translation of texts that are written as static text in visualizations in the project. It contains a table with these texts. If you write a text in a visualization in an element under the property “Texts”, CODESYS automatically adds a line in the table. You cannot write any new text here, you can only edit an existing text.

In addition CODESYS makes the following commands available, in order to consolidate the “GlobalTextList”:
- “Check Visualization Text IDs”
- “Update Visualization Text IDs”
- “Remove Unused Text List Entries”

The object is located in the POUs view and exists once at the most
Object 'GVL' - Global Variable List

Symbol: ☰

A global variable list is used for the declaration, editing and display of global variables.

A GVL is added to the application or the project with the command "Project ➔ Add Object ➔ Global Variable List".

If you insert a GVL under an application in the Device tree, the variables are valid within this application. If you add a GVL in the POU's view, the variables are valid for the entire project.

You can apply settings for the editor of the object in the dialog "Tools ➔ Options" in the categories "Declaration Editor" and "Text Editor".

If the target system supports network functionality, you can convert the variables of a GVL into network variables and thus use them for data exchange with other devices in the network. To do this you must define corresponding properties for the GVL in the "Network Variables" tab of the "Properties" dialog.

See also
- ☰ Chapter 1.3.1.9 “Programming of applications” on page 171
- ☰ Chapter 1.3.1.25.3.10.10 “Dialog 'Properties' - 'Network Variables'” on page 1052
- ☰ Chapter 1.3.1.25.3.13 “Dialog 'Options'” on page 1073

Object 'GVL' - Global Variable List (task-local)

Symbol: ☰

A global variable list (task-local) is used for the declaration, editing and display of global variables. For this special global variable list, the declared variables in the list can be written by one task only. All other tasks have only read-only access. This makes sure that the values of these variables are always consistent, even for multicore projects.
The object is available for compiler version 3.5.13.0 with the corresponding device description.

| Task with write access | Task that has exclusive write access to the variables. |

See also
- % Chapter 1.3.1.9.2.5 “Using task-local global variable lists” on page 177
- % Chapter 1.3.1.9.2.4 “Declaring global variables” on page 176
- % Chapter 1.3.1.9.17.2 “Multicore” on page 241
- % Chapter 1.3.1.25.1.21 “Object ‘GVL’ - Global Variable List” on page 769
- % Chapter 1.3.1.25.1.37.5 “Tab ‘Task Groups’” on page 838

Object 'Persistent variable list'

Symbol: ¶

The object contains the declaration of global persistent variables in the declaration section
VAR_GLOBAL PERSISTENT RETAIN .. END_VAR. The variables are stored in special non-volatile memory.

The persistence editor shows the variables as a list in the usual way. The displayed list does not influence the persistence behavior of the variables, but only the list stored internally in the process image. The list there contains all variables ever declared in chronological order. Variables that you have removed are marked with a placeholder and continue to exist as a gap.

The declaration section can also contain instance paths, which refer to locally declared persistent variables and were created with the command “Declarations ➤ Add All Instance Paths”.

NOTICE!
Before you decide how to set up persistence for an application, it would be helpful for you to be familiar with the use cases described in the “Data Persistence” section. Moreover, it is helpful if you can differentiate between the mechanisms of persistent variables, retain variables, variables of the Persistence Manager, and recipe variables.

Commands

The following commands are provided in the persistence editor:
- Command “Declarations ➤ Add All Instance Paths”
- Command “Declarations ➤ Reorder List and Clear Gaps”

See also
- TODO
- % Chapter 1.3.1.25.2.17.1 “Command ‘Reorder List and Clean Gaps’” on page 1011
- % Chapter 1.3.1.9.20.1 “Preserving data with persistent variables” on page 252
- % Chapter 1.3.1.9.20.2 “Preserving data with retain variables” on page 254
- % Chapter 1.3.1.25.2.17.4 “Command ‘Add all instance paths’” on page 1012

Object 'Image Pool'

The “Image Pool” object contains a table with image ID assignments.
“ID” | ID of the image; you reference this ID, for example in the visualization of the image.
---|---
“File name” | File path of the image; if you click for more settings ( ), the “Select Image” dialog box opens.
“Image” | Show a thumbnail of the image.
“Link type” | Opens the “Select Image” dialog box, where you define the link type.

**Dialog box 'Select Image'**

| “Image file” | Name and directory of the image file (example: "C:\Programme\images \logo.bmp") CODESYS supports the following image formats: BMP, EMF, GIF, ICO, JPG, PNG, SVG, and TIFF. Please note that a controller may not support all formats. Whether or not you can use images formatted as scalable vector graphics (*.svg) depends on the operating system. Any necessary information is located in the device description of the hardware vendor. |

**Table 79: “File Handling”**

| “Remember the link” | CODESYS saves only the link. CODESYS automatically updates any changes to an image file in the image pool. You must ensure that the path of the image file does not change. When saving the project as an archive, CODESYS embeds the image file in the project archive. |
| “Remember the link and embed into project” | CODESYS copies the image to the image pool and the link information is retained. In this way, CODESYS recognizes any changes to the image file and then update the image pool as needed. This behavior is controlled with the options in the next table. Embedded image files increase the memory requirement of the project. |
| “Embed into project” | CODESYS copies the image to the image pool. If the image file is changed again afterwards, then it is not updated in the project. For libraries, you must embed the image in the project. Embedded image files increase the memory requirement of the project. |

**Table 80: “Change Tracking”**

These options are available only if you have selected the “Remember the link and embed into project” check box as described above.

| “Reload the file automatically” | CODESYS automatically updates the image file in the project without prompting. |
| “Prompt whether to reload the file” | If the image file has changed, you may be prompted whether or not the image file should be updated. |
| “Do nothing” | CODESYS does not update the image file in the image pool. |

See also

- Chapter 1.3.1.9.10 “Using image pools” on page 221

**Object 'Library Manager’**

Symbol: 🎨

The Library Manager lists all libraries that were integrated in the project for creating applications. It provides information about the type of library, its properties, and its contents.
You can expand or collapse the list of integrated libraries, as well as edit library properties for non-dependent libraries.

The Library Manager consists of three views:

● Upper view: List of integrated libraries
● Lower left view: Tree structure with all modules of the library selected in the upper view
● Lower right view: Documentation for the module selected in the tree

See also

● Chapter 1.3.1.19 “Using libraries” on page 371

<table>
<thead>
<tr>
<th>Displayed in gray fonts</th>
<th>The library was added to the project automatically by means of a plug-in.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Displayed in black fonts</td>
<td>The library was added to the project automatically by means of the “Add Library” command.</td>
</tr>
<tr>
<td>“Name”</td>
<td>Display of the integrated library in the following syntax:</td>
</tr>
<tr>
<td></td>
<td>“&lt;placeholder name&gt; = &lt;library name&gt;, &lt;version&gt; &lt;company&gt;” :</td>
</tr>
<tr>
<td></td>
<td>“&lt;placeholder name&gt;”: If it is a placeholder library for a library, then the placeholder name is before a “ = ”.</td>
</tr>
<tr>
<td></td>
<td>“&lt;library name&gt;”: Name of the library that is used for management in the library repository.</td>
</tr>
<tr>
<td></td>
<td>“&lt;version&gt;”: Version that was referenced at the first time it was integrated.</td>
</tr>
<tr>
<td></td>
<td>“&lt;company&gt;”: Vendor (optional)</td>
</tr>
<tr>
<td>“Namespace”</td>
<td>Namespace for unique access to the contents of the library.</td>
</tr>
<tr>
<td></td>
<td>It is prepended to a module identifier for this purpose:</td>
</tr>
<tr>
<td></td>
<td>&lt;namespace&gt;.&lt;library module identifier&gt;</td>
</tr>
<tr>
<td></td>
<td>The namespace usually coincides with the library name.</td>
</tr>
<tr>
<td></td>
<td>Note: If the library has the property LanguageModelAttribute</td>
</tr>
<tr>
<td></td>
<td>&quot;qualified-access-only&quot;, then you must access the library module in the</td>
</tr>
<tr>
<td></td>
<td>application code by means of the namespace. Qualified (unique) access is</td>
</tr>
<tr>
<td></td>
<td>enforced.</td>
</tr>
<tr>
<td></td>
<td>You can modify the standard namespace for local use (within the project) in the “Properties” dialog.</td>
</tr>
<tr>
<td>“Effective version”</td>
<td>Version of the library after the resolution. This version is used in the project.</td>
</tr>
<tr>
<td></td>
<td>Requirement: The Library Manager exists in the “Devices” view and a placeholder library is selected.</td>
</tr>
<tr>
<td></td>
<td>Example: 3.5.10.0</td>
</tr>
<tr>
<td></td>
<td>A placeholder library that is integrated below an application is resolved by assigning a special resolution to the placeholder library in the “Placeholders” dialog. Then the selected library is loaded. Other resolutions are ignored. If no special resolution is given, then a check is performed as to whether or not a resolution is specified in the device description and library profile of the application. The first search hit is applied.</td>
</tr>
</tbody>
</table>
Symbol with tooltip to notify about the current device-dependent resolution of the selected library.

Example when the Library Manager is in the “Devices” view: “This placeholder is explicitly redirected to this version (see the Placeholders dialog)”

Example when the Library Manager is in the “POUs” view: “In the 'Device_1' device, the placeholder is resolved to 'VisuElemAlarms, 1.0.0.0 (System)’”

A placeholder library that is integrated in the “POUs” view is resolved by checking depending on the application whether or not a resolution is specified in the device description. Afterwards, the library profile is checked. The first detected resolution is used. If you have assigned a special resolution to the placeholder library in the “Placeholders” dialog, then this will **always be ignored**. The result is shown in the tooltip of the symbol.

Library that is signed with a trusted certificate (compatible with CODESYS >= V3 SP15)

Library that is signed with a trusted certificate, but references at least one unsigned library

Library that is signed with a private key and token (compatible with CODESYS < V3 SP15)

Library that is not signed, or signed with an untrusted or expired certificate. In the case of an untrusted certificate, the “Trust Certificate” command is provided in the context menu.

Library that is defined as optional and not currently available

Library whose status is being determined

Licensed library for which no valid license is currently available

Library symbol for a library that cannot be loaded because its signature (encryption) could not be verified

---

**Table 81: Commands in the Library Manager**

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Add Library”</td>
<td>Opens the dialog for selecting a library. All libraries installed in the library repository are offered.</td>
</tr>
<tr>
<td>“Delete Library”</td>
<td>Removes the presently selected library from the project</td>
</tr>
<tr>
<td>“Properties”</td>
<td>Opens the dialog for the display and editing of the properties of the presently selected library</td>
</tr>
<tr>
<td>“Details”</td>
<td>Opens a dialog with details for the presently selected library (general information, contents, properties, license information)</td>
</tr>
<tr>
<td>“Try to Reload Library”</td>
<td>If you select a library marked as not found, you can attempt to load it into the project again using this command.</td>
</tr>
<tr>
<td>“Download Missing Libraries”</td>
<td>CODESYS scans for the missing libraries in the download servers specified in the project options. After that you can download and install the library.</td>
</tr>
<tr>
<td>“Placeholders”</td>
<td>The “Placeholders” dialog opens. The current resolution is displayed and you can edit them here.</td>
</tr>
<tr>
<td>“Library Repository”</td>
<td>Opens the “Library Repository” dialog for installing and uninstalling libraries and for defining library locations</td>
</tr>
<tr>
<td>“Icon legend”</td>
<td>Opens the “Information” dialog with a legend of the icons that display the current status of a library in the list of integrated libraries (see above)</td>
</tr>
</tbody>
</table>
“Trust Certificate” 
Only in the context menu of a library selected in the Library Manager, in which the library has been signed with an untrusted certificate. The command turns the untrusted certificate into a trusted certificate and the prepended icon changes from 🚫 to 🌟.

“Export Library” 
Only in the context menu of a library selected in the Library Manager: Opens the default dialog for saving the library file in the file system.

See also
- Chapter 1.3.1.25.2.14.1 “Command ‘Add Library’” on page 1004
- Chapter 1.3.1.25.2.14.5 “Command ‘Export Library’” on page 1008
- Chapter 1.3.1.25.2.14.3 “Command ‘Properties’” on page 1006
- Chapter 1.3.1.25.2.8.2 “Command ‘Library Repository’” on page 951
- Chapter 1.3.1.25.2.14.4 “Command ‘Placeholders’” on page 1008
- Chapter 1.3.1.25.3.13.3 “Dialog ‘Options’ – ‘Library Download’” on page 1075
- Chapter 1.3.1.25.1.31 “Object ‘Project Information’” on page 811
- “Deprecated: Signing a library project (only for version compatibility with CODESYS < SP15)” on page 133

Tree structure of all modules of a selected library

All library modules that were integrated with the library are listed in the tree structure. Requirement: A library is selected in the upper view.

The usual sorting and search functions are available in the menu bar.

Documentation for the library module selected in the lower left view

Tab “Inputs/Outputs” Interface (inputs/outputs) of the library module
Tab “Graphical” Graphical display of the module
Tab “Documentation” Documentation for the library module.
Note: As a library developer, you must follow the rules for documentation inclusion in 'Guidelines for library development'.
Tab “Parameter List” Requirement: The library project contains a parameter list.
You can change the values of these parameters in the column “Value (editable)”.

See also
- Chapter 1.3.1.19.1 “Information for library developers” on page 372

Object ‘Network Variable List (Sender)’

Symbol: 🌐

A network variable list (sender) is used for declaring and listing global variables that should be sent to network variable lists (receiver) of other devices or network projects.

You add the object to the device tree by clicking “Add Object ➤ Network Variable List (Sender)” of an application.

You can configure the protocol and transfer parameters in the “Add Network Variable List (Sender)” dialog box or “Properties” dialog box of the object in the “Network Variables” tab.
**Dialog Box 'Add Network Variable List (Sender)'**

**Function:** This dialog box defines the network properties for the sender NVL. When you close the dialog box, CODESYS adds the sender NVL of the application to the device tree.

**Call:** Main menu “Project ➔ Add Object ➔ Network Variable List (Sender)” while the application is selected in the device tree.

This dialog box corresponds to the “Network Variables” tab in the “Properties” of the network variable list object.

See also
- Chapter 1.3.1.25.3.10.10 “Dialog 'Properties' - ‘Network Variables’” on page 1052
- Chapter 1.3.1.25.1.27 “Object 'Network Variable List (Receiver)’” on page 775
- Chapter 1.3.1.10.3.1 “Configuring a network variable exchange” on page 278

**Object 'Network Variable List (Receiver)'**

**Symbol:** 📚

The object is used for listing the received network variables and displaying the information: network and transmit information and sender.

You add the object to an application by clicking “Add Object ➔ Network Variable List (Receiver)’”.

The network variable list (receiver) shows the received network variables, which were declared in network variable list (sender) of another device or project. You cannot change the network variables in the object editor.

The object editor consists of two parts:
- Information about the sender and transfer log of the list
- List of declarations of network variables

**Dialog Box 'Add Network Variable List (Receiver)’**

**Function:** This dialog box defines the receiver NVL to a sender NVL and adds the receiver NVL to the application object in the device tree.

**Call:** Main menu “Project ➔ Add Object ➔ Network Variable List (Receiver)” (when the application object is selected).

| “Task” | Task of the current application that controls the variables to be received. |
| “Sender” | Drop-down list |
| | • Available sender NVLs of another device in the project |
| | • “Import from file”: Required if the necessary sender NVL is defined in another project. For this, the necessary sender NVL must have been generated in another project as “GVL export file *.gvl” in the properties dialog of the NVL in the ”Link To File” tab. |
| “Import from file” | File name in “GVL export file *.gvl” format if you have selected “Import from file” for “Sender”. |

See also
- Chapter 1.3.1.10.3.1 “Configuring a network variable exchange” on page 278
- Chapter 1.3.1.25.1.26 “Object 'Network Variable List (Sender)’” on page 774

**Object 'POU’**

**Symbol:** 📚
An object of the type “POU” is a Program Organization Unit in a CODESYS project. You write source code for your controller program in POUs.

There are the following types of POUs:
- Program
- Function
- Function block

A “POU” object is inserted by using the command “Project ➔ Add Object” in the Device tree or in the “POUs” view. When adding a POU you define the POU type and the implementation language.

You can also add other programming objects (method, action, etc.) to these objects.

**Calling POUs**

Certain POUs can call other POUs. Recursions are not permitted.

When calling POUs via the namespace, CODESYS browses the project for the POU to be called in accordance with the following order:

1. Current application
2. “Library Manager” of the current application
3. “POUs” view
4. “Library Manager” in the “POUs” view

The term “POU” is also used in CODESYS for the “POUs” view in which CODESYS manages the global objects in the project.

See also
- Chapter 1.3.1.25.1.28 “Object ‘POU’” on page 775
- Chapter 1.3.1.25.3.10 “Dialog ‘Properties’” on page 1046

**Dialog 'Add POU’**

**Function:** The dialog is used to configure a new POU according to the IEC 61131-3 standard. This means that a POU can be a program, a function, or a function block.

**Call:** Menu bar: “Project ➔ Add Object”; context menu in the “Devices” view when an application is selected; context menu in the “POUs” view

| “Name” | Name of POU |
Table 82: “Type”

<table>
<thead>
<tr>
<th>“Function Block”</th>
<th>“Program”</th>
</tr>
</thead>
<tbody>
<tr>
<td>● “Extends”: Specification or selection of a base function module in the sense of object-oriented programming. Specified with the EXTENDS keyword in the function block declaration.</td>
<td></td>
</tr>
<tr>
<td>● “Implements”: Specification or selection of an interface in the sense of object-oriented programming. Specified with the IMPLEMENTS keyword in the function block declaration.</td>
<td></td>
</tr>
<tr>
<td>● “Final”: Derived access is not permitted. This means that you cannot extend the function block with another function block. This allows for optimized code generation.</td>
<td></td>
</tr>
<tr>
<td>● “Abstract”: Identifies that the function block has a missing or incomplete implementation and cannot be instantiated. Abstract FBs are used exclusively as base function blocks and the implementation typically occurs in a derived FB.</td>
<td></td>
</tr>
<tr>
<td>● “Access specifier”</td>
<td></td>
</tr>
<tr>
<td>– “PUBLIC”: Corresponds to the specification of no access modifier</td>
<td></td>
</tr>
<tr>
<td>– “INTERNAL”: Access to the function block is restricted to the namespace (library).</td>
<td></td>
</tr>
<tr>
<td>● “Method implementation language”: When you select the “Implements” option, you can select an implementation language here for all method objects that CODESYS generates by means of the implementation of the interface. The “Method implementation language” does not depend on the implementation language of the function block.</td>
<td></td>
</tr>
</tbody>
</table>

| “Function” | Note: Not available when “Sequential Function Chart (SFC)” is selected as the “Implementation language”. |
| “Return type:” | Data type of the return value |

| “Implementation language” | Implementation language of the POU |

See also
- ☝ Chapter 1.3.1.25.1.28.2 “Object ‘Function Block’” on page 778
- ☝ Chapter 1.3.1.25.1.28.1 “Object ‘Program’” on page 777
- ☝ Chapter 1.3.1.25.1.28.3 “Object ‘Function’” on page 781
- ☝ Chapter 1.3.1.9.23.1 “Extension of function blocks” on page 263
- ☝ Chapter 1.3.1.9.23.2 “Implementing interfaces” on page 265

Object ‘Program’

A program is a POU that supplies one or more values during execution. After execution of the program, all values are retained until the next execution. The order of calling the programs within an application is defined in task objects.

A program is added to the application or the project using the command “Project ➔ Add Object ➔ POU”. In the Device tree and in the “POUs” view the program POUs have the suffix “(PRG)”.

The editor of a program consists of the declaration part and the implementation part.

The uppermost line of the declaration part contains the following declaration:

PROGRAM <program> 

Calling a program

Programs and function blocks can call a program. A program call is not permitted in a function. There are no instances of programs.
If a POU calls a program and values of the program change as a result, these changes are retained until the next program call. The values of the program are also retained even if the repeat call takes place by another POU. This differs from the call of a function block. When calling a function block only the values of the respective instance of the function block change. The changes only need to be observed if a POU calls the same instance again.

You can also set the input or output parameters for a program directly when calling.

**Syntax:** `<program>({<input variable> := <value>, <output value> => <value>})`:

If you insert a program call via the input assistant and the "Insert with arguments" option in the input assistant is activated at the same time, CODESYS adds input and/or output parameters to the program call in accordance with the syntax.

### Examples

**Calls:**

**IL:**

```
CAL PLC_PRG()

in1:= 2

LD PLC_PRG.out2

ST erg
```

With assignment of the parameters:

```
CAL PLC_PRG()

in1:= 2,

out2=> erg)
```

**ST:**

```
PLC_PRG()

erg := PLC_PRG.out2;
```

With assignment of the parameters:

```
PLC_PRG(in1:=2, out1=>erg);
```

### See also

- [Chapter 1.3.1.25.1.28 “Object 'POU'” on page 775](#)
- [Chapter 1.3.1.9.17 “Task configuration” on page 239](#)

### Object 'Function Block'

A function block is a POU that supplies one or more values during execution.

The object is added to the application or the project using the command "Project ➔ Add Object ➔ POU". In the Device tree or in the "POUs" view the function block POU's have the suffix "(FB)".

You always call a function block via an instance that is a copy of the function block.

The editor of a function block consists of the declaration part and the implementation part.
After execution the values of the output variables and the internal variables are retained until the next execution. This means that the function block does not necessarily supply the same output values with the same input variables when called repeatedly.

In addition to the functionality described in IEC 61131-3, you can also use function blocks in CODESYS for the following functionalities of object-oriented programming:

- Extension of a function block
- Implementation of interfaces
- Methods
- Properties

The uppermost line of the declaration part contains the following declaration:

```plaintext
FUNCTION_BLOCK <access specifier> <function block> | EXTENDS <function block> | IMPLEMENT <comma-separated list of interfaces>
```

**Calling a function block**

The call always takes place via an instance of the function block. When calling a function block only the values of the respective instance change.

Declaration of the instance:

```plaintext
<instance> : <function block>;
```

You access a variable of the function block in the implementation part as follows:

```plaintext
<instance>.<variable>
```

**NOTICE!**

Please note the following:

- From outside of the function block instance you can access only the input and output variables of a function block, not the internal variables.
- Access to a function block instance is limited to the POU in which the instance is declared, unless you have declared the instance as global.
- You can assign the desired values to the function block variables when calling the instance.
Access to function block variables:
The function block “FB1” has the input variable iVar1 of the type INT and the output variable out1. In the following the variable iVar1 is called from the program “Prog”.

```plaintext
PROGRAM Prog
VAR
  inst1:FB1;
END_VAR

  inst1.iVar1 := 33;  (* FB1 is called and the value 33 is assigned to the variable iVar1 *)
  inst1();            (* FB1 is called, that's necessary for the following access to the output variable *)
  ires := inst1.out1  (* the output variable out1 of the FB1 is read *)
```

In FBD:

```
inst1

FB1

33 - iVar1  out1 - ires
```

Assigning variable values during the call:
In the text languages IL and ST you can directly assign values to the input and/or output variables when calling the function block.

The assignment of a value to an input variable takes place with `:=`
The assignment of a value to an output variable takes place with `=>`

Example
The instance CMD_TMR of the timer function block is called with assignments for the input variables IN and PT. Subsequently, the output variable Q of the timer is assigned to the variable A.

```
CMD_TMR (IN := %IX5, PT := 100=);
A := CMD_TMR.Q;
```

If you insert a function block instance via the “input assistant” and the option “Insert with arguments” is activated in the “Input Assistant” dialog box, CODESYS inserts the call with all input and output variables. You then only need to insert the desired value assignment. In the above example CODESYS inserts the call as follows: CMD_TMR (IN:= ,PT:= , Q=> ).

See also
- § Chapter 1.3.1.25.1.28 “Object ‘POU’” on page 775
- § Chapter 1.3.1.9.23.1 “Extension of function blocks” on page 263
- § Chapter 1.3.1.9.23.2 “Implementing interfaces” on page 265
- § Chapter 1.3.1.25.1.28.5 “Object ‘Method’” on page 784
- § Chapter 1.3.1.25.1.28.8 “Object ‘Property’” on page 791
Object 'Function'

A function is a POU that supplies precisely one data element when executed and whose call in textual languages can occur as an operator in expressions. The data element can also be an array or a structure.

The object is added to the application or the project by clicking “Project ➤ Add Object ➤ POU”. In the device tree or in the “POUs” view, function POUs have the “(FUN)” suffix.

**NOTICE!**

Functions have no internal status information, which means that functions do not save the values of their variables until the next call. Calls of a function with the same input variable values always supply the same output value. Therefore, functions must not use global variables and addresses!

The editor of a function consists of the declaration part and the implementation part.

The top line of the declaration part contains the following declaration:

FUNCTION <function> : <data type>

Below that, you declare the input and function variables.

The output variable of a function is the function name.

**NOTICE!**

If you declare a local variable in a function as RETAIN, this has no effect. In this case, CODESYS issues a compiler error.

**NOTICE!**

You cannot mix explicit and implicit parameter assignments in function calls in CODESYS V3. This means that you have to use either only explicit or only implicit parameter assignments in function calls. The order of the parameter assignments when calling a function is arbitrary.

Calling a function

In ST, you can use the call of a function as an operand in expressions.

In SFC, you can use a function call only within step actions or transitions.
Examples

Function with declaration part and a line implementation code

According to the IEC 61131-3 standard, functions can have additional outputs. You declare the additional outputs in the function between the keywords `VAR_OUTPUT` and `END_VAR`. The function is called according to the following syntax:

```
<function> (<function output variable1> => <output variable 1>,
<function output variable n> => <output variable n>)
```

Function calls:

**ST:**

```
result := POU_Funct(5,3,22);
```

**AWL:**

```
LD
POU_Funct
3
ST
```

**FBD:**

```
POU_Funct := ivar1 + ivar2 * ivar3;
```

Example

The `fun` function is defined with two input variables `in1` and `in2`. The output variable of the `fun` function is written to the locally declared output variables `loc1` and `loc2`.

```
fun(in1 := 1, in2 := 2, out1 => loc1, out2 => loc2);
```
Object 'Interface'

Symbol: ☠

Keyword: INTERFACE

An interface is a means of object-oriented programming. The object ITF describes a set of method and property prototypes. In this context, prototype means that the methods and properties contain only declarations and no implementation.

This allows different function blocks having common properties to be used in the same way. An object "ITF" is added to the application or the project with the command "Project ➤ Add Object ➤ Interface".

Table 83: “Adding an interface”

<table>
<thead>
<tr>
<th>&quot;Inheritance&quot;</th>
<th>Interface name</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Name&quot;</td>
<td></td>
</tr>
<tr>
<td>&quot;Extends&quot;</td>
<td>☑️: Extends the interface that you enter in the input field or via the input assistant ☑️. This means that all methods of the interface that extend the new interface are also available in the new interface.</td>
</tr>
</tbody>
</table>

You can add the objects “Interface property” and “Interface Method” to the object "ITF". Interface methods may contain only the declarations of input, output and input/output variables, but no implementation.

So that you can also use an interface in the program, there must be a function block that implements this interface.

This means:

- the function block contains the interface in its IMPLEMENTS list in its declaration part
- the function block contains an implementation for all methods and property prototypes of the interface

A function block can implement one or more interfaces. You can use the same method with identical parameters, but different implementation code in different function blocks.

Please note the following:

- You may not define variables within an interface. An interface has no implementation part and no actions. Only a collection of methods is defined, in which you may define only input, output and input/output variables.
- CODESYS always treats variables declared with the type of an interface as references.
- A function block that implements an interface must contain implementation code for the methods of the interface. You have named the methods exactly as in the interface and the methods contain the same input, output and input/output variables as in the interface.
NOTICE!

Interface references and online change

The following can happen with a compiler version < 3.4.1.0: if a function block changes its data because variables are added or deleted, or because the type of variables changes, then CODESYS copies all instances of the function block to a new memory location. In this case, however, an interface reference refers not to the new memory location, but still to the old one.

In case of compiler versions >= 3.4.1.0, CODESYS automatically re-addresses the interface references so that CODESYS also references the correct interface in case of an online change. CODESYS requires additional code and more time for this, so that jitter problems can occur depending on the number of objects concerned. Therefore, CODESYS displays the number of variables and interface references concerned before the execution of the online change and you can then decide whether the online change should be executed or aborted.

Example

Definition of an interface and its use in a function block

You have inserted the interface "ITF" below the application. The interface contains the methods "Method1" and "Method2". "ITF", "Method1" and "Method2" contain no implementation code. You insert the required variable declarations only in the declaration part of the methods.

If you subsequently insert a function block in the device tree that implements the interface "ITF", CODESYS automatically also inserts the methods "Method1" and "Method2" under the function block. Here you can implement function-block-specific code in the methods.

Object 'Method'

Symbol: METHOD

Keywords: METHOD
Methods are an extension of the IEC 61131-3 standard and a tool for object-oriented programming which is used for data encapsulation. A method contains a declaration and an implementation. However, unlike a function, a method is not an independent POU, and it is subordinated to a function block or program. A method can access all valid variables of the superordinate POU.

You can use interfaces for the organization of methods.

You can add a method below a program or a function block. Click “Project ➤ Add Object ➤ Method” to open the “Add Method” dialog.

### Declaration
- The variables of a method contain temporary data that are valid only during the execution of the method (stack variables). All variables that are declared and implemented in a method are reinitialized each time the method is called.
- Like functions, methods can have additional outputs. You must assign these additional outputs in the method call.
- Depending on the declared access specifier, a method can be called only within its own namespace (INTERNAL), only within its own POU and its derivatives (PROTECTED), or only within its own POU (PRIVATE). For PUBLIC, the method can be called from anywhere.

Interface methods can have declared input, output, and VAR_IN_OUT variables, but do not contain an implementation.

See also
- Chapter 1.3.1.25.1.28.6 “Object 'Interface Method’” on page 788

### Implementation
- Access to function block instances or program variables is permitted in the implementation of the method.
- The THIS pointer allows for access to its own function block instance. Therefore, the pointer is permitted only in methods that are assigned to a function block.
- A method cannot access VAR_TEMP variables of the function block.
- A method can call itself recursively.

#### NOTICE!
When you copy a method below a POU and add it below an interface, or move the method there, the contained implementation is removed automatically.

### Calling a method

#### Syntax for calls:
```
<return value variable> := <POU name> . <method name> ( <method input name> := <variable name> , <further method input name> := <variable name> )* ;
```

For the method call, you assign transfer parameters to the input variables of the method. Respect the declaration when doing this. It is enough to specify the names of the input variables without paying attention to their order in the declaration.
METHOD PUBLIC DoIt : BOOL
  VAR_INPUT
    iInput_1 : DWORD;
    iInput_2 : DWORD;
    sInput_3 : STRING(12);
  END_VAR
  bFinishedMethod := fbInstance.DoIt(sInput_3 := 'Hello World ',
    iInput_2 := 16#FFFF, iInput_1 := 16);

When the method is called, the return value of the method is assigned, for example, to variables declared locally. When you omit the names of the input variables, you have to pay attention to the declaration order.

Example
Declaration
METHOD PUBLIC DoIt : BOOL
  VAR_INPUT
    iInput_1 : DWORD;
    iInput_2 : DWORD;
    sInput_3 : STRING(12);
  END_VAR
Call
  bFinishedMethod := fbInstance.DoIt(16, 16#FFFF,'Hello World ');

Recursive method call
Within the implementation, a method can call itself, either directly by means of the THIS pointer, or by means of a local variable for the assigned function block.

- THIS^. <method name> ( <parameter transfer of all input and output variables> )
  Direct call of the relevant function block instance with the THIS pointer
- VAR fb_Temp : <function block name>; END_VAR
  Call by means of a local variable of the method that temporarily instantiates the relevant function block

A compiler warning is issued for a recursive call. If the method is provided with the pragma {attribute 'estimated-stack-usage' := 'estimated_stack_size_in_bytes'}), then the compiler warning is suppressed. For an implementation example, see the "Attribute 'estimated-stack-usage'" chapter.

To call methods recursively, it is not enough to specify only the method name. If only the method name is specified, then a compiler error is issued: "Program name, function or function block instance expected instead of"

See also
- § Chapter 1.3.1.9.23.4 “Calling methods” on page 267
- § Chapter 1.3.1.24.6.2.10 “Attribute ‘estimated-stack-usage’” on page 608
- § Chapter 1.3.1.24.2.15 “THIS” on page 502

Special methods of a function block
| **FB_Init** | Declarations automatically implicit, but explicit declaration also possible.  
Contains initialization code for the function block, as is defined in the declaration part of the function block. |
| **FB_Reinit** | Explicit declaration is necessary.  
Call after the instance of the function block was copied (as during an online change). It reinitializes the new instance module. |
| **FB_Exit** | Explicit declaration is necessary.  
Call for each instance of the function block before a new download or a reset or during an online change for all shifted or deleted instances. |

### Properties
Provides **Set** and/or **Get** accessor methods.

**See also**
- Chapter 1.3.1.24.10 “Methods ‘FB_Init’, ‘FB_Reinit’, and ‘FB_Exit’” on page 653
- Chapter 1.3.1.25.1.28.8 “Object ‘Property’” on page 791
- Chapter 1.3.1.25.1.28.7 “Object ‘Interface Property’” on page 788

### Dialog 'Add Method'
**Function:** Defines a method below the selected POU when the dialog is closed.

**Call:** Menu bar: “Project ➔ Add Object ➔ Method”; context menu

**Requirement:** A program (PRG) or a function block (FUNCTION_BLOCK) is selected in the “POUs” view or the “Devices” view.

The interface of a method inserted below a basic function block is copied when a method with the same name is inserted below a derived function block.

| **“Name”** | Example: `meth_DoIt`.  
The standard methods `FB_Init` and `FB_Exit` are offered in a list box if they are not already inserted below the POU. If it is a derived function block, then the list box also offers all of the methods of the basic function block. |
| **“Return type”** | Default data type or structured data type of return value  
Example: `BOOL` |
| **“Implementation language”** | Example: “Structured Text (ST)” |
| **“Access specifier”** | Controls access to data.  
- “PUBLIC” or not specified: Access is not restricted.  
- “PRIVATE”: Access is restricted to the program, function block, or GVL. The object is marked as (private) in the POU or device view. The declaration contains the keyword PRIVATE.  
- “PROTECTED”: Access is restricted to the program, function block, or GVL with its derivations. The declaration contains the keyword PROTECTED. The object is marked as (protected) in the POU or device view.  
- “INTERNAL”: Access to the method is restricted to the namespace (library). The object is marked as (internal) in the POU or device view. The declaration contains the keyword INTERNAL. |
| **“Abstract”** | ☑: Identifies that the method does not have an implementation and the implementation is provided by the derived FB |
| **“Add”** | Adds a new method below the selected object. |
When you do object-oriented programming and want to use inheritance for blocks, you have the
following support: When you insert a method, action, etc. below an inherited block, the “Add
Object” dialog box includes a combo box with a list of methods, actions, etc. used in the base
block. In this way, you can easily accept a method definition of the base and adapt it accordingly
for the inherited method of the block. Methods and attributes with the PRIVATE access modifier
are not available in this selection because they should not be inherited. When accepted into the
inherited block, methods and attributes with the PUBLIC access modifier automatically have a
blank access modifier field. (Functionally, this means the same thing.)

See also
- Chapter 1.3.1.9.23.1 “Extension of function blocks” on page 263
- Chapter 1.3.1.9.23 “Object-oriented programming” on page 263

Object 'Interface Method'

Symbol: 🟢

This object is used for object-oriented programming.

The object “Interface Method” is added to an interface via the command “Project ➔ Add Object”.

If a method is inserted underneath an interface, you can add and instance only variable declara-
tions (input, output and input/output variables) in this method.

You can only add program code to the method if a function block 'implements' the interface to
which the method belongs. CODESYS then inserts the method underneath the function block.

See also
- Chapter 1.3.1.25.1.28.4 “Object 'Interface'” on page 783
- Chapter 1.3.1.25.1.28.5 “Object 'Method'” on page 784
- Chapter 1.3.1.9.23.2 “Implementing interfaces” on page 265

Object 'Interface Property'

Symbol: 🛠

Interface properties are an extension of the IEC 61131-3 standard and a tool for object-oriented
programming. An interface property declares the accessor methods Get and Set (no imple-
mentation code). Therefore, a function block that implements an interface also inherits their
interface properties.

You can add an interface property to the device tree for an interface. Then an interface is
extended with the accessor methods Get and Set. The Get accessor is for read access. The
Set accessor is for write access. You can delete an unneeded accessor. Click “Project ➔ Add Object ➔ Interface Property” to add an accessor. The “Add Interface Property” dialog
opens.

See also
- Chapter 1.3.1.25.1.28.4 “Object 'Interface'” on page 783
- Chapter 1.3.1.25.1.28.8 “Object 'Property'” on page 791
This interface `itf_A` has the property `Literal_A` with the accessor methods `Get` and `Set`. The function blocks `fb_A1` and `fb_A2` implement the interface `itf_A` and therefore inherit its interface property. Each FB has its own implementation.

**Interface itf_A**

```plaintext
INTERFACE itf_A
VAR
END_VAR
PROPERTY Literal_A : STRING
```

**FB fb_A1**

```plaintext
FUNCTION_BLOCK fb_A1 IMPLEMENTS itf_A
VAR
  str_1 : STRING;
  str_2 : STRING;
  iCnt : INT;
END_VAR
iCnt := iCnt + 1;
str_1 := 'Function block A1';
```
VAR

Literal_A := CONCAT (str_1, ' and property.');

VAR

str_2 := Literal_A;

FUNCTION_BLOCK fb_A2 IMPLEMENTS itf_A
VAR
str_1 : STRING;
str_2 : STRING;
iCnt : INT;
END_VAR

iCnt := iCnt + 1;
str_1 := 'Function block A2';

PROGRAM PLC_PRG
VAR
iCnt : INT;
my_1 : fb_A1;
my_2 : fb_A2;
strName_1 : STRING;
strName_2: STRING;
END_VAR

iCnt := iCnt + 1;
my_1();
my_2();
strName_1:= my_1.Literal_A;
strName_2:= my_2.Literal_A;
my_1.Literal_A := 'Hello 1';
my_2.Literal_A := 'World 2';

This leads to the following monitoring of PLC_PRG when the application is in runtime mode:
Properties are an extension of the IEC 61131-3 standard and a tool for object-oriented programming. Properties are used for data encapsulation because they allow for external access to data and act as filters at the same time. For this purpose, a property provides the accessor methods Get and Set which allows for read and write access to the data of the instance below the property.

You can add a property with accessor methods below a program, a function block, or a global variable list. Click “Project ➔ Add Object ➔ Property” to open the “Add Property” dialog.

You can add an interface property below an interface.

When you copy a property that is inserted below a POU and add it below an interface, or if you move the property there, the included implementations are removed automatically.

See also

- Chapter 1.3.1.25.1.28.7 “Object ‘Interface Property’” on page 788

**Dialog 'Add Property'**

**Function:** Creates a new property below the selected POU when the dialog is closed.

**Call:** Menu bar: “Project ➔ Add Object ➔ Property”; context menu

**Requirement:** A program (PRG), a function block (FUNCTION_BLOCK), or a global variable list (GVL) is selected in the “POUs” view or the “Devices” view.

<table>
<thead>
<tr>
<th>“Name”</th>
<th>Name (identifier) of the property</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Return type”</td>
<td>Default type or structured type of return value</td>
</tr>
<tr>
<td>Example: BOOL</td>
<td></td>
</tr>
<tr>
<td>“Implementation language”</td>
<td>Example: “Structured Text (ST)”</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Expression</th>
<th>Type</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>iCnt</td>
<td>INT</td>
<td>-25684</td>
</tr>
<tr>
<td>my_1</td>
<td>fb_A1</td>
<td></td>
</tr>
<tr>
<td>str_1</td>
<td>STRING</td>
<td>'Function block A1'</td>
</tr>
<tr>
<td>str_2</td>
<td>STRING</td>
<td>'Hello 1'</td>
</tr>
<tr>
<td>my_2</td>
<td>fb_A2</td>
<td></td>
</tr>
<tr>
<td>str_1</td>
<td>STRING</td>
<td>'Function block A2'</td>
</tr>
<tr>
<td>str_2</td>
<td>STRING</td>
<td>'World 2'</td>
</tr>
<tr>
<td>iCnt</td>
<td>INT</td>
<td>-31303</td>
</tr>
<tr>
<td>strName_1</td>
<td>STRING</td>
<td>'Function block A1 and property.'</td>
</tr>
<tr>
<td>strName_2</td>
<td>STRING</td>
<td>'Function block A2'</td>
</tr>
<tr>
<td><strong>Access specifier</strong></td>
<td>Controls access to data</td>
<td></td>
</tr>
<tr>
<td>---------------------</td>
<td>-------------------------</td>
<td></td>
</tr>
<tr>
<td>“PUBLIC” or unspecified</td>
<td>Access is not restricted.</td>
<td></td>
</tr>
<tr>
<td><strong>PRIVATE</strong></td>
<td>Access is restricted to the program, function block, or GVL. The object is marked as (private) in the POU or device view. The declaration contains the keyword PRIVATE.</td>
<td></td>
</tr>
<tr>
<td><strong>PROTECTED</strong></td>
<td>Access is restricted to the program, function block, or GVL with its derivations. The object is marked as (protected) in the POU or device view. The declaration contains the keyword PROTECTED.</td>
<td></td>
</tr>
<tr>
<td><strong>INTERNAL</strong></td>
<td>Access is restricted to the namespace (library). The object is marked as (internal) in the POU or device view. The declaration contains the keyword INTERNAL.</td>
<td></td>
</tr>
</tbody>
</table>

**Editor 'Property'**

You can program the data access in the editor. The code can contain additional local variables. However, it must not contain any additional input variables or (as opposed to a function or method) output variables.

| **Abstract** | ✔: Identifies that the property does not have an implementation and the implementation is provided by the derived FB |
| **Add** | Adds a new property below the selected object and below that the accessor methods Get and Set. Note: When you select a property, you can also add a previously removed accessor explicitly by clicking “Add Object”. |
Function block

```plaintext
FUNCTION_BLOCK fb_A
VAR_INPUT
END_VAR
VAR_OUTPUT
END_VAR
VAR
  iA : INT;
  iB : INT;
END_VAR
  iA := iA + 1;
```

Property Value_A

```plaintext
PROPERTY PUBLIC Value_A : INT
```

Accessor method fb_A.Value_A.Get

```plaintext
VAR
  Value_A := iA;
END_VAR
```

Accessor method fb_A.Value_A.Set

```plaintext
VAR
  iB := Value_A;
```

Get and Set accessor

When the `Set` accessor is called, the property is written. It is then used in the same way as an input parameter. When the `Get` accessor is called, the property is read. It is then used in the same way as an output parameter. Access is restricted by means of access specifiers, and the objects are marked accordingly.

When a property is accessed as read only or write only, you can delete the unneeded accessor.

You can add accessors explicitly by selecting a property and clicking “Add Object”. A dialog opens, either “Add Get Accessor” or “Add Set Accessor”. There you can set the implementation language and the access.

<table>
<thead>
<tr>
<th>&quot;Implementation language&quot;</th>
<th>Example: “Structured Text (ST)”</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>&quot;Access specifier&quot;</th>
<th>Access is not restricted.</th>
</tr>
</thead>
<tbody>
<tr>
<td>PUBLIC or unspecified</td>
<td>Access is restricted to the program, function block, or GVL. The object is marked as (private) in the POU or device view. The declaration contains the keyword.</td>
</tr>
<tr>
<td>PRIVATE</td>
<td>Access to the property is restricted to the program, function block, or GVL and its derivations. The declaration contains the keyword. The object is marked as (protected) in the POU or device view.</td>
</tr>
</tbody>
</table>
Access to the method is restricted to the namespace (the library). The object is marked as (internal) in the POU or device view. The declaration contains the keyword.

“Add” Adds the accessor methods Get or Set below the selected property.

Monitoring of properties in online mode

The following pragmas are provided for the monitoring of properties in online mode. You insert them at the top position of the property definition:

- `{attribute 'monitoring' := 'variable'}`
  Each time the property is accessed, CODESYS saves the actual value to a variable and displays the value of this variable. This value can become outdated if no more access to the property takes place in the code.

- `{attribute 'monitoring' := 'call'}`
  Each time the value is displayed, CODESYS calls the code of the Get accessor. If this code contains a side effect, then the monitoring executes the side effect.

You can monitor a property with the help of the following functions.

- Inline monitoring
  Requirement: The “Enable inline monitoring” option is selected in the “Text Editor” category of the “Options” dialog.

- Watch list

See also:

- § Chapter 1.3.1.9.23.4 “Calling methods” on page 267
- § Chapter 1.3.1.24.6.2.21 “Attribute ‘monitoring’” on page 619

Input assistance when creating inheriting blocks

When you do object-oriented programming and want to use inheritance for blocks, you have the following support: When you insert a method, action, etc. below an inherited block, the “Add Object” dialog box includes a combo box with a list of methods, actions, etc. used in the base block. In this way, you can easily accept a method definition of the base and adapt it accordingly for the inherited method of the block. Methods and attributes with the PRIVATE access modifier are not available in this selection because they should not be inherited. When accepted into the inherited block, methods and attributes with the PUBLIC access modifier automatically have a blank access modifier field. (Functionally, this means the same thing.)

See also:

- § Chapter 1.3.1.9.23.1 “Extension of function blocks” on page 263
- § Chapter 1.3.1.9.23 “Object-oriented programming” on page 263

Object ‘Action’

Symbol: 🌈

In an action, you implement additional program code, which you can further implement as in another language as the base implementation. This base implementation is a function block or a program where you inserted the action.

An action does not have its own declaration and it works with the data from the base implementation. This means that the action uses the input and output variables and the local variables from its base implementation.

Add an “Action” to a function block or program by clicking “Project ➔ Add Object ➔ Action”.

Object 'Action'
Table 85: “Add Action”

<table>
<thead>
<tr>
<th>“Name”</th>
<th>Name of the action</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Implementation language”</td>
<td>Drop-down list of implementation language</td>
</tr>
</tbody>
</table>

Input assistance when creating inheriting blocks

When you do object-oriented programming and want to use inheritance for blocks, you have the following support: When you insert a method, action, etc. below an inherited block, the “Add Object” dialog box includes a combo box with a list of methods, actions, etc. used in the base block. In this way, you can easily accept a method definition of the base and adapt it accordingly for the inherited method of the block. Methods and attributes with the *PRIVATE* access modifier are not available in this selection because they should not be inherited. When accepted into the inherited block, methods and attributes with the *PUBLIC* access modifier automatically have a blank access modifier field. (Functionally, this means the same thing.)

See also

- § Chapter 1.3.1.9.23.1 “Extension of function blocks” on page 263
- § Chapter 1.3.1.9.23 “Object-oriented programming” on page 263

Calling an action

Syntax:

<program>.<action> or <FB instance>.<action>

To call an action from only within the base implementation, you only have to provide the action name.

Examples

Calling a “Reset” action from another POU

The call is not executed from the base implementation.

Declaration:

```plaintext
PROGRAM PLC_PRG
VAR
  Inst : Counter;
END_VAR
```

Calling a “Reset” action from an IL POU

```plaintext
CAL Inst.Reset(In := FALSE)
LD Inst.Out
ST ERG
```

Calling a “Reset” action from an ST POU

```plaintext
Inst.Reset(In := FALSE);
Erg := Inst.out;
```

Calling a “Reset” action from an FBD POU

![Diagram](image_url)

*Actions are used frequently in the SFC implementation language.*

See also

- § Chapter 1.3.1.24.1.4.7.2 “SFC element 'Action’” on page 454
Object 'Transition'

Symbol: 

This object can be used as a transition element in a program block implemented in SFC.

See also

-  Chapter 1.3.1.24.1.4.7.1 "SFC elements 'Step' and 'Transition’" on page 452

Input assistance when creating inheriting blocks

When you do object-oriented programming and want to use inheritance for blocks, you have the following support: When you insert a method, action, etc. below an inherited block, the “Add Object” dialog box includes a combo box with a list of methods, actions, etc. used in the base block. In this way, you can easily accept a method definition of the base and adapt it accordingly for the inherited method of the block. Methods and attributes with the PRIVATE access modifier are not available in this selection because they should not be inherited. When accepted into the inherited block, methods and attributes with the PUBLIC access modifier automatically have a blank access modifier field. (Functionally, this means the same thing.)

See also

-  Chapter 1.3.1.9.23.1 “Extension of function blocks” on page 263
-  Chapter 1.3.1.9.23 “Object-oriented programming” on page 263

Object 'POUs for Implicit Checks'

You can add these special POUs to an application to equip them with implicit monitoring functions. At runtime, these functions check the limits of arrays or subrange types, the validity of pointer addresses, and division by zero. Please note: This option can be disabled for devices that are already equipped with these kinds of monitoring blocks by a special implicit library.

The command “Add Object ➔ POU for Implicit Checks” is used for adding to the application. The command opens the “Add POU for Implicit Checks” dialog where you can select a monitoring function type (see table below). Depending on the monitoring function, you have to adapt the implementation code or create it yourself from scratch.

To prevent multiple inclusions, monitoring functions that have already been inserted are disabled in the “Add POU for Implicit Checks” dialog.

**NOTICE!**

To get the feature for monitoring functions, do not edit their declaration part. However, you are permitted to add local variables.

After removing an implicit monitoring function (example: Check Bounds) from the project, only a download is possible, not an online change. A corresponding message is issued.

By default, CODESYS does not run implicit checks for function blocks from libraries used in the application. However, you can extend the check to the libraries by opening the “Properties” dialog of the application and specifying the compiler definition checks_in_libs in the “Compiler-Defines” field in the “Build” tab. This definition affects implementation libraries (*.library) only, not protected libraries (*.compiled-library).

You can use the "no_check" attribute to deactivate the check for special POUs in the project.
### Table 86: “Available Functions”

<table>
<thead>
<tr>
<th>Monitoring function</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Check Bounds”</td>
<td>“Bound Checks”</td>
<td>Appropriate handling of bound violations; such handling includes setting flags or changing field indices.</td>
</tr>
<tr>
<td>“CheckDivDInt”</td>
<td>“Division checks”</td>
<td>Monitors the divisor value to avoid division by zero.</td>
</tr>
<tr>
<td>“CheckDivLInt”</td>
<td></td>
<td></td>
</tr>
<tr>
<td>“CheckDivReal”</td>
<td></td>
<td></td>
</tr>
<tr>
<td>“CheckDivLReal”</td>
<td></td>
<td></td>
</tr>
<tr>
<td>“CheckRangeSigned”</td>
<td>“Range checks”</td>
<td>Monitors the range limit of a subrange type in runtime mode. Valid for data types DINT/UDINT.</td>
</tr>
<tr>
<td>“CheckRangeUnsigned”</td>
<td></td>
<td>Monitors the range limit of a subrange type in runtime mode. Valid for data types LINT/ULINT.</td>
</tr>
<tr>
<td>“CheckLRangeSigned”</td>
<td>“L-range checks”</td>
<td></td>
</tr>
<tr>
<td>“CheckLRangeUnsigned”</td>
<td></td>
<td></td>
</tr>
<tr>
<td>“CheckPointer”</td>
<td>“Pointer checks”</td>
<td>You are responsible for filling in this function completely with implementation code. Refer to the help page for &quot;POU 'CheckPointer'&quot;. The function should monitor whether the passed pointer reference a valid memory address, and whether the orientation of the referenced memory area matches the variable type to which the pointer refers. If both conditions are fulfilled, then the pointer is returned. If not, then CheckPointer should complete an appropriate error handling. CheckPointer monitors the same way as variables of type REFERENCE TO.</td>
</tr>
</tbody>
</table>

See also
- Chapter 1.3.1.9.22 “Using POUs for implicit checks” on page 262
- Chapter 1.3.1.25.1.29.1 “POU 'CheckBounds'” on page 797
- Chapter 1.3.1.25.1.29.2 “POU 'CheckDivInt'” on page 801
- Chapter 1.3.1.25.1.29.3 “POU 'CheckDivLInt'” on page 801
- Chapter 1.3.1.25.1.29.5 “POU 'CheckDivLReal'” on page 803
- Chapter 1.3.1.25.1.29.4 “POU 'CheckDivReal'” on page 802
- Chapter 1.3.1.25.1.29.10 “POU 'CheckPointer'” on page 809
- Chapter 1.3.1.25.1.29.6 “POU 'CheckRangeSigned'” on page 804
- Chapter 1.3.1.25.1.29.8 “POU 'CheckRangeUnsigned'” on page 807
- Chapter 1.3.1.25.1.29.7 “POU 'CheckLRangeSigned'” on page 806
- Chapter 1.3.1.25.1.29.9 “POU 'CheckLRangeUnsigned'” on page 808
- Chapter 1.3.1.25.3.10.4 “Dialog ‘Properties’ - ‘Build’” on page 1048
- Chapter 1.3.1.24.6.2.23 “Attribute ‘no_check’” on page 622

**POU 'CheckBounds'**

**Functions for Bound Checks:**

- **CheckBounds**

The task of this monitoring function is to handle bound violations appropriately. Examples of reactions to violations include setting error flags and changing the value of the array index. The check is performed only for one variable array index. An incorrect constant array index causes a compiler error. CODESYS calls the function implicitly when values are assigned to an ARRAY variable.

After inserting the function, you receive automatically generated code in the declaration and implementation parts. See below.
Declaration part

```plaintext
// Automatically generated code: DO NOT EDIT
FUNCTION CheckBounds : DINT
VAR_INPUT
    index, lower, upper: DINT;
END_VAR
```

Implementation

```plaintext
// This automatically generated code is a suggested implementation.
IF  index < lower THEN
    CheckBounds := lower;
ELSIF  index > upper THEN
    CheckBounds := upper;
ELSE
    CheckBounds := index;
END_IF

(* It is also possible to set a breakpoint, log messages or e.g. to
halt on an exception:
Add CmpApp.library, SysExcept.library and SysTypes2_Itf as newest.
Declaration:
VAR
    _pApp : POINTER TO CmpApp.APPLICATION;
    _result   : SysTypes.RTS_IEC_RESULT;
END_VAR

Implementation:
    _pApp := AppGetCurrent(pResult:=_result);
    IF  index < lower THEN
        CheckBounds := lower;
        IF _pApp <> 0 THEN
            AppGenerateException(pApp:=_pApp,
                ulException:=RtsExceptions.RTSEXCPT_ARRAYBOUNDS);
        END_IF
    ELSIF  index > upper THEN
        CheckBounds := upper;
        IF _pApp <> 0 THEN
            AppGenerateException(pApp:=_pApp,
                ulException:=RtsExceptions.RTSEXCPT_ARRAYBOUNDS);
        END_IF
    ELSE
        CheckBounds := index;
    END_IF
*)
```

When the “CheckBounds” function is called, it receives the following input parameters:
- **index**: Index of the array element
- **lower**: Lower limit of the array range
- **upper**: Upper limit of the array range

The return value is the index of the array element, as long as it is within a valid range. If not, then the CODESYS returns either the upper or lower limit, depending on which threshold was violated.
In the sample program below, the index falls short of the defined lower limit of the `a` array.

```
PROGRAM PLC_PRG
VAR
  a: ARRAY[0..7] OF BOOL;
  b: INT:=10;
END_VAR
  a[b]:=TRUE;
```

In this example, the `CheckBounds` function causes `a` to change the upper limit of the array range index to 10. The value `TRUE` is assigned then to the element `a[7]`. In this way, the function corrects array access outside of the valid array range.
Add the following libraries in the library manager of the application:

- CmpApp.library and SysExcept.library as placeholder libraries
- SysTypes2_Itfs.library with “Newest version always”

Add a “CheckBounds” object below the application and modify the specified code as shown below.

**FUNCTION CheckBounds : DINT**

**VAR_INPUT**

index, lower, upper: DINT;

**END_VAR**

**VAR**

_pApp : POINTER TO CmpApp.APPLICATION;
_Result : ISystypes2.RTS_IEC_RESULT;

**END_VAR**

// This automatically generated code is a suggested implementation.
_pApp := AppGetCurrent(pResult := _Result);
IF  index < lower THEN CheckBounds := lower;
IF _pApp <> 0 THEN
    AppGenerateException(pApp := _pApp, ulException := RtsExceptions.RTSEXCPT_ARRAYBOUNDS);
END_IF
ELSIF index > upper THEN CheckBounds := upper;
IF _pApp <> 0 THEN
    AppGenerateException(pApp:=_pApp, ulException:=RtsExceptions.RTSEXCPT_ARRAYBOUNDS);
END_IF
ELSE
    CheckBounds := index;
END_IF

Program a “MAIN_PRG” object below the application with the contents shown below.

**PROGRAM MAIN_PRG**

**VAR**

xInit    : BOOL;
arData   : ARRAY[0..7] OF BYTE;
i        : INT;
dwAdr       : DWORD;

**END_VAR**

IF NOT xInit THEN // Required for CheckBounds
    xInit := TRUE;
END_IF

// Set i to a value > 7 or < 0
// Generates an exception in CheckBounds, user-defined
arData[i] := 11;

When you load and start this application, an exception will be thrown when array bounds are violated. Processing stops in “CheckBounds” so that the type of error can be detected.

See also

- § Chapter 1.3.1.9.22 “Using POUs for implicit checks” on page 262
- § Chapter 1.3.1.25.1.19.8 “Tab ‘Log’” on page 750
POU 'CheckDivInt'

To prevent division by zero, you can use the functions CheckDivInt, CheckDivLint, CheckDivReal, and CheckDivLReal. If you include these functions in the application, then they are called before each division operation in the code.

CAUTION!

To obtain the feature for monitoring functions, do not edit the declaration section. However, you are permitted to add local variables.

The default implementation of CheckDivReal:

```
// This is automatically generated code: DO NOT EDIT
FUNCTION CheckDivReal : REAL
VAR_INPUT
  divisor:REAL;
END_VAR

IF divisor = 0 THEN
  CheckDivReal:=1;
ELSE
  CheckDivReal:=divisor;
END_IF;
```

The DIV operator uses the output of the CheckDivReal function as a divisor. In the sample program below, CheckDivReal prevents division by 0 by changing the implicit value of the divisor d from "0" to 1 before the division operation is executed. Therefore, the division result is 799.

```
PROGRAM PLC_PRG
VAR
  erg:REAL;
  v1:REAL:=799;
  d:REAL:=0;
END_VAR
  erg:= v1 / d;
```

See also

- Chapter 1.3.1.9.22 “Using POUs for implicit checks” on page 262

POU 'CheckDivLInt'

To prevent division by zero, you can use the functions CheckDivInt, CheckDivLint, CheckDivReal, and CheckDivLReal. If you include these functions in the application, then they are called before each division operation in the code.
CAUTION!
To obtain the feature for monitoring functions, do not edit the declaration section. However, you are permitted to add local variables.

The default implementation of CheckDivReal:

Declaration section:

// This is automatically generated code: DO NOT EDIT
FUNCTION CheckDivReal : REAL
VAR_INPUT
    divisor:REAL;
END_VAR

Implementation section:

// This automatically generated code is a suggested implementation.
IF divisor = 0 THEN
    CheckDivReal:=1;
ELSE
    CheckDivReal:=divisor;
END_IF;

The DIV operator uses the output of the CheckDivReal function as a divisor. In the sample program below, CheckDivReal prevents division by 0 by changing the implicit value of the divisor d from "0" to 1 before the division operation is executed. Therefore, the division result is 799.

PROGRAM PLC_PRG
VAR
    erg:REAL;
    v1:REAL:=799;
    d:REAL:=0;
END_VAR
    erg:= v1 / d;

See also
- Chapter 1.3.1.9.22 “Using POU for implicit checks” on page 262

POU 'CheckDivReal'

Functions for preventing division by zero: CheckDivInt, CheckDivLint, CheckDivReal, and CheckDivLReal

To prevent division by zero, you can use the functions CheckDivInt, CheckDivLint, CheckDivReal, and CheckDivLReal. If you include these functions in the application, then they are called before each division operation in the code.

CAUTION!
To obtain the feature for monitoring functions, do not edit the declaration section. However, you are permitted to add local variables.
The default implementation of CheckDivReal:

// This is automatically generated code: DO NOT EDIT
FUNCTION CheckDivReal : REAL
VAR_INPUT
  divisor: REAL;
END_VAR

Implementation section:

// This automatically generated code is a suggested implementation.
IF divisor = 0 THEN
  CheckDivReal := 1;
ELSE
  CheckDivReal := divisor;
END_IF;

The DIV operator uses the output of the CheckDivReal function as a divisor. In the sample program below, CheckDivReal prevents division by 0 by changing the implicit value of the divisor d from "0" to 1 before the division operation is executed. Therefore, the division result is 799.

PROGRAM PLC_PRG
VAR
  erg: REAL;
  v1: REAL := 799;
  d: REAL := 0;
END_VAR
  erg := v1 / d;

See also
● % Chapter 1.3.1.9.22 “Using POUs for implicit checks” on page 262

POU 'CheckDivLReal'

Functions for preventing division by zero:
CheckDivInt, CheckDivLint, CheckDivReal, and CheckDivLReal

To prevent division by zero, you can use the functions CheckDivInt, CheckDivLint, CheckDivReal, and CheckDivLReal. If you include these functions in the application, then they are called before each division operation in the code.

CAUTION!
To obtain the feature for monitoring functions, do not edit the declaration section. However, you are permitted to add local variables.
The default implementation of CheckDivReal:

// This is automatically generated code: DO NOT EDIT
FUNCTION CheckDivReal : REAL
VAR_INPUT
  divisor:REAL;
END_VAR

Implementation section:

// This automatically generated code is a suggested implementation.
IF divisor = 0 THEN
  CheckDivReal:=1;
ELSE
  CheckDivReal:=divisor;
END_IF;

The DIV operator uses the output of the CheckDivReal function as a divisor. In the sample program below, CheckDivReal prevents division by 0 by changing the implicit value of the divisor d from "0" to 1 before the division operation is executed. Therefore, the division result is 799.

PROGRAM PLC_PRG
VAR
  erg:REAL;
  v1:REAL:=799;
  d:REAL:=0;
END_VAR
erg:= v1 / d;

See also
- § Chapter 1.3.1.9.22 “Using POU's for implicit checks” on page 262

POU 'CheckRangeSigned'

Function for monitoring the range limits of a subrange type of type DINT.

This monitoring function is responsible for the appropriate handling violations to range limits. Examples of reactions to violations include setting error flags and changing values. The functions are called implicitly when a value is assigned to a subrange type variable.

CAUTION!

To obtain the feature for monitoring functions, do not edit the declaration section. However, you are permitted to add local variables.

When the function is called, it receives the following input parameters:
- value: Value that should be assigned to the subrange type variables
- lower: Lower range limit
- upper: Upper range limit

The return value is the assignment value as long as it is within the valid range. If not, then either the upper or lower limit is returned, depending on which threshold was violated.

For example, the assignment i := 10*y is replaced implicitly by i := CheckRangeSigned(10*y, -4095, 4095);
If \( y \) is "1000", then "10*1000=10000" is not assigned to \( i \) like in the original code. Instead, the upper range limit of "4095" is assigned.

The same is true for \text{CheckRangeUnsigned} function.

\begin{center}
\begin{tabular}{|c|}
\hline
\textbf{NOTICE!} \\
If functions are not available, then the subrange is not checked for the respective variables at runtime. In this case, you can assign any value between \(-2147483648\) and \(+2147483648\) (or between 0 and \(4294967295\)) to a variable of subrange type \text{DINT/UDINT}. You can assign any value between 
\(-9223372036854775808\) and \(+9223372036854775807\) (or between 0 and 
\(18446744073709551615\)) to a variable of a subrange type \text{LINT/ULINT}. \\
\hline
\end{tabular}
\end{center}

\begin{center}
\begin{tabular}{|c|}
\hline
\textbf{CAUTION!} \\
Linking area monitoring functions can lead to endless loops. For example, an endless loop can occur if the counter variable of a \text{FOR} loop is a subrange type and the counting range for the loop exits the defined subrange. \\
\hline
\end{tabular}
\end{center}

\begin{center}
\begin{tabular}{|l|}
\hline
\textbf{Example of an endless loop:} \\
\text{VAR} \\
\hspace{1cm} ui : UINT (0..10000); \\
\hspace{1cm} ... \\
\hspace{1cm} END_VAR \\
\hspace{1cm} FOR ui:=0 TO 10000 DO \\
\hspace{2cm} ... \\
\hspace{1cm} END_FOR \\
The program never exits the \text{FOR} loop because the \text{CheckRangeSigned} monitoring function prevents \text{ui} from being set to a value greater than 10000. \\
\hline
\end{tabular}
\end{center}

\begin{center}
\begin{tabular}{|l|}
\hline
\textbf{Example for \text{CheckRangeSigned}} \\
The assignment of a value to a \text{DINT} variable of a signed subrange type is a condition for automatically calling the \text{CheckRangeSigned}. This function restricts the assignment value to the subrange as defined in the variables declaration. The default implementation of the function in \text{ST} is as follows: \\
\hspace{1cm} Declaration section: \\
\hspace{2cm} // This is automatically generated code: DO NOT EDIT \\
\hspace{2cm} FUNCTION CheckRangeSigned : DINT \\
\hspace{2cm} VAR_INPUT \\
\hspace{3cm} value, lower, upper: DINT; \\
\hspace{2cm} END_VAR \\
\hspace{1cm} Implementation: \\
\hspace{2cm} // This automatically generated code is a suggested implementation. \\
\hspace{2cm} IF (value < lower) THEN \\
\hspace{3cm} CheckRangeSigned := lower; \\
\hspace{2cm} ELSEIF (value > upper) THEN \\
\hspace{3cm} CheckRangeSigned := upper; \\
\hspace{2cm} ELSE \\
\hspace{3cm} CheckRangeSigned := value; \\
\hspace{2cm} END_VAR \\
\hline
\end{tabular}
\end{center}
POU 'CheckLRangeSigned'

Function for monitoring the range limits of a subrange type of type LINT.

For an implementation example of range monitoring, refer to the help page for the CheckRangeSigned function.

This monitoring function is responsible for the appropriate handling violations to range limits. Examples of reactions to violations include setting error flags and changing values. The functions are called implicitly when a value is assigned to a subrange type variable.

**CAUTION!**
To obtain the feature for monitoring functions, do not edit the declaration section. However, you are permitted to add local variables.

When the function is called, it receives the following input parameters:

- **value**: Value that should be assigned to the subrange type variables
- **lower**: Lower range limit
- **upper**: Upper range limit

The return value is the assignment value as long as it is within the valid range. If not, then either the upper or lower limit is returned, depending on which threshold was violated.

For example, the assignment `i := 10*y` is replaced implicitly by `i := CheckRangeSigned(10*y, -4095, 4095);`

If `y` is "1000", then "10*1000=10000" is not assigned to `i` like in the original code. Instead, the upper range limit of "4095" is assigned.

The same is true for CheckRangeUnsigned function.

**NOTICE!**
If functions are not available, then the subrange is not checked for the respective variables at runtime. In this case, you can assign any value between -2147483648 and +2147483648 (or between 0 and 4294967295) to a variable of subrange type DINT/UDINT. You can assign any value between -9223372036854775808 and +9223372036854775807 (or between 0 and 18446744073709551615) to a variable of a subrange type LINT/ULINT.

**CAUTION!**
Linking area monitoring functions can lead to endless loops. For example, an endless loop can occur if the counter variable of a FOR loop is a subrange type and the counting range for the loop exits the defined subrange.
The program never exits the FOR loop because the `CheckRangeSigned` monitoring function prevents `ui` from being set to a value greater than 10000.

See also
- § Chapter 1.3.1.9.22 “Using POU for implicit checks” on page 262
- § Chapter 1.3.1.25.1.29.6 “POU 'CheckRangeSigned’” on page 804

**POU 'CheckRangeUnsigned’**

Function for monitoring the range limits of a subrange type of type UDINT.

For an implementation example of range monitoring, refer to the help page for the `CheckRangeSigned` function.

This monitoring function is responsible for the appropriate handling violations to range limits. Examples of reactions to violations include setting error flags and changing values. The functions are called implicitly when a value is assigned to a subrange type variable.

**CAUTION!**

To obtain the feature for monitoring functions, do not edit the declaration section. However, you are permitted to add local variables.

When the function is called, it receives the following input parameters:
- **value**: Value that should be assigned to the subrange type variables
- **lower**: Lower range limit
- **upper**: Upper range limit

The return value is the assignment value as long as it is within the valid range. If not, then either the upper or lower limit is returned, depending on which threshold was violated.

For example, the assignment `i := 10*y` is replaced implicitly by `i := CheckRangeSigned(10*y, -4095, 4095);`

If `y` is "1000", then "10*1000=10000" is not assigned to `i` like in the original code. Instead, the upper range limit of "4095" is assigned.

The same is true for `CheckRangeUnsigned` function.

**NOTICE!**

If functions are not available, then the subrange is not checked for the respective variables at runtime. In this case, you can assign any value between -2147483648 and +2147483648 (or between 0 and 4294967295) to a variable of subrange type DINT/UDINT. You can assign any value between -9223372036854775808 and +9223372036854775807 (or between 0 and 18446744073709551615) to a variable of a subrange type LINT/ULINT.
CAUTION!
Linking area monitoring functions can lead to endless loops. For example, an endless loop can occur if the counter variable of a FOR loop is a subrange type and the counting range for the loop exits the defined subrange.

Example of an endless loop:

```plaintext
VAR
  ui : UINT (0..10000);
...
END_VAR

FOR ui:=0 TO 10000 DO
  ...
END_FOR
```

The program never exits the FOR loop because the `CheckRangeSigned` monitoring function prevents `ui` from being set to a value greater than 10000.

See also
- Chapter 1.3.1.9.22 “Using POU for implicit checks” on page 262
- Chapter 1.3.1.25.1.29.6 “POU ‘CheckRangeSigned’” on page 804

POU 'CheckLRangeUnsigned'

Function for monitoring the range limits of a subrange type of type ULINT.

For an implementation example of range monitoring, refer to the help page for the `CheckRangeSigned` function.

This monitoring function is responsible for the appropriate handling violations to range limits. Examples of reactions to violations include setting error flags and changing values. The functions are called implicitly when a value is assigned to a subrange type variable.

CAUTION!
To obtain the feature for monitoring functions, do not edit the declaration section. However, you are permitted to add local variables.

When the function is called, it receives the following input parameters:
- `value`: Value that should be assigned to the subrange type variables
- `lower`: Lower range limit
- `upper`: Upper range limit

The return value is the assignment value as long as it is within the valid range. If not, then either the upper or lower limit is returned, depending on which threshold was violated.

For example, the assignment `i := 10*y` is replaced implicitly by `i := CheckRangeSigned(10*y, -4095, 4095);`

If `y` is "1000", then "10*1000=10000" is not assigned to `i` like in the original code. Instead, the upper range limit of "4095" is assigned.

The same is true for `CheckRangeUnsigned` function.
NOTICE!
If functions are not available, then the subrange is not checked for the respective variables at runtime. In this case, you can assign any value between -2147483648 and +2147483648 (or between 0 and 4294967295) to a variable of subrange type DINT/UDINT. You can assign any value between -9223372036854775808 and +9223372036854775807 (or between 0 and 18446744073709551615) to a variable of a subrange type LINT/ULINT.

CAUTION!
Linking area monitoring functions can lead to endless loops. For example, an endless loop can occur if the counter variable of a FOR loop is a subrange type and the counting range for the loop exits the defined subrange.

Example of an endless loop:

```
VAR
  ui : UINT (0..10000);
...
END_VAR

FOR ui:=0 TO 10000 DO
  ...
END_FOR
```

The program never exits the FOR loop because the CheckRangeSigned monitoring function prevents ui from being set to a value greater than 10000.

See also
- % Chapter 1.3.1.9.22 “Using POUs for implicit checks” on page 262
- % Chapter 1.3.1.25.1.29.6 “POU ‘CheckRangeSigned’” on page 804

POU 'CheckPointer'

Monitoring function for pointers (Checkpoint)

Use this function to monitor the memory access of pointers in runtime mode. As opposed to other monitoring functions, a standard suggestion does not exist for the implementation of CheckPointer. You must define an implementation according to your own requirements.

The CheckPointer function should check whether the returned pointer references a valid memory address; monitors whether the orientation of the referenced memory range matches the variable type that the pointer refers to. If both conditions are fulfilled, then the pointer is returned. If not, then the function should complete an appropriate error handling.

CAUTION!
To obtain the feature for monitoring functions, do not edit the declaration section. However, you are permitted to add local variables.

NOTICE!
An implicit monitoring function call does not occur for THIS pointer and SUPER pointer.
NOTICE!

For compiler version 3.5.7.40 and later, the implicit check function "Checkpointer" also acts on REFERENCE variables in the same way as on pointer variables.

// Automatically generated code: DO NOT EDIT
FUNCTION CheckPointer : POINTER TO BYTE
VAR_INPUT
  ptToTest : POINTER TO BYTE;
  iSize : DINT;
  iGran : DINT;
  bWrite: BOOL;
END_VAR

// Not a standard implementation. Please add your own code here.
CheckPointer := ptToTest;

When the function is called, CODESYS provides the following input parameters:

- ptToTest: Target address of the pointer
- iSize: Size of the referenced variable; the data type of iSize must be compatible with INT and cover the dimensional scope of the variables.
- iGran: Granularity of the referenced size; this is the largest non-structured data type contained in the referenced variables; the data type of iGran must be compatible with INT
- bWrite: Access type (TRUE = write access, FALSE = read access); the data type of bWrite must be BOOL

When the result of the check is positive, the unchanged pointer is returned (ptToTest).

See also

- "Chapter 1.3.1.9.22 “Using POU for implicit checks” on page 262"

Object 'Project Settings'

Symbol: 📖

Function: This object contains the configuration of the project.

Call:

- “Project ➔ Project Settings”
- Double-click on the object in the device tree

CODESYS saves the project settings directly in the project. If, for example, you transfer a project to another system, the "Project Settings" object is also transferred with it without a project archive being required.

The project settings are valid project-wide and offer setting possibilities for various categories such as “AS” or “Users and Groups”. The available categories vary, depending on which software packages you have installed via the package manager.

See also

- "Chapter 1.3.1.3.6.2 “Making project settings” on page 134"
- "Chapter 1.3.1.25.2.8.1 “Command ‘Package Manager’” on page 949"
- "Chapter 1.3.1.25.3.11.1 ‘Dialog ‘Project Settings’ - ‘SFC’” on page 1059"
- "Chapter 1.3.1.25.3.11.2 ‘Dialog Box ‘Project Settings’ - ‘Users and Groups’” on page 1060"
Object 'Project Information'

Symbol: 🔄

Function: The object contains the properties, meta-information, and project information. With this, you can check the authorship and integrity of the project.

Call
- Double-click the object in the device tree
- Menu bar: “Project ➔ Project Information”

Requirement: CODESYS creates the object when you click “Project ➔ Project Information”, and the dialog opens.

CODESYS saves the project information directly in the project. For example, if you transfer a project to another system, then the “Project Information” object is also transferred. You do not need a project archive.

Tab ‘File’
The tab displays the properties of the project file and their attributes. You cannot edit these attributes. They correspond to the file properties of Windows Explorer.

Tab ‘Summary’
The tab contains general information and meta-information of the project file. CODESYS uses this information to create keys on the “Properties” tab. For example, if the name Company_A is specified in “Company”, then the Company key with the value Company_A is provided on the “Properties” tab.

NOTICE!
If you save your project as a library project, then you should pay attention to the guidelines for library developers (Library Development Summary).

For a library project, a “Company”, a “Title”, and a “Version” must be specified to install the library.

<table>
<thead>
<tr>
<th>“Company”</th>
<th>Name of the company (example: Company_A).</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Title”</td>
<td>Title of the project (example Automation_A).</td>
</tr>
<tr>
<td>“Version”</td>
<td>Version of the project (example: 0.0.0.1).</td>
</tr>
<tr>
<td>“Released”</td>
<td>✅: Activates protection from modification. Result: If you edit the project now, then a dialog prompt opens to confirm whether you really want to change the project. If you reply to this prompt one time by clicking “Yes”, then no additional prompts appear for more editing actions.</td>
</tr>
</tbody>
</table>
“Library Categories”
Categories of the library project, according to which you can sort in the “Library Repository” dialog. If no category is specified, then the category “Other” is assigned to the library.

The categories originate from one or more external description files in XML format. However, they can also originate from a library project that has already been created.

Requirement: The project is a library project.

The “Library Categories” dialog opens where you can add library categories.

“Default namespace”
If you do not define a standard namespace here, then the name of the library file is applied automatically as the namespace.

“Author”
Author of the project (example: Arnold Best).

“Description”
Example: For internal use only

“Library compatibility”
List box for setting the CODESYS version for which the currently open library project is compatible, including all later versions). The setting is relevant with regard to the certificate-based signing of compiled libraries which is possible only as of V3 SP15.

See also
● § Chapter 1.3.1.25.2.1.7 “Command ‘Save Project as Compiled Library’” on page 856
● § Chapter 1.3.1.19.1 “Information for library developers” on page 372

Table 87: Dialog “Library Categories”

<table>
<thead>
<tr>
<th>List of categories</th>
<th>List of the categories that are assigned to the library project. They can originate from several sources. After you specify all desired categories, click “OK” to confirm.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Button “Add”</td>
<td>The “From Description File” and “From Other Library” commands appear.</td>
</tr>
<tr>
<td>Button “Remove”</td>
<td>CODESYS removes the selected category.</td>
</tr>
<tr>
<td>Command “From Description File”</td>
<td>The “Select Description File” dialog opens for you select a description file (*.libcat.xml). The file contains command categories. When you click “Open”, CODESYS accepts the categories.</td>
</tr>
<tr>
<td>Command “From Other Library”</td>
<td>The “Select Library” dialog opens, where you select a library with command categories to be accepted. When you click “Open”, CODESYS accepts the categories.</td>
</tr>
<tr>
<td>Button “OK”</td>
<td>CODESYS provides the categories as project information and displays it in the “Library Categories” field.</td>
</tr>
</tbody>
</table>

See also
● § Chapter 1.3.4.1 “Guidelines for creating libraries” on page 1105
● § Chapter 1.3.1.25.2.8.2 “Command ‘Library Repository’” on page 951

Tab ‘Properties’
On this tab, you can define keys that you can control externally from user-specific programs.

NOTICE!
If you have opened a library project, then note the description of the relevant keys in the guidelines for library developers (Library Development Summary).

If you have opened a symbol library as a project, then the key VisuSymbolLibrary = TRUE must be defined. It identifies the library as a symbol library.
### Key
Name of the key. Specify any string of text for the new key, or select an existing key from the "Properties" table.

### Type
Data type of the key. Possible types: "Text", "Date", "Number", "Boolean", "Version".

### Value
Value of the key in permitted format
- "Text": Any character string
- "Date": Example: Friday, January 1, 2016 00:00:00. Minimum entry for the date: 1.1
- "Number": Integer in Integer32 format with or without a sign (example: -32500).
- "Boolean": True or False, capitalization irrelevant.
- "Version": Examples: 1.1, 1.0.1.0, maximum four figures.

### Add
Adds the new defined key to the "Properties" table.

### Modify
Saves the change made for the key selected in the "Properties" table.

### Remove
Removes the key selected in the "Properties" table.

### Properties
List of the properties that are defined as keys. CODESYS creates keys automatically for the information in the "Summary" tab.
Click a key to edit it in the input fields above the list.

See also
- Chapter 1.3.4.1 “Guidelines for creating libraries” on page 1105
- Using the Symbol Library in the Visualization
- Chapter 1.3.1.25.3.10.16 “Dialog 'Properties' - 'Image Pool'” on page 1057

### Tab 'Statistics'
The dialog provides statistical information about the number of objects of the individual type or use in the project.

### Tab 'Licensing'
The dialog is for the license protection of libraries.

CAUTION!
You can protect only compiled libraries in this way.

### Table 88: "Variables"

| "Activate dongle licensing" | ☑: The library requires a dongle with a license to use it. |
| "Firm code" | License information that must be supplied from the dongle for using the library later. |
| "Product code" | |
| "Activation URL" | |
| "Activation mail" | |

See also
- Chapter 1.3.4.1 “Guidelines for creating libraries” on page 1105
- Chapter 1.3.1.21 “Managing packages and licenses” on page 375
Tab 'Signing'  
When a certificate-signed library is created (possible as of CODESYS V3 SP15) and library compatibility with CODESYS < V3 SP15 is not set, the settings on this tab are disabled. In this case, the signing is done by means of a certificate that has to be assigned to the user profile in the security screen.

One method, which is not recommended but may be necessary in some case for compatibility with versions < V3 SP15, is the less secure signing of a library by means of a vendor-specific, one-time key in this dialog. Requirement: This key is available as a private key file (*.libpk) with an associated token. The user of the library also has to obtain this key in order to be able to check whether the last signing was actually performed by the library vendor.

<table>
<thead>
<tr>
<th>“Activate signing”</th>
<th>CODESYS signs the library project with a single-use, manufacturer-specific key.</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Private key file”</td>
<td>Location of the private key file *.libpk (example: D:\for lib developers only\mycomp_libkey.libpk).</td>
</tr>
</tbody>
</table>
| “Public key token” | Example: 427A5701DA3CF3CF  
Requirement: A private key file is specified, and CODESYS has read and entered the token. |
| “Create Private Key File” | CODESYS creates a new private key file. |

See also
- Chapter 1.3.1.25.2.3.18 “Command ‘Security Screen’” on page 891
- “Deprecated: Signing a library project (only for version compatibility with CODESYS < SP15)” on page 133

Options for creating blocks for accessing project information
### Automatically generate 'Project Information' POUs

Note: The functions that are created with this option can be used only if the runtime system supports the **WSTRING** data type. If this is not the case, then you can use the functions that were created automatically for the with the individual items of the project information, at least in the application for accessing properties. These functions are not registered in the runtime system.

- CODESYS creates POUs of the **FUNCTION** type in the “POUs” view, allowing programmatic access to the project properties in the application. The function blocks GetCompany, GetTitle and GetVersion are created for the properties “Company”, “Title” and “Version”.

The following function blocks are available for user-defined properties:

- GetBooleanProperty: BOOL (TRUE/FALSE)
- GetNumberProperty: DINT (numeric value)
- GetTextProperty: WSTRING (character string)
- GetTextProperty2: POINTER TO WSTRING (unlimited length)
- GetVersionProperty: VERSION (version number as character string)

Note: Do not activate this option for standard libraries, because this can cause problems on smaller systems due to the additional memory requirements.

<table>
<thead>
<tr>
<th>Automatically generate 'Library Information' POUs</th>
</tr>
</thead>
</table>

- CODESYS creates POUs of the **FUNCTION** type in the “POUs” view, allowing programmatic access to the project properties in the application.

For the “Version” and “Released” properties, the following functions are created:
- GetLibVersion (version number as character string),
- GetLibVersionNumber (version number as numeric value), and
- IsLibReleased (TRUE/FALSE).

Note: These functions are not registered in the runtime system. The option is available as an alternative solution is the runtime system does not support the **WSTRING** data type, therefore not permitting you to use the functions created with the “Automatically generate 'Project Information' POUs” option.

---

**See also**

- Chapter 1.3.1.3.6.1 “Retrieving and editing project information” on page 132
- Chapter 1.3.1.3.6.2 “Making project settings” on page 134

---

### Object 'Redundancy Configuration' and Redundancy Editor

<table>
<thead>
<tr>
<th>1.3.1.25.1.32.1</th>
<th>Tab 'Redundancy Status' .................................................. 816</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.3.1.25.1.32.2</td>
<td>Tab 'Applications' .......................................................... 817</td>
</tr>
<tr>
<td>1.3.1.25.1.32.3</td>
<td>Tab 'Files' ....................................................................... 817</td>
</tr>
<tr>
<td>1.3.1.25.1.32.4</td>
<td>Tab 'Log' ........................................................................ 817</td>
</tr>
<tr>
<td>1.3.1.25.1.32.5</td>
<td>Tab 'PLC Shell' .................................................................. 817</td>
</tr>
<tr>
<td>1.3.1.25.1.32.6</td>
<td>Tab 'Redundancy Settings' - 'Redundancy Link' ................. 817</td>
</tr>
<tr>
<td>1.3.1.25.1.32.7</td>
<td>Tab 'Redundancy Settings' - 'General' ............................. 818</td>
</tr>
<tr>
<td>1.3.1.25.1.32.8</td>
<td>Tab 'Registered Areas' ...................................................... 819</td>
</tr>
</tbody>
</table>

The “Redundancy Configuration” object contains the editor for the CODESYS Redundancy feature. Here you configure two configured and redundant PLC devices that execute the same application. At any one time, just one of the PLCs controls the execution, while the other is in standby mode. If the active PLC fails, then the other PLC takes control without any interruption in the processing.

You add the object below the application in the device tree by clicking “Add Object”.

Depending on the device description, the editor includes one or all of the following tabs:
● "Redundancy Status": Configuration of the communication between both PLCs. Status change and status bar. Synchronization:
● “Applications”: Information about the application on PLC2. Corresponds to the tab with the same name in the device editor of the primary controller.
● “Files”: File transfer with PLC2. Corresponds to the tab with the same name in the device editor of the primary controller.
● “Log”: Display of the log messages from PLC2. Corresponds to the tab with the same name in the device editor of the primary controller.
● “PLC Shell”: Transmission of PLC shell commands to PLC2. Corresponds to the tab with the same name in the device editor of the primary controller.
● “Redundancy Settings”:
  – “Connection”: IP addresses and ports of both PLC devices.
  – “General”: Timeout, task, available services, and fieldbus settings.
● “Registered Areas”: Definition of the data areas to be controlled by CODESYS Redundancy (programming objects, address ranges).

See also
● § Chapter 1.3.1.25.1.19.2 “Tab ‘Communication Settings’” on page 741
● § Chapter 1.3.1.17.1 “Configuring redundant PLC devices” on page 363
● § Chapter 1.3.1.25.2.4.1 “Command ‘Add Object’” on page 897

Tab 'Redundancy Status'

In this tab of the “Redundancy Configuration” dialog, you define the network path to the other PLC. PLC1 and PLC2 are displayed in a graphic with information about the current status of the PLC, about the gateway used, and about the device itself (name, type, version, address, and vendor).

At runtime, you can use the following buttons to change the status of the PLC devices or start synchronization, each depending on the situation.

| “Activate” | Requirement: Simulation mode.  
| Switches the PLC to “Active” status. |
| “Switch” | Switches the PLC from “Active” status to “Passive” status. |
| “Synchronize” | Synchronizes the passive PLC with the active PLC (values, data, addresses in the areas registered for redundancy). |
| “Simulation” | Switches the PLC from “Passive” status to “Simulation” status. |

The mode of each PLC is indicated by one of the following circle symbols in the lower right corner of the device graphic. They are also indicated as text in “Redundancy state” and in the value of eRedundancyState of a variable type RedundancyState (if used in the application):
### Status

<table>
<thead>
<tr>
<th>Status</th>
<th>Value of eRedundancyState</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unknown</td>
<td>No assignment; Displayed when there is no online connection to the controller.</td>
</tr>
</tbody>
</table>

See also
- Chapter 1.3.1.17 “Using redundant PLC devices, CODESYS Redundancy” on page 362

### Tab 'Applications'

This tab in the “Redundancy Configuration” dialog corresponds to the tab with the same name in the device editor of the PLC. It contains information about the applications located on the other PLC.

See also
- Chapter 1.3.1.25.1.19.4 “Tab 'Applications’” on page 747
- Chapter 1.3.1.17 “Using redundant PLC devices, CODESYS Redundancy” on page 362

### Tab 'Files'

This tab in the “Redundancy Configuration” dialog corresponds to the tab with the same name in the device editor of the PLC. It is used for exchanging files with the second PLC.

See also
- Chapter 1.3.1.17 “Using redundant PLC devices, CODESYS Redundancy” on page 362
- Chapter 1.3.1.25.1.19.7 “Tab 'Files’” on page 750

### Tab 'Log'

This tab in the “Redundancy Configuration” dialog corresponds to the tab with the same name in the device editor of the PLC. It is used for displaying log book messages from the second PLC.

See also
- Chapter 1.3.1.17 “Using redundant PLC devices, CODESYS Redundancy” on page 362
- Chapter 1.3.1.25.1.19.8 “Tab 'Log’” on page 750

### Tab 'PLC Shell'

This tab in the “Redundancy Configuration” dialog corresponds to the tab with the same name in the device editor of the PLC. It is used for retrieving information from the second PLC.

See also
- Chapter 1.3.1.17 “Using redundant PLC devices, CODESYS Redundancy” on page 362
- Chapter 1.3.1.25.1.19.10 “Tab 'PLC Shell’” on page 753

### Tab 'Redundancy Settings' - 'Redundancy Link'

This tab in the “Redundancy Configuration” dialog is used for configuring the connection between both redundant PLC devices. It is used for synchronizing and triggering status changes. If possible, use a separate port especially for this communication only.
Table 89: “Redundancy Link” / “1st Redundancy Link”

<table>
<thead>
<tr>
<th>“IP address PLC 1”</th>
<th>Example: 192.168.101.176</th>
</tr>
</thead>
<tbody>
<tr>
<td>“IP address PLC 2”</td>
<td>Example: 192.168.101.87</td>
</tr>
<tr>
<td>“Port”</td>
<td>Defines the IP port for the connection between both redundant PLC devices. Example: 1205</td>
</tr>
</tbody>
</table>

“Use two redundant links”  
☑️ An additional network connection for redundant control is available. If one network connection fails, then the second connection can be used.

Table 90: “2nd Redundancy Link”

<table>
<thead>
<tr>
<th>“IP address PLC 1”</th>
<th>Example: 192.168.101.178</th>
</tr>
</thead>
<tbody>
<tr>
<td>“IP address PLC 2”</td>
<td>Example: 192.168.101.21</td>
</tr>
<tr>
<td>“Port”</td>
<td>Defines the IP port for the second connection between both redundant PLC devices. Example: 1205</td>
</tr>
</tbody>
</table>

“Read”  
Clicking this button reads the current redundancy link settings for PLC1 into the dialog.

“Write”  
Clicking this button transmits the available redundancy link settings to both PLC devices.

See also
- Chapter 1.3.1.17 “Using redundant PLC devices, CODESYS Redundancy” on page 362

Tab 'Redundancy Settings' - 'General'
This tab in the “Redundancy Configuration” dialog is used for configuring watchdogs, specifying the used task, and desired services and settings of the fieldbus for redundant operation.

Table 91: “Application”

<table>
<thead>
<tr>
<th>“Timeout (ms)”</th>
<th>Time, when exceeded that the PLC switches to stand-alone mode, if it has not received a message from the second PLC. In this case, the task execution is delayed by this time interval. The setting is stored in the RTS configuration file. Recommendation: Set this timeout value higher than the maximum task jitter time plus the maximum transmission time for redundancy messages.</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Redundancy task name”</td>
<td>Task that is synchronized between both PLC devices. Only one task can be synchronized. The setting must be stored in the RTS configuration file (button or manually).</td>
</tr>
</tbody>
</table>

Table 92: “Services”

The CODESYS online services selected here are executed on both PLC devices simultaneously in the same cycle.

| “Force variable” | ☑️ The services for forcing variable values are executed on both PLCs. |
| "Write variable" | ☑️ The services for writing variable values are executed on both PLCs. |
Table 93: “Fieldbus Settings”

| “EtherCAT” | EtherCAT is used as the fieldbus.  
  ● “Source address (MAC) PLC1”: This corresponds to the EtherCAT master address.  
  ● “Source address (MAC) PLC2”: This must be different from the address above. It must correspond to the PLC2 adapter that is used for EtherCAT.  
The NIC settings for the adapter must be set on both PLCs. Automatically searching for the respective PLC is possible by means of the “Scan” button. |
| “Profibus” | Profibus is used as the fieldbus. |

| “Read” | Clicking this button reads the current redundancy link settings for PLC1 into the dialog. |
| “Write” | Clicking this button transmits the available redundancy link settings to both PLC devices. |

See also

- § Chapter 1.3.1.17 “Using redundant PLC devices, CODESYS Redundancy” on page 362

Tab 'Registered Areas'

This tab in the “Redundancy Configuration” dialog is used for defining the data area that should be controlled by the redundant configuration.

By default, all data areas are controlled by CODESYS Redundancy. They are transmitted from the active PLC to the passive PLC during the synchronization phase. However, if you exclude areas from the control as "private" or "local", then select them in the available dialog. Examples of private and local data: diagnostic data of a fieldbus, file handles, and other operating system handles that are valid for the local PC only.

Defining the data area is separated on two tabs:

- Tab “Variables”: All global variable lists and program POUs of the application are provided automatically. Deactivate the check boxes of those that should not be controlled redundantly. Also note the “Select all / Clear all” check box under the selection view.
- Tab “Addresses”: Here you can specify an area of direct addresses that should be controlled by CODESYS Redundancy. The “Add” button opens a dialog for defining the start and size of a “Register address range”. With the “Edit” and “Delete” buttons, you can change or remove existing entries.

NOTICE!

On a redundant system, be careful when using file handles and operating system handles, as they can lead to different data on each of the PLCs.
NOTICE!

The method of mapping a device input to an existing project variable is not recommended. If you do use it, then keep in mind that variables mapped in this way are not stored in the input or output data areas. This means that they are not included in the synchronization of the redundant PLCs.

See also
- Chapter 1.3.1.17 “Using redundant PLC devices, CODESYS Redundancy” on page 362
- “Linking a device input with an existing project variable (“mapping”)” on page 164

Object 'Recipe Manager'
Tab 'Storage'
The recipe manager provides functions for maintaining user-defined variable lists, known as recipe definitions. The recipe definitions can be stored in recipe files on the PLC.

<table>
<thead>
<tr>
<th>“Storage Type”</th>
<th>“Textual”: CODESYS saves the recipe in a readable Format with the configured columns and delimiters.</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Binary”: CODESYS saves the recipe in a non-readable binary format. This format requires less storage space.</td>
<td></td>
</tr>
<tr>
<td>Note: You can read binary recipes again only if you have not changed the variable lists.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>“File path”</th>
<th>&lt;directory name&gt;\</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example: AllRecipes\</td>
<td></td>
</tr>
<tr>
<td>Relative path on the runtime system The path must end with a slash (&quot;`).</td>
<td></td>
</tr>
<tr>
<td>The directory is created on the target system in the directory for the runtime files (PlcLogic).</td>
<td></td>
</tr>
<tr>
<td>Example of the file path in the runtime system: PlcLogic/AllRecipes</td>
<td></td>
</tr>
<tr>
<td>CODESYS saves a file in this directory for each recipe when downloading to the PLC. The requirement is that you select the option “Recipe management in the PLC”.</td>
<td></td>
</tr>
<tr>
<td>The files are loaded to the recipe manager each time the application is restarted.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>“File extension”</th>
<th>File extension for the recipe file in the format .&lt;file extension&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>The resulting default name for recipe files is in the form &lt;recipe&gt;.&lt;recipe definition&gt;.&lt;file extension&gt;.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>“Separator”</th>
<th>Delimiters between the individual values in the saved file</th>
</tr>
</thead>
</table>

| “Available Columns” | Defines which information is saved in which order in the recipe file |
| “Selected Columns” | |
| “Save as Default” | CODESYS uses the settings on the tab throughout the entire project for all other recipe managers. |

Tab 'Common'

| “Recipe management in the PLC” | Must be selected for the user program or visualization elements to load recipes in runtime mode. If you transfer recipes to the PLC exclusively via the CODESYS program interface, then you can clear this option. |
Table 94: “Save Recipe”

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
</table>
| “Save recipe changes to recipe files automatically” | When “Recipe management in the PLC” is selected, there is the following option for saving the recipe:  
☑️ We recommend this option because it helps the recipe manager operate “normally”. The recipe files on the PLC are updated automatically in runtime mode whenever a recipe is changed. |

Table 95: “Load Recipe”

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Download only for exact match of the variable list”</td>
<td>☒️: The recipe is only downloaded if the file on the PLC contains all variables from the variable list of the recipe definition of the application and these are sorted in the same order. Additional entries at the end are ignored. If the required match does not exist, then the error status ERR_RECIPE_MISMATCH is set (RecipeManCommands GetLastError).</td>
</tr>
<tr>
<td>“Download variables with matching names”</td>
<td>☒️: The recipe values are downloaded only for those variables that have the same name in the recipe definition of the application as in the recipe file on the PLC. If the variable lists differ in composition and sorting, then no error status is set. In this way, recipe files can also be downloaded if variables in the file or in the recipe definition have been deleted.</td>
</tr>
</tbody>
</table>

Table 96: “Write Recipe”

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Limit the variable to min/max when recipe value is out of the range”</td>
<td>☐️: If the recipe contains a value that is outside of the value range specified in the definition, then the defined minimum or maximum value is written to the PLC variable instead of this value.</td>
</tr>
<tr>
<td>“Do not write to a variable when the recipe value is out of the min/max range”</td>
<td>☑️: If the recipe contains a value that is outside of the value range specified in the definition, then no value is written to the PLC variable. It retains its current value.</td>
</tr>
</tbody>
</table>

Table 97: “Read Recipe”

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
</table>
| “Check recipe for changes” | Always use the function block RecipeManCommands from RecipeManagement.library to read recipes. Never call the method cyclically. This is because each call can be written to the file system, which is time-intensive and burdens the controller. For example, the number of write cycles is limited for a Raspberry Pi.  
☑️: With each method call, the current PLC variable values are first read into the recipe. Then the system checks whether the values have changed. Only if the values have changed is the recipe saved. This means that the recipe file is overwritten with the current recipes. This option can be used in order to update the recipe file in the local file system only if recipe values have changed on the PLC. However, it affects performance because it generates additional code for checking.  
☐️: With each method call, the current PLC variable values are first read into the recipe. Then the recipe is written to the recipe file in the local file system. Note: As the file system is written to each call, the controller can be very burdened. |
Table 98: Option “Save recipe changes to recipe files automatically” is activated.

<table>
<thead>
<tr>
<th>Menu Commands</th>
<th>Behavior of the recipes defined in the project</th>
<th>Behavior of the defined recipes at runtime</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Online ➔ Reset Warm”</td>
<td>The recipes of all recipe definitions are downloaded with the values from the current projects.</td>
<td>Dynamically generated recipes remain unchanged.</td>
</tr>
<tr>
<td>“Online ➔ Reset Cold”</td>
<td></td>
<td></td>
</tr>
<tr>
<td>“Online ➔ Download”</td>
<td></td>
<td></td>
</tr>
<tr>
<td>“Online ➔ Reset Origin”</td>
<td>The application is removed from the PLC. If a download is done again afterwards, then the recipes are restored as for an online reset warm.</td>
<td></td>
</tr>
</tbody>
</table>

Shut down and restart the PLC

After a restart, the recipes are downloaded again from the automatically created files. This will restore the same state as before shutdown.

“Online ➔ Online change”

The recipe values remain unchanged. In runtime mode, a recipe can be changed only via the function block command RecipeManCommands.

“Debug ➔ Stop”

The recipes remain unchanged when the PLC is stopped or started.

“Debug ➔ Start”

The recipe values remain unchanged. In runtime mode, a recipe can be changed only via the function block command RecipeManCommands.

Table 99: Option “Save recipe changes to recipe files automatically” is deactivated.

<table>
<thead>
<tr>
<th>Actions</th>
<th>Recipes defined in the project</th>
<th>Recipes defined in runtime mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Online ➔ Reset Warm”</td>
<td>The recipes of all recipe definitions are downloaded with the values from the current projects. However, these are set in the memory only. To save recipes to a file, you must execute the command “Save Recipe” explicitly.</td>
<td>Dynamically generated recipes are lost.</td>
</tr>
<tr>
<td>“Online ➔ Reset Cold”</td>
<td></td>
<td></td>
</tr>
<tr>
<td>“Online ➔ Download”</td>
<td></td>
<td></td>
</tr>
<tr>
<td>“Online ➔ Reset Origin”</td>
<td>The application is removed from the PLC. When a download is performed afterwards, the recipes are restored.</td>
<td>Dynamically generated recipes are lost.</td>
</tr>
</tbody>
</table>

Shut down and restart the PLC

After the restart, the recipes are downloaded again from the automatically created files. This will restore the same state as before shutdown.

“Online ➔ Online Change”

The recipe values remain unchanged. In runtime mode, a recipe can be changed only via the function block command RecipeManCommands.

“Debug ➔ Stop”

The recipes remain unchanged when the PLC is stopped or started.

“Debug ➔ Start”

The recipe values remain unchanged. In runtime mode, a recipe can be changed only via the function block command RecipeManCommands.

See also

- Chapter 1.3.1.13.2 “Changing values with recipes” on page 331
- Chapter 1.3.1.25.2.19.9 “Command ‘Read and Save Recipe’” on page 1018
- Chapter 1.3.1.25.1.34 “Object ‘Recipe Definition’” on page 823
- Method Calls of the 'Recipe Management' Library
Object 'Recipe Definition'

In the recipe definition (1), you define different data sets for the variables, which are termed recipes (2).

You can toggle the display of the recipe definition between the flat list view (3) and the structured view (4). In the structured view, CODESYS groups variables according to structure.

<table>
<thead>
<tr>
<th>“Type”</th>
<th>Entered automatically</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Name”</td>
<td>Optional</td>
</tr>
<tr>
<td>“Minimal Value”</td>
<td>If the variable value is less than the “Minimal Value” or greater than the “Maximal Value”, then CODESYS sets the value to the “Minimal Value” or “Maximal Value”.</td>
</tr>
<tr>
<td>“Maximal Value”</td>
<td></td>
</tr>
<tr>
<td>“Comment”</td>
<td>Additional information, for example the unit of the value.</td>
</tr>
<tr>
<td>“Current Value”</td>
<td>Current variable value; not shown in online mode</td>
</tr>
</tbody>
</table>

Table 100: “Additional commands in the context menu in the structured view”

| “Add Sibling” | Adds a sibling variable to the recipe definition. |
| “Add Child”   | Adds a child variable to the recipe definition. |

See also

- Chapter 1.3.1.13.2 “Changing values with recipes” on page 331

Object 'TextList'

This object is for the creation, management and translation of texts. It contains a table with texts, to which you can add new texts. A text that you have written here can be selected in a visualization in an element in the property “Dynamic texts”. During operation the visualization displays this text dynamically in the selected language.

If the object is assigned to an alarm group and is located under the object “Alarm Configuration”, CODESYS adds the texts of the alarm group to the table. You can also add texts.

<table>
<thead>
<tr>
<th>“ID”</th>
<th>Unambiguous identifier of the text</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Standard”</td>
<td>Source text as character string, for example Information A: %i possibilities . Add a line break by pressing the keyboard shortcut [Ctrl] + [Enter]. Double-click in the field in order to edit the text.</td>
</tr>
</tbody>
</table>

The table contains as many language columns as you have added. A language column is named with a language code that you entered when creating the column with the command “Insert Language”.

2020/12/10
**Object 'Symbol Configuration'**

You can use the symbol configuration for creating symbol descriptions for project variables. Click **"Project ➔ Add Object"** to add a symbol configuration object to the device tree. Then define specific presets. See dialog below: **"Add Symbol Configuration"**.

If you use a **"DeviceApplication"**, the following applies: The device application itself cannot have a symbol configuration. However, symbols from the device application are offered for selection in the symbol configurations of their child applications.

Double-click the **"Symbol Configuration"** object to open the symbol configuration editor.

**Dialog 'Add Symbol Configuration'**

**Function:** This dialog is used to define the defaults for a **"Symbol Configuration"** object.

**Call:** **"Project ➔ Add Object ➔ Symbol Configuration"** menu; context menu of the application object.

<table>
<thead>
<tr>
<th><strong>Include comments in XML</strong></th>
<th>Exports the symbol file with the comments assigned to the variables.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Support OPC UA features</strong></td>
<td>Note: Availability and editability of this option depend on the device.</td>
</tr>
<tr>
<td>✔️</td>
<td>When downloading the symbol configuration, additional information is also downloaded to the controller. The information below is necessary for operating the OPC UA server.</td>
</tr>
<tr>
<td></td>
<td>• Base types of inherited function blocks</td>
</tr>
<tr>
<td></td>
<td>• Contents of attributes that were assigned via compiler pragmas</td>
</tr>
<tr>
<td></td>
<td>• Scopes (example: VAR_INPUT, VAR_OUTPUT, VAR_IN_OUT)</td>
</tr>
<tr>
<td><strong>Add library placeholder in DeviceApplication (recommended, but may trigger download)</strong></td>
<td>This option is relevant when a <strong>&quot;DeviceApplication&quot;</strong> is used:</td>
</tr>
<tr>
<td></td>
<td>The library references for the symbol configuration libraries, which are available in the Library Manager below the application, are also inserted into the Library Manager of the device application. The advantage of this is that the library data must no longer be downloaded with the application to the controller. They are downloaded one time with the device application as the other global data. For more information, see the chapter on using a device application.</td>
</tr>
</tbody>
</table>
Table 101: “Client Side Data Layout”

<table>
<thead>
<tr>
<th>Setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Compatibility layout”</td>
<td>This setting is used for the compatibility of old projects. The data layout created for the client is matched as much as possible to the layout created internally by the compiler.</td>
</tr>
<tr>
<td>“Optimized layout”</td>
<td>Recommended for new projects. Calculates the output layout in optimized form detached from the internal compiler layout. Does not generate any gaps for unpublished elements and strictly fulfills the requirements for memory alignment of the data types. Requires compiler version 3.5.7.0 or later.</td>
</tr>
</tbody>
</table>

See also

- Chapter 1.3.1.16 “Using device applications” on page 358

Symbol configuration editor

The editor includes a table with selected variables and a menu bar for editing.

Table 102: Menu bar

<table>
<thead>
<tr>
<th>Button</th>
<th>Description</th>
</tr>
</thead>
</table>
| ![Folder View](folder_icon) “View” | You can use this button for activating and deactivating the following categories of variables used in the configuration editor:
  - ![Folder Unconfigured from Project](unconfigured_icon) “Unconfigured from Project”: Variables that have not been added to the symbol configuration, but are provided in the project.
  - ![Folder Unconfigured from Libraries](unconfigured_icon) “Unconfigured from Libraries”: Variables that have not been added to the symbol configuration, but are provided in the project.
  - ![Folder Symbols Exported via Attribute](unconfigured_icon) “Symbols Exported via Attribute”: This filter also lists the variables that have already been marked for export in the symbol file by means of the `{attribute 'symbol' := 'read'}` pragma. These symbols are marked in gray. The “Attribute” column shows which access rights are set by the pragma. |
| ![Folder Build](build_icon) “Build” | Compiles the project. Requirement for current preparation of variables in the configuration editor. |
## "Settings"

- "Support OPC UA Features":
  Note: Availability and editability of this option depend on the device.
  - When downloading the symbol configuration, additional information is also downloaded to the controller. The information below is necessary for operating the OPC UA server. This currently includes the following information:
    - Base types of inherited function blocks
    - Contents of attributes that were assigned via compiler pragmas
    - Scopes (example: VAR_INPUT, VAR_OUTPUT, VAR_IN_OUT)
- "Include Comments in XML"
  - Exports the symbol file with the comments assigned to the variables.
- "Include Node Flags in XML"
  - The namespace node flags provide additional information about the origin of a node in the namespace. The node flags always get included in the symbol table when OPC UA is activated. However, its inclusion in the XML file can be deactivated as some defective parsers have problems with it.
- "Configure Comments and Attributes"
  - Opens the "Comments and Attributes" dialog. Here you configure the details of what should be included in the symbol configuration and XML file with respect to comments and attributes.
- "Configure Synchronisation with IEC Tasks":
  - Opens the dialog "Properties - <device name>", "Options" tab.
  - This setting allows for the symbolic clients (e.g. visualizations or database links based on the PLCHandler) to have consistent read/write access synchronized with the IEC tasks. For a detailed description of this setting, see the "Setting: Configuring Synchronisation with IEC Tasks" section below.
  - Note: Variable access which is synchronous with the IEC tasks can increase the jitter for all IEC applications on this device. Synchronized consistent access can interrupt the real-time capability.
- Drop-down list for defining the data layout type for the client of the symbol configuration:
  - "Optimized Layout": Recommend for new projects. Calculates the output layout in optimized form detached from the internal compiler layout. Does not generate any gaps for unpublished elements and strictly fulfills the requirements for memory alignment of the data types. Requires compiler version 3.5.7.0 or later.
  - "Compatibility Layout": This setting is used for the compatibility of old projects. The data layout created for the client is matched as much as possible to the layout created internally by the compiler. Due to the configuration possibilities of the symbol configuration which have grown over time, problematic offsets can still result.
  - Causes of offsets:
    - Memory gaps due to internal pointers or references in function blocks and structure components that are not released for symbol configuration.
    - Memory gaps that occur differently in 32-bit and 64-bit systems depending on the data type, such as __XINT / __XWORD.
    - Fields that are at uneven addresses. Some clients are not set up for this.
    - Unintentional memory misalignment, which occurs when using the attributes 'pack_mode' or 'relative_offset'.
- "Use Empty Namespaces by Default (V2 Compatibility)": Required when using a CODESYS V2-compatible OPC server configuration.
  - Behavior same as in CODESYS V2.3.
    - Program variables are exported without an application name
      (Application.PLC_FRG.MyVar → PLC_FRG.MyVar)
    - Global variables are exported additionally without the GVL name
      (Application.GVL.MyGlobVar → .MyGlobVar)
**“Enable Direct I/O Access”:** This feature is potentially dangerous and **not intended for operation in production**. Activate only for error checking and tests, or when commissioning the machinery (for example, for checking cables connections).

- [ ] In the symbol configuration, you can also use access to direct I/O addresses that correspond to IEC syntax (for example, "%IX0.0"). Access to input addresses (I) is read-only*. Access to output addresses (Q) and memory addresses (M) can be read-write.

*Information: In simulation mode, write access to symbols is also possible for input addresses.

Because external clients for protocols such as OPC or OPC UA do not always support IEC syntax for direct addresses, access is also provided using an array syntax in the namespace __MIO of the implicit code. For example, you can also access __MIO.MIO_IX[2].x3 instead of %IX2.3.

However, the symbols for array access are hidden in browsers because some clients cannot handle the large number of nodes (several thousand depending on the size of the I/O ranges).

**“Support Calls of Functions, FBs, Methods and Programs”:**

- Note: Availability and editability of this option depend on the device.

- [ ] The access rights “execute” can be set in the symbol table for symbols of POUs of type function, function block, method, or program. The “Support OPC UA Features” option also has to be selected in the “Settings”.

**“Include Call Information in XML”:**

- [ ] The information about called functions, function blocks, methods, or programs is also listed in the XML file of the symbol configuration. The option is enabled only if the option “Support Calls of Functions, FBs, Methods and Programs” is supported by the device.

**“Enable Symbol Sets”:**

- [ ] A toolbar with buttons and a drop-down list is displayed above the symbol table. You can use this to configure symbol sets for client-specific assignment of access rights to the controller. See "Toolbar for symbol set configuration" below.

**Download**

If you use a device that supports its own application file for the symbol configuration, then this button is also available in the toolbar. If you change the symbol configuration in online mode, then you can load the new <application name>._symbols file immediately to the PLC.

**Tools**

**“Save the XSD Scheme File”:** This command opens the standard dialog for saving a file in the file system. With this command, you can prepare the XSD format of the symbol file, for example for use in external programs.

**“Add Library Placeholder to Device Application (Recommended, but May Trigger a Download)”**: This option is relevant when a “Device Application” is used. The library references for the symbol configuration libraries, which are available in the Library Manager below the application, are also inserted into the Library Manager of the device application. For more information, see the description of the same-named option in the “Add Symbol Configuration” dialog.
Table 103: Symbol table

<table>
<thead>
<tr>
<th>“Access Rights”</th>
<th>You can change the access rights for a symbol by clicking the symbol in the “Access Rights” column. Icons for access rights (in ascending order)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>● ☒ Read only</td>
</tr>
<tr>
<td></td>
<td>● ☐ Write only</td>
</tr>
<tr>
<td></td>
<td>● ☑ Read and write</td>
</tr>
<tr>
<td></td>
<td>● ☐ : Execute</td>
</tr>
<tr>
<td></td>
<td>This permission allow for execute access to functions, function blocks, methods, and programs.</td>
</tr>
<tr>
<td></td>
<td>Requirements for the assignment: The device provides the options “Support calls of Functions, FBs, Methods and Programs” and “Support OPC UA Features”. Both options are activated in the “Settings”.</td>
</tr>
<tr>
<td></td>
<td>Note: In case the controller has a user management, you can use symbol sets to define client-specific access rights to the same symbols.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>“Maximal”</th>
<th>Maximum access rights for this symbol</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Attribute”</td>
<td>If the access right was assigned by attribute, then a corresponding icon is displayed here.</td>
</tr>
<tr>
<td>“Type”</td>
<td>Alias data types are also displayed in CODESYS V3.5 SP6 and later. Example: MY_INT : INT for a variable declared with the data type MY_INT (type INT).</td>
</tr>
<tr>
<td>“Members”</td>
<td>You can add variables of a structured data type also by selecting a check box for symbol configuration in the “Symbols” column. This causes CODESYS to export all member variable symbols. However, in the “Members” column, you can click the ellipsis button (…) to select only specific structural components. Note: This selection applies to all instances of this data type for which symbols are exported. If a member of a structured type cannot be selected, then an asterisk (∗) is displayed in the check boxes of the members to indicate that all exportable members of that type are exported.</td>
</tr>
</tbody>
</table>

Table 104: Toolbar for symbol set configuration

<table>
<thead>
<tr>
<th>“List box”</th>
<th>Already defined symbol sets</th>
</tr>
</thead>
<tbody>
<tr>
<td>✪ “Add a new symbol set”</td>
<td>Opens the “Add a New Symbol Set” dialog for specifying a name for this set</td>
</tr>
<tr>
<td>✯ “Add a duplicate from selected symbol set”</td>
<td>Opens the “Add a Duplicate from Selected Symbol Set” dialog. A copy is created for the set selected in the drop-down list. You can change the default name (&lt;group name&gt;_duplicate).</td>
</tr>
<tr>
<td>✤ “Rename the selected Symbol Set”</td>
<td>Opens the “Rename the Selected Symbol Set” dialog for specifying another name for the set selected in drop-down list.</td>
</tr>
<tr>
<td>✏ “Delete selected Symbol Set”</td>
<td>Opens a dialog prompting whether or not the symbol set selected in the drop-down list should be deleted.</td>
</tr>
<tr>
<td>“Configure Symbol Rights”</td>
<td>Opens the “Symbol Rights” tab of the device editor. When logged in there, you can assign different access rights for each user group (client) to the symbol set selected in the drop-down list.</td>
</tr>
</tbody>
</table>

See also
- ☆ Chapter 1.3.1.25.1.19.15 “Tab 'Symbol Rights'” on page 766
### Table 105: “Symbol Table Contents”

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
</table>
| “Enable extended OPC UA information”       | Note: Availability and editability of this option depend on the device. Additional information that can be evaluated by OPC UA servers is included in the symbol table. This includes inheritance information of user-defined data types and the namespace node flags. Additional information, such as comments and attributes, can also be included if the OPC UA setting is active. When the OPC UA setting is enabled, attributes are included in the symbol table according to the following rule:  
  - In compiler versions V3.5.5.0 to V3.5.7.X, all attributes are included according to the setting “Select simple names”.  
  - In compiler version V3.5.8.X, all attributes are included according to the setting “Include all attributes”.  
  - In compiler version V3.5.9.0 and later, you can customize the attributes that are included. |
| “Include comments”                          | Requirement: “Enable extended OPC UA information” is activated. Comments and attributes are also saved in the symbol table.                                                                                       |
| “Include attributes”                        | ✔️: Comments and attributes are also saved in the symbol table.                                                                                                                                              |
| “Also include comments and attributes for type nodes” | Requirement: “Include comments” is activated.  
  ✔️: The information for type nodes is also included (user-defined types, such as `STRUCT` and `ENUM` elements).  
  ☐: Only directly exported variables have comments and attributes. |

### Table 106: “XML Symbol File Contents”

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Include namespace node flags”</td>
<td>✔️: The namespace node flags provide additional information about the origin of a node in the namespace. The node flags always in the symbol table when OPC UA is activated. However, its inclusion in the XML file can be deactivated as some defective parsers have problems with it.</td>
</tr>
<tr>
<td>“Include comments”</td>
<td>✔️: Comments can also be saved in the XML file. In compiler versions V3.5.5.x to V3.5.8.0, this includes the setting “Prefer docu comments”.</td>
</tr>
<tr>
<td>“Include attributes”</td>
<td>✔️: Attributes can also be saved in the symbol file.</td>
</tr>
</tbody>
</table>
| “Also include comments and attributes for type nodes” | Requirement: “Include comments” is activated.  
  ✔️: The information for type nodes is also included (user-defined types, such as `STRUCT` and `ENUM` elements).  
  ☐: Only directly exported variables have comments and attributes. |

### Table 107: “Select Comments”

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
</table>
| Requirement: “Include comments” is activated. | The options determines the comments that are saved in the symbol configuration.  
  “Include docu comments”  
  “Include normal comments”  
  “Always include both types of comments”  
  “Prefer docu comments, fallback to normal ones”  
  “Prefer normal comments, fallback to docu comments” |
Table 108: “Filter Attributes (CaseInsensitive)”

<table>
<thead>
<tr>
<th>Requirement: “Include attributes” is activated.</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Include all attributes”</td>
</tr>
<tr>
<td>“Include attributes starting with”</td>
</tr>
<tr>
<td>“Filter attributes with regular expression”</td>
</tr>
<tr>
<td>“Match simple identifiers”</td>
</tr>
<tr>
<td>Defines the attributes that are saved in the symbol configuration.</td>
</tr>
<tr>
<td>Exists primarily due to the backward compatibility to older versions in order to emulate the old behavior.</td>
</tr>
</tbody>
</table>

Setting: Configure synchronization with IEC tasks

For synchronously consistent access, the symbolic client waits in the runtime system when processing a read or write request until a time is found when no IEC task is executed. When this gap is detected, restarting the IEC tasks is prevented until all values of the variable list have been copied. Then the IEC tasks are planned again as usual. Synchronized access can cause a delayed starting of IEC tasks, which is shown as increased jitter. As all applications in the runtime system are managed by a common scheduler, this potential impairment of the real-time behavior affects all applications on the device. All applications of the device are affected, regardless of whether or not they include a symbol configuration or they have been downloaded to the controller from one or more CODESYS projects. Therefore, the runtime system permits synchronized consistent access only if this it allows all applications that are downloaded to the controller at the time of access.

The setting is located in the editor of the symbol configuration of the “Settings” menu. In addition, the setting is also located in the context menu of the controller when you click the “Properties” command and then select the “Options” tab in the opened dialog.

For applications without symbol configuration, the setting can only be found in the properties dialog.

NOTICE!

After changing the setting, all applications downloaded to the device by means of a download or online change must be reloaded and all boot applications updated.

In which cases is synchronized consistent access necessary?

As a rule, there is no need for consistent values for visualized values because it is mostly irrelevant from which IEC task cycle the changed values originate. It is completely irrelevant for seldom changed values. Even when writing there are almost no hard consistency demands because typically the machine must be in a kind of standby mode (for example when writing recipes) in which there is no direct access to the values written as recipes.

In contrast, consistent values are particularly necessary for database links to save production data. For clocked machines, however, these values must be synchronous with the production timing (one value set per produced product) and not consistent with reference to one or more IEC tasks. With reference to the machine clocking, the consistency must be already ensured by the IEC application. For this purpose, the values that arise during a production cycle are typically collected in a global variable list. At the end of the cycle, the symbolic client is notified by means of an additional variable (BOOL or counter) that the machine cycle has ended and the values are valid. Now the client has the chance to archive the values from the production cycle. Depending on necessity, the successful reading can also be displayed in the opposite direction by means of a released variable, so that the production can also be halted in case the production data cannot be archived. Synchronized consistent access is not necessary and helpful for this use case because the synchronization takes place at the application level.
In contrast, synchronized consistent access by symbolic clients is typically applied in the process industry with continuously running systems without production clocking when, for example when process values are written consistently and cyclically in a fixed time frame of 60s. This can take place either by synchronization on the application level similar to clocked machines (see above) or by synchronization of the synchronized consistent symbolic access. The advantage of the latter is that no logic has to be implemented in the IEC program and access is controlled entirely by the client.

**CAUTION!**

Due to the increased jitter, the synchronized consistent monitoring is not suitable for motion or real-time critical applications. For these reasons, synchronized consistent access should be released and used only if it is absolutely necessary.

If a client uses synchronous consistent access released by this setting, then it has and effect on the client. Depending on the scheduler of the runtime system, the response time can jitter more here for read/write access because the system might still have to wait for an execution gap of the IEC tasks. Read and/or write access can still fail when IEC tasks run for a long time (in the range of several 100 ms) or the CPU load is close to 100% for an extended period of time with one or more IEC tasks (in the range of several 100 ms). Therefore, the availability of the values also depends on the load of the controller by the IEC application.

Moreover, the client can minimize the effects on itself and on the runtime system if it observes the following in the definition of the variable lists to be read or written:

- Synchronized consistent access only to those variables that are absolutely and consistently required.
- Separate variable lists for variables that have to be consistent and for variables that could be inconsistent.
- Divide variable lists with several consistent variables into several smaller lists.
- Select read intervals for cyclic reading of values as large as possible.

Entries marked in red in the symbol table show variables that they are configured for export to the symbol file but are currently invalid in the application. The cause for this can be that the declaration has been removed from the block.

In version 3.5.8.0 and later, a warning appears in the editor if variables that have configured symbols are not used in the IEC code or are not mapped in the case of I/O variables. In addition, the compiler indicates variables that are referenced from outdated library versions in the symbol configuration.

**NOTICE!**

Object variables that are not used in the program code remain uncompiled by default and are therefore not available in the symbol configuration. However, CODESYS provides variables from uncompiled objects in the symbol configuration when one of the following conditions is met:

- The “Link always” POU property is selected.
- The `attribute 'linkalways'` pragma is used.

See also

- Chapter 1.3.1.10 “Working with control networks” on page 270
- Chapter 1.3.1.25.3.10.18 “Dialog ‘Properties’ - ‘Options’” on page 1058
- Chapter 1.3.1.25.3.10.4 “Dialog ‘Properties’ - ‘Build’” on page 1048
- Chapter 1.3.1.24.6.2.20 “Attribute ‘linkalways’” on page 618
Example of data layout types

The following examples from an IEC application will show how gaps can result in the client-side memory layout caused by unpublished symbols, internal "invisible" pointers, or a "pack mode" definition in the device description. With the "Optimized Layout" setting, the gaps are avoided. The symbol file contains different information about the size and offset of memory locations, depending on the selected layout setting.

Example: Large structure

// Example of a big structure, where not all members get published :

```plaintext
STRUCT
    {attribute 'symbol':='readwrite'}
    PublicNumber : INT;

    {attribute 'symbol':='none'}
    InternalData : ARRAY[0..100] OF BYTE;

    {attribute 'symbol':='readwrite'}
    SecondNumber : INT;

    {attribute 'symbol':='none'}
    MoreData : ARRAY[0..100] OF BYTE;
END_STRUCT
END_TYPE
```

Resulting entries in the symbol file; pay attention to size and byteoffset:

```xml
<TypeUserDef name="T_LargeStructure" size="208" nativesize="208" typeclass="Userdef" pouclass="STRUCTURE" iecname="LargeStructure">
    <UserDefElement iecname="PublicNumber" type="T_INT" byteoffset="0" vartype="VAR" />
    <UserDefElement iecname="SecondNumber" type="T_INT" byteoffset="104" vartype="VAR" />
</TypeUserDef>
```

```
<TypeUserDef name="T_LargeStructure" size="4" nativesize="208" typeclass="Userdef" pouclass="STRUCTURE" iecname="LargeStructure">
    <UserDefElement iecname="PublicNumber" type="T_INT" byteoffset="0" vartype="VAR" />
    <UserDefElement iecname="SecondNumber" type="T_INT" byteoffset="2" vartype="VAR" />
</TypeUserDef>
```

Symbol file, large structure, compatibility layout option

Symbol file, large structure, optimized layout option

Example: Structure with uneven addresses

// The following mechanisms can cause memory misalignment:
// - {attribute 'relative_offset':='...'} at a member
// - {attribute 'pack_mode':='...'} at a structure declaration
// - target setting 'memory-layout\pack-mode' in the device description

```plaintext
{attribute 'pack_mode':='1'}
TYPE UnevenAddresses:
    STRUCT
        {attribute 'relative_offset':='3'}
        {attribute 'symbol':='readwrite'}
        PublicNumber : INT;
```
Resulting entries in the symbol file; pay attention to size and byteoffset:

```xml
<TypeUserDef name="T_UnevenAddresses" size="13" nativesize="13" typeclass="Userdef" pouclass="STRUCTURE" iecname="UnevenAddresses">
  <UserDefElement iecname="PublicNumber" type="T_INT" byteoffset="3" vartype="VAR" />
  <UserDefElement iecname="PublicValue" type="T_LREAL" byteoffset="5" vartype="VAR" />
</TypeUserDef>

<TypeUserDef name="T_UnevenAddresses" size="16" nativesize="13" typeclass="Userdef" pouclass="STRUCTURE" iecname="UnevenAddresses">
  <UserDefElement iecname="PublicNumber" type="T_INT" byteoffset="0" vartype="VAR" />
  <UserDefElement iecname="PublicValue" type="T_LREAL" byteoffset="8" vartype="VAR" />
</TypeUserDef>
```

// Each POU contains some implicit variables, which do not get published. Depending on the data type these might cause memory gaps of different sizes.

```xml
<TypeUserDef name="T_POU" size="24" nativesize="24" typeclass="Userdef" pouclass="FUNCTION_BLOCK" iecname="POU">
  <UserDefElement iecname="in" type="T_INT" byteoffset="16" vartype="VAR_INPUT" />
  <UserDefElement iecname="out" type="T_INT" byteoffset="18" vartype="VAR_OUTPUT" />
</TypeUserDef>

<TypeUserDef name="T_POU" size="4" nativesize="24" typeclass="Userdef" pouclass="FUNCTION_BLOCK" iecname="POU">
  <UserDefElement iecname="in" type="T_INT" byteoffset="0" vartype="VAR_INPUT" />
</TypeUserDef>
```

Example:

**Function block**

Each POU contains some implicit variables, which do not get published. If it is a data type such as `__XWORD`, then different sizes of memory gaps result in the client-side data layout, depending on whether the system is 64-bit or 32-bit.

Resulting entries in the symbol file for 64-bit and 32-bit; pay attention to size and byteoffset:
Object 'Task Configuration'

Symbol: 📖

The object is used to define and display the basic settings for the task configuration.

The "Task Configuration" object must be included exactly one time in each application.

"Task Configuration" tabs and functions

- "Properties": Display of the basic settings
- "System Events": Linking of POU calls with system events
- "Monitor": Display of the status and current statistics for the cycles times in online mode
- "Variable Usage": Overview of the tasks that access the variables and how they do it
- "Task Groups": Definitions of the tasks groups and their assignment to CPUs
- "CPU Load": Graphical representation of the CPU load in online mode

See also

- Chapter 1.3.1.9.17.1 “Creating a task configuration” on page 240
- Chapter 1.3.1.25.1.37.1 “Tab ‘Properties’” on page 835
- Chapter 1.3.1.25.1.37.2 “Tab ‘System Events’” on page 835
- Chapter 1.3.1.25.1.37.3 “Tab ‘Monitor’” on page 837
- Chapter 1.3.1.25.1.37.4 “Tab ‘Variable Usage’” on page 838
- Chapter 1.3.1.25.1.37.5 “Tab ‘Task Groups’” on page 838
- Chapter 1.3.1.25.1.37.6 “Tab ‘CPU Load’” on page 839
- Chapter 1.3.1.25.1.38 “Object ‘Task’” on page 839
Tab 'Properties'

Object: “Task Configuration”
In this tab, you define the basic settings of the task configuration as predefined by the target system, such as the maximum values for tasks and watchdog parameters.

Tab 'System Events'

Object: “Task Configuration”
In the “System Events” tab, you define which events call which functions and whether or not the configuration is currently activated. You use this tab when a system event (instead of a task) should call a project function.

<table>
<thead>
<tr>
<th>“Add Event Handler”</th>
<th>Opens the “Add Event Handler” dialog box</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Remove Event Handler”</td>
<td>Deletes the selected list assignment</td>
</tr>
<tr>
<td>“Event Info”</td>
<td>Shows information from the corresponding event library</td>
</tr>
<tr>
<td>“Open Event Function”</td>
<td>Opens the editor of the new function for the selected assignment You have selected the implementation language of the new function in the “Add Event Handler” dialog box.</td>
</tr>
</tbody>
</table>

Assignment of functions to call for events with: “Name”, “Description”, “Function to call”, and “Active” (activate/deactivate configuration).

Table 109: “Add Event Handler”

Adds a new assignment "Event – Function to call" to the list

| “Event” | The list of choices depends on the target device. CODESYS marks unavailable events with a red symbol in front of the name.  
A list of all possible system events is located at the end of this section. |
| “Function to call” | Function name (“POU”, type “FUNCTION”)  
You must enter the name of the new function. CODESYS inserts the function to the device tree after you confirm the dialog box. |
| “Scope” | ● “Application”: Function is available for the application  
● “POUs”: Function is available for the entire project |
| “Implementation language” | Implementation language for the new function |
| “Description” | Short description of the selected event |

Features in “Online Mode”
The list of assignments from called functions to events also includes the following information: “Event Status”, “Call Count”, and the “Online Reset” button.

| “Event Status” | 0: No error has occurred.  
Does not equal 0: error. You must consult the respective runtime system documentation. |
<p>| “Call Count” | Displays how often the event has occurred or the associated function has been called. |
| “Online Reset” | CODESYS reinitializes the event lists and resets the counter for the events/function calls. Incorrectly initialized events are displayed with a red status cell. |</p>
<table>
<thead>
<tr>
<th>Event</th>
<th>Description</th>
<th>Task</th>
<th>Debugging</th>
</tr>
</thead>
<tbody>
<tr>
<td>PrepareStart</td>
<td>Call before starting the application</td>
<td>Communication Task</td>
<td>no</td>
</tr>
<tr>
<td>StartDone</td>
<td>Call after starting the application</td>
<td>Communication Task</td>
<td>no</td>
</tr>
<tr>
<td>PrepareStop</td>
<td>Call before stopping the application</td>
<td>Communication Task</td>
<td>no</td>
</tr>
<tr>
<td>StopDone</td>
<td>Call after stopping the application</td>
<td>Communication Task</td>
<td>no</td>
</tr>
<tr>
<td>PrepareReset</td>
<td>Call before resetting the application</td>
<td>Communication Task</td>
<td>no</td>
</tr>
<tr>
<td>ResetDone</td>
<td>Call after resetting the application</td>
<td>Communication Task</td>
<td>no</td>
</tr>
<tr>
<td>PrepareOnline-Change</td>
<td>Call before online change of the application</td>
<td>Communication Task</td>
<td>no</td>
</tr>
<tr>
<td>OnlineChangeDone</td>
<td>Call after online change of the application</td>
<td>Communication Task</td>
<td>no</td>
</tr>
<tr>
<td>PrepareDownload</td>
<td>Call before downloading the application</td>
<td>Communication Task</td>
<td>no</td>
</tr>
<tr>
<td>DownloadDone</td>
<td>Call after downloading the application</td>
<td>Communication Task</td>
<td>no</td>
</tr>
<tr>
<td>PrepareDelete</td>
<td>Call before deleting the application</td>
<td>Communication Task</td>
<td>no</td>
</tr>
<tr>
<td>DeleteDone</td>
<td>Call after deleting the application</td>
<td>Communication Task</td>
<td>no</td>
</tr>
<tr>
<td>PrepareExit</td>
<td>Call before exiting the application</td>
<td>Communication Task</td>
<td>no</td>
</tr>
<tr>
<td>ExitDone</td>
<td>Call after exiting the application</td>
<td>Communication Task</td>
<td>no</td>
</tr>
<tr>
<td>CodeInitDone</td>
<td>Event is sent after Code Init. Called within the safe section task and only for online change (for example, the copy code is executed here for online change).</td>
<td>Communication Task</td>
<td>no</td>
</tr>
<tr>
<td>Exception</td>
<td>The event is sent if an exception has occurred in the context of an application.</td>
<td>Exception Handling Task (of runtime system) or task itself (if the runtime system does not support exception handling)</td>
<td>Depends on the task</td>
</tr>
<tr>
<td>Login</td>
<td>Logs in a client to this application</td>
<td>Communication Task</td>
<td>no</td>
</tr>
<tr>
<td>Logout</td>
<td>Logs out a client from this application</td>
<td>Communication Task</td>
<td>no</td>
</tr>
<tr>
<td>BeforeReadingInputs</td>
<td>Call before reading the inputs</td>
<td>IEC Task</td>
<td>yes</td>
</tr>
<tr>
<td>AfterReadingInputs</td>
<td>Call after reading the inputs</td>
<td>IEC Task</td>
<td>yes</td>
</tr>
<tr>
<td>Event</td>
<td>Description</td>
<td>Task</td>
<td>Debugging</td>
</tr>
<tr>
<td>---------------------</td>
<td>------------------------------------------------------------</td>
<td>-------------------------------------------</td>
<td>-----------</td>
</tr>
<tr>
<td>BeforeWritingOutputs</td>
<td>Call before writing the inputs</td>
<td>IEC Task</td>
<td>yes</td>
</tr>
<tr>
<td>AfterWritingOutputs</td>
<td>Call after writing the inputs</td>
<td>IEC Task</td>
<td>yes</td>
</tr>
<tr>
<td>DebugLoop</td>
<td>Event is sent in cycles to the debug loop if the IEC task stops at a breakpoint.</td>
<td>Communication Task</td>
<td>no</td>
</tr>
<tr>
<td>PrepareShutdown</td>
<td>Event is sent immediately before the runtime system is downloaded.</td>
<td>Runtime System Main Loop</td>
<td>no</td>
</tr>
<tr>
<td>PrepareExitComm</td>
<td>Event is sent during download before exiting the communication server.</td>
<td>Runtime System Main Loop</td>
<td>no</td>
</tr>
<tr>
<td>PrepareExitTasks</td>
<td>Event is sent during download before exiting all tasks.</td>
<td>Runtime System Main Loop</td>
<td>no</td>
</tr>
</tbody>
</table>

**Tab 'Monitor'**

Object: “Task Configuration”

This tab shows the following information in online mode: status for tasks in the task configuration and some current measurements of the cycles and cycle times. CODESYS updates the values in the same time interval as for the monitoring of values from the PLC.

Reset the displayed values by right-clicking the task and then clicking the “Reset” command in the context menu.

<table>
<thead>
<tr>
<th>“Task”</th>
<th>Task name (as defined in the task configuration)</th>
</tr>
</thead>
</table>
| “Status”                      | • “Not created”: The task has not been started since the last update (especially for event tasks).  
                              | • “Generated”: The task is recognized in the runtime system, but not yet in operation.  
                              | • “Valid”: The task is operating normally.  
                              | • “Exception”: The task has produced an exception status. |
| “IEC Cycle Count”             | Number of cycles executed since starting the application where the IEC code was executed (0 if the target system does not support the counter function) |
| “Cycle Count”                 | Number of executed cycles since logging in to the PLC. It depends on the target system whether cycles are also counted where the application is not running. In these cases, the “Cycle Count” may be greater than the “IEC Cycle Count”. |
| “Last Cycle Time (µs)”        | Last measured cycle time [µs] |
| “Average Cycle Time (µs)”     | Average cycle time over all cycles [µs] |
| “Max. Cycle Time (µs)”        | Maximum measured cycle time over all cycles [µs] |
| “Min. Cycle Time (µs)”        | Minimum measured cycle time over all cycles [µs] |
“Jitter (µs)”

Last measured jitter [µs]

Jitter is the difference between the time when the task should be started and the time when the task is actually started. For example, if the task should be started every 1000 µs and it actually starts after 1100 µs, then the jitter is 100 µs.

“Min. Jitter (µs)”

Minimum measured jitter [µs]

“Max. Jitter (µs)”

Maximum measured jitter [µs]

“Core”

Number of processor cores on which the task is currently running.

Example: 2

Note: The controller has a multi-core CPU.

Tab 'Variable Usage'

Object: “Task Configuration”

The "Variable Usage" tab provides an overview of all variables and their usage. There you can see the tasks where variables are accessed.

When using multicore, write access (w) to a variable should take place only in a task because otherwise it can cause inconsistencies.

In the context menu, you can hide individual tasks and show the cross-reference list to variables.

<table>
<thead>
<tr>
<th>“Variables”</th>
<th>Name of the variable</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Type”</td>
<td>Data type</td>
</tr>
<tr>
<td>“Number”</td>
<td>Number of tasks that access these variables.</td>
</tr>
<tr>
<td>“&lt;task name&gt;”</td>
<td>Access to the variable (r: read, w: write, rw: read/write)</td>
</tr>
</tbody>
</table>

See also

- Chapter 1.3.1.9.17.2 “Multicore” on page 241
- Chapter 1.3.1.9.17 “Task configuration” on page 239

Tab 'Task Groups'

Object: “Task configuration”

You define task groups in the “Task Groups” tab. Task groups can be distributed over the individual processor cores in multicore systems. Tasks in a task group always run on the same processor core.

<table>
<thead>
<tr>
<th>“Add Group”</th>
<th>The button adds a new task group named NewGroup_&lt;no&gt;.</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Remove Group”</td>
<td>Deletes the selected task group.</td>
</tr>
<tr>
<td>“Group Name”</td>
<td>The name can be changed by double-clicking in the field.</td>
</tr>
<tr>
<td>-------------</td>
<td>---------------------------------------------------</td>
</tr>
<tr>
<td>“Core”</td>
<td>Determines the processor core for process the tasks of this group.</td>
</tr>
<tr>
<td></td>
<td>● “Free floating”: All tasks are bound dynamically to different processor cores. The user does not have any influence over this. The operating system is responsible for the distribution.</td>
</tr>
<tr>
<td></td>
<td>● “Sequentially pinned”: All tasks are bound and fixed to different processor cores. The user does not have any influence over this.</td>
</tr>
<tr>
<td></td>
<td>● “Fixed pinned”: All tasks are bound to one processor core. By default, the runtime system determines the processor core.</td>
</tr>
<tr>
<td></td>
<td>● “&lt;core number&gt;”: Fixed defined processor core. If the processor core is not available, then an error message is issued.</td>
</tr>
</tbody>
</table>

See also
- "Chapter 1.3.1.9.17.2 “Multicore” on page 241"
- "Chapter 1.3.1.9.17 “Task configuration” on page 239"

Tab 'CPU Load'

Object: “Task Configuration”

The “CPU Load” tab is available in online mode for multicore devices only. The load of the individual CPUs is presented in the trace editor.

You open the trace configuration by double-clicking the legend in the window on the right side. Adding more variables is not possible here.

See also
- "Chapter 1.3.1.9.17.2 “Multicore” on page 241"
- "Displaying the CPU load with DeviceTrace objects in the CODESYS project (example)” on page 343"

Object 'Task'

Symbol: 🎉

In this object, you define the conditions for starting and calling the task.

You insert the object below “Task Configuration” in the device tree.

Tab 'Configuration'

Object: “Task”

<table>
<thead>
<tr>
<th>“Priority”</th>
<th>Possible values: 0..16, where 0 is the highest priority</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Task group”</td>
<td>Assigned task group. This assignment is shown in parentheses in the device tree. Task groups can be assigned to specific processor cores in multicore. The task group is shown in parentheses after the task in the device tree.</td>
</tr>
</tbody>
</table>

Table 110: “Type”

<table>
<thead>
<tr>
<th>“Cyclic”</th>
<th>CODESYS processes the task in cycles. The cycle time of the task is defined in the input field “Interval”.</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Event”</td>
<td>CODESYS starts processing the task as soon as the global variable defined in the input field “Event” contains a rising edge.</td>
</tr>
<tr>
<td>“Freewheeling”</td>
<td>CODESYS starts processing the task again automatically in a continuous loop at program start and at the end of a complete pass. Cycle time is not defined.</td>
</tr>
<tr>
<td>------------------------</td>
<td>----------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>“Status”</td>
<td>CODESYS starts task processing as soon as the variable defined in the “Event” input field yields the Boolean value TRUE.</td>
</tr>
<tr>
<td>“External”</td>
<td>CODESYS starts processing the task as soon as the event defined in the “Event” input field occurs. The target system determines which events are supported and offered in the dropdown list. (Not to be confused with system events).</td>
</tr>
<tr>
<td>“Interval”</td>
<td>Task-cycle time Required for the types “Cyclic” or “External Event” when the event requires a given time. Time interval after which the task should be restarted. If you enter a number here, then you can select the desired unit in the dropdown list after the input field. If you select “ms”, then an entry is automatically displayed in TIME format, for example t#200ms, as soon as the window is in focus again. You can also enter the task cycle time directly in TIME format. Entries in [µs] format are always displayed as a pure number.</td>
</tr>
<tr>
<td>“Parent task”</td>
<td>Only for projects with a “Device Application” You can use this task type in order to couple the processing of an application with a task of a “Device application”. In practice, the bus master task is managed primarily in the task configuration of the device application and a coupling with this allows for a harmonization with the timing of the bus master.</td>
</tr>
</tbody>
</table>

**NOTICE!**

For fieldbuses, a fixed cycle matrix is necessary to assure a determined behavior. Therefore, you should not use “Type” “Freewheeling” for a bus cycle task.

**NOTICE!**

Please note the following difference between the processing types “Status” and “Event”. If the given event yields TRUE, then the start condition of a task of type “Status” is fulfilled. In contrast, the start of a task of type “Event” requires a switch of the event from FALSE to TRUE. If the scanning rate of the task scheduler is too low, then the rising edge of the event can remain unnoticed.

**NOTICE!**

Please note: When setting the task cycle time, identify which bus system is currently being used. For example, the task cycle time in a CAN bus system must match the currently set transmission rate and the number of frames used in the bus. In addition, the times set for heartbeat, node guarding, and sync should always be a multiple of the task cycle time. If not, then CAN frames can be lost.
Table 11: "Watchdog"

Defines the time monitoring for a task. If the target system supports an advanced watchdog configuration, then the following settings may be predefined in the device description.

- Upper and lower limit
- Default watchdog time
- Time percentage

The default watchdog settings depend on the device.

"Enable"

The watchdog is active.

If the task exceeds the currently set “Time” of the watchdog, then the task is halted with an error status (exception). The application in whose task the error occurred and its child applications are also halted. In this way, all tasks of the affected applications are also halted. Then the currently defined “Sensitivity” is also taken into account. If you activate the option "Update I/Os" in the “PLC Settings” of the PLC, then CODESYS resets the outputs to the defined default values.

Possible cases:

- Several consecutive timeouts:
  - Sensitivity: 0, 1 - exception in cycle 1
  - Sensitivity: 2 - exception in cycle 2
  - Sensitivity: n - exception in cycle n
- Single timeout: Exception if the cycle time of the current cycle is longer than (time * sensitivity). Example: Time=t#10ms, Sensitivity=5 (i.e., exception as soon as the one-time task runs longer than 50 ms)

"Time (e.g. t#200ms)"

Watchdog time

Defines the watchdog for a task (with “Sensitivity”); description as for “Enable”.

Depending on the target system, the monitoring interval is given as a percentage of the task interval if possible. In this case, the drop-down list for the unit is disabled and displays “%”.

"Sensitivity"

Number

Defines the watchdog for a task (with the watchdog time); description as for “Enable”.

Using the functions from the library CmpIecTask.library, you can deactivate a watchdog for specific PLC cycles. This is useful for cycles that demand more time due to initialization.

Example

Deactivating/reactivating the watchdog:

```plaintext
hIecTask := RTS_IEC_HANDLE //Declaration of the variable hIecTask
hIecTask := IecTaskGetCurrent(0);
IecTaskDisableWatchdog(hIecTask); // Watchdog disabled
...
IecTaskEnableWatchdog(hIecTask); Watchdog enabled
```

List of POUs that control the task:

The calling order corresponds to the POU order in the list (from top to bottom).

"Add Call"

Defines a new program call
Object 'Trace'

Symbol: 

An object of type "Trace" is used for configuring and displaying application-specific trace data in one or more charts. At application runtime, value curves of trace variables, which you can monitor in the trace editor in CODESYS, are recorded on the controller. The requirement is that setting of a trace configuration, the transfer of the trace configuration to the controller, and the starting of the trace recording. The recorded data is transferred to the development system and displayed in diagrams according to the configuration. You can navigate through the data when tracing.

If the controller supports a trace manager, then you can use the 'DeviceTrace' object type in the trace manager to access all traces that are running on the controller.

Double clicking the trace object opens the trace editor. The corresponding toolbar contains the most important trace commands. The trace variable list shows the variable whose value curve is recorded.

- (1): Toolbar of the trace editor
- (2): Trace editor
At application runtime, the runtime system buffer of the trace component is filled with the recorded samples. The data is transferred to the development system and stored in its trace editor buffer. The trace editor accesses this data and visualizes it in diagrams as a graph over time. If you close the trace editor, the trace editor buffer will be freed up.

Use menu commands for controlling the trace. In addition, you can use menu commands, keyboard shortcuts, and mouse input for navigating through the data.

**See also**
- Chapter 1.3.1.13.3 “Operating the data recording” on page 341
- Chapter 1.3.1.13.3.5 “Navigating into trace data” on page 343

**Trace variable list**

The trace variable list provides an overview of the current trace configuration. When you double-click a trace variable, the dialog “Trace configuration” also opens with its variable settings.

<table>
<thead>
<tr>
<th><strong>Context menu in the trace variable list</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>“Add Variable”</strong></td>
</tr>
<tr>
<td><strong>“Visible”</strong></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>
"Display Settings" Opens the “Trace Configuration” dialog. Select a configuration item in the tree view “Trace record” or “Presentation (Diagrams)”.

“Configuration” Opens the “Trace Configuration” dialog. The “Variable Settings” are displayed on the right.

See also
- Chapter 1.3.1.25.3.15.2 “Dialog ‘Trace Configuration’” on page 1092

Navigating in the diagram

Table 113: Via mouse input

| User input via mouse                                      | Symbol mouse cursor during user input | Effect |&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbs
Keyboard shortcuts | Effect
---|---
[-] | Compressed time axis (like the symbol $\frac{\text{old}}{\text{new}}$).
[+] | Stretches the x-axis (like the symbol $\frac{\text{old}}{\text{new}}$).
[Ctrl]+[-] | Compresses the y-axis of the selected diagram.
[Ctrl]+[+] | Stretches the y-axis of the selected diagram.
[Tab] | Selects the next lower diagram.

Requirement: One or two trace cursors are activated.

- [Left arrow] Moves the black trace cursor.
- [Right arrow] Moves the gray trace cursor.

requirement: Two trace cursors are activated.

- [Shift]+[Left arrow] Moves the black trace cursor.
- [Shift]+[Right arrow] Moves the gray trace cursor.

See also

- Chapter 1.3.1.25.2.21.5 “Command ‘Cursor’” on page 1025
- Chapter 1.3.1.25.2.21.9 “Command ‘Mouse Zooming’” on page 1029
- Chapter 1.3.1.13 “Application at runtime” on page 322

Object 'DeviceTrace'

Symbol: 📊

A “DeviceTrace” object shows trace data in one or more diagrams, as does a “Trace” object. The difference is that a “DeviceTrace” directly accesses traces that are running on the controller. The object is inserted below the device in the device tree. Therefore there is no immediate dependency on the applications in the CODESYS project.

You can use the DeviceTrace for visualizing the processor load of a multicore controller.

For more information about the editor and its operation, refer to the help page for the “Trace” object.

See also

- Chapter 1.3.1.25.1.39 “Object ‘Trace’” on page 842
- “Runtime system component CmpTraceMgr, “Trace manager”” on page 335
- Chapter 1.3.1.13.3.4 “Accessing all traces on the controller” on page 342
- Chapter 1.3.1.9.17.2 “Multicore” on page 241

Object 'Trend Recording Manager'

Symbol 📈

A “Trend Recording Manager” object makes it possible to save data at runtime in a database for a long period of time. This data is recorded with the “CmpTraceMgr” runtime system component. In the device tree, this object is used as a node for trend recordings that are created below an application. It is available below an application only one time.
Object 'Trend Recording'

Symbol: 📊

A "TrendRecording" object is always located below a "Trend Recording Manager" and enables editing of the trace configuration. At runtime, CODESYS loads the configuration that is available to the CmpTraceMgr runtime system component. You can configure an application with any number of trend recordings.

**NOTICE!**

**Timeout for trend recording**

During a trend recording, it can happen that the application task triggers a timeout that is caught with an exception when transitioning from “Running” to “Stop”. Causes can be that file operations with the SQLite database are taking too long or that too many variables are being recorded. This usually happens on a target device with weak performance.

You can avoid the occurrence of an exception:
- Configure the trend recording with less memory demand so that the amount of data that is stored is adapted to the target system.
- Reduce the number of variables.

The editor includes the configuration for trend recording. The tree view shows the trend configuration and enables navigation there.

The top entry contains the trend name. When this entry is selected, the “Record Settings” appear next to it. An entry is located here for each variable that data was recorded continuously. When a variable is selected, the “Variable Settings” appear next to it.

### Table: Trend Recording Actions

<table>
<thead>
<tr>
<th>Action</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Add Variable&quot;</td>
<td>When you click the link, a new entry appears in the trend configuration with its blank configuration below the “Variable Settings” group.</td>
</tr>
<tr>
<td>&quot;Delete Variable&quot;</td>
<td>The selected variable is removed. Requirement: A variable is selected.</td>
</tr>
</tbody>
</table>

See also
- Command ‘Edit trend recording’ § Chapter 1.3.5.19.2.17 “Command ‘Edit Trend Recording’” on page 1577

**'Recording settings'**

The data is recorded in the runtime system component using the function that is also used for trace. The settings that appear here are the same. The options that are not required here are deactivated.

The settings that affect the trigger are deactivated. Only a trace configuration for a trace editor can configure triggering.

<table>
<thead>
<tr>
<th>Action</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Task&quot;</td>
<td>Task where data was recorded. Click the &quot;down&quot; symbol (▼) to open a drop-down list with all tasks available in the project. In general the trend recording runs in the same task as the main program. For example: MainTask</td>
</tr>
</tbody>
</table>

See also
- § Chapter 1.3.1.13.4 “Data recording with trend” on page 344
- § Chapter 1.3.1.25.2.4.1 “Command ‘Add Object’” on page 897
- § Chapter 1.3.1.25.1.42 “Object ‘Trend Recording’” on page 846
| "Record condition" | Recording condition for which the application records data:  
| | - IEC variable of type INT The condition is fulfilled for TRUE.  
| | - Bit access to an integer variable. The condition is fulfilled for 1.  
| | As read access to a property.  
| The contents of a pointer are not permitted. |
| "Comment" | Comment (example: Data recording of sensor A) |
| "Resolution" | Resolution that the application saves the time stamp  
| | Note: If the task where the trend object is executed has a cycle time of 1 ms or less, then you should set the resolution of the time stamp to “1 μs”. |
| "Trend storage" | The “Trend storage” dialog opens. |
| "Advanced" | The “Advanced Trend Settings” dialog opens. |

See also  
- § Chapter 1.3.1.25.3.16 “Dialog Box ‘Trend storage’” on page 1097  
- § Chapter 1.3.1.25.3.17 “Dialog Box ‘Advanced Trend Settings’” on page 1097

| "Variable settings" |
| "Variable" | Variable for recorded value. |
| | - IEC variable with valid type.  
| | - Property  
| | - Reference  
| | - Contents of the pointer  
| | - Array element of a valid data type  
| | - Enumeration of a valid data type  
| | Valid data types are all standard types, except STRING, WSTRING, and ARRAY. |
| "Parameter" | Parameter for recorded value. |
| | The “Input Assistant” dialog lists all valid system parameters in the “Parameters” category of the “Categories” tab. |
| | Click the symbol to toggle between “Variable” and “Parameter”. |
| "Attached y axis" | Y-axis of the trend diagram that displays the “Variable”. The drop-down list provides the standard y-axis and the configured y-axes.  
| | Requirement: This option is visible only when the “Trend” visualization element has configured additional y-axes in the “Edit Display Settings” dialog. |
| "Display variable name" | ☑: The visualization shows the name of the IEC variable in the trend diagram in runtime mode. Either alone or in parentheses after the “Description”  
| | ☐: The name of the IEC variable is shown and does not appear in parentheses after the “Description”.  
| | Requirement: If any text is typed in “Description”, then you can disable the option. |
**Description**

Text for the tooltip (example: Sensor A): When a visualization user focuses on the variable in the trend diagram, the visualization shows the text as a tooltip. The text is typed into the “GlobalTextList” object and can be localized there.

When the “Display Variable Name” property is activated, then the text is supplemented with the variable name in parentheses. Example: Sensor A (PLC_PRG.iSensor_A)

If “Description” does not contain any text, then “Display Variable Name” is enabled. The name is then alone without parentheses (for example, PLC_PRG.iSensor_A).

If a legend is assigned to the trend, then the trend variable is labeled in the legend and shown as the trend is configured here.

<table>
<thead>
<tr>
<th>Curve type</th>
<th>“Line”</th>
<th>“Area”</th>
</tr>
</thead>
</table>

**Graph color**

Color of the curve in the trend diagram

<table>
<thead>
<tr>
<th>Line type</th>
<th>“Line”: Values are linked to form a line.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>“Step”: Values are linked in the form of steps.</td>
</tr>
<tr>
<td></td>
<td>“None”: Values are not linked.</td>
</tr>
</tbody>
</table>

Requirement: The “Curve type” is “Line”.

<table>
<thead>
<tr>
<th>Filling type</th>
<th>“No filling”</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>“Plain color”</td>
</tr>
<tr>
<td></td>
<td>“Gradient”</td>
</tr>
</tbody>
</table>

Requirement: The “Curve type” is “Area”.

<table>
<thead>
<tr>
<th>Filling color</th>
<th>✅: The area is filled with the selected color.</th>
</tr>
</thead>
</table>

Requirement: The “Curve type” is “Area”.

<table>
<thead>
<tr>
<th>Transparency</th>
<th>Transparency value (0 to 255) for defining the transparency of the selected color.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example 255:</td>
<td>The color is opaque.</td>
</tr>
<tr>
<td>Example 0:</td>
<td>The color is completely transparent.</td>
</tr>
</tbody>
</table>

Requirement: The “Curve type” is “Area”.

<table>
<thead>
<tr>
<th>Line width</th>
<th>Value in pixels</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example: 1</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Line style</th>
<th>The display of the line is solid, dash, dot, dash-dot, or dash-dot-dot.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Point type</th>
<th>Display as scatter chart</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>“Dot”: Value as a dot.</td>
</tr>
<tr>
<td></td>
<td>“Cross”: Value as a cross.</td>
</tr>
<tr>
<td></td>
<td>“None”: No dot display</td>
</tr>
</tbody>
</table>

Hint: Select “None” for larger size data.

<table>
<thead>
<tr>
<th>Activate minimum warning</th>
<th>✅: Warning when below the lower limit.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Critical lower limit</td>
<td>If the variable value is below the limit, then the variables are displayed with the alert color in the trend diagram.</td>
</tr>
<tr>
<td>Warning minimum color</td>
<td>Warning color on falling below the limit</td>
</tr>
<tr>
<td>Activate maximum warning</td>
<td>✅: A warning is given if the upper limit is exceeded.</td>
</tr>
<tr>
<td>Critical upper limit</td>
<td>If the variable value exceeds the limit, then the variables are displayed with the alert color in the trend diagram.</td>
</tr>
<tr>
<td>Warning maximum color</td>
<td>Warning color on exceeding the limit</td>
</tr>
</tbody>
</table>
Object 'Trend Recording Task'

Symbol 

If you design a visualization with a “Trend” element, then CODESYS automatically extends the “Task Configuration” with a “Trend Recording Task”. The task is below an application one time at most and calls the VisuTrendStorageAccess.GlobalInstances.g_TrendRecordingManager.CyclicCall program to run the trend recording manager.

See also

- Chapter 1.3.1.25.2.4.1 “Command ‘Add Object’” on page 897
- Chapter 1.3.1.13.4 “Data recording with trend” on page 344

Object 'Unit Conversion'

Symbol 

A “Unit Conversion” object is used to define a conversion rule. The following table lists all defined conversion rules. You can edit a conversion rule in the input fields listed below the table.

Table of conversion rules

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Setting</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Name</strong></td>
<td>&lt;name&gt; : &lt;name&gt;_Impl is the name of the conversion rule. CODESYS automatically implements the entry as a function block &lt;name&gt;_Impl and instances it as &lt;name&gt;.</td>
<td></td>
</tr>
<tr>
<td><strong>Type</strong></td>
<td>Type of conversion rule</td>
<td></td>
</tr>
<tr>
<td></td>
<td>“Single scaling (offset)”: adds an offset to the input variable. Result := Input + Offset</td>
<td></td>
</tr>
<tr>
<td></td>
<td>“Single scaling (factor)”: multiplies the input variable by a factor. Result := Input * Factor</td>
<td></td>
</tr>
<tr>
<td></td>
<td>“Linear scaling 1 (factor and offset)”: converts the input variable with a factor and offset. Result := Input * Factor + Offset</td>
<td></td>
</tr>
<tr>
<td></td>
<td>“Linear scaling 2 (Base and target range)”: converts the input variable for the output value to be within a target range. CODESYS calculates the functional linear equation internally.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>“User defined conversion”: configures a user-defined conversion rule with IEC operators. The input variable is rValue.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>“Switchable conversion”: defines a conversion rule that CODESYS executes independent of any specified language or variable.</td>
<td></td>
</tr>
<tr>
<td><strong>Setting</strong></td>
<td>Displays the configured conversion rule.</td>
<td></td>
</tr>
</tbody>
</table>

If you design a visualization with a “Trend” element, then CODESYS automatically extends the “Task Configuration” with a “Trend Recording Task”. The task is below an application one time at most and calls the VisuTrendStorageAccess.GlobalInstances.g_TrendRecordingManager.CyclicCall program to run the trend recording manager.

See also

- Chapter 1.3.1.25.2.4.1 “Command ‘Add Object’” on page 897
- Chapter 1.3.1.13.4 “Data recording with trend” on page 344

Object 'Unit Conversion'

Symbol 

A “Unit Conversion” object is used to define a conversion rule. The following table lists all defined conversion rules. You can edit a conversion rule in the input fields listed below the table.

Table of conversion rules

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Setting</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Name</strong></td>
<td>&lt;name&gt; : &lt;name&gt;_Impl is the name of the conversion rule. CODESYS automatically implements the entry as a function block &lt;name&gt;_Impl and instances it as &lt;name&gt;.</td>
<td></td>
</tr>
<tr>
<td><strong>Type</strong></td>
<td>Type of conversion rule</td>
<td></td>
</tr>
<tr>
<td></td>
<td>“Single scaling (offset)”: adds an offset to the input variable. Result := Input + Offset</td>
<td></td>
</tr>
<tr>
<td></td>
<td>“Single scaling (factor)”: multiplies the input variable by a factor. Result := Input * Factor</td>
<td></td>
</tr>
<tr>
<td></td>
<td>“Linear scaling 1 (factor and offset)”: converts the input variable with a factor and offset. Result := Input * Factor + Offset</td>
<td></td>
</tr>
<tr>
<td></td>
<td>“Linear scaling 2 (Base and target range)”: converts the input variable for the output value to be within a target range. CODESYS calculates the functional linear equation internally.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>“User defined conversion”: configures a user-defined conversion rule with IEC operators. The input variable is rValue.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>“Switchable conversion”: defines a conversion rule that CODESYS executes independent of any specified language or variable.</td>
<td></td>
</tr>
<tr>
<td><strong>Setting</strong></td>
<td>Displays the configured conversion rule.</td>
<td></td>
</tr>
</tbody>
</table>
**“Condition”**
- "TRUE": CODESYS always executes the conversion.
- "Language": If the language in the visualization is the language defined here, then CODESYS executes the conversion. The current visualization language is located in the `VisuElems.CurrentLanguage` variable.
- "Variable": If the comparison is TRUE, then CODESYS executes the conversion rule. CODESYS can pass the comparison for a constant, variable, or IEC expression.
  
You can edit the comparison below the table in the “Condition Setting”.

**“Condition Setting”**
If you select "TRUE" as the "Condition", then the field is hidden.
If you configure "Language" as the "Condition", then the field shows the current configuration, for example en, de.
If you select "Variable" as the "Condition", then the field shows the current configuration, for example `PLC_PRG.bActual=PLC_PRG.bSet`.
You can edit the current condition setting below the table in the input fields for “Condition Setting”.

**Input field 'Single scaling (offset)'**
The input variable is added with an offset.

**"Offset"**
- as a number, including REAL
- as an IEC variable

**Input field 'Single scaling (factor)'**
The input variable is multiplied by the factor.

**"Factor"**
- as a number, including REAL
- as an IEC variable

**Input field 'Linear scaling 1 (factor and offset)'**
The input variable is converted with the linear equation defined below.

**"Factor"**
- as a number, including REAL
- as an IEC variable

**"Offset"**
- as a number, including REAL
- as an IEC variable

**Input field 'Linear scaling 2 (Base and target range)'**
The input variable is converted to be within a target range. CODESYS internally creates a linear equation from the following input values.

**"Base start value"**
Lowest possible value for the input variable.
- as a number, including REAL
- as an IEC variable

**"Base end value"**
Highest possible value for the input variable.
- as a number, including REAL
- as an IEC variable
"Target start value" | Lowest possible value for the output variable.  
| as a number, including REAL  
| as an IEC variable

"Target end value" | Highest possible value for the output variable.  
| as a number, including REAL  
| as an IEC variable

Example | Conversion of electric current from a 10-bit input signal to an amperage range of 4-20 mA

| "Base start value" | 0 |
| "Base end value" | 1024 |
| "Target start value" | 4.0 |
| "Target end value" | 20.0 |

Example | Conversion rule as mathematical function of rValue The input variable is rValue.

Example | Reverse function of the function defined in \"Convert\"

Example | Use this conversion rule when you want to apply a conversion that is language-specific or variable-dependent.

Example | The Conv_A_LanguageDependent conversion rule that defines which conversion rule is executed for the English or German language.

<table>
<thead>
<tr>
<th>&quot;Name&quot;</th>
<th>&quot;Type&quot;</th>
<th>&quot;Setting&quot;</th>
<th>&quot;Condition&quot;</th>
<th>&quot;Condition setting&quot;</th>
</tr>
</thead>
<tbody>
<tr>
<td>Conv_A_LanguageDependent</td>
<td>&quot;Switchable conversion&quot;</td>
<td>Conv_AInInch, Conv_AInMM</td>
<td>&quot;Language&quot;</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>&quot;Switchable conversion name&quot;</th>
<th>&quot;Condition setting&quot;</th>
</tr>
</thead>
<tbody>
<tr>
<td>Conv_AInInch</td>
<td>en</td>
</tr>
<tr>
<td>Conv_AInMM</td>
<td>de</td>
</tr>
</tbody>
</table>
See also
- § Chapter 1.3.1.9.19 “Unit conversion” on page 246

### 1.3.1.25.2 Menu Commands

By default the most important commands are already provided in the CODESYS user interface. If you want to customize the menu configuration individually, choose command “Tools ➔ Customize ➔ Menu”.

When you have installed any packages or add-ons, additional menus and commands might be available.

---

#### Menu 'File'

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.3.1.25.2.1.1</td>
<td>Command ‘New Project’</td>
<td>852</td>
</tr>
<tr>
<td>1.3.1.25.2.1.2</td>
<td>Command ‘Open Project’</td>
<td>853</td>
</tr>
<tr>
<td>1.3.1.25.2.1.3</td>
<td>Command ‘Close Project’</td>
<td>854</td>
</tr>
<tr>
<td>1.3.1.25.2.1.4</td>
<td>Command ‘Save project’</td>
<td>854</td>
</tr>
<tr>
<td>1.3.1.25.2.1.5</td>
<td>Command ‘Save Project as’</td>
<td>855</td>
</tr>
<tr>
<td>1.3.1.25.2.1.6</td>
<td>Command ‘Save Project and Install into Library Repository’</td>
<td>856</td>
</tr>
<tr>
<td>1.3.1.25.2.1.7</td>
<td>Command ‘Save Project as Compiled Library’</td>
<td>856</td>
</tr>
<tr>
<td>1.3.1.25.2.1.8</td>
<td>Command ‘Save/Send Archive’</td>
<td>857</td>
</tr>
<tr>
<td>1.3.1.25.2.1.9</td>
<td>Command ‘Extract Archive’</td>
<td>858</td>
</tr>
<tr>
<td>1.3.1.25.2.1.10</td>
<td>Command ‘Source Upload’</td>
<td>859</td>
</tr>
<tr>
<td>1.3.1.25.2.1.11</td>
<td>Command ‘Source Download’</td>
<td>860</td>
</tr>
<tr>
<td>1.3.1.25.2.1.12</td>
<td>Command ‘Print’</td>
<td>860</td>
</tr>
<tr>
<td>1.3.1.25.2.1.13</td>
<td>Command ‘Print Preview’</td>
<td>860</td>
</tr>
<tr>
<td>1.3.1.25.2.1.14</td>
<td>Command ‘Page Setup’</td>
<td>860</td>
</tr>
<tr>
<td>1.3.1.25.2.1.15</td>
<td>Command ‘Recent Projects’</td>
<td>861</td>
</tr>
<tr>
<td>1.3.1.25.2.1.16</td>
<td>Command ‘Exit’</td>
<td>861</td>
</tr>
</tbody>
</table>

---

**Command ‘New Project’**

Symbol: ☰, Shortcut: [Ctrl] + [N]

**Function:** This command opens the “New Project” dialog box for the creation of a new project file.

**Call:** “File” menu

---

**'New Project' dialog box**

**Function:** Selection of a project category and a project template.

**Call:** “File ➔ New Project”

Depending on the template, you obtain a project that is automatically equipped with a certain range of objects.

---

**Table 115: “Categories”**

<table>
<thead>
<tr>
<th>Libraries</th>
<th>Projects</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

---
Table 116: “Templates”

<table>
<thead>
<tr>
<th>“Projects” category:</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>“Empty project”</td>
<td>Contains only the “Project Settings” object.</td>
</tr>
<tr>
<td>“Standard project”</td>
<td>Contains a basic range of objects and libraries. A wizard assists with the creation — see below.</td>
</tr>
<tr>
<td>“Standard project with Application Composer”</td>
<td>Contains a basic range of objects and libraries for working with the Application Composer. A wizard assists with the creation.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>“Libraries” category:</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>“CODESYS container library”</td>
<td>Library that contains only further libraries, but no function blocks of its own.</td>
</tr>
<tr>
<td>“CODESYS interface library”</td>
<td>Library only for the definition of the interface of a software component. Thus contains only objects that do not generate any code (constants, structures, interfaces, etc.).</td>
</tr>
<tr>
<td>“Empty library”</td>
<td>Contains only the “Project Settings” object.</td>
</tr>
<tr>
<td>“External CODESYS library”</td>
<td>Target-system-specific library which is implemented as part of the runtime system (in ANSI C or C++).</td>
</tr>
<tr>
<td>“Name”</td>
<td>Name of the project to be created. Depending on the template, a standard name appears. The numerical suffix ensures the uniqueness of the name in the file system. You can change the file name, taking into consideration the file path conventions of the operating system. Periods are not permitted in names. CODESYS automatically adds the appropriate file extension to the selected template.</td>
</tr>
</tbody>
</table>
| “Location” | Location for the new project file. 

opens a dialog box for browsing the file system. 

Displays the history of previously entered paths. |
| “OK” | CODESYS opens a new project. An error symbol next to the input field draws attention to missing specifications. If you place the mouse pointer on it, a tooltip appears, informing you what to do. |

“Standard Project” dialog box

**Function:** Wizard for the creation of a standard project.

**Call:** Command “File ➤ New Project”; in the “New Project” dialog box, select the “Projects” category and the “Standard project” template and click on “OK”.

| “Device” | Selection list with PLC devices. The selected device is inserted as an object in the Devices view below the root node. |
| “PLC_PRG in” | Selection list with the programming languages. The automatically inserted program PLC_PRG is created in the selected language. |

See also

● Chapter 1.3.1.3.1 “Creating standard projects” on page 124

Command ‘Open Project’

**Symbol:** ⌘ **Shortcut:** [Ctrl] + [O]

**Function:** This command opens the standard dialog box for opening a file. Here you can browse the file system for a CODESYS project file and open it in the development system.

**Call:** “File” menu
Depending on the selected file format

### 'Open Project' dialog box

<table>
<thead>
<tr>
<th>&quot;File type&quot;</th>
<th>Selection list for filtering the type of file</th>
</tr>
</thead>
<tbody>
<tr>
<td>● &quot;All supported files&quot;</td>
<td>files in all supported formats.</td>
</tr>
<tr>
<td>● Project files: CODESYS project files from V3, file extension &quot;project&quot;</td>
<td></td>
</tr>
<tr>
<td>● Project archive files: CODESYS project archives from V3, file extension &quot;projectarchive&quot;</td>
<td></td>
</tr>
<tr>
<td>● Library files: Library projects from V3, file extension &quot;library&quot;</td>
<td></td>
</tr>
</tbody>
</table>

| "Open" | CODESYS opens the selected project file. If necessary, it is converted first. |

### Command 'Close Project'

**Function:** This command closes the currently opened project. CODESYS remains opened.

**Call:** "File" menu. In addition implicitly when opening a new/other project, while another project is still open.

If the project contains unsaved changes, a query appears, asking whether the project should be saved.

If you have not yet explicitly saved the project, a query appears asking whether you wish to delete the project files.

### Command 'Save project'

**Symbol:** 📖, shortcut [Ctrl] + [S]

**Function:** this command saves the project file.

**Call:** "File" menu

This command saves the project file with the current project name, which appears in the title bar of the main window. If the project has been changed since it was last saved, the project name is given an asterisk.

The command is not available if the project is read-only.

Write protection exists if

- the project is identified in the project information (summary) as 'Released'
- the option "Open read-only" was selected in the dialog box "Open Project" when opening the project

Write protection is indicated by a line in the top right corner of the main window. A mouse-click on this line brings up a menu with commands for the possible actions:

- "Save project under a different file name on the disk": a mouse-click on this option leads to 'Save file as...' and
- "Exit read-only mode": appears only if the option "Open read-only" was selected when opening the project.
- "Remove read-only attribute from the project on the disk": appears only if the project file had been provided with the property 'Read-only' on the disk at the time of opening.
- "Remove identification 'Released' in the project information": appears only if this attribute is currently set.

**Backup copy**
Optionally a backup copy of the project file can be created. If the option "Create backup copy" is activated in the option dialog box 'Load and Save', the project is additionally copied to a file <projectname.backup> each time the project is saved.

See also
- § Chapter 1.3.1.25.2.1.5 “Command 'Save Project as'” on page 855
- § Chapter 1.3.1.6.8 “Save project” on page 150
- § Chapter 1.3.1.25.3.13.13 “Dialog ‘Options’ – ‘Load and Save’” on page 1083

Command 'Save Project as'

This command opens the standard Windows dialog box for saving a file. The project can be stored with the desired location and file type.

<table>
<thead>
<tr>
<th>“File type”</th>
<th>For both normal projects and library projects, this drop-down list contains the respective versions of the development system for which the project can be saved. If the current project contains add-ons that are not available in the selected memory format (profile), then the “Extend Profile” dialog box opens.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>- “Project files (CODESYS v&lt;version&gt;) (*.project)”: The project is saved as a CODESYS project file &quot;&lt;project name&gt;.project &quot; for the currently used or selected version of the development system.</td>
</tr>
<tr>
<td></td>
<td>- “Library files (CODESYS v&lt;version&gt;) (*.library)”: The project is saved as a CODESYS library file &quot;&lt;project name&gt;.library&quot; for the currently used or selected version of the development system.</td>
</tr>
<tr>
<td></td>
<td>If the project should be opened later in an older version, then it makes sense to save for precisely this version, as you will then be informed immediately about possible data loss.</td>
</tr>
</tbody>
</table>

Before saving a project as a library:
- Make sure that the rules for creating libraries have been followed. |
- If it is to be possible to configure global constants provided by the library at a later time in an application, then you must define them in a parameter list. A parameter list is a special type of global variable list. |
- When saving the project, no automatic check for errors is performed. |
- Unlike CoDeSys V2.3, there is no distinction between 'external' and 'internal' libraries. Now you can define in the properties of each individual project object whether or not it should be treated as 'external'. |
- Consider whether the library created is to be installed in the system library repository immediately. If so, then use the command ‘Save project and install in the library repository’. |
- If you want to protect the library project from later changes, then set the “Released” attribute in the “Project Information” dialog box. At the next attempt to save the project, a corresponding message will be displayed and the user must respond to the write protection with deliberate actions. |
- If you save the project as a version of the development system other than the one currently in use, then you will be informed first about possible data loss. |

Dialog box 'Extend Profile'

In this dialog box, the selected profile (memory format) can be extended by the add-ons that are contained in the current project. The profile is saved temporarily and then deleted after being saved or exported.

<table>
<thead>
<tr>
<th>“Add to profile”</th>
<th>☑: The current profile is extended by the add-on so that the add-on data of the current project is also saved.</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Add-on”</td>
<td>The add-on of the current project that is not contained in the selected memory format.</td>
</tr>
</tbody>
</table>


<table>
<thead>
<tr>
<th>“Version”</th>
<th>Version of the “Add-on” included in the current profile. If several versions are installed, then the version can be selected.</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Save profile”</td>
<td>Opens the “Enter Profile Name” dialog box. In this dialog box, specify the name for the new profile. The new profile is saved permanently at $ProgramData$/PRODUCT$/CustomInformationalProfiles.</td>
</tr>
<tr>
<td>“Use saved profile”</td>
<td>The profile which was permanently saved in “Save profile” is used for saving or exporting the current project.</td>
</tr>
</tbody>
</table>

See also

- § Chapter 1.3.1.25.2.1.4 “Command 'Save project’” on page 854
- § Chapter 1.3.1.6.8 “Save project” on page 150
- § Chapter 1.3.1.25.3.13.13 “Dialog ‘Options’ – ‘Load and Save’” on page 1083
- § Chapter 1.3.1.19.1 “Information for library developers” on page 372
- § Chapter 1.3.1.25.2.1.6 “Command ‘Save Project and Install into Library Repository’” on page 856

Command 'Save Project and Install into Library Repository’

**Function:** this command saves the project as a library in the 'system' library repository.

**Call:** Main menu “File”.

With this command CODESYS saves the project as a library in the 'system' library repository. This is an extension to the saving of a project as a library file using the “Save Project as” command. The library is installed on the local system and is immediately available for insertion into a project.

See also

- § Chapter 1.3.1.25.2.1.5 “Command 'Save Project as’” on page 855

Command 'Save Project as Compiled Library’

**Function:** This command saves a library project in encrypted form.

**Call:** Menu bar: “File”

The command opens the standard dialog for saving a file in the file system. The “Compiled CODESYS Libraries” file type is already preset. The file extension is .compiled-library-v3 or .compiled-library (CODESYS < SP15). In this format, then source code of the library POU is not visible when the library is used in a project.

If the “Enforce signing of compiled libraries” option is selected in the “Security Screen” view, then a library project has to be provided with a digital certificate-based signature when being saved. In the “Project Information”, on the “Summary” tab, a “Library compatibility” with a CODESYS version >= V3 SP15 is set by default. In this case, the project file is stored with the file extension .compiled-library-v3 when being saved as a compiled and signed library. If you still have not specified a suitable valid certificate for your user profile in the “Security Screen”, then a dialog prompt opens next for you to do this. Afterwards, you can execute the save command again.

In all other respects, compiled library files behave just like *.library files, and therefore they can be installed and referenced with the same steps.

We recommend the use of compiled libraries signed with certificates. Besides the protection of the source code and the unauthorized use of a library, less memory is also used which results in shorter loading times.
If you have the corresponding help files with translations, then as of CODESYS V3 SP15 you can extend the library documentation with the translation into other languages. This is done as follows:

Place the files created for the new languages _lmd_<language>.aux in a directory <library name>.lmd parallel to the library project <library name>.compiled-library-v3. If the files are correct, then they are included in the compiled library file when saving the library project by means of the "Save Project as Compiled Library" command.

Example: The directory standard.lmd is exists parallel to the library file standard.compiled-library-v3 and contains the file _lmd_fr.aux with the French translation of the library documentation. After the compiled library is saved, the French version of the documentation is also available in the Library Manager.

See also

- “Tab "Summary"” on page 811
- Chapter 1.3.1.25.2.18 “Command ‘Security Screen’” on page 891
- Chapter 1.3.1.19.1 “Information for library developers” on page 372
- Chapter 1.3.1.25.2.1.6 “Command ‘Save Project and Install into Library Repository’” on page 856

**Command 'Save/Send Archive'**

**Function:** This command opens the dialog “Project Archive” for the configuration of project archives.

**Call:** Menu bar: “File ➪ Project Archive”

An archive file (*.projectarchive) contains all files contained and referenced in the currently opened project. It can either be saved or dispatched as an e-mail attachment. The dispatch by email is very helpful for providing an employee with all project-relevant files. The file can be simply unpacked again with the command “Extract Archive”.

**NOTICE!**

The archiving function is not intended for the storage of a project, but rather for the simple summarizing of all project-relevant files.

See also

- Chapter 1.3.1.6.9 “Saving/Sending the project archive” on page 152
- Chapter 1.3.1.25.2.1.9 “Command ‘Extract Archive’” on page 858

**Dialog 'Project Archive'**

The dialog displays all the categories that can be added to the project archive. In this dialog, complete categories or individual objects from the categories can be added to the project archive by setting a check mark (√).

Entries that are display as red in the list require your attention. Move the mouse pointer over this library for more information.
"Additional files"  Opens the dialog "Additional Files". Here, further files can be added to the archive with the "Add" button.

"Comment"  Opens the "Comment" dialog. Here, comments can be added to the archive.

"Save"  Creates the archive file and saves it. The storage location and the archive name are specified in the subsequent dialog.

"Send"  Creates a temporary archive file that is attached to an empty e-mail. A correct installation of the MAPI (Messaging Application Programming Interface) is required for the successful execution of this operation. Failure is documented by the display of a corresponding error message. The temporary archive is automatically deleted after sending the e-mail.

### Command 'Extract Archive'

**Function:** The command extracts a project archive, that was created with the command "Save/ Send Archive". You have to configure which objects of the archive CODESYS shall extract and in which directory of the file system they will be copied.

**Call:** Main menu "File ➔ Project Archive"

The file extension of an archive is .projectarchive.

After the archive is selected, the dialog "Extract Project Archive" opens to configure the extract parameters.

#### Dialog Box 'Extract Project Archive'

This dialog box shows the contents of the project archive. You can exclude complete categories or single objects from categories by clearing the check boxes (✓) from the extraction.

**Table 118: "Locations"**

<table>
<thead>
<tr>
<th>Location</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Extract into the same folder where the archive is located&quot;</td>
<td>The archive is extracted to the same directory.</td>
</tr>
<tr>
<td>&quot;Extract into the following folder&quot;</td>
<td>The contents of the archive are extracted to the given path.</td>
</tr>
<tr>
<td>&quot;Advanced&quot;</td>
<td>Opens the &quot;Advanced&quot; dialog box for you to define where special and additional files from the archive are extracted.</td>
</tr>
</tbody>
</table>

**Table 119: "Contents"**

<table>
<thead>
<tr>
<th>Content</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Items&quot;</td>
<td>Shows the contents of the archive structured in object categories.</td>
</tr>
<tr>
<td>✓</td>
<td>The object is extracted.</td>
</tr>
<tr>
<td>□</td>
<td>The object is not extracted.</td>
</tr>
<tr>
<td>&quot;Comment&quot;</td>
<td>Comment that was entered when creating the project archive</td>
</tr>
<tr>
<td>&quot;Extract&quot;</td>
<td>If an extracted file has the same name as an existing file in the target directory, then a dialog box opens, prompting whether the local file should be replaced. The decision can be applied automatically to any additional conflicting names. In this case, you have to select the ✓ &quot;Apply to all objects and files&quot; check box.</td>
</tr>
</tbody>
</table>
Table 120: "Repositories"

| "Install devices into" | Drop-down list with currently available repositories. Select the repositories, in which CODESYS shall install the devices and the libraries of the archive. |
| "Install libraries into" |

Table 121: "Additional Files"

By default the "additional files" are set to "Do not extract". Select the entries in the table and chose one of the following options:

| "Extract into project folder" | Folder of the project file |
| "Extract into folder" | User defined folder |
| "Do not extract" | Default |

See also
- Chapter 1.3.1.25.2.1.8 "Command 'Save/Send Archive'" on page 857

Command 'Source Upload'

**Function:** This command loads the project source code (as project archive) from the controller.

**Call:** Main menu "File".

**Requirement:** The network path for the controller must be configured.

After you execute the command, an overview opens with all devices in the network. Select a controller from this overview. The dialog box "Extract Project Archive" then opens with export settings.

See also
- Chapter 1.3.1.11.7 "Downloading source code to and from the PLC" on page 307
- Chapter 1.3.1.25.2.1.11 "Command 'Source Download'" on page 860

Dialog Box 'Extract Project Archive'

This dialog box shows the contents of the project archive. You can exclude complete categories or single objects from categories by clearing the check boxes (☐) from the extraction.

Table 122: “Locations”

| "Extract into the same folder where the archive is located" | The archive is extracted to the same directory. |
| "Extract into the following folder" | The contents of the archive are extracted to the given path. |
| "Advanced" | Opens the "Advanced" dialog box for you to define where special and additional files from the archive are extracted. |

Table 123: “Contents”

| "Items" | Shows the contents of the archive structured in object categories. |
| "Comment" | Comment that was entered when creating the project archive. |
| ☑ | The object is extracted. |
| ☐ | The object is not extracted. |
If an extracted file has the same name as an existing file in the target directory, then a dialog box opens, prompting whether the local file should be replaced. The decision can be applied automatically to any additional conflicting names. In this case, you have to select the **Apply to all objects and files** check box.

**Command 'Source Download'**

**Function:** This command loads the project source code (as project archive) to the controller.

**Call:** Main menu “File”.

**Requirement:** The network path for the controller must be configured.

After you execute the command, an overview opens with all devices in the network. Select a controller from this overview. Then the Archiv.prj project archive is downloaded to this controller. You can click “Source Upload” to upload the complete source code to the CODESYS development system at a later time.

If you are already connected to a controller (in online mode), then the “Source Download to Connected Device” command is also available for this process.

See also
- Chapter 1.3.1.11.7 “Downloading source code to and from the PLC” on page 307
- Chapter 1.3.1.25.2.1.10 “Command ‘Source Upload’” on page 859
- Chapter 1.3.1.25.2.6.7 “Command ‘Source Download to Connected Device’” on page 929

**Command 'Print'**

**Symbol:**  

**Function:** This command opens the default Windows dialog box for printing documents.

**Call:** Main menu “File”

See also
- Chapter 1.3.1.25.3.11.6 “Dialog ‘Project Settings’ - ‘Page Setup’” on page 1064

**Command 'Print Preview'**

**Function:** This command opens a print preview for the currently open element.

**Call:** Main menu “File”

**Requirement:** An object is open in the editor.

See also
- Chapter 1.3.1.25.3.11.6 “Dialog ‘Project Settings’ - ‘Page Setup’” on page 1064
- Chapter 1.3.1.25.2.1.12 “Command ‘Print’” on page 860

**Command 'Page Setup'**

**Symbol:**  

**Function:** This command opens the “Page Setup” dialog box for configuring the layout of the printed version of the project contents.

**Call:** Main menu “File ➤ Page Setup”
Command ‘Recent Projects’

Function: Opens the list of the projects used recently, from which you can select a project to open.

Call: “File” menu

Command ‘Exit’

Shortcut: [Alt]+[F4]

Function: this command exits from the programming system. If a project is currently opened that has been changed since it was last saved, a dialog box opens asking whether the project should be saved.

Call: “File” menu
Menu 'Edit'

| 1.3.1.25.2.2.1 | Standard Commands....................................................... 862 |
| 1.3.1.25.2.2.2 | Command 'Find', 'Find in Project'..................................... 863 |
| 1.3.1.25.2.2.3 | Command 'Replace', 'Replace in Project'.............................. 864 |
| 1.3.1.25.2.2.4 | Command 'Find Next'.......................................................... 865 |
| 1.3.1.25.2.2.5 | Command 'Find Next (Selection)'........................................ 865 |
| 1.3.1.25.2.2.6 | Command 'Find Previous'.................................................. 865 |
| 1.3.1.25.2.2.7 | Command 'Find Previous (Selection)'................................... 866 |
| 1.3.1.25.2.2.8 | Command 'Insert File as Text'........................................... 866 |
| 1.3.1.25.2.2.9 | Command 'Overwrite Mode'................................................ 866 |
| 1.3.1.25.2.2.10 | Command 'View Whitespace'................................................ 866 |
| 1.3.1.25.2.2.11 | Command 'View Indentation Guides'..................................... 867 |
| 1.3.1.25.2.2.12 | Command 'Go to Line'........................................................ 867 |
| 1.3.1.25.2.2.13 | Command 'Make Uppercase'................................................ 867 |
| 1.3.1.25.2.2.14 | Command 'Make Lowercase'................................................ 867 |
| 1.3.1.25.2.2.15 | Command 'Go to Matching Bracket'...................................... 868 |
| 1.3.1.25.2.2.16 | Command 'Select to Matching Bracket'................................ 868 |
| 1.3.1.25.2.2.17 | Command 'Expand All Folds'.............................................. 868 |
| 1.3.1.25.2.2.18 | Command 'Collapse All Folds'........................................... 868 |
| 1.3.1.25.2.2.19 | Command 'Comment Out Selected Lines'................................ 869 |
| 1.3.1.25.2.2.20 | Command 'Uncomment Selected Lines'.................................. 869 |
| 1.3.1.25.2.2.21 | Command 'Enable Inline Monitoring'................................... 869 |
| 1.3.1.25.2.2.22 | Command 'Toggle Bookmark'............................................... 869 |
| 1.3.1.25.2.2.23 | Command 'Next Bookmark (Active Editor)'............................... 870 |
| 1.3.1.25.2.2.24 | Command 'Next Bookmark'.................................................. 870 |
| 1.3.1.25.2.2.25 | Command 'Previous Bookmark (Active Editor)'.......................... 870 |
| 1.3.1.25.2.2.26 | Command 'Previous Bookmark'............................................ 870 |
| 1.3.1.25.2.2.27 | Command 'Clear All Bookmarks (Active Editor)'....................... 871 |
| 1.3.1.25.2.2.28 | Command 'Clear All Bookmarks'......................................... 871 |
| 1.3.1.25.2.2.29 | Command 'Browse Cross-References'................................... 871 |
| 1.3.1.25.2.2.30 | Command 'Browse Call Tree'............................................... 871 |
| 1.3.1.25.2.2.31 | Command 'Auto Declare'.................................................. 872 |
| 1.3.1.25.2.2.32 | Command 'Input Assistant'................................................ 874 |
| 1.3.1.25.2.2.33 | Command 'Go to Source Position'....................................... 875 |
| 1.3.1.25.2.2.34 | Command 'Next Message'................................................... 875 |
| 1.3.1.25.2.2.35 | Command 'Previous Message'............................................... 875 |
| 1.3.1.25.2.2.36 | Command 'Go to Definition'............................................... 875 |
| 1.3.1.25.2.2.37 | Command 'Go To Reference'............................................... 876 |
| 1.3.1.25.2.2.38 | Command 'Go to Instance'.................................................. 876 |
| 1.3.1.25.2.2.39 | Command 'Refactoring' - 'Rename <...>'............................... 876 |
| 1.3.1.25.2.2.40 | Command 'Refactoring' - 'Update Referenced Pins'................... 877 |
| 1.3.1.25.2.2.41 | Command 'Refactoring' - 'Add Variable'................................ 878 |
| 1.3.1.25.2.2.42 | Command 'Refactoring' - 'Remove <variable>'.......................... 879 |
| 1.3.1.25.2.2.43 | Command 'Refactoring' - 'Reorder Variables'.......................... 880 |

**Standard Commands**

CODESYS provides the following standard commands:

- **Undo**: ⬤, shortcut: \[Ctrl\] + \[Z\]
- **Redo**: ⬤, shortcut: \[Ctrl\] + \[Y\]
• Cut: \(\text{⌘}X\), shortcut: [Ctrl] + [X]
• Copy: \(\text{⌘}C\), shortcut: [Ctrl] + [C]
• Paste: \(\text{⌘}V\), shortcut: [Ctrl] + [V]
• Delete: \(\text{⌘}X\), shortcut: [Ctrl]
• Select all: shortcut: [Ctrl] + [A]

Not all editors support the “Insert” command. In some editors it can be used with limitations. Graphical editors only support the command if the pasted elements will create a valid construct.

In object trees like POU's or device view the command refers to the currently selected object. Multi selection is possible.

Command 'Find', 'Find in Project'
Symbol \(\text{⌘}F\), keyboard shortcut: [Ctrl]+[F]
Symbol \(\text{⌘}F\), keyboard shortcut [Ctrl]+[Shift]+[F]

Function: These commands scan the project or parts of it for a specified character string.

Call: Menu bar: “Edit \(\rightarrow\) Find Replace”

This command opens the “Find” dialog where the searched character string is specified and the search options are defined.

Dialog 'Find'

| “Search for” | Character string to be searched. |
| “Match case” | ✓: The search considers uppercase and lowercase. |
| “Match whole word” | ✓: Only character strings are found that exact matches. |
| “Search up” | ✓: The specified search range runs upwards. |
| “Use regular expressions” | Use the \(\text{⌘}\) button to receive support when specifying regular expressions. |
| “Search in” | ☑: Drop-down list with the areas of the project to be searched: |
| | • “Active editor” |
| | • “All open editors” |
| | • “Selected objects & Subobjects” |
| | • “Entire project” |
| | • “Entire project & Uncompiled libraries” |
| | • “Selection only” |
| “Find next” | Start the search |
| “Find all” | All search results are listed in the message view with their object path, project name, object name, and object position. Possible additional information for position: “(Decl)” = Declaration part of the object; “(Impl)” = Implementation part of the object |
| “Replace” | Switches to the “Replace” dialog |

The color of the search result markings can be customized in the options of the text editor. This is done by means of the parameter “Selection color” - “Inactive” in the “Text Area” tab.
Dialog for setting the objects to be searched

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Entire project&quot;</td>
<td>All editable positions in all objects of the project are searched.</td>
</tr>
<tr>
<td>&quot;Entire project and all uncompiled libraries&quot;</td>
<td>All editable positions in all objects of the project, including integrated uncompiled libraries, are searched.</td>
</tr>
<tr>
<td>&quot;Within the following objects&quot;</td>
<td>Only the editable positions within the objects defined here are searched:</td>
</tr>
<tr>
<td>- &quot;Scheme&quot;</td>
<td>The &quot;Save&quot; command saves the current search configuration by the specified name. All saved schemes are available in the drop-down list.</td>
</tr>
<tr>
<td>- &quot;Object types&quot;</td>
<td>The object is searched.</td>
</tr>
<tr>
<td>- &quot;Name filter&quot;</td>
<td>Name filter for the searched objects. The placeholder &quot;*&quot; can be used.</td>
</tr>
<tr>
<td>Example: Filter &quot;<em>CAN</em>&quot;: All objects are searched that have &quot;CAN&quot; in the name.</td>
<td></td>
</tr>
<tr>
<td>&quot;All open editors&quot;</td>
<td>All editors are searched that are currently open in a window.</td>
</tr>
<tr>
<td>&quot;Active editor&quot;</td>
<td>Only the editor is searched where the cursor currently is.</td>
</tr>
<tr>
<td>&quot;Selection only&quot;</td>
<td>Only the text is searched that is currently selected in an object.</td>
</tr>
</tbody>
</table>

See also

- § Chapter 1.3.1.25.2.2.3 “Command 'Replace', 'Replace in Project'” on page 864
- § Chapter 1.3.1.9.15 “Searching and replacing in the entire project” on page 236

Command 'Replace', 'Replace in Project'

Symbol  
keyboard shortcut: [Ctrl]+[H]  
keyboard shortcut: [Ctrl]+[Shift]+[H]  
Function: These commands scan the project or parts of it for a specified character string and replaces it.  
Call: Menu bar: “Edit ➔ Find Replace”  
Requirement: The application is in online mode.  
This command opens the "Replace" dialog where the search and replace character strings are specified and the search options are defined.

Table 124: In addition to the options of the “Search” dialog, the following settings are still possible:

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Replace with&quot;</td>
<td>Input field for the new character string.</td>
</tr>
<tr>
<td>&quot;Replace&quot;</td>
<td>Each next found string is highlighted in the editor and replaced (step-by-step replace).</td>
</tr>
<tr>
<td>&quot;Replace all&quot;</td>
<td>All found strings are replaced at one time without them being displayed in the editors.</td>
</tr>
<tr>
<td>&quot;Leave changed objects open after &quot;Replace all&quot;&quot;</td>
<td>The editors of the found objects remain open.</td>
</tr>
</tbody>
</table>
Replacement in referenced libraries is not possible.

See also
- % Chapter 1.3.1.25.2.2.2 “Command 'Find', 'Find in Project’” on page 863
- % Chapter 1.3.1.9.15 “Searching and replacing in the entire project” on page 236

Command 'Find Next'

Symbol ⌘, keyboard shortcut [F3]

**Function:** During the search for a certain string within the project, this command selects the next match at its position in the respective editor.

**Call:** Menu “Edit ➔ Search Replace”

**Requirement:** You have already started searching the project for a certain string by using the commands “Find” or “Replace”.

See also
- % Chapter 1.3.1.25.2.2.2 “Command 'Find', 'Find in Project’” on page 863
- % Chapter 1.3.1.25.2.2.3 “Command 'Replace', 'Replace in Project’” on page 864
- % Chapter 1.3.1.9.15 “Searching and replacing in the entire project” on page 236

Command 'Find Next (Selection)'

Keyboard shortcut [Ctrl] + [F3]

**Function:** The command searches the project for the next string matching the string which is currently selected or in which you have currently placed the cursor.

**Call:** Menu “Edit ➔ Find Replace”

**Requirement:** You have the cursor placed in an editable string in your project, or you have selected an editable string.

See also
- % Chapter 1.3.1.25.2.2.2 “Command 'Find', 'Find in Project’” on page 863
- % Chapter 1.3.1.25.2.2.3 “Command 'Replace', 'Replace in Project’” on page 864
- % Chapter 1.3.1.9.15 “Searching and replacing in the entire project” on page 236

Command 'Find Previous'

Symbol ⌘, keyboard shortcut [Shift] + [F3]

**Function:** During the search for a certain string within the project, this command selects the next match at its position in the respective editor.

**Call:** Menu “Edit ➔ Search Replace”

**Requirement:** You have already started searching the project for a certain string by using the commands “Find” or “Replace”.

See also
- % Chapter 1.3.1.25.2.2.2 “Command 'Find', 'Find in Project’” on page 863
- % Chapter 1.3.1.25.2.2.3 “Command 'Replace', 'Replace in Project’” on page 864
- % Chapter 1.3.1.9.15 “Searching and replacing in the entire project” on page 236
Command 'Find Previous (Selection)'

Keyboard shortcut: [Ctrl] + [Shift] + [F3]

**Function:** The command searches the project for the previous string matching the string which is currently selected or in which you have currently placed the cursor.

**Call:** Menu “Edit ➔ Find Replace”

**Requirement:** You have the cursor placed in an editable string in your project, or you have selected an editable string.

See also
- ° Chapter 1.3.1.25.2.2.2 “Command ‘Find’, ‘Find in Project’” on page 863
- ° Chapter 1.3.1.25.2.2.3 “Command ‘Replace’, ‘Replace in Project’” on page 864
- ° Chapter 1.3.1.9.15 “Searching and replacing in the entire project” on page 236

Command 'Insert File as Text'

**Function:** This command copies the contents of a text file to the active editor as the current cursor position.

**Call:** The command is not in any menu by default. You can add it to a menu by using the dialog box from “Tools ➔ Customize” (command category “Text Editor”).

**Requirement:** The file must have the extension .txt. The command is available in a text editor only.

Many development environments and text processing applications provide the option of exporting code and text as a plain text file. This command can copy the contents of this file to the editor.

See also
- ° Chapter 1.3.1.25.3.13.20 “Dialog ‘Options’ - ‘Text Editor’” on page 1086

Command 'Overwrite Mode'

**Shortcut:** [Insert]

**Function:** This command activates the overwrite mode.

**Call:** Menu “Edit ➔ Advanced”

**Requirement:** A text editor is opened.

If the overwrite mode is activated, characters in front of the cursor are overwritten when entering new characters. If the overwrite mode is deactivated, characters are inserted and existing characters in front of the cursor are retained.

See also
- ° Chapter 1.3.1.25.3.13.20 “Dialog ‘Options’ - ‘Text Editor’” on page 1086

Command 'View Whitespace'

**Symbol:** \( \_a \_b \)

**Function:** This command causes control characters for spaces and tabs to be shown.

**Call:** Menu “Edit ➔ Advanced”

**Requirement:** A text editor is opened.

CODESYS visualizes spaces by a period and tabs by an arrow.
See also

- Chapter 1.3.1.25.3.13.20 “Dialog ‘Options’ - ‘Text Editor’” on page 1086

Command 'View Indentation Guides'

Function: This command activates the indentation help lines.
Call: Menu “Edit ➔ Extended”
Requirement: A text editor is opened.
If the indentation help lines are activated, a broken line is inserted for each manual indentation in the code. This facilitates the overview of the different levels in the code. You can insert manual indentations with the [Tab] key.
See also

- Chapter 1.3.1.25.3.13.20 “Dialog ‘Options’ - ‘Text Editor’” on page 1086

Command 'Go to Line'

Function: With this command the cursor jumps to a defined line in the code.
Call: Menu “Edit ➔ Extended”
Requirement: A text editor is opened.
This command opens a dialog box with an input field “Line number”.
See also

- Chapter 1.3.1.25.3.13.20 “Dialog ‘Options’ - ‘Text Editor’” on page 1086

Command 'Make Uppercase'

Shortcut: [Ctrl]+[Shift]+[U]
Function: This command converts all lowercase letters in the selected code into uppercase letters.
Call: Menu “Edit ➔ Advanced”
Requirement: A text editor is opened and code is selected, or the declaration editor is opened and variable declarations are selected.
See also

- Chapter 1.3.1.25.3.13.20 “Dialog ‘Options’ - ‘Text Editor’” on page 1086

Command 'Make Lowercase'

Shortcut: [Ctrl]+[U]
Function: This command converts all uppercase letters in the selected code into lowercase letters.
Call: Menu “Edit ➔ Advanced”
Requirement: a text editor is opened and code is selected, or the declaration editor is opened and variable declarations are selected.
See also

- Chapter 1.3.1.25.3.13.20 “Dialog ‘Options’ - ‘Text Editor’” on page 1086
Command 'Go to Matching Bracket'

**Function:** This command makes the cursor jump to the other part of the selected code parenthesis.

**Call:** Menu “Edit ➔ Advanced”

**Requirement:** A text editor is opened and the cursor is positioned at an opening or closing code parenthesis. If you position the cursor at a code parenthesis, CODESYS displays the corresponding parenthesis in color, provided you have activated the option “Associated parentheses” in the CODESYS options in the “Text Editor” category, “Text Area” tab.

See also
- “Tab ‘Text Area’” on page 1087

Command 'Select to Matching Bracket'

**Function:** This command selects the entire code section within the currently selected code parentheses.

**Call:** Menu “Edit ➔ Extended”

**Requirement:** A text editor is opened and the cursor is positioned at an opening or closing code parenthesis. If you position the cursor at a code parenthesis, CODESYS displays the corresponding parenthesis in color, provided you have activated the option “Associated parentheses” in the project options in the “Text Editor” category, “Text Area” tab.

See also
- Chapter 1.3.1.25.3.13.20 “Dialog ‘Options’ - ‘Text Editor’” on page 1086

Command 'Expand All Folds'

**Function:** This command expands all collapsed code segments in the textual editor or result locations in the cross-reference list so that the code and all search locations are displayed in full again.

**Requirement:** A textual editor is active and indentation is activated in the “Options” (“Text Editor” category); or the cross-reference list is active.

**Call:** Textual editors: main menu “Edit ➔ Advanced”, or right-click. In the cross-reference list: right-click.

See also
- Chapter 1.3.1.25.3.13.20 “Dialog ‘Options’ - ‘Text Editor’” on page 1086
- Chapter 1.3.1.25.2.2.18 “Command ‘Collapse All Folds’” on page 868
- “Right-click commands in the cross-reference list” on page 888

Command 'Collapse All Folds'

**Function:** This command collapses all expanded code segments in the textual editor or result locations in the cross-reference list. In this way, only the uppermost level of code and only the root node of the result locations displayed.

**Requirement:** A textual editor is active and indentation is activated in the “Options” (“Text Editor” category); or the cross-reference list is active.

**Call:** In textual editors: main menu “Edit ➔ Advanced”, or right-click. In the cross-reference list: right-click.
See also
- § Chapter 1.3.1.25.3.13.20 “Dialog ‘Options’ - ‘Text Editor’” on page 1086
- § Chapter 1.3.1.25.2.2.17 “Command 'Expand All Folds’” on page 868
- § “Right-click commands in the cross-reference list” on page 888

Command 'Comment Out Selected Lines’

Symbol //; keyboard shortcut: [Ctrl]+[O]

Function: The command inserts comment marks ('//') at the beginning of the selected lines.

Call: Menu bar: “Edit ➔ Advanced”, context menu

Requirement: In the ST editor, either the cursor is located in a line of the implementation or multiple lines are selected.

See also
- § Chapter 1.3.1.25.2.2.20 “Command ‘Uncomment Selected Lines’” on page 869

Command 'Uncomment Selected Lines'

Symbol //; keyboard shortcut: [Ctrl]+[I]

Function: The command removes any comment marks ('//') at the beginning of the selected lines.

Call: Menu bar: “Edit ➔ Advanced”, context menu

Requirement: In the ST editor, either the cursor is located in a line of the implementation or multiple lines are selected.

See also
- § Chapter 1.3.1.25.2.2.19 “Command 'Comment Out Selected Lines’” on page 869

Command 'Enable Inline Monitoring'

Function: This command enables or disables the inline monitoring function. This works the same way as the check box with the same name in the CODESYS options (“Text Editor” category).

Requirement: A text editor is active.

Call: Context menu of the text editor in the “Advanced” submenu.

See also
- § “Tab 'Monitoring’” on page 1088
- § Chapter 1.3.1.13.1 “Monitoring of values” on page 323

Command 'Toggle Bookmark'

Symbol , keyboard shortcut [Ctrl]+[F12]

Function: The command sets or removes a bookmark at the current position.

Call: Menu bar: “Edit ➔ Bookmarks”

Requirement: A POU is open in the editor and the cursor is at a program line.

See also
- § Chapter 1.3.1.9.14.3 “Setting and using bookmarks” on page 234
Command 'Next Bookmark (Active Editor)'
Symbol: ✌; keyboard shortcut: [F12]
Function: The command jumps to the next bookmark in the active editor.
Call: Menu bar: “Edit ➔ Bookmarks”
Requirement: A POU is open in the editor and the cursor is positioned in the POU.
See also
- Chapter 1.3.1.25.2.2.24 “Command 'Next Bookmark’” on page 870
- Chapter 1.3.1.9.14.3 “Setting and using bookmarks” on page 234

Command 'Next Bookmark'
Symbol: ✌
Function: The command jumps to the next bookmark in the “Bookmarks” view and in the project, and opens the respective POU. The order of jumping to bookmarks corresponds to the order of bookmarks in the table of the “Bookmarks” view.
Call:
- “Next Bookmark” button in the “Bookmarks” view
- The command is not in any menu by default. You can add it to a menu by using the dialog from “Tools ➔ Customize” (command category “Bookmarks”).
Requirement:
- A project is open.
- The “Bookmarks” view is open.
See also
- Chapter 1.3.1.25.2.2.23 “Command 'Next Bookmark (Active Editor)’” on page 870

Command 'Previous Bookmark (Active Editor)'
Symbol: ⬄; keyboard shortcut: [Shift]+[F12]
Function: The command jumps to the previous bookmark in the active editor.
Call: Menu bar: “Edit ➔ Bookmarks”
A POU is open in the editor and the cursor is positioned in the POU.
See also
- Chapter 1.3.1.25.2.2.26 “Command 'Previous Bookmark’” on page 870
- Chapter 1.3.1.9.14.3 “Setting and using bookmarks” on page 234

Command 'Previous Bookmark'
Symbol: ⬄
Function: The command jumps to the previous bookmark in the “Bookmarks” view and in the project, and opens the respective POU. The order of jumping to bookmarks corresponds to the order of bookmarks in the table of the “Bookmarks” view.
Call:
- “Previous Bookmark” button in the “Bookmarks” view
- The command is not in any menu by default. You can add it to a menu by using the dialog from “Tools ➔ Customize” (command category “Bookmarks”).
Requirement:
● A project is open.
● The “Bookmarks” view is open.

See also
● § Chapter 1.3.1.25.2.2.25 “Command ‘Previous Bookmark (Active Editor)’” on page 870
● § Chapter 1.3.1.9.14.3 “Setting and using bookmarks” on page 234

Command 'Clear All Bookmarks (Active Editor)'
Symbol:  
Function: The command deletes all bookmarks in the active editor.
Call: Menu bar: “Bookmarks”
Requirement: A POU is open in the editor and the cursor is positioned in the POU.
See also
● § Chapter 1.3.1.25.2.2.28 “Command ‘Clear All Bookmarks’” on page 871
● § Chapter 1.3.1.9.14.3 “Setting and using bookmarks” on page 234

Command 'Clear All Bookmarks'
Symbol:  
Function: The command deletes all bookmarks in the open project.
Call: The command is not in any menu by default. You can add it to a menu by using the dialog from “Tools ➔ Customize” (command category “Bookmarks”).
Requirement: A POU is open in the editor and the cursor is positioned in the POU.
See also
● § Chapter 1.3.1.25.2.2.27 “Command ‘Clear All Bookmarks (Active Editor)’” on page 871
● § Chapter 1.3.1.9.14.3 “Setting and using bookmarks” on page 234

Command 'Browse Cross-References'
Symbol:  
Function: This command shows all occurrences of a variable in the view “Cross-Reference List”.
Call: Main menu “Edit ➔ Browse”; cross-reference list: toolbar.
Requirement: A POU is open in the editor and the cursor is at a variable, or the view “Cross-Reference List” is open and a variable is in the “Name” input field.
See also
● § Chapter 1.3.1.9.14.1 “Using the cross-reference list to find occurrences” on page 232

Command 'Browse Call Tree'
Symbol:  
Function: The command opens the view “Call Tree”, which displays the calls of a module and also its callers.
Call:
● Menu “Edit ➔ Browse”
● Context menu, see below: Requirement
**Requirement:** A module is opened in the editor and the cursor is placed in a variable, or a module is selected in the “Devices” view or in the “POUs” view.

See also

- Chapter 1.3.1.25.2.3.16 “Command ‘Call tree’” on page 889

**Command ‘Auto Declare’**

Keyboard shortcut: [Shift]+[F2]

**Function:** The command opens the “Auto Declare” dialog, which supports the declaration of a variable.

**Call:** Menu bar: “Edit”

**Requirement:** An object or a device of the project is opened in the editor.

Due to the auto-declaration function, the “Auto Declare” dialog also appears when the cursor is located in the implementation part of a POU in a line containing the name of an undeclared variable. The requirement for this is that you must have selected the command “Tools ➔ Options” and activated the option “Declare unknown variables automatically (AutoDeclare)” in the category “SmartCoding”.

Thanks to the smart tag function, the “Declare variable” command also appears when you place the cursor over an undeclared variable in the implementation part of the ST editor and then click ⏩

**Dialog ‘Auto Declare’**

<table>
<thead>
<tr>
<th>“Scope”</th>
<th>Scope of the variable that is not declared yet.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Example: VAR (default setting for local variables)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>“Name”</th>
<th>Variable name that is not declared yet</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Example: bIsValid</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>“Type”</th>
<th>Example: BOOL</th>
</tr>
</thead>
</table>

- ▼: Lists the standard data types.
- ➔
  - “Input Assistant”: Opens the “Input Assistant” dialog
  - “Array Assistant”: Opens the “Array” dialog

<table>
<thead>
<tr>
<th>“Object”</th>
<th>Object where the new variable is declared. By default, the object that you are editing now.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Example: fbA</td>
</tr>
</tbody>
</table>

- ▼: Lists that objects where the variable can be declared.

If no objects are available for the selected “Scope”, the entry “<create object>” appears. If you select the entry “<create object>”, the dialog “Add Object” appears for the generation of a suitable object.

<table>
<thead>
<tr>
<th>“Initialization”</th>
<th>Example: FALSE</th>
</tr>
</thead>
</table>

If you do not specify an initialization value, then the variable is initialized automatically.

قدرة: Opens the “Initialization Value” dialog. This procedure is helpful for the initialization of structured variables.
"Address" Memory address of the application for the variable that is not declared yet.
Example: %IX1.0
Note:
Possible only for the following scopes:
● Local variable (VAR)
● Global variable (VAR_GLOBAL)
● Or for a persistent variable (PERSISTENT).

"Flags" Attribute keywords
● CONSTANT: Keyword for a constant.
● RETAIN: Keyword for a remanent variable.
● PERSISTENT: Keyword for a persistent variable ( stricter than RETAIN).
The selected attribute keyword is added to the variable declaration.

"Comment" Example: New input In1
In the tabular declaration editor the comment entered is displayed in the column "Comment", while in the textual declaration editor it is displayed above the variable declaration.

"Apply changes using refactoring" If you exit the dialog, then the variable is not declared yet, but then it opens the “Refactoring” dialog. You can continue editing your changes here.
The option appears for the following scopes:
● Input variable (VAR_INPUT)
● Output variable (VAR_OUTPUT)
● VAR_IN_OUT variables (input variable and output variable)

"OK" The variable is declared and appears in the declaration.
Example:

VAR RETAIN
  // New input In1
  xIn1 AT %IX1.0: BOOL := FALSE;
END_VAR

Dialog 'Array'

"Ranks and base type specification" Definition of the field sizes (“Dimension”) by entering the lower and upper limits and the "Base type" of the array. You can enter the basic type directly or with the help of the dialogs "Input Assistant" or “Array” if you click the button.

"Result" Display of the defined array.

NOTICE!
CODESYS only re-initializes variables if you have modified the initialization values of the variables.

Dialog 'Initialization Value'

List of the variables with name (“Expression”), “Initialization Value” and “Data Type”.
Modified initialization values are displayed in bold lettering.

Input field below the list Input of an initialization value for the selected variable(s).
"Apply value to selected lines" | Change of the initialization value of the selected line(s) according to the value of the input field.

"Reset selected lines to default values" | Establishment of the standard initialization values.

"OK" | CODESYS applies the initialization values in the "Auto Declare" dialog.

See also
- % Chapter 1.3.1.9.2.2 “Using the 'Declare variable' dialog box” on page 174
- % “Smart tag functions” on page 211
- % Chapter 1.3.1.9.12.2 “AT declaration” on page 228
- % Chapter 1.3.1.9.16 “Refactoring” on page 236
- % Chapter 1.3.1.9.20 “Data persistence” on page 249
- % “Dialog box 'Refactoring'” on page 878
- % Chapter 1.3.1.24.4.12 “Addresses” on page 566
- % Chapter 1.3.1.25.3.13.17 “Dialog 'Options' - 'Refactoring'” on page 1085
- % Chapter 1.3.1.24.1.3.1 “ST editor” on page 430

Command 'Input Assistant'
Symbol: 📚; keyboard shortcut: [F2]
Function: This command opens the “Input Assistant” dialog which helps you to select one of the possible programming elements at the current cursor position.
Call: Menu bar: “Edit”; context menu.
Requirement: A POU is open in the editor and the cursor is at a program line.

Dialog 'Input Assistant' - Tab 'Categories'
The input assistant provides all program elements that you can insert at the current cursor position in the editor.
The elements are sorted by “Categories”. In the category “Variables”, you can also set a “Filter” for the scope, for example “Local variables”, “Global variables”, or “Constants”.

"Structured view" | : The elements are displayed in a structure tree. You can show/hide the columns “Type”, “Address”, and “Origin” by right-clicking the column title and selecting/clearing the column name in the dropdown list.
| : The elements are displayed in a flat structure.

"Show documentation" | : The dialog is extended with the “Documentation” field.

"Insert with arguments" | : Elements that include arguments (for example, functions) also insert with these arguments at the cursor position.
Example: If you insert the function block fb1, which contains an input variable fb1_in and an output variable fb1_out, “with arguments”, then this appears in the editor as follows: fb1(fb1_in:= , fb1_out=> ).

"Insert with namespace prefix" | : Inserts the selected element with the appended namespace. In the case of library modules, the check box remains disabled if the requirement for a namespace has been defined in the library properties.

If you create objects with the same name in the same category, whether globally (“POUs” view) or assigned to an application (“Devices” view), then only one entry appears in the input assistant. The usage conforms to the usual call priority (application assigned before global).
Dialog 'Input Assistant' - Tab 'Text Search'

This tab allows you to search for specific objects. When you begin typing a search string into the search field, the names of all objects are listed whose names include the search string. Double-click an object to insert it at the current cursor position in the editor.

<table>
<thead>
<tr>
<th>“Filters”</th>
<th>Limits the search to a specific variable category</th>
</tr>
</thead>
</table>

See also
- Chapter 1.3.1.9.6 “Using input assistance” on page 208
- Dialog box 'Properties’’ on page 1006
- Chapter 1.3.1.25.2.2.32 “Command 'Input Assistant’’ on page 874

Command 'Go to Source Position'

Function: The command sets the cursor to the position in the source code that causes the message.

Call: Main menu “Edit”, context menu of the message in the message view.

Requirements: A message is selected in the message view.

Use the command “Next Message” or “Previous Message” to display the source code position of the next or previous message.

See also
- Chapter 1.3.1.25.2.2.34 “Command 'Next Message’’ on page 875
- Chapter 1.3.1.25.2.2.35 “Command 'Previous Message’’ on page 875
- Chapter 1.3.1.25.2.3.5 “Command 'Messages’’ on page 882

Command 'Next Message'

Keyboard shortcut: [F4]

Function: This command selects the next message in the messages view.

Call: Main menu “Edit”.

If the last message in the list has been reached, then the marking jumps to the beginning.

See also
- Chapter 1.3.1.25.2.2.35 “Command 'Previous Message’’ on page 875

Command 'Previous Message'

Keyboard shortcut: [Shift]+[F4]

Function: This command selects the previous message in the messages view.

Call: Main menu “Edit”

If the first message in the list has been reached, then the marking jumps to the end.

See also
- Chapter 1.3.1.25.2.2.34 “Command 'Next Message’’ on page 875

Command 'Go to Definition'

Symbol: 🔍
Function: This command shows the definition locations of a variable or function.

Call: Main menu “Edit ➔ Browse”

Requirement: A POU is open in the editor and the cursor is at a variable or function.

See also
● Chapter 1.3.1.9.14.2 “Finding declarations” on page 234

Command 'Go To Reference'

Symbol: 

Function: The command opens the declaration location of the variable that is referenced by the pointer currently in focus in online mode.

Call:
● Context menu in the declaration part or implementation code
● Menu bar: “Edit ➔ Browse”

Requirement: Online mode. A POU is open in the editor and the cursor is at a pointer. The referenced variable is stored in static memory.

See also
● Chapter 1.3.1.24.5.12 “Pointers” on page 577

Command 'Go to Instance'

Symbol: 

Function: This command opens the instance of a function block in a new window.

Call: Menu bar: “Edit ➔ Browse for Symbol”

Requirement: The application is in online mode. A POU is open in the editor and the cursor is at an instance of a function block.

The command is not available for temporary instances or instances from compiled libraries.

See also
● Chapter 1.3.1.9.14.2 “Finding declarations” on page 234

Command 'Refactoring' - 'Rename <...>,'

Function: This command opens a dialog box for renaming an object or variable across the project.

Call: Main menu “Edit ➔ Refactoring” or right-click.

Requirement: An object is selected in the device tree or in the “POUs” view, or the cursor is placed before or on a variable identifier in the declaration section of a programming object.

You can rename the following:
● Variables
● POUs
● GVLs
Dialog box 'Rename'

<table>
<thead>
<tr>
<th>“Current name”</th>
<th>Name of the object or variable</th>
</tr>
</thead>
<tbody>
<tr>
<td>“New name”</td>
<td>Input field for a new name.</td>
</tr>
<tr>
<td></td>
<td>If the name already exists, then CODESYS reports this directly below this input field.</td>
</tr>
<tr>
<td>“OK”</td>
<td>Can be activated if you have typed a valid name in “New name”.</td>
</tr>
<tr>
<td></td>
<td>Opens the “Refactoring” dialog box.</td>
</tr>
<tr>
<td></td>
<td>The affected objects and occurrences are highlighted in both views.</td>
</tr>
<tr>
<td></td>
<td>You can determine how to handle the occurrences in each view by right-clicking the occurrences and clicking the available commands.</td>
</tr>
</tbody>
</table>

Dialog box 'Refactoring'

This dialog box displays all occurrences in the project. The affected objects and occurrences are highlighted in both views.

| Right view | Displays the occurrence within an object where “Current name” occurs. |
| Left view  | Device tree of the project with the object. |

You can determine how to handle the occurrences in each view by right-clicking the occurrences and clicking the available commands.

| “Reject this change” | Reject the single change in view on the right. |
| “Accept this object” | Accept all changes in the affected object. |
| “Reject this object” | Reject all changes in the affected object. |
| “Accept whole project” | Accept all changes in the project. |
| “Reject whole project” | Reject all changes in the project. |

CODESYS highlights the accepted changes in yellow and the rejected changes in gray.

See also

- § Chapter 1.3.1.9.16 “Refactoring” on page 236

Command 'Refactoring' - 'Update Referenced Pins'

NOTICE!

Currently, this command applies only to the CFC, FBD, LD, and IL editors. It is a combination of the “Reset Pins” and “Update Parameters” commands.

Function: This command modifies the pins according to the latest block declaration in all affected occurrences of the block.

Call: Main menu “Edit ➔ Refactoring” or right-click.
**Requirement:** The cursor is placed in the name of the block in the first line of the block declaration or in the device tree.

See also

- Chapter 1.3.1.9.16 “Refactoring” on page 236
- Chapter 1.3.1.25.2.12.24 “Command 'Reset Pins'” on page 986
- Chapter 1.3.1.25.2.13.38 “Command 'Update Parameters'” on page 1002

**Command 'Refactoring' - 'Add Variable'**

**Symbol:**

**Function:** This command enables the declaration of variables in a POU, as well as the automatic update to the occurrence of the POU.

**Call:** Main menu “Edit ➤ Refactoring”, or right-click.

**Requirements:** The declaration part is in focus.

The command opens the default dialog box for declaring variables.

See also

- “Dialog 'Auto Declare'” on page 872

**Dialog box 'Refactoring'**

After clicking “OK” to close the declaration dialog, the “Refactoring” dialog box opens with two frames.
Before you decide which changes to accept at which locations, select the required option from the drop-down list (3) at the upper right part of the window:

### Examples

1. By refactoring, the `fun` block receives a new input variable `input3` with the initialization value 1. The change has the following effect:
   
   **Before:**
   
   ```
   fun(a + b, 3, TRUE);
   fun(input1:= a + b , input2 :=3 , inputx := TRUE);
   ```
   
   **After:**
   
   ```
   fun(a + b, 3, 1, TRUE);
   fun(input1:= a + b , input2 :=3 , inputx := TRUE);
   ```

2. By refactoring, the "fun" block receives a new input variable `input3` with the placeholder text "_REFACTOR_":
   
   **Before:**
   
   ```
   inst(input1 := a + b, input2 := 3, inputx := TRUE);
   fun(a + b, 3, TRUE);
   ```
   
   **After:**
   
   ```
   inst(input1 := a + b, input2 := 3, input3 := _REFACTOR_, inputx := TRUE);
   fun(a + b, 3, _REFACTOR_, TRUE);
   ```

See also

- % Chapter 1.3.1.9.16 "Refactoring" on page 236
- % Chapter 1.3.1.25.2.2.39 "Command 'Refactoring' - 'Rename <...>’" on page 876

### Command 'Refactoring' - 'Remove <variable>,'

**Symbol:** æ
t

**Function:** This command removes an input or output variable from the POU and all occurrences of the POU.

**Call:** Main menu “Edit ➔ Refactoring”, or right-click.
Requirements: In the declaration part of the POU, the cursor is located in the identifier of the variable to be removed.

Then, the command opens a dialog box with information about the removal. After you confirm this, the “Refactoring” dialog box opens. For a description of the “Refactoring” dialog box, refer to the “Edit ➔ Refactoring ➔ Rename” help page.

When you accept the changes in the “Refactoring” dialog box, the respective input and output parameters are deleted at the occurrence locations of the affected POU.

In CFC, only the connection is removed between the removed input or output to the block. The input or output itself remains in the chart.

Example in ST
In a POU, refactoring removes the input4 input variable. The occurrences are updated automatically:

Before removal:
```plaintext
inst(input1 := a + b, input2 := 3, input4 := 1, input5 := TRUE);
fun(a + b, 3, 1, TRUE);
```

After removal:
```plaintext
inst(input1 := a + b, input2 := 3, input5 := TRUE);
fun(a + b, 3, TRUE);
```

See also
- ☰ Chapter 1.3.1.9.16 “Refactoring” on page 236
- ☰ Chapter 1.3.1.25.2.2.39 “Command 'Refactoring' - 'Rename <...>'” on page 876

Command 'Refactoring' - 'Reorder Variables'

Symbol: [Reorder]

Function: This command allows changing the order of variables in the declaration editor for the selected scope: VAR_INPUT, VAR_OUTPUT, or VAR_IN_OUT.

Call: “Edit ➔ Refactoring”; context menu of the focused scope in the declaration editor.

Requirement: One of the above scopes is selected in the declaration, and more than one variable is declared in it.

The command opens the “Reorder” dialog box with a list of all declarations of the selected scope. You can drag a selected declaration up or down to another position.

See also
- ☰ Chapter 1.3.1.9.16 “Refactoring” on page 236
Menu 'View'

1.3.1.25.2.1 Standard Menu in View 'Devices', 'POUs', 'Modules'............. 881
1.3.1.25.2.2 Command 'Devices'....................................................... 881
1.3.1.25.2.3 Command 'POUs'.......................................................... 882
1.3.1.25.2.4 Command 'Modules'...................................................... 882
1.3.1.25.2.5 Command 'Messages'.................................................. 882
1.3.1.25.2.6 Command 'Element properties'......................................... 883
1.3.1.25.2.7 Command 'ToolBox'..................................................... 883
1.3.1.25.2.8 Command 'Watch' - 'Watch <n>'.................................... 883
1.3.1.25.2.9 Command 'Watch' - 'Watch All Forces'............................ 883
1.3.1.25.2.10 Command 'Add All Forces to Watchlist'.......................... 884
1.3.1.25.2.11 Command 'Bookmarks'................................................ 884
1.3.1.25.2.12 Command 'Breakpoints'.............................................. 885
1.3.1.25.2.13 Command 'Cross Reference List'.................................... 886
1.3.1.25.2.14 Command 'Browse Cross References in Classic View'......... 888
1.3.1.25.2.15 Command 'Call Stack'.................................................. 888
1.3.1.25.2.16 Command 'Call tree'.................................................... 889
1.3.1.25.2.17 Command 'Memory'..................................................... 891
1.3.1.25.2.18 Command 'Security Screen'......................................... 891
1.3.1.25.2.19 Command 'Settings of Memory Reserve for Online
   Change' .................................................................................... 894
1.3.1.25.2.20 Command 'Start Page'.................................................. 895
1.3.1.25.2.21 Command 'Full Screen'................................................ 896
1.3.1.25.2.22 Command 'Properties'.................................................. 896

Standard Menu in View 'Devices', 'POUs', 'Modules'

The views "Devices", "POUs" and "Modules" provide the button in the top right corner to open a menu with the following commands:

- Open in editor: Opens the selected object in the corresponding editor.
- Find object: Opens the dialog "Find Object" for the object tree. Starting to enter a search string all matching objects will be displayed with their path. Use the button "Open" to open the selected search result in the editor.
- Sort by type: Sorts the objects in the view alphabetic by type.
- Sort by name: Sorts the objects in the view alphabetic by name.
- Sort ascending: Displays the chosen sorting in ascending order.
- Sort descending: Displays the chosen sorting in descending order.
- Track active editor: CODESYS selects the object, that is opened in the active editor, in the device tree of the view.

See also
- Chapter 1.3.1.25.2.3.2 “Command 'Devices'” on page 881
- Chapter 1.3.1.25.2.3.3 “Command 'POUs'” on page 882
- Chapter 1.3.1.25.2.3.4 “Command 'Modules'” on page 882

Command 'Devices'

Symbol: view: [Alt] + [D]

Function: The command opens the view "Devices" in the CODESYS main window. The view contains the project's "device tree", where you configure your applications

Button opens the standard menu for navigating in the tree view.

Call: Menu “View”
Command ‘POUs’

Symbol: 🟢, Shortcut: [Alt] + [1]

**Function:** This command opens the “POUs” view in the CODESYS main window. POUs located here are available in the entire project.

**Call:** Menu “View”

See also

- § Chapter 1.3.1.8.1 “Device tree and device editor” on page 155
- § Chapter 1.3.1.25.2.3.3 “Command ‘POUs’” on page 882
- § Chapter 1.3.1.25.2.3.1 “Standard Menu in View ‘Devices’, ‘POUs’, ‘Modules’” on page 881

Command 'Modules'

Symbol: 🟢

**Function:** This command opens the “Modules” view and shows the modules of the application composer in a tree structure.

**Call:** Main menu “View”

See also

- § Chapter 1.3.1.25.2.3.1 “Standard Menu in View ‘Devices’, ‘POUs’, ‘Modules’” on page 881
- Application Composer Module Tree

Command 'Messages'

Symbol: 🟢

**Function:** This command opens the “Messages” view.

**Call:** Menu bar: “View”.

View ‘Messages’

<table>
<thead>
<tr>
<th>Message category</th>
<th>The messages are categorized by component or functionality for selection from a drop-down list. Filter the message display by selecting a category.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Message type</td>
<td>Click the symbol of the message type to show or hide messages. CODESYS displays the number of messages next to each symbol.</td>
</tr>
</tbody>
</table>
|                  | - 🟢: Error  
|                  | - 🟢: Warning  
|                  | - 🟢: Message  |
| ✗                | Deletes all messages in the selected message category. |
| ✗                | Deletes all messages in all message categories. |
| “Description”    | Message text with the reported object and the location in the object. |
| “Project”        | Message text with the reported object and the location in the object. |
| “Object”         | Double-click a message in the table to jump to the source text location. |
| “Position”       | |
### Table 125: “Commands in the context menu”

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Next Message&quot;</td>
<td>The source text position of the next message is displayed.</td>
</tr>
<tr>
<td>&quot;Previous Message&quot;</td>
<td>The source text position of the previous message is displayed.</td>
</tr>
<tr>
<td>&quot;Go to Source Position&quot;</td>
<td>The source position of the selected message is displayed.</td>
</tr>
</tbody>
</table>

---

**Command 'Element properties'**

**Symbol:** 🗒️

**Function:** This command opens the “Element Properties” view.

**Call:** Main menu “View”

This command opens the properties view for the open object. This view is available only for a few objects, for example visualization and POU (SFC).

The properties are displayed in a structured table. You change the property values by clicking into the value fields. You can filter or sort the properties view.

---

**Command 'ToolBox'**

**Symbol:** 🎨

**Function:** This command opens the “ToolBox” view.

**Call:** Main menu “View”

This command opens the toolbox view for the open object. By default, this view is available for graphical editors and visualizations. It includes the graphical programming elements that you can drag into the editor.

---

**Command 'Watch' - 'Watch <n>'**

**Symbol:** 📈

**Function:** This command opens the "Watch <n>" view. You can populate a watchlist with variables from your project in order to monitor, force, or write these variable values in an individual view in online mode. The value "n" can be 1, 2, 3, or 4 for a total of up to four watchlists.

**Call:** Main menu “View”

See also

- [Chapter 1.3.1.13.1.2 “Using watch lists” on page 330](#)

---

**Command 'Watch' - 'Watch All Forces'**

**Symbol:** 📈

**Function:** The command opens the view “Watch All Forces”, which is a special form of a watch list.

**Call:** Menu “View ➔ Watch ➔ Watch all Forces”

**Requirement:** A project is in offline mode or in online mode.

The view contains all variables currently prepared for forcing, and all forced variables of the application in 1 list. The list allows the same actions as other watch lists. Additionally in the selection menu “Unforce...” the following commands are available:
● “Unforce and Keep All Selected Values”: For all selected list entries the variables will be set to the forced value and the forcing will be released.
● “Unforce and Restore All Selected Values”: For all selected list entries the variables will be reset to the values they had before they got forced, and the forcing will be released.

See also
● Chapter 1.3.1.12.4 “Forcing and writing variables” on page 315
● Chapter 1.3.1.13.1.2 “Using watch lists” on page 330

Command 'Add All Forces to Watchlist'

Function: The command adds all variables of the active application, which are currently prepared for forcing, or which are already forced, to the watchlist. Please regard, that this works only for docked watch list views.

Call: Context menu of view “Watch”

Requirement: Online mode, a watch list view is active.

There is a special watch list: “Watch All Forces”. This view shows automatically all variables currently prepared for forcing or already being forced. It provides additional commands for releasing any forces.

See also
● Chapter 1.3.1.13.1.2 “Using watch lists” on page 330
● Chapter 1.3.1.25.2.3.8 “Command ‘Watch’ - ‘Watch <n>’” on page 883
● Chapter 1.3.1.25.2.3.9 “Command ‘Watch’ - ‘Watch All Forces’” on page 883
● Chapter 1.3.1.12.4 “Forcing and writing variables” on page 315

Command 'Bookmarks'

Symbol: 📚

Function: This command opens the “Bookmarks” view.

Call: Menu bar: “View”.

| “Previous Bookmark” | Jumps to the bookmark that above the selected bookmark in the table and opens the respective POU in the editor. |
| “Next Bookmark” | Jumps to the bookmark that below the selected bookmark in the table and opens the respective POU in the editor. |
| X | Deletes the selected bookmark from the table and in the respective POU. |

List of bookmarks in the project with the following information: “Bookmark”, “Object”, and “Position”.

You can edit the bookmark order per drag&drop.

When you double-click a row, CODESYS opens the respective “Object” in the editor and jumps to this bookmark.

| “Bookmark” | Name of the bookmark as assigned by CODESYS in ascending numerical order: “Bookmark_0”, “Bookmark_2” etc. |
| “Bookmark” | If the bookmark is selected and you click in the field, then it is editable and you can modify the bookmark name. |
"Object" Name and project path of the POU where the bookmark is set
Example: POU_Add [PLC_1: SPS-Logic: Application]

"Position" Position of the bookmark in the POU
Example: Row 3, Column 1 (Impl)

(Impl): in the implementation part of the POU
(Decl): in the declaration part of the POU

See also
- Chapter 1.3.1.9.14.3 “Setting and using bookmarks” on page 234
- Chapter 1.3.1.25.2.2.24 “Command 'Next Bookmark’” on page 870
- Chapter 1.3.1.25.2.2.26 “Command ‘Previous Bookmark’” on page 870

Command 'Breakpoints'
Symbol: 🛡
Function: This command opens the “Breakpoints” view.
Call: Menu bar: “View”.
This view shows an overview of all defined breakpoints for an application. You have access to all breakpoint commands within this view.

Table 126: Table of current breakpoints

| "Application" | Select the required application from the list. |
| "POU" | Name of the function block that will receive the breakpoint |
| "Location" | Location of the breakpoint in the POU |
| ● Text editor: Line number and column number |
| ● Graphical editor: Network number or element number |
| For function blocks, "(Impl)" indicates that the breakpoint is located in the implementation of the function block, not in an instance. |
| "Instance Path" | Complete object path of the breakpoint location. |
| "Tasks" | Names of tasks that will be effective when the breakpoint is executed. If there are no restrictions, then "(all)" is displayed here. |
| "Condition" | ● "Break always": No additional enable condition defined; the breakpoint is always enabled. |
| ● Boolean expression. The expression must yield TRUE for the breakpoint to be enabled. |
| "Hit Count Condition" | When the breakpoint should be in effect (depending on the hit count) |
| "Current Hit Count" | How often the breakpoint has already been reached up to now during the execution |

Table 127: Toolbar

| "New Breakpoint" | This command opens the “Breakpoint Properties” dialog. |
| "New Breakpoint" | This command opens the “New Breakpoint” dialog. |
| "Clear Breakpoint" | Removes the breakpoint (not the same as disable) |

Command 'Breakpoints'
Symbol: 🛡
Function: This command opens the "Breakpoints" view.
Call: Menu bar: “View”.
This view shows an overview of all defined breakpoints for an application. You have access to all breakpoint commands within this view.

Table 126: Table of current breakpoints

| "Application" | Select the required application from the list. |
| "POU" | Name of the function block that will receive the breakpoint |
| "Location" | Location of the breakpoint in the POU |
| ● Text editor: Line number and column number |
| ● Graphical editor: Network number or element number |
| For function blocks, "(Impl)" indicates that the breakpoint is located in the implementation of the function block, not in an instance. |
| "Instance Path" | Complete object path of the breakpoint location. |
| "Tasks" | Names of tasks that will be effective when the breakpoint is executed. If there are no restrictions, then "(all)" is displayed here. |
| "Condition" | ● "Break always": No additional enable condition defined; the breakpoint is always enabled. |
| ● Boolean expression. The expression must yield TRUE for the breakpoint to be enabled. |
| "Hit Count Condition" | When the breakpoint should be in effect (depending on the hit count) |
| "Current Hit Count" | How often the breakpoint has already been reached up to now during the execution |

Table 127: Toolbar

| "New Breakpoint" | This command opens the “Breakpoint Properties” dialog. |
| "New Breakpoint" | This command opens the “New Breakpoint” dialog. |
| "Clear Breakpoint" | Removes the breakpoint (not the same as disable) |
| **“Enable/Disable Breakpoint”** | Toggles the status of the breakpoint or execution point between "enabled" and "disabled"
- ● Breakpoint enabled
- ○ Breakpoint disabled
- ● Execution point enabled
- ○ Execution point disabled
- ● Data breakpoint enabled
- ○ Data breakpoint disabled
- ● Data execution point enabled
- ○ Data execution point disabled

As opposed to "Clear breakpoint", a disabled breakpoint remains in the list and can be enabled again.

| “Properties” | The “Breakpoint Properties” dialog opens for editing the breakpoint parameters. This dialog is the same as “New Breakpoint”. In online mode, you can change the breakpoint into an execution point.

| “Go to Source Position” | Opens the online view of the affected block. The cursor is set at the breakpoint location.

| “Clear All Breakpoints” | Deletes all breakpoints and execution points in the application. The list is cleared. Not to be confused with "deactivate".

| “Enable All Breakpoints” | Enables all currently disabled breakpoints and execution points.

| “Disable All Breakpoints” | Disables all currently enabled breakpoints and execution points. The points remain in the list and can be enabled again.

See also
- % Chapter 1.3.1.25.3.5 “Dialog ‘Breakpoint Properties’” on page 1041
- % Chapter 1.3.1.25.2.7.4 “Command ‘New Breakpoint’” on page 939
- % Chapter 1.3.1.25.2.7.5 “Command ‘New Data Breakpoint’” on page 939
- % Chapter 1.3.1.25.2.7.7 “Command ‘Enable Breakpoint’” on page 940
- % Chapter 1.3.1.25.2.7.8 “Command ‘Disable Breakpoint’” on page 940
- % Chapter 1.3.1.25.2.7.9 “Command ‘Toggle Breakpoint’” on page 940

Command 'Cross Reference List'

**Symbol:** ☑

**Function:** This command opens the “Cross Reference List” view.

**Call:** Menu bar: “View”, or “Edit ➔ Browse ➔ Browse Cross References”.

This view shows a list of cross-references for a symbol in the project. The symbol can be a variable, a POU (program, function block, function), or a user-specific data type (DUT). The cross-reference list offers two basic types of searches:

- **Text search:** By specifying a symbol name, the cross-references of all symbols in the project are displayed with their names. If multiple symbols with the same name are found, then the display can be limited to individual declarations by means of the context menu.
- **Declaration search:** The symbol can be selected by means of the input assistant or by specifying a qualified path (for example, Device.Application.PLC_PRG.i or __POOL.POU.a). Then only the occurrence locations of this symbol are displayed, even if there exist other symbols with the same name.
### Input Field
Symbol name (variable name, POU name, DUT name). Input options:
- Selection of a declared symbol by means of the input assistant (🔍 button).
- Manual input of the symbol name. Triggering of the search by pressing the ✉️ button or the [Enter] key.

For the text search, you can use the placeholders "*" (any number of characters) or "?" (exactly any one character) in combination with a partial string of a variable identifier.

Use the percent sign "%" to search for IEC addresses. Examples: "%MW8", "%M".

More options outside of cross-reference list view:
- Use the command “Browse for Symbol ➔ Browse Cross References” if the name of a declared symbol is selected in an editor, or if the cursor is in the name field. A search is also possible if the object is selected in the device tree or POU pool.
- Automatic if the name of a declared symbol is selected in an editor, or if the cursor is in the name field. A automatic search is also possible if the object is selected in the device tree or POU pool.

Requirement: CODESYS option “Automatically list selection in cross reference view” is activated (category “SmartCoding”).

The following input is valid:
- Variable name, simple or qualified. Examples: "iVar", "PLC_PRG.iVar".
- POU name: Examples: "PLC_PRG", "myFB".
- DUT name: Example: "mySTRUCT".
- Strings combined with placeholders: asterisk (*) for any character or question mark (?) for exactly one characters).

Example: "iVar*" applies to iVar1, iVar_g1b2, iVar45, etc. "iVar?" refers to iVar1, iVar2, iVarX, and so on, but not iVar_g1b2, iVar45 and so on...
- "%<IEC address>": CODESYS searches for variables that are assigned to this address and direct memory access. Example: "%QB0", "%Q0 := 2."
"POU" | Block name; also a task name if a block call in the task configuration.
---|---
"Variable" | Only the variable name (for example, iVar)
---|---
"Access" | Type of access to the variable at the occurrence location: "Declaration"/ "Read"/ "Write"/ "Call".
Special case for pointers: An assignment type p := ADR(var1) is displayed as write | address when searching for var1. The reason for this: Any write access to p is not displayed when searching for var1. Write access is also possible by means of pointer variables.
---|---
"Type" | Data type of the variable
---|---
"Address" | IEC address if variables are assigned Example: "AT %QB0".
---|---
"Position" | Location of the occurrence in the POU editor, for example line number, network number, declaration part, or implementation part. Example: "line 1, column 1 (Impl)".
---|---
"Object" | POU name plus complete path of the occurrence location in brackets (if this is found in the "Devices" view). Example: "PLC_PRG [Device:Plc Logic:Application]"
---|---
"Comment" | Comments if available in the declaration of the variable

The search yields all result locations in the project and in included, uncompiled libraries.

### Right-click commands in the cross-reference list

- **Show source position**: Opens the respective POU and marks the occurrence: for root entries, the declaration, and for subordinate entries, the respective occurrence location. As an alternative, you can double-click a line.

- **Limit Results to Selected Declaration**: Limits the display of results to the selected symbol declaration if multiple declarations are found.

- **Expand All**: In the list, only the root nodes of the result locations are shown.

- **Collapse All**: In the list, every single result location is shown.

See also

- Chapter 1.3.1.25.2.29 “Command 'Browse Cross-References'” on page 871
- Chapter 1.3.1.9.14.1 “Using the cross-reference list to find occurrences” on page 232
- Chapter 1.3.1.25.2.23.3 “Command 'Limit Results to Current Declaration'” on page 1038
- Chapter 1.3.1.25.2.2.18 “Command 'Collapse All Folds'” on page 868
- Chapter 1.3.1.25.2.2.17 “Command 'Expand All Folds'” on page 868

### Command 'Browse Cross References in Classic View'

**Symbol:** 🔄

**Function:** This command opens the “Classic Cross Reference List” view.

**Call:** The command is not in any menu by default. You can add it to a menu by using the dialog box from “Tools ➔ Customize” (command category “Browse Project”).

The view corresponds to the “Cross Reference List” view before CODESYS V3.5 SP6.

### Command 'Call Stack'

**Symbol:** 🔄

**Function:** This command opens the “Call Stack” view.

**Call:** Main menu “Debug”.

---

888 3ADR010583, 1, en_US 2020/12/10
This view is very useful when you want to step into programs. It shows the current location with the complete call path.

<table>
<thead>
<tr>
<th>“Application”</th>
<th>Name of the active application that controls the current POU</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Task”</td>
<td>Name of the task that controls the current POU</td>
</tr>
<tr>
<td>“POU”</td>
<td>Name of the POU where program execution has halted</td>
</tr>
<tr>
<td></td>
<td>The first line in the list describes the current execution location (marked with a yellow arrow). If this location is in a block that is called by another block, then the call location is described in the second line. In turn, if the caller is called by yet another block, then that call location is described in the third line, and so on.</td>
</tr>
<tr>
<td>“Location”</td>
<td>Position within the POU where program execution has halted</td>
</tr>
<tr>
<td></td>
<td>● Line and column numbers for textual editors</td>
</tr>
<tr>
<td></td>
<td>● Network or element numbers for graphical editors</td>
</tr>
<tr>
<td>“Instance path”</td>
<td>Instance where program execution has halted</td>
</tr>
</tbody>
</table>

The call stack is also available in offline mode and normal online mode when you are not currently using any debugging functions. In this case, it receives the last displayed location during a stepped execution, but it is displayed in gray.

The “Call Tree” view, in contrast to the “Call Stack”, at any time provides information on the calls of a POU.

See also
● § Chapter 1.3.1.12.2 “Using breakpoints” on page 310
● § Chapter 1.3.1.25.2.3.16 “Command ‘Call tree’” on page 889

Command 'Call tree'

Symbol: 

Function: This command opens the “Call Tree” view.

Call:
● “View” menu
● Context menu of a callable block in the “Devices” or “POUs” view.

View ‘Call tree’
The call tree is available at all times, even before compiling the application. It is a static representation of the caller and the calls of the block that you specify explicitly. Therefore, the tree always contains two root nodes above the respective call order is displayed as successive indented entries. Recursive calls are detected quickly in this tree representation.

Example of a call tree (1) for the (2) PLC_PRG block:
- (3) Node “<block name> is called by:”
- (4) Node “<block name> calls:”

<table>
<thead>
<tr>
<th>“Block name”</th>
<th>Name of the program block; specified manually, by dragging from another view, or by means of the button ( \text{\textcolor{red}{\textbullet}} ). The drop-down list includes the last specified block names.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Toolbar and keyboard usage</td>
<td></td>
</tr>
<tr>
<td>: Find block</td>
<td>CODESYS searches for the block specified in “Block name” and displays its caller and calls.</td>
</tr>
<tr>
<td>Use block from the input assistant</td>
<td>The “Input Assistant” dialog box opens for selecting a block call or instance call. The call tree is refreshed automatically after the selection.</td>
</tr>
<tr>
<td>Show source code position of the selected block</td>
<td>CODESYS jumps to the occurrence location of the block in the source code of the program.</td>
</tr>
<tr>
<td>[F4]: Show source code position of the next block</td>
<td>The selection in the call tree jumps to the next or previous block in the call structure. At the same time, the associated source code position is opened in the respective editor. Note: Double-clicking an entry in the call tree also opens the associated source code position.</td>
</tr>
<tr>
<td>[Shift]+[F4]: Show source code position of the previous block</td>
<td></td>
</tr>
</tbody>
</table>

- Display of the call tree:

| “Symbol” | “<block name> is called by”: The call order is displayed for below this node. The bottom entry in this tree structure shows the start of the calls. “<block name> calls”: The calls from this block are displayed below this node. The bottom entry in this tree structure shows the end of the call chain. |
| “Position” | For the root node in the call tree: Line numbers of the declaration (“Decl”) of the block. For the caller or calls below the root node: Line number, column number, and network number of the position, depending on the implementation language. |

- Context menu for the entry selected in the tree:

| “Collapse All” | The expanded entries in the call tree are collapsed, except for the two root nodes. |
| “Show Source Position” | CODESYS jumps to the occurrence location of the block in the source code of the program. |
| “Set as New Root Node” | The entry selected in the call tree is displayed in “Block name”. The tree is refreshed automatically for the new root nodes. |
The “Call list” view is provided for immediate information when stepping through a program, as opposed to the static call tree that provides call information about a block. The call list always shows the full call path of the current position that is reached.

See also
- § Chapter 1.3.1.12.3 “Stepping through a program” on page 313
- § Chapter 1.3.1.25.2.3.15 “Command ‘Call Stack’” on page 888
- § Chapter 1.3.1.25.2.2.30 “Command ‘Browse Call Tree’” on page 871

Command 'Memory'

Symbol:

Function: In CODESYS V3.5 version earlier than SP11, the command opens the “Memory” view.

Call: Menu bar: “View → Memory”.

As of SP11, the command provides the notice that you must install the CODESYS Memory Tools package (available in the CODESYS Store) in order to use the memory view. After installation, you can open the “Memory” view by clicking “View → Show Memory View”.

Command 'Security Screen'

Symbol:

Function: The command opens the “Security Screen” view.

Call:
- “View” menu
- ☰ icon or ☰ in the status bar
  The icon is displayed in blue when a valid certificate is specified for the digital signature. When only one client certificate is specified for the encrypted communication, the icon remains gray, resulting in the client certificate providing no increased security for the user.

The following security features of CODESYS are configured and displayed in the view:
- Personal user certificate
- Encrypted communication
- Encryption and signatures of IEC projects
- Encryption and signature of download, online change, and boot application
- Security level

If the CODESYS Security Agent add-on product is installed, then the “Security Screen” view provides an additional “Devices” tab. This allows for the configuration of certificates for the encrypted communication with controllers.

Tab 'User'

On this tab, certificates are configured that are required for the encrypted communication and the digital signature of the user. Only certificates with private keys can be specified here. The user profile is saved as an XML file in the user options.
“User Profile and Certificate Selection”  
By default, the login name for Windows is specified as the user profile.

| List box with existing user profiles |  O: Opens the “User Profiles” dialog. Here you specify the name for a new user profile.  
|                                       | ✗: Deletes the selected user profile. This user profile is no longer displayed in the list box. |

“Digital Signature”  
O: Opens the “Certificate Selection” dialog for selecting the certificate for the digital signature.  
One certificate can be selected. The certificate must have a private key.  
✗: Deletes the displayed certificate.  
One certificate can be selected. The certificate must have a private key.

“Project File Decryption”  
O: Opens the “Certificate Selection” dialog for selecting the certificate for decrypting project files.  
One certificate can be selected. The certificate must have a private key.  
✗: Deletes the displayed certificate.

See also  
●  § Chapter 1.3.1.25.3.18 “Dialog ‘Certificate Selection’” on page 1098

Table 128: “Security Level”

<table>
<thead>
<tr>
<th>“Activate the Use of Certificates for Enhanced Security”</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Enforce encrypted communication”</td>
</tr>
</tbody>
</table>
| “Enforce encryption of project files” | ☑: All project files of the user are encrypted with a certificate. When the project is saved, it is encrypted with the certificate specified in the project settings (“Project Settings ➔ Security” dialog). The selected certificate is displayed on the “Project” tab in the “Project file encryption” group.  
To open this project, the certificate to be encrypted must be specified in “Project file decryption” with a private key. |
| “Enforce signing of project files” | ☑: All project files of the user are signed with a certificate. In “Digital Signature”, a certificate must be specified with a private key.  
When a project is saved, a signature file <project name>.project.p7s is generated in the project directory containing the signature. |
| “Enforce encryption of downloads, online changes and boot applications” | ☑: The data that is downloaded to the controller must be encrypted with a controller certificate.  
This certificate is defined directly either in the properties dialog of the application on the “Encryption” tab, or in the security screen, on the “Project” tab, in the “Encryption of Boot Application, Download and Online Change” group.  
Controller certificates are located in the local Windows Certificate Store in the “PLC Certificates” directory. If the certificates of your controller are not available in the directory, then they first have to be loaded from the controller and installed to the directory. For instructions, see the “Controller Certificates” chapter. |
**“Enforce signing of downloads, online changes and boot applications”**

![Image](image1.png)

The online code (downloads, online changes, and boot applications) must be signed with a certificate with a personal key. The certificate is selected from the "Digital signature" area.

**Requirement:** The “Encryption of boot application, download and online change” option is selected.

**“Enforce signing of compiled libraries”**

![Image](image2.png)

The "File ➔ Save Project as Compiled Library" command generates a signed library `<library name>.compiled-library-v3`.

**Requirements**
- A certificate with a private key that supports code signing is available.
- A library compatibility >= CODESYS V3 SP15 is set in the project information.

**See also**
- "Chapter 1.3.1.25.2.1.7 “Command ‘Save Project as Compiled Library’” on page 856"
- "Chapter 1.3.1.19.1 “Information for library developers” on page 372"

**Tab ‘Project’**

All project-specific settings are configured on this tab. These elements are active only when a primary project is loaded.

**“Project file encryption”**

**“Technology”**

![Image](image3.png)

Opens the “Project Settings ➔ Security” dialog

When you select the “Encryption” project setting and then “Certificates” in the dialog, you can choose a corresponding certificate by clicking ![Image](image4.png). For more information, see the description of the “Project Settings: Security” dialog.

**“Certificates of Users Sharing this Project”**

Area for listing the certificates that encrypt the project file.

**“Encryption of Boot Application, Download and Online Change”**

List of the applications of the controller

Double-clicking an application in the list opens the “Properties ➔ Encryption” dialog. Depending on the settings of the "Security Level" on the "User" tab of the security screen, the following fields are available in the open properties dialog:
- "Encryption" tab with active "Certificates" area
- "Encryption" tab with "Encryption Technology" list box.

In the "Properties ➔ Encryption" dialog, click the ![Image](image5.png) button to select the controller certificate for "Encryption of Boot Application, Download and Online Change". For more information, see the description of the "Properties: Encryption" dialog.

Controller certificates are located in the local Windows Certificate Store in the "PLC Certificates" directory. If the certificates of your controller are not available in the directory, then they first have to be loaded from the controller and installed to the directory. For instructions, see the "Protecting and Saving a Project" - "Encryption with Certificates" chapter.

**See also**
- "Chapter 1.3.1.25.3.11.7 “Dialog ‘Project Settings’ - ‘Security’” on page 1064"
- "Chapter 1.3.1.25.3.10.3 “Dialog ‘Properties’ - ‘Encryption’” on page 1047"
- "Chapter 1.3.1.6.7 “Encrypting projects with certificates” on page 149"
Tab 'Devices'

This tab is available only after you have installed the CODESYS Security Agent add-on. For a description of this tab, see the help for the CODESYS Security Agent.

Command 'Settings of Memory Reserve for Online Change'

**Function:** This command opens the “Online Change Memory Reserve” view.

**Call:** Menu bar: “View”.

In the view, memory reserves are configured for the function blocks during the online change.

| “Scan Application” | - Searches the selected application for function blocks and displays them in the “Function blocks” area  
| - Updates the “Function blocks” area after the application is built again.  
| - Updates the “Function blocks” area after an online change. |

Drop-down list with the applications of the open project: Selection of the application whose function blocks should be displayed and/or edited in this view.

<table>
<thead>
<tr>
<th>Table 129: “Function Blocks”</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>“All”</strong></td>
</tr>
<tr>
<td><strong>“Pool”</strong></td>
</tr>
<tr>
<td><strong>“No memory-reserve”</strong></td>
</tr>
<tr>
<td><strong>“&lt;memory reserve&gt; bytes”</strong></td>
</tr>
</tbody>
</table>

Information about the function blocks:

Multiple selection is also possible when selecting a POU for the configuration of the memory reserve.

| **“Function block”** | Name of the function block |
|---|
| **“Size”** | Size of the function block  
Size of an instance of a function block  
Specified in bytes |
| **“Number of instances”** | Number of instances of a function block in the project |
| **“Memory reserve”** | Display of the memory reserve for each instance of the function block |
| **“Additional memory for all instances”** | Product of “Number of instances” and “Memory reserve” |
| **“Remaining memory reserve”** | Number of bytes that are available as reserve. |
Table 130: “Settings”

| **Memory reserve (in bytes)** | Input field for the memory reserve for the selected function block. Specified in bytes
| Requirement: the application is not located on the controller yet or you have allowed the memory reserve to be changed by clicking the “Edit” button in the “Allow editing” area. |
| **Apply for Selection** | The “Memory reserve (in bytes)” is assigned to the function block and the table column “Memory Reserve” is updated. In multiple selection, the specified value is assigned to each function block. In order to update the columns “Size”, “Number of Instances”, “Additional Memory for All Instances”, and “Remaining Size of the Memory Reserve”, click “Build ➔ Build”, and then click the “Scan Application” button. |

Table 131: “Enable Editing”

| **Enable** | The input field “Memory reserve (in bytes)” is editable. This button is modified in “Editable”. |

Table 132: “Information”

| **Number of FBs** | Total number of function blocks in the application |
| **Additional memory for all instances** | Sum of the memory reserves of all function block instances of the application. Specified in bytes |

See also

- § Chapter 1.3.1.9.4 “Configuring the memory reserve for an online change” on page 206
- § Chapter 1.3.1.25.2.6.6 “Command ‘Online Change’” on page 928

Command ‘Start Page’

Symbol: 🗯

Function: This command opens the “Start Page” view.

Call: Main menu “View”

The view includes some basic commands and a list of recently opened projects. In addition, the CODESYS homepage is displayed.

If you access the Internet through a proxy, then you can save the authentication data in the project options (“Proxy Settings”) so you do not have to provide this data every time you use this command.

By moving the mouse pointer over the list of recently opened projects, you can remove or pin individual projects in the list. Pinned projects remain in this list until you remove the pin.

In the project options (“Load and Save”), you can configure whether this start page should open automatically when you start CODESYS.

See also

- § Chapter 1.3.1.25.3.13.13 “Dialog ‘Options’ – ‘Load and Save’” on page 1083
- § Chapter 1.3.1.25.3.13.16 “Dialog ‘Options’ - ‘Proxy Settings’” on page 1084
Command 'Full Screen'

Symbol: ☰, keyboard shortcut [Ctrl]+[Shift]+[F12]

**Function:** This command switches the CODESYS display to full screen mode.

**Call:** Main menu “View”

Choosing this command displays the main window of the CODESYS user interface in full-screen mode. You can return to the previous setting by choosing the command again or with the keyboard shortcut [Ctrl]+[Shift]+[F12].

Command 'Properties'

Symbol: ☐

**Function:** This command opens the properties of the currently selected object in the POUs tree or device tree.

**Call:** Main menu “View”
Menu ‘Project’

1.3.1.25.2.4.1 Command ‘Add Object’............................................................. 897
1.3.1.25.2.4.2 Command ‘Add Folder’.............................................................. 897
1.3.1.25.2.4.3 Command ‘Insert Device’........................................................... 899
1.3.1.25.2.4.4 Command ‘Plug Device’............................................................ 899
1.3.1.25.2.4.5 Command ‘Scan for Devices’.................................................... 901
1.3.1.25.2.4.6 Command ‘Update Device’........................................................ 901
1.3.1.25.2.4.7 Command ‘Edit Object’............................................................ 901
1.3.1.25.2.4.8 Command ‘Edit Object with’....................................................... 902
1.3.1.25.2.4.9 Command ‘Edit Object (Offline)’................................................ 902
1.3.1.25.2.4.10 Command ‘Set Active Application’........................................... 902
1.3.1.25.2.4.11 Command ‘Project information’................................................ 902
1.3.1.25.2.4.12 Command ‘Project Settings’.................................................... 902
1.3.1.25.2.4.13 Command ‘Project Environment’............................................. 903
1.3.1.25.2.4.14 Command ‘Project Localization’ - ‘Create Localization Template’........................................................................................................ 903
1.3.1.25.2.4.15 Command ‘Project Localization’ - ‘Manage Localizations’........... 904
1.3.1.25.2.4.16 Command ‘Project Localization’ - ‘Toggle Localization’.............. 905
1.3.1.25.2.4.17 Command ‘Document’ ............................................................... 905
1.3.1.25.2.4.18 Command ‘Compare’................................................................. 905
1.3.1.25.2.4.19 Command ‘Commit Accepted Changes’.................................... 910
1.3.1.25.2.4.20 Command ‘Export’................................................................. 910
1.3.1.25.2.4.21 Command ‘Import’................................................................. 910
1.3.1.25.2.4.22 Command ‘Export PLCopenXML’............................................ 911
1.3.1.25.2.4.23 Command ‘Import PLCopenXML’............................................ 911
1.3.1.25.2.4.24 Command ‘User management’ – ‘Log in User’.......................... 911
1.3.1.25.2.4.25 Command ‘User management’ – ‘Log out User’.......................... 912
1.3.1.25.2.4.26 Command ‘User management’ – ‘Rights…’............................... 912
1.3.1.25.2.4.27 Command ‘Insert Device’........................................................... 912
1.3.1.25.2.4.28 Command ‘Generate EtherCAT XML’........................................ 913
1.3.1.25.2.4.29 Command ‘Generate Sercos SCI XML’...................................... 913
1.3.1.25.2.4.30 Command ‘Disable Device’ – ‘Enable Device’......................... 913
1.3.1.25.2.4.31 Command ‘Edit I/O Mapping’.................................................. 914
1.3.1.25.2.4.32 Command ‘Import Mappings from CSV’.................................... 914
1.3.1.25.2.4.33 Command ‘Export Mapping to CSV’......................................... 915
1.3.1.25.2.4.34 Command ‘Read PLC Parameter File to Configuration’............. 915
1.3.1.25.2.4.35 Command ‘Online Config Mode’............................................. 915

Command ‘Add Object’

Symbol: 

**Function:** This command opens a submenu with objects that contain all objects that can be inserted, depending on the current position in the “Devices” or “POUs” view.

**Call:** “Project” menu, context menu in the “Devices” or “POUs” view.

**Requirement:** If CODESYS is to insert the object in the device tree, select an already existing object under which the new one is can be inserted indented. If CODESYS is to insert the object in the POUs tree, set the focus in any free place in the CODESYS window.

Command ‘Add Folder’

Symbol: 

2020/12/10  3ADR010583, 1, en_US  897
Function: This command opens a dialog box for defining a new folder in the Devices or POUs view.

Call: “Project” menu, context menu in the Devices or POUs view

You cannot structure the arrangement of device nodes and device objects through folders that you have created yourself.

This command inserts the folder below the object that has just been selected in the tree. If no object is selected, CODESYS inserts the folder right at the top in the tree directly under the root node.

Command ‘Insert Device’

Function: this command opens the dialog box “Add Device” for the selection of a device object that is to be inserted in the device tree below the currently selected object.

Call: Context menu of a device object in the device tree.

Requirement: An object is selected in the device tree below which a device object can be inserted.

See also
- § Chapter 1.3.1.8 “Configuring I/O links” on page 155
- § Chapter 1.3.1.8.1 “Device tree and device editor” on page 155

Dialog box ‘Add device’

Function: Depending on the currently selected position in the device tree, the dialog box offers a selection of the devices that can be inserted at this point. In addition, it contains the commands also available in the context menu: “Insert Device”, “Add Device”, “Plug Device”, “Update Device”.

Requirement: The devices are installed in the device repository on the local system.

“Name” Name with which the device is to appear in the device tree. Must be a valid IEC identifier.

Table 133: “action”

| “Add device”      | CODESYS inserts the selected device indented below the selected object in the device tree. |
| “Insert device”   | CODESYS inserts the selected device at the same level as the selected object below it in the device tree. |
| “Plug device”     | CODESYS inserts the selected device in the selected slot. If the slot is already occupied, the existing module is replaced by the new one. |
| “Update device”   | CODESYS replaces the device selected in the device tree by the one selected. Please note: Depending on the device, this may cause the configuration already done in the device editor to be overwritten with the default values! |
### String for full text search
This field is editable after clicking in it. For any character string entered, only those devices that include the character string are displayed in the lower view. The matched string is highlighted in yellow for these devices.

### Vendor:
Drop-down list with manufacturers whose available devices are displayed.

### Group by category
- **☑**: The available devices (newest version) are sorted by category. The category is defined in the device description file.
- **☐**: The available devices appear flat and alphabetically sorted.

### Display all versions (for experts only)
- **☑**: In addition, all other available versions of the devices can also be selected.
- **☐**: Only the newest version of each device is available for selection

### Display outdated versions
- **☑**: In addition, outdated versions of the devices can also be selected. Outdated versions result, for example, from the update of plug-ins.
- **☐**: Outdated device versions are not displayed.

The information provided by the device description file is displayed:
- device name, vendor, categories, version, order number and a short description, device-specific bitmap.

See also
- ☐ Chapter 1.3.1.25.2.4.27 “Command ‘Insert Device’” on page 912
- ☐ Chapter 1.3.1.25.2.4.4 “Command ‘Plug Device’” on page 899
- ☐ Chapter 1.3.1.25.2.4.6 “Command ‘Update Device’” on page 901

### Command ‘Plug Device’

**Function**: Like the command “Add Device”, this command opens the dialog box “Add Device” for the selection of a device object that is to be inserted in the device tree in the currently selected slot.

**Call**: Context menu of the slot of a device object in the device tree.

**Requirement**: The slot of a device object is selected in the device tree.

An empty slot is identified by the symbol and the entry "<empty> (empty)". An occupied slot is given the symbol and the name of the device.

In the case of an occupied slot, this command replaces the existing module with the new one.

See also
- ☐ Chapter 1.3.1.25.2.4.3 “Command ‘Insert Device’” on page 898
- ☐ Chapter 1.3.1.8 “Configuring I/O links” on page 155
- ☐ Chapter 1.3.1.8.1 “Device tree and device editor” on page 155

### Command ‘Scan for Devices’

**Function**: This command establishes a brief connection to the hardware and determines the devices in the network. You can then adopt the devices found into the device tree of your project.

**Call**: Main menu “Project”, context menu of a device object in the device tree.

**Requirement**: The settings for communication with the control device are correct. The gateway and the PLC are running. The device supports the scan function.
To use the scan functionality, you must log in once in order to load the library to the controller.

The command refers to the master controller selected in the device tree. For example, an already inserted PROFINET IO controller can be selected and the command used to determine the I/O devices and I/O modules assigned to it.

After scanning, the dialog box “Scan for Devices” appears with the results.

Table 134: “Scanned Devices”

| “Device name, Device type, Address, Station name, etc.” | Data about the scanned device depending on network type. When you change a value in the list of scanned devices, the value is shown in italics. This indicates that the new value has been changed in the editor in CODESYS, but not in the device. When you download the value to the device, it is shown normally. Value that indicate differences between the project and the scanned device are shown in orange. If several device descriptions are available for the scanned device, then the name is displayed in bold. The selection of the matching device description is resolved differently for different fieldbuses. Details can be found in the corresponding fieldbus sections. If a device description cannot be found, then the following message is shown: "Attention! The device was not found in the repository." Depending on the bus system, additional information is displayed, such as manufacturer number and product number. The device cannot be inserted into the project without the installed device description. |

| “Show differences to project” | □: The table in the dialog also shows additional configured devices (in the device tree of the project). □: The table shows all scanned devices. The configured devices are not shown. |

| “Scan Devices” | Starts a new search. |

| “Copy all devices to project” | The device that is selected in the table is inserted into the device tree in the project. If nothing is selected, then all scanned devices are shown. |

NOTICE!

If you insert devices, which are available in the device tree, to the device tree with “Copy All Devices to Project”, then the following should be noted. The data of the “Process Data” and “<...> I/O Mapping” tabs of the existing devices can be overwritten with the data of the recently inserted devices.

Table 135: “Configured Devices”

This part of the dialog is visible only when you select the “Show differences to project” option. Differences between the scanned and configured devices are color-coded. Devices in green are identical on both sides. Device in red are available only in the view of the scanned or configured devices.

If you have selected a device in both views, then the scanned devices are inserted above the selected configured device.
If you have selected a device in both views, then the scanned devices are inserted below the selected configured device.

If you have selected a device in both views, then the configured devices are replaced by the selected scanned device.

All scanned devices are copied to the project.

Deletes the selected configure device.

The dialogs for the scan differ depending on the type of device. Please refer to the help pages for the respective device editor.

See also

- "Scanning the current hardware and applying devices into the project" on page 160

Command 'Update Device'

Function: Like the command “Add Device”, this command opens the dialog box “Add Device” for the selection of a device object. This object is inserted in the device tree in place of the currently selected object.

Call: Context menu of a device object in the device tree.

Requirement: An object is selected in the device tree below which a device object can be inserted.

With this command you can insert either a different version of a device or a different type of device in place of the previous one.

The symbolic device name used in the device tree is retained, but the device type specified in parentheses behind it changes if a different type has been selected. Thus if only the device version is changed, the object entry appears unchanged.

If the device type does not change, the configuration tree indented below the device entry concerned is retained. In this case the configuration settings also remain the same. Inconsistencies in the configuration resulting from the device update are reported by CODESYS at the next compilation of the application. This also concerns implicitly inserted libraries, which CODESYS does not remove accordingly during a device update.

See also

- Chapter 1.3.1.8 "Configuring I/O links" on page 155
- Chapter 1.3.1.25.2.4.3 "Command 'Insert Device'" on page 898

Command 'Edit Object'

Function: This command opens the object in its editor.

Call: Main menu “Project”, context menu.

Requirement: An object is selected in the device tree or in the “POUs” view.
Command 'Edit Object with'

**Function:** When multiple objects are available for an object, this command opens a dialog box for selecting an editor.

If only one editor is available for an object, then this command opens the object in that editor.

**Call:** Main menu “Project” or shortcut menu (right-click)

**Requirement:** An object is selected in the device tree or in the “POUs” view.

In the standard installation of CODESYS, there is no object that has multiple available editors.

Command 'Edit Object (Offline)'

**Function:** The command opens the object offline in the editor.

**Call:** Main menu “Project”, Context menu

**Requirement:** The application is in online mode. An object is selected in the device tree or in the “POUs” view.

The command allows you to edit objects in online mode. After editing you transfer the changes to the controller by use of the command “Online ➔ Online Change” or “Online ➔ Load”.

See also

- § Chapter 1.3.1.25.2.6.6 “Command ‘Online Change’” on page 928
- § Chapter 1.3.1.25.2.6.5 “Command ‘Load’” on page 927

Command 'Set Active Application'

**Function:** This command sets the selected application as the active application.

**Call:** Main menu “Project”, or right-click the “Application” object.

**Requirement:** The project has at least two applications. The selected application is not active.

Online actions apply only to the active application. The name of an active application is displayed in bold typeface in the device tree.

Command 'Project information'

**Symbol:**  

**Function:** This command opens the dialog box “Project Information”.

**Call:** Main menu “Project”

When you execute the command in the project for the first time, CODESYS creates the “Project Information” object.

See also

- § Chapter 1.3.1.3.6.1 “Retrieving and editing project information” on page 132
- § Chapter 1.3.1.3.6.2 “Making project settings” on page 134
- § Chapter 1.3.1.25.1.31 “Object ‘Project Information’” on page 811

Command 'Project Settings'

**Symbol:**  

**Function:** This command opens the “Project Settings” dialog box.

**Call:** “Project” menu or double-click on the object “Project Settings” in the “POUs” view
Requirement: A project is open.

See also
- Chapter 1.3.1.3.6.2 “Making project settings” on page 134
- Chapter 1.3.1.25.3.11 “Dialog 'Project Settings'” on page 1059
- Chapter 1.3.1.25.1.30 “Object 'Project Settings'” on page 810

Command 'Project Environment'

Function: This command opens the “Project Environment” dialog box.

Call: “Project” menu

Requirement: A project is open.

this command is for checking the currentness of software and files integrated in the project and enables them to be updated.

See also
- Chapter 1.3.1.25.3.12.1 “Dialog 'Project Environment' – 'Library Versions'” on page 1070
- Chapter 1.3.1.25.3.12.6 “Dialog 'Project Environment' – 'C Code Modules'” on page 1072
- Chapter 1.3.1.25.3.12.2 “Dialog 'Project Environment' - 'Compiler Version'” on page 1070
- Chapter 1.3.1.25.3.12.3 “Dialog 'Project Environment' - 'Device Versions'” on page 1071
- Chapter 1.3.1.25.3.12.4 “Dialog 'Project Environment' – 'Visualization Profile'” on page 1071
- Chapter 1.3.1.25.3.12.5 “Dialog 'Project Environment' – 'Visualization Styles'” on page 1072
- Chapter 1.3.1.25.3.12.7 “Dialog 'Project Environment' – 'Visualization Symbols'” on page 1073

Command 'Project Localization' - 'Create Localization Template'

Function: This command opens the “Create Localization Template” dialog. Define here which information should be exported from the project to a translation template (*.pot file).

Call: Menu bar: “Project ➤ Project Localization”.

Requirement: A project is open.

Dialog 'Create Localization Template'

This dialog is used for selecting the textual information that should be used in the localization template.

Table 136: “Include the Following Information”

<table>
<thead>
<tr>
<th>“Names”</th>
<th>Texts, such as dialog captions and object names in the device tree</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Identifier”</td>
<td>Variable identifier (example: Counter)</td>
</tr>
<tr>
<td>“Strings”</td>
<td>Example: 'count' in the following declaration: strVar: STRING :=</td>
</tr>
<tr>
<td></td>
<td>'count';</td>
</tr>
<tr>
<td>“Comments”</td>
<td>Comment texts in the POU</td>
</tr>
</tbody>
</table>
**"Position information"**

Selection of which positions of the selected text categories in the project should be included in the translation file. The position information is located in the first line(s) of a segment for a translation. Example:

```
#: D:\Proj1.project\Project_Settings:1
msgid "Project settings"
msgstr ""
```

- **“All”**: All detected positions of the text are listed.
- **“First appearance”**: In the translation file, the position is included in the project where the text to be translated appears for the first time.
- **“None”**

**“Generate”**

This button opens the dialog for saving a file. The translation template is created in a text file of type *.pot (portable object template). Each further generation creates a completely new template file.

See also

- Help about CODESYS Visualization: Multi-language capability

---

**Command ‘Project Localization’ - ’Manage Localizations’**

**Function**: This command opens the “Manage localizations” dialog. Select the desired localization language in the dialog or the original version of the project. You can still accept the localization files *.<language>.po into the project or remove them.

**Call**: Menu bar: “Project ➔ Project localization”.

**Requirement**: A project is open.

---

**Dialog 'Manage localizations'**

**“Available Localizations”**

List of the localization files available in the project. Example:

```
proj1-de.po
proj1-en.po
<original version>
```

The original version is always available. The project can be edited only in the original version.

**“Add”**

This button opens the dialog for selecting an additional po file from the file system.

**“Remove”**

This button removes the po file, which is selected on the left side, from the project.

**“Default localization”**

☑: The selected localization is for the default localization. The entry is display in bold.

**“Switch Localization”**

Use this button to switch to the selected localization.

**“OK”**

The project is displayed in the language that is provided by the file selected below the files. If you select “<original version>”, then the project appears in the editable non-localized version.

See also

- Help about CODESYS Visualization: Multi-language capability ᵀᵉᵃⁿᵉᵉ Chapter 1.3.5.6 “Setting up multiple languages” on page 1143
Command 'Project Localization' - 'Toggle Localization'

Symbol: 

**Function:** This command switches between the currently set project localization and the original version.

**Call:** Menu bar: “Project ➤ Project Localization”; button in the “Manage Localizations” dialog; button on the toolbar.

**Requirement:** A project is open. A default localization for the project is defined in the “Manage Localizations” dialog.

See also
- Help about CODESYS Visualization: Multi-language capability
- Chapter 1.3.1.25.2.4.15 “Command 'Project Localization' - 'Manage Localizations’” on page 904

Command 'Document'

Symbol: 

**Function:** This command opens the “Document Project” dialog box, where you can define the project documentation. This includes the selection of objects in the open project that you want to print.

**Call:** Main menu “Project”

<table>
<thead>
<tr>
<th>Table 137: “Document Project” dialog box</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>“Please select the objects which are to be printed”</strong></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td><strong>“Title page”</strong></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td><strong>“Table of contents”</strong></td>
</tr>
<tr>
<td><strong>“Preview”</strong></td>
</tr>
<tr>
<td><strong>“Select”</strong></td>
</tr>
<tr>
<td><strong>“Deselect”</strong></td>
</tr>
<tr>
<td><strong>“OK”</strong></td>
</tr>
</tbody>
</table>

See also
- Chapter 1.3.1.25.3.11.6 “Dialog 'Project Settings' - 'Page Setup’” on page 1064
- Chapter 1.3.1.25.2.1.12 “Command 'Print’” on page 860

Command 'Compare'

Symbol: 

Table 137: “Document Project” dialog box
**Function:** This command opens the “Project Comparison” dialog. In this dialog, you define the reference project to compare with the current project. You configure the comparison process by means of options. When the dialog is exited, the comparison starts and the result is shown in the view “Project Compare - Differences”.

**Call:** Menu bar: “Project ➔ Compare”.

**Requirement:** A project is open.

See also
- ❘ Chapter 1.3.1.5 “Comparing projects” on page 137
- ❘ Chapter 1.3.1.25.2.4.19 “Command ‘Commit Accepted Changes’” on page 910

### Dialog 'Project Comparison'

**Table 138: “Compare the currently open project with:”**

<table>
<thead>
<tr>
<th>“Project on disk”</th>
<th>Path of the reference project on the file system.</th>
</tr>
</thead>
</table>
| “Project in a source control database” | “Host”: Name of the host where the source code management is located.  
“Port”: Number of the port for connecting to the source code management.  
“Location”: Path of the reference project.  
Requirement: The project is linked to source code management (for example, CODESYS SVN). |

**Table 139: “Compare Options”**

| “Ignore whitespace” | ☑: Whitespace differences between the current project and the reference project are ignored. |
| “Ignore comments” | ☑: Comments in the programming code are excluded from the comparison. |
| “Ignore properties” | ☑: Object properties are excluded from the comparison. |

| “OK” | Starts the project compare and displays the result in the view “Project compare - Differences”.

### View 'Project Comparison' - 'Differences'

The project compare view opens when you click “OK” to close the “Project Compare” dialog.
(1) Object tree of the current project
(2) Object tree of the reference project
(3) Command 'Accept Block', command 'Accept Single'
(4) Compare options, configured in 'Project Compare' dialog
   Compare statistics: added, deleted, and changed objects

Table 140: Toolbar

<table>
<thead>
<tr>
<th>Icon</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Switches to the detailed compare view “Project Comparison” - ‘&lt;object name&gt;’</td>
</tr>
<tr>
<td></td>
<td>Differences” for the object selected in the tree. Alternative: Double-click</td>
</tr>
<tr>
<td></td>
<td>the object.</td>
</tr>
<tr>
<td></td>
<td>Selects the next bottom object in the device tree where differences were</td>
</tr>
<tr>
<td></td>
<td>detected.</td>
</tr>
<tr>
<td></td>
<td>Selects the next top object in the device tree where differences were</td>
</tr>
<tr>
<td></td>
<td>detected.</td>
</tr>
<tr>
<td></td>
<td>“Accept Block”</td>
</tr>
<tr>
<td></td>
<td>The block (selected object with all subordinate objects and units) is</td>
</tr>
<tr>
<td></td>
<td>selected for acceptance from the reference block to the current block.</td>
</tr>
<tr>
<td></td>
<td>Repeated clicking of “Accept Block” undoes the effects of its last change.</td>
</tr>
<tr>
<td></td>
<td>“Accept Single”</td>
</tr>
<tr>
<td></td>
<td>The object is selected in the current object for acceptance from the</td>
</tr>
<tr>
<td></td>
<td>reference line.</td>
</tr>
<tr>
<td></td>
<td>Requirement: The properties, access rights, or contents of the objects</td>
</tr>
<tr>
<td></td>
<td>selected in the object tree are different.</td>
</tr>
<tr>
<td></td>
<td>Opens the “Accept” dialog.</td>
</tr>
</tbody>
</table>
Table 141: Display of differences with colors, and symbols

<table>
<thead>
<tr>
<th>Font Style</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Black font</td>
<td>Objects are identical.</td>
</tr>
</tbody>
</table>

Object name with 🔄: Child objects of the object are different

<table>
<thead>
<tr>
<th>Font Style</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gray highlight</td>
<td>Objects are different.</td>
</tr>
<tr>
<td>Gray highlight + bold blue font</td>
<td>Object is only in the reference project.</td>
</tr>
<tr>
<td>Gray highlight + bold green font</td>
<td>Object is only in the open project (not in reference project).</td>
</tr>
<tr>
<td>Gray highlight + red font + 🔄</td>
<td>Object has different properties.</td>
</tr>
<tr>
<td>Gray highlight + red font + 🔄</td>
<td>Access rights of object and reference object are different.</td>
</tr>
<tr>
<td>Gray highlight + bold red font + 🔄</td>
<td>Implementation of objects is different. Double-click the line to display the object-specific compare view.</td>
</tr>
</tbody>
</table>

Yellow highlight | Object is activated for acceptance. |

Yellow highlight + 🔄: Adding the reference object to the open project is activated.

Yellow highlight + ✗: Deleting the object (in the open project) is activated.

Yellow highlight + 🔄: Acceptance of the properties of the reference project is activated.

Yellow highlight + red font + 🔄: Acceptance of the access rights of the reference project is activated.

Gray highlight + bold red font + 🔄: Acceptance of the implementation of the reference project is activated.

"Compare options" | Defined comparison options in the "Project Comparison" dialog.

"Compare statistics" | Number of additions, deletions, and changes in the current project, as compared to the reference project. "Change" means differences of an object available in both projects.

The dialog prompt opens: "Do you want to commit the changes which you made in the diff view?"

"Yes": The contents, properties, or access rights of the objects highlighted in yellow are modified in the project. Now they correspond to the reference project. Then the project compare view is closed completely.

View 'Project Comparison' - '<object name> Differences'

Function: Detail compare view

Call in the project compare view:
- Select an object that is identified as having different contents which you need to view in detail. Click 🔄.
- Double-click the object.

Table 142: Toolbar

<table>
<thead>
<tr>
<th>Toolbar</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>🔄</td>
<td>Switch back to the project compare view.</td>
</tr>
<tr>
<td>✗</td>
<td>Selects the next line below in the code where differences were detected.</td>
</tr>
</tbody>
</table>
Selects the next line above in the code where differences were detected.

"Accept Block" The block (with all subordinate lines) is selected for acceptance of the reference blocks into the current project.

A block in the detailed compare view consists of the unit where the cursor is located and all corresponding units that have the same difference markers. A unit is a line, network, or element. Subsequent lines of a line are examples of corresponding units.

Repeated clicking of "Accept Block" undoes the effects of its last change.

"Accept Single" The line is selected in the current object for acceptance of the reference line.

Switches between the default display where different units (lines, networks, elements) are displayed in red and another display: The units are displayed as recently added in the open project. In the reference project, they are displayed as deleted.

Available within the detailed compare view only.

Note: Depending on the display, detected differences in the statistics are counted as changed, inserted, or deleted.

Table 143: Display of differences with colors, and symbols

<table>
<thead>
<tr>
<th>Style</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Black font</td>
<td>Objects are identical.</td>
</tr>
<tr>
<td>Gray highlight + bold blue font</td>
<td>Code is only in the reference project.</td>
</tr>
<tr>
<td>Gray highlight + bold green font</td>
<td>Code is only in the current project (not in reference project).</td>
</tr>
<tr>
<td>Yellow highlight</td>
<td>The object is activated for acceptance.</td>
</tr>
</tbody>
</table>

The dialog prompt opens: “Do you want to commit the changes which you made in the diff view?”

“Yes”: The code highlighted in yellow is accepted into the project. The code corresponds to the reference project. Then the detailed view is closed and the project view is displayed. You can continue working with project compare.

Dialog 'Accept'

Table 144: “Which meta data should be accepted?”

<table>
<thead>
<tr>
<th>Meta data</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Access rights&quot;</td>
<td>[✓]: Access rights that are selected for acceptance.</td>
</tr>
<tr>
<td>&quot;Accepted groups&quot;</td>
<td>Grouping with access rights accepted by the reference project. A group is accepted if it is present in both projects with different access rights. Example: Group A</td>
</tr>
<tr>
<td>&quot;Unaccepted groups (missing in a project)&quot;</td>
<td>The group is not accepted if it is not present in one of the two projects.</td>
</tr>
<tr>
<td>&quot;Properties&quot;</td>
<td>[✓]: Properties activated for accept</td>
</tr>
<tr>
<td></td>
<td>Requirement: The properties of the reference object and object are different.</td>
</tr>
</tbody>
</table>
“OK” Settings are accepted.

Command 'Commit Accepted Changes'

Symbol: ✚

Function: This command commits the accepted differences from the project comparison to the current project.

Call: “Project ➔ Commit Accepted Changes”.

Requirement: Changes from the project comparison have been accepted.

The changes are only copied to the project. This command does not save them to the hard disk.

See also
• ☹ Chapter 1.3.1.5.2 “Opening the Detailed Compare View” on page 138

Command 'Export'

Function: This command opens a dialog box for exporting objects from a project to an XML file.

Call: Menu bar: “Project”.

Dialog 'Export'

This dialog box lists all objects from the device tree, POU tree, and module tree that CODESYS can export.

<table>
<thead>
<tr>
<th>One file per subtree</th>
<th>✓: CODESYS generates a separate export file for each subtree that is located directly under the root node and includes selected files.</th>
<th>◻: CODESYS generates one export file for all selected objects.</th>
</tr>
</thead>
</table>

“Saved version”

This version should correspond to the target version where the export file will later be imported.
If the current project contains plug-ins or add-ons that are not available in the selected memory format (profile), then the “Extend Profile” dialog box opens. In this dialog box, the selected profile can be extended with the add-ons.

See also
• ☹ Chapter 1.3.1.25.2.1.5 “Command ‘Save Project as’” on page 855
• ☹ Chapter 1.3.1.25.2.4.21 “Command ‘Import’” on page 910
• ☹ Chapter 1.3.1.4.1 “Exporting and importing projects” on page 135

Command 'Import'

Function: This command opens a dialog box for importing objects from an XML file.

Call: Menu “Project”

Requirement: A project is open.
Dialog box 'Import PLCopenXML'

The dialog box lists all objects from the export file that CODESYS can import at this point.

<table>
<thead>
<tr>
<th>“Currently selected target objects”</th>
<th>Object that is selected in the Device tree</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Insertable items”</td>
<td>Displays all objects of the export file that CODESYS can insert below the selected object.</td>
</tr>
<tr>
<td>“Show contents”</td>
<td>Displays the contents of the export file in a tree structure</td>
</tr>
</tbody>
</table>

Command 'Export PLCopenXML'

**Function:** This command opens a dialog box for exporting objects from a project into an XML file in the PLCopen format.

**Call:** Menu “Project”

Dialog box 'Export PLCopenXML'

The dialog box lists all objects from the Device tree that CODESYS can export into an XML file in accordance with the PLCopen format.

*The PLCopenXML scheme does not permit VAR_GLOBAL and VAR_GLOBAL CONSTANT POUs to be in the same variable list. Therefore, if you wish to export both, you must first divide the variables into two separate variable lists.*

See also

- § Chapter 1.3.1.25.2.4.23 “Command 'Import PLCopenXML’” on page 911

Command 'Import PLCopenXML'

**Function:** This command opens a dialog box for importing objects from an XML file in PLCopen format.

**Call:** Menu “Project”

**Requirement:** A project is open.

Dialog box 'Import PLCopenXML'

The dialog box lists all objects from the PLCopen export file that CODESYS can import at this point.

<table>
<thead>
<tr>
<th>“Currently selected target object”</th>
<th>Object that is selected in the Device tree</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Insertable items”</td>
<td>Displays all objects of the export file that CODESYS can insert below the selected object.</td>
</tr>
</tbody>
</table>

*The PLCopenXML scheme does not permit VAR_GLOBAL and VAR_GLOBAL CONSTANT POUs to be in the same variable list. Therefore, if you wish to export both, the variables must first be divided into two separate variable lists.*
This command opens the dialog box “Login”. Here you specify the project that you wish to edit and enter the login data for a user account with the corresponding rights. In addition, you can open the password manager from this dialog box.

The command is available in the menu “Project ➔ User Management”.

See also

● § Chapter 1.3.1.6.6 “Logging in via user account and password manager” on page 147

Command 'User management' – 'Log out User'

Symbol: 

The user currently logged in to the project is logged out again with this command. This takes place without a dialog box or message, unless no user is currently logged in.

The command is available in the menu “Project ➔ User Management”.

If the user is currently logged in to several projects or to libraries integrated in them (it does not have to be the same user account), then the dialog box “Logout” opens, in which the specific project or library project can be selected from which the current user is to be logged out.

The status bar always displays the user who is currently logged into the project.

A double-click on the field “Current user” in the status bar enables quick access to the “Login” or “Logout” dialog box.

See also

● § Chapter 1.3.1.6.6 “Logging in via user account and password manager” on page 147
● § Chapter 1.3.1.25.2.4.24 “Command 'User management' – 'Log in User'” on page 911

Command 'User management' – 'Rights...'

This command opens the dialog box “Rights”, in which you define the actions that may be carried out, the user groups that may carry them out and the project objects on which they may be carried out.

The command is available in the menu “Project ➔ User Management”.

See also

● § Chapter 1.3.1.6.5 “Protecting objects in the project by access rights” on page 146

Command 'Insert Device'

Function: Like the command “Add Device”, this command opens a dialog box “Insert Device” for the selection of a device object. This object is inserted in the device tree at the same level as the currently selected object.

Call: Context menu of a device object in the device tree.

Requirement: An object is selected in the device tree below which a device object can be inserted at the same level.
Command 'Generate EtherCAT XML'

The command is not integrated in the standard main menu. You can add it via the dialog box "Tools ➔ Customize" from the category "Devices".

**Function:** This command opens the standard dialog box for saving a file in the local file system. You can define a name and a storage location for an xml file, in which CODESYS is to store the EtherCAT configuration of the EtherCAT master currently selected in the device tree. This may be necessary in order to operate an external EtherCAT stack.

**Call:** Context menu of an EtherCAT master device object in the device tree.

See also
- Chapter 1.3.1.8 "Configuring I/O links" on page 155

Command 'Generate Sercos SCI XML'

The command is not integrated in the standard menu. You can add it via the dialog box "Tools ➔ Customize" from the category "Devices".

**Function:** This command opens the standard dialog box for saving a file in the local file system. You can define a name and a location for an xml file in which CODESYS then stores the configuration data of the sercos master currently selected in the device tree. This may be necessary in order to operate an external sercos stack.

**Call:** Context menu of a sercos master device object in the device tree.

See also
- Chapter 1.3.1.8 "Configuring I/O links" on page 155
- Chapter 1.3.1.1.2.1 "Customizing menus" on page 103

Command 'Disable Device' – 'Enable Device'

**Function:** This command switches back and forth between the enabled (activated) and disabled (deactivated) states of a device in the bus system.

**Call:** Context menu of a device object in the device tree.

**Requirement:** The project is in offline mode. The bus driver must support the function.

A disabled device is not taken into account and is not addressed. Note that with some bus systems the deactivation of a node can lead to the master stopping.

The entry of a disabled device in the tree appears in light-gray lettering. When logging in, disabled devices are additionally marked with a red triangle ▲.

See also
- Chapter 1.3.1.8 "Configuring I/O links" on page 155
Command 'Edit I/O Mapping'

**Function:** This command opens the "Edit I/O Mapping" dialog box. This displays all I/O mappings of the currently selected device object, including I/O mappings of all additional device objects that are inserted in the device tree below this object.

**Call:** Context menu of a device object in the device tree.

### Dialog box 'Edit I/O mapping'

You can edit the I/O mapping in this dialog box in exactly the same way as in the dialog box "I/O mapping" of the individual device editors. The respective other dialog boxes are directly updated accordingly.

| "Search" | Input field for a search string for the mapping table. The search results are marked in yellow. |
| "Filter" | Drop-down list for filtering I/O assignments displayed listed in the mapping table: |
|          | ● “Show all” |
|          | ● “Show outputs only” |
|          | ● “Show inputs only” |
|          | ● “Show unmapped variables only” |
|          | ● “Show mapped variables only” |
|          | ● “Show mappings to existing variables only” |
|          | ● “Show mappings to new variables only” |

In the context menu you will find among other things the following commands:

"Export Mappings to CSV": Stores the mappings of a device and its sub-devices in an external file. To do this you select the device in the device tree or in the mapping list.

"Import Mappings from CSV": Inserts mappings from a file created beforehand by export.

See also

- ☞ Chapter 1.3.1.8 “Configuring I/O links” on page 155
- ☞ Chapter 1.3.1.25.2.4.31 “Command ‘Edit I/O Mapping’” on page 914
- ☞ Chapter 1.3.1.25.2.4.33 “Command ‘Export Mapping to CSV’” on page 915
- ☞ Chapter 1.3.1.25.2.4.32 “Command ‘Import Mappings from CSV’” on page 914

Command 'Import Mappings from CSV'

**Function:** The command opens the standard dialog for searching for a file in the local file system. The filter is set to file format .csv, for the selection of a file to which the I/O mapping configuration of a device was previously exported. CODESYS reads the configuration back into the current project.

**Call:** Context menu of a device object in the “Devices” view.

**Requirement:** A project with a device with I/O mapping configuration is opened. The device matches the exported .csv file.

**NOTICE!**

You can edit the .csv file externally before import. Please note: The entries of the file to the I/O mapping of the device are assigned by the device name and the parameter name. Parameter names that are not unique are numbered sequentially in this file (@<n>).

Fields without contents in the .csv file are ignored at import. To remove an existing entry in the I/O mapping by importing, you have to add a space in the respective field in the .csv file.
Command 'Export Mapping to CSV'

**Function:** This command opens the standard dialog for saving a file in the local file system. The filter is set to file format **csv**. After specifying a name and a location, CODESYS stores the I/O mapping configuration in a CSV file.

**Call:** Context menu of a device object in the “Devices” view.

**Requirement:** A device object with an I/O mapping configuration is selected in the device tree.

Parameter names that are not unique are numbered sequentially in this file (@<n>).

Command 'Import Mappings from CSV'

See also
- Chapter 1.3.1.25.2.4.31 “Command ‘Export Mapping to CSV’” on page 914
- Chapter 1.3.1.8.3 “Configuring devices and I/O mapping” on page 162

Command 'Read PLC Parameter File to Configuration'

**Function:** This command reads the configuration file **IoConfig.par** of the PLC and stores the values in the project. Such a file is created if the parameters of the PLC have been changed by another device, for example via a visualization. Then these parameters are changed only in the memory of the PLC, but not in the configuration of the project.

**Call:** Context menu of the PLC device object

**Requirement:** You have made the command available using the dialog in “Tools ➔ Customize”.

See also
- “Read PLC parameter file to configuration” on page 163

Command 'Online Config Mode'

**Function:** This command is for switching the online configuration mode on and off. At switch-on it establishes a connection to the PLC and loads an implicitly created application “**HiddenOnlineConfigModeApp**” to the PLC. Depending on the device, CODESYS goes into simple online configuration mode or a dialog box appears for selecting between simple and advanced online configuration mode.

**Call:** Context menu of the PLC object in the device tree

**Requirement:** The communication settings for the PLC device are correctly set.

Simple online configuration mode:

This command creates the implicit application **HiddenOnlineConfigModeApp** and loads it to the controller. The application automatically initializes all inputs and outputs of the controller once. After that you can access the I/Os as follows:

- Read I/Os
- Write outputs
- Diagnosis (in the device tree and on the “Status” tab of the device editor)
- Scan (of the current hardware)
- Interactive online functions, if supported (for example, writing asynchronous messages)

Advanced online configuration mode (parameter mode):

If there are already applications on the PLC and the controller supports it, the command first opens the dialog box “Devices”, which displays the applications existing on the controller. From this dialog you can connect via the button “Parameter mode” to the PLC and then access the values of the device parameters without having to log in with a real application.

**Writing and forcing in the I/O mapping**

In online configuration mode the writing and forcing of values on the “I/O Mapping” tab works differently to the way it works in real online mode. The outputs are written immediately after insertion into the table. There is no “Prepared Value” column; instead, the initial values can be changed directly after a double-click on the column “Current Value”.

---

**Dialog box 'Config application mode'**

This dialog box appears after the command “Online Config Mode” if the device supports the advanced online configuration mode and there are already real applications on the controller.

<table>
<thead>
<tr>
<th>“Parameter mode”</th>
<th>The controller configuration in the project is compared with that on the device. If they correspond, CODESYS establishes a connection to the PLC. Unlike the simple online configuration mode it permits the reading and – if supported by the driver – the writing of parameters in the generic device editor. The applications already loaded to the device remain unchanged in this case!</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Config application mode”</td>
<td>CODESYS switches to the ‘simple online configuration mode’.</td>
</tr>
</tbody>
</table>

See also

- “Checking the controller configuration with the help of the 'Online Config Mode' command” on page 161

---

**Menu 'Build'**

1.3.1.25.2.5.1 Command 'Build'................................................................. 916
1.3.1.25.2.5.2 Command 'Rebuild'.............................................................. 917
1.3.1.25.2.5.3 Command ‘Generate Code’.................................................. 917
1.3.1.25.2.5.4 Command ‘Generate Runtime System Files’.......................... 917
1.3.1.25.2.5.5 Command 'Clean'................................................................... 918
1.3.1.25.2.5.6 Command 'Clean All'............................................................ 919
1.3.1.25.2.5.7 Command 'Check All Pool Objects'........................................ 919
1.3.1.25.2.5.8 Command 'Check All Application Objects'............................ 919
1.3.1.25.2.5.9 Command 'Check Library Compatibility'............................... 920
1.3.1.25.2.5.10 Command 'C Integration' - 'Update C Sources'.................... 920
1.3.1.25.2.5.11 Command 'C Integration – Open in IDE'.............................. 920
1.3.1.25.2.5.12 Command 'C Integration' - 'Export C Source(s)'.................... 921
1.3.1.25.2.5.13 Command 'C Integration – Create Stub Implementation in C'.............................................................................. 921
1.3.1.25.2.5.14 Command 'Create IEC Interface'.......................................... 921
1.3.1.25.2.5.15 Command 'Generate Disassembly File'................................ 922

---

**Command 'Build'**

Symbol: ⌘, keyboard shortcut: [F11].

---
**Function:** This command starts the build process for the active application.

**Call:** Main menu “Build”.

During the compile process, CODESYS performs a syntactic validation of all objects in the application. However, code is not generated like at login to the target system or download of the application. The compile process is always performed automatically when you login with a changed program.

When the check is complete, CODESYS displays any error messages or warnings in the messages view (“Build” category).

If the program has not been changed since it was compiled without errors the last time, then it is not recompiled. The message “The application is current” appears. If the syntactic validation is repeated, then you must execute the “Rebuild” command.

See also

- Chapter 1.3.1.25.2.5.2 “Command ‘Rebuild’” on page 917

**Command ‘Rebuild’**

**Function:** This command starts the build process for the active application, even if the last build contained errors.

**Call:** Main menu “Debug”.

See also

- Chapter 1.3.1.25.2.5.1 “Command ‘Build’” on page 916

**Command ‘Generate Code’**

**Function:** This command starts the code generation for the active application.

**Call:** Menu bar: “Build”

When generating code with this command, code is generated as when downloading the application to the PLC, but the code is not transferred to the PLC. Other source code tests are performed as well for checking the code for bugs that were not detected by the compiler and for fixing any bugs before the code is used in online mode.

See also

- Chapter 1.3.1.11.4 “Generating application code” on page 303

**Command ‘Generate Runtime System Files’**

**Function:** This command generates a C stub file and an M4 interface file from the current library project. These files are used as the basis for creating an external library file.

**Call:** Main menu “Build”.

This command opens the “Generate Files for Runtime System” dialog box.

| "Output directory" | Directory where CODESYS creates the runtime system files. Click the “more” button to open the default dialog box for browsing the file system. |
Table 145: “Which files do you want to create?”

**M4 interface file**

- Interface file `<project name>Itf.m4` with definitions. Example of m4 file:

```
/*
 * $interface$ SysFile</interface>
 */

- `extern C`{
  
  typedef enum
  {
    FS_OK = 0,
    FS_NO_FILE = 1,
    FS_ILLEGAL_POS = 2,
    FS_FULL = 3,
    FS_EMPTY = 4,
    FS_BUFFER_FULL = 5
  } filestatus;

  /*
   * ANSI C file access modes
   */
  #define AM_READ   0
  #define AM_WRITE  1
  #define AM_APPEND 2

  /*
   * ANSI C file access modes
   */
  #define AM_READ_Plus  0
  #define AM_WRITE_Plus 1
  #define AM_APPEND_Plus 2

  typedef enum
  {
    AM_READ = 0,
    AM_READ_Plus = 1,
    AM_APPEND = 2,
    AM_READ_Plus = 3
  } filestatus;
```

**C stub file**

- Stub file for reprogramming the library in C. Example of stub file:

```
#include "Cmpstd.h"
#include "Cmerrors.h"
#include "CmpItf.h"
#include "SysFileDep.h"

void CDCC CDCCEXT sysfilegetpath(sysfilegetpath_struct *p)
{
}

void CDCC CDCCEXT sysfilerename(sysfilerename_struct *p)
{
}

void CDCC CDCCEXT sysfilewriter(syfilewriter_struct *p)
```

**Command ‘Clean’**

**Function:** This command deletes the build information for the active application.

**Call:** Main menu “Build”.

During the last download, the build information was created and saved to a file (*.compileinfo). After a cleaning process, an online change is no longer possible for the affected application. The application must be fully downloaded to the controller again.

See also
- Chapter 1.3.1.25.2.5.6 “Command ‘Clean All’” on page 919

Command ‘Clean All’

**Function:** This command deletes the build information for all applications in the project.

**Call:** Main menu “Build”.

During the last download, the build information was created in the local file system and saved to a file (*.compileinfo).

This command requires a download before another login. An online change is no longer possible. As compared to the “Clean” command (only the active application), CODESYS regenerates the language model for all objects, which is very time-consuming.

**NOTICE!**
Reconsider carefully whether or not executing this command is really necessary. If you only want to rebuild and download the active application, then execute the “Clean” command.

See also
- Chapter 1.3.1.25.2.5.5 “Command ‘Clean’” on page 918

Command ‘Check All Pool Objects’

**Function:** The command causes a build run, that is a syntax check, for all pool objects, which are managed in the POU s view and thus are available in the entire project. Especially this is useful when creating libraries.

**Call:** By default the command is not available in any menu. You can add it to a menu using the dialog “Tools ➔ Customize”, command category “Build”.

**NOTICE!**
The command does not trigger code generation. Also no compile information file will be stored in the project directory.

See also
- Chapter 1.3.1.25.3.14.1 “Dialog ‘Customize’ - ‘Menu’” on page 1089
- Chapter 1.3.1.19.1 “Information for library developers” on page 372

Command ‘Check All Application Objects’

**Function:** This command starts a build operation for all objects of the active application, even for the POU s that are not used by the application. After the build operation, the errors that were found in the unused objects are also displayed in the message window.

**Call:** The command is not in any menu by default. You can add it to a menu by using the dialog from “Tools ➔ Customize” (command category “Build”).

**Requirement:** An application of the open project is active.
NOTICE!
The command does not result in code generation. In addition, no file is created in the project directory with information about the build operation.

See also
- Chapter 1.3.1.25.3.14.1 “Dialog ‘Customize’ - ‘Menu’” on page 1089

Command 'Check Library Compatibility'

Function: The command triggers a check whether the currently opened library project is compatible with the last installed version of this library (next lower version number).

Call: By default the command is not available in any menu. You can add it to a menu by using the “Tools ➔ Customize” dialog, command category “Build”.

Requirement: A library project is opened.

The check regards differences in the implemented interfaces of a method. So, after the check you will get displayed error messages in the messages window in the following cases:
- Adding or removing inputs or outputs of function blocks, functions or methods
- Changing the data type of inputs or outputs
- Modifying the implemented interfaces of a method

See also
- Chapter 1.3.1.25.3.14.1 “Dialog ‘Customize’ - ‘Menu’” on page 1089
- Chapter 1.3.1.19.1 “Information for library developers” on page 372

Command 'C Integration' - 'Update C Sources'

Function: this command opens the dialog “Update C Sources” for updating the objects in the project that have changed in the source directory on the disk.

Call: Menu bar: “Build”; context menu.

Requirement: An object “C Code Module” or “C Implemented Library” is selected. When adding the C-code module in the dialog “Add C Code Module”, you have activated the option “Check folder for source code changes”.

<table>
<thead>
<tr>
<th>“File”</th>
<th>File that has changed on the disk.</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Action”</td>
<td>Action that is executed in CODESYS if you click “Update”.</td>
</tr>
<tr>
<td>“Update options”</td>
<td>“Remove IEC interfaces due to changed header files”</td>
</tr>
<tr>
<td></td>
<td>“Export source files to the monitored project folder”</td>
</tr>
<tr>
<td>“Refresh”</td>
<td>CODESYS updates the listed files.</td>
</tr>
</tbody>
</table>

See also
- Chapter 1.3.1.25.1.16 “Object ‘C Code Module’” on page 737
- Chapter 1.3.1.25.2.5.11 “Command ‘C Integration – Open in IDE’” on page 920
- Chapter 1.3.1.9.11 “Integrating C Modules” on page 222

Command 'C Integration – Open in IDE'

**Call**: Main menu “Create”, context menu

**Requirement**: You have opened an object “C Code Module” and the associated IDE is not opened.

If the IDE is closed, CODESYS checks whether the files have been changed and, in such a case, a dialog box appears for confirming the update of the C-code module in CODESYS.

See also

- Chapter 1.3.1.25.3.10.5 “Dialog ‘Properties’ – ‘Build’ (C-integration)” on page 1049
- Chapter 1.3.1.25.2.5.10 “Command ‘C Integration’ - ‘Update C Sources’” on page 920
- Chapter 1.3.1.9.11 “Integrating C Modules” on page 222

**Command ‘C Integration’ - 'Export C Sources'**

**Function**: The command exports all C-code files of a C-code module and saves them in the folder that you select in the dialog “Find Folder”.

**Call**: Menu bar: “Build”; context menu.

**Requirement**: A C code module is selected in the device tree.

See also

- Chapter 1.3.1.9.11 “Integrating C Modules” on page 222

**Command ‘C Integration – Create Stub Implementation in C’**

**Function**: This command creates C-stubs for the selected POU and stores them in the “Extensions” folder in the objects "iec_external.c" and "iec_external.h".

**Call**: Main menu “Build”, context menu

**Requirement**: A POU that is inserted under the object of the type “C Code Module” is selected in the device tree. The application has been compiled without errors.

See also

- Chapter 1.3.1.25.1.17 “Object ‘C Code File’” on page 739
- Chapter 1.3.1.9.11 “Integrating C Modules” on page 222

**Command ‘Create IEC Interface’**

**Function**: The command creates corresponding IEC objects from the selected file with the format *.h or *.hpp and stores these IEC objects in the folder “IEC interface”.

**Call**: Main menu “Build”, context menu

**Requirement**: You have selected an imported C-code file of the format *.h or *.hpp in the device tree below the object “C Code Module”.

If you select the command and the header file is free of errors, the dialog box “C Functions” opens with a list of the functions of the file that is to be exported.

**Dialog box ‘Create IEC interface’**

| “Function” | List of the functions  
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>You select the functions for which a corresponding IEC object is to be created.</td>
</tr>
</tbody>
</table>

| “Import” | CODESYS generates corresponding IEC objects for the selected C-functions and stores them in the folder “IEC interface” below the object “C Code Module”. |
Command 'Generate Disassembly File'

**Function:** This command generates a disassembly file `<project name>.asm` from the current project and saves it in the file directory in the project folder.

**Call:** The command is not in any menu by default. You can add it to a menu by using the dialog box from "Tools ➔ Customize" (command category "Build").

See also
- Chapter 1.3.1.9.11 “Integrating C Modules” on page 222

Menu 'Online'

<table>
<thead>
<tr>
<th>Menu Command</th>
<th>Command Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.3.1.25.2.6.1 Command 'Choose active application'</td>
<td>- Command with a list box to select an active application.</td>
<td>922</td>
</tr>
<tr>
<td>1.3.1.25.2.6.2 Command 'Login'</td>
<td>- Select an application to log in to.</td>
<td>923</td>
</tr>
<tr>
<td>1.3.1.25.2.6.3 Command 'Logout'</td>
<td>- Exit from the selected application.</td>
<td>926</td>
</tr>
<tr>
<td>1.3.1.25.2.6.4 Command 'Create Boot Application'</td>
<td>- Create a boot application.</td>
<td>927</td>
</tr>
<tr>
<td>1.3.1.25.2.6.5 Command 'Load'</td>
<td>- Load a selected application.</td>
<td>927</td>
</tr>
<tr>
<td>1.3.1.25.2.6.6 Command 'Online Change'</td>
<td>- Change the selected application.</td>
<td>928</td>
</tr>
<tr>
<td>1.3.1.25.2.6.7 Command 'Source Download to Connected Device'</td>
<td>- Download an application.</td>
<td>929</td>
</tr>
<tr>
<td>1.3.1.25.2.6.8 Command 'Multiple Download'</td>
<td>- Multiple download for selected applications.</td>
<td>931</td>
</tr>
<tr>
<td>1.3.1.25.2.6.9 Command 'Reset Cold'</td>
<td>- Reset to cold state.</td>
<td>931</td>
</tr>
<tr>
<td>1.3.1.25.2.6.10 Command 'Reset Warm'</td>
<td>- Reset to warm state.</td>
<td>931</td>
</tr>
<tr>
<td>1.3.1.25.2.6.11 Command 'Reset Origin'</td>
<td>- Reset to origin state.</td>
<td>932</td>
</tr>
<tr>
<td>1.3.1.25.2.6.12 Command 'Reset Origin Device'</td>
<td>- Reset the origin device.</td>
<td>933</td>
</tr>
<tr>
<td>1.3.1.25.2.6.13 Command 'Logoff Current Device User'</td>
<td>- Log off the current device user.</td>
<td>934</td>
</tr>
<tr>
<td>1.3.1.25.2.6.14 Command 'Add Device User'</td>
<td>- Add a new device user.</td>
<td>934</td>
</tr>
<tr>
<td>1.3.1.25.2.6.15 Command 'Remove Device User'</td>
<td>- Remove a device user.</td>
<td>935</td>
</tr>
<tr>
<td>1.3.1.25.2.6.16 Command 'Change Password Device User'</td>
<td>- Change the password for a device user.</td>
<td>935</td>
</tr>
<tr>
<td>1.3.1.25.2.6.17 Command 'Stop Execution on Handled Exceptions'</td>
<td>- Stop execution on handled exceptions.</td>
<td>936</td>
</tr>
<tr>
<td>1.3.1.25.2.6.18 Command 'Connect to Device'</td>
<td>- Connect to the selected device.</td>
<td>936</td>
</tr>
<tr>
<td>1.3.1.25.2.6.19 Command 'Disconnect from Device'</td>
<td>- Disconnect from the selected device.</td>
<td>936</td>
</tr>
<tr>
<td>1.3.1.25.2.6.20 Command 'Wink'</td>
<td>- Blink the device indicator.</td>
<td>937</td>
</tr>
<tr>
<td>1.3.1.25.2.6.21 Command 'Simulation'</td>
<td>- Simulate the selected application.</td>
<td>937</td>
</tr>
<tr>
<td>1.3.1.25.2.6.22 Command 'Operating Mode'</td>
<td>- Enter operating mode.</td>
<td>937</td>
</tr>
</tbody>
</table>

Command 'Choose active application'

**Symbol:** App [Device PLC Logic]

The command is implemented as a list box from which you can set an application active. By default, the list box is located on the toolbar.

**Function:** The list box displays the currently active application with its device path.

**Call:** The list box contains all applications that are organized in the "Devices" view. By clicking an entry in the list box, you activate the selected application.

**Requirement:** The project has multiple applications.
When you call commands in the “Build” oder “Online” menus, these commands are applied to the active application. This is displayed in the list box, and also displayed in bold in the device tree. In particular, this applies to the “Build ➔ Build” and “Online ➔ Login” commands.

You can also access these commands using the command icons on the toolbar where the list box is located. When the command icons are called, they are also applied to the active application.

However, if you call a command from the context menu of a device object in the device tree, then the command is applied to the corresponding object. For example, by calling 📲, you can establish a connection to an application on the device which is not active.

See also

- ☰ Chapter 1.3.1.25.2.4.10 “Command ‘Set Active Application’” on page 902
- ☰ Chapter 1.3.1.25.2.5.1 “Command ‘Build’” on page 916
- ☰ Chapter 1.3.1.25.2.6.2 “Command ‘Login’” on page 923

Command ‘Login’

Symbol: 📲, shortcut: [Alt]+[F8]

**Function:** The command connects the application to the target system (PLC to simulated device) and starts the online mode.
Call: Menu bar: “Online”; context menu of an “Application” object

Requirement: The application contains no errors and the communication settings are configured.

A dialog prompt opens if the communication settings are incorrect. You can then switch directly to the “Communication Settings” of the PLC.

If you click “Login” from the online menu, then the currently active application is connected to the target system. If you choose this command from the context menu (right-click) while an application is selected in the device tree, then that application is logged in, even if it is not set as the active application.

If an online user management is configured on the target device, then you are prompted for user data when you login. The “Device User Login” dialog opens for this.

CAUTION!
Check controller accessibility
For security reasons, controllers should not be accessible from the Internet or untrusted Networks under any circumstances! In particular, the TCP/IP programming ports (usually UDP-Ports 1740..1743 and TCP-Ports 1217 + 11740 or the controller specific ports) should not be accessible from the internet without protection. In case Internet access to the controller is needed, using a safe mechanism is absolutely mandatory, such as VPN and password protection of the controller.

see also: § Chapter 1.3.1.11.3 “Handling of device user management” on page 300

NOTICE!
If a safety controller is inserted below a controller, then this command can interrupt the communication connections temporarily.

Connections of the safety controller to other safety controllers (via safety network variables), to field devices, and to the development system are affected. The safe field devices or other safety controller can enter the safe state as a reaction. The connection to the development system is affected only when a safety controller that is connected to the main controller via a fieldbus.

For more information, see the "Subordinate Safety Controllers" chapter.

Possible situations when logging in:

- A later version of the device description (than in the project) is on the PLC. A warning prompt is displayed with the option to cancel the process.
- The application does not exist on the PLC: You are prompted to confirm the download.
- The application is already on the PLC and has not been changed since the last download. The login continues without any more prompts.
- The application exists on the PLC, but it has been changed since the last download. You are prompted to select one of the following options:
  - Login with online change (Note the information about online changes in the help page "Command ‘Online Change’").
  - Login with download
  - Login without any change

The position also provides the option of updating the boot application on the PLC.

- An unknown version of the application exists on the PLC. CODESYS prompts you to replace it.
A version of the application exists on the PLC and is running. CODESYS prompts you to login anyway and overwrite the currently running application.

The application on the PLC is currently halted at a breakpoint. You are logged out and the program has been changed: CODESYS prompts you with a warning that the PLC will be stopped completely if an online change or download occurs. This happens also if several tasks exist and the breakpoint affects only one of them.

Click “Details” in the dialogs above to open the “Application Information” dialog.

See also

- Chapter 1.3.1.25.2.6.6 “Command ‘Online Change’” on page 928
- Chapter 1.3.1.10.5 “Subordinate safety controller” on page 294

The dialog provides two tabs with comparative information about the application changed in the development system and its previous version currently located on the PLC. There are two tabs:

- “Application information”: The application properties of the “Application in the IDE” (Integrated Development Environment) are compared with those of the “Application in the PLC”. Project name, Last modification, IDE version, Author, Description. In addition, CODESYS shows the objects that have changed since the last download.
- “Application contents”: When the “Download application info” is selected, the contents of the applications on both the (1) development system and (2) PLC can be compared. The “Download application info” option is located on the “Application Build Options” tab of the application properties.

If the code in the development system is not current, then (3) “Application not up to date. Generate code now?” appears at the bottom left of the dialog. Execute this command to update the application source code.

This detailed information can help you to better assess the effects of login in the current situation and to make a decision about downloading the new application.

The comparison can also be displayed in the device editor (“Applications” tab) by clicking “Content”.

See also

- Chapter 1.3.1.25.1.19.4 “Tab ‘Applications’” on page 747
- Chapter 1.3.1.5 “Comparing projects” on page 137
Unknown applications on the PLC

If one or more applications are already on the PLC, but are not in the project, then CODESYS opens a dialog with a list of these applications. You can then define whether an application should be deleted before loading the current application from the PLC. This also applies to child applications that are on the PLC, but have been deleted from the project in the meantime.

Compiling the project before login

If an application program has not been compiled since the last change, then CODESYS compiles the project before login. This operation is the same as the “Build” command when logged out.

If compile errors occur, then a dialog prompt opens. The errors are printed to the message view (“Build” category). You can then decide whether or not you login without downloading the program to the PLC.

See also

- Chapter 1.3.1.25.2.5.1 “Command ‘Build’” on page 916

Error at login

If an error occurs when logging in to the PLC, then CODESYS cancels the loading operation with an error message. The error dialog gives you the options of showing the error details. If an exception was thrown and the text *SOURCEPOSITION* is included in the log, then you can display the affected function in the editor by clicking “Show in Editor”. The cursor jumps to the line containing the error.

Messages during the download operation

If CODESYS downloads the project to the PLC at login, then the following information is printed to the message view:

- Generated code size
- Size of the global data
- Resulting memory requirement on the PLC
- List of the affected blocks (for online change)

In online mode, you cannot change the settings of the devices or modules. You have to logout of the application for changing device parameters. Depending on the bus system, there may be some special parameters that you can also change in online mode.

CODESYS saves the view configuration separately in online and offline mode. In addition, views are closed that cannot be used in any operating mode. For this reason, the view can change automatically at login.

Command 'Logout'

Symbol: ☐, keyboard shortcut: [Ctrl]+[F8].

Function: This command disconnects the application from the target system (controller or simulated device) and returns to offline mode.

Call: Main menu “Online”, or context menu of the “Application” object.

Command 'Create Boot Application'

Function: This command generates a boot application.

Call: Main menu “Online”.

926 3ADR010583, 1, en_US 2020/12/10
A boot application is the application that is started automatically when the controller is switched on or started.

In offline mode, you can save the boot application in any directory. In online mode, CODESYS save the boot application to the target device. The file name is <application name>.app.

See also
- [Chapter 1.3.1.11.6 “Generating boot applications” on page 305](#)
- [Chapter 1.3.1.25.1.1 “Object ‘Application’” on page 710](#)

## Command 'Load'

**Function:** This command causes a compilation of the active application with subsequent download to the controller.

**Call:** Menu bar: “Online”.

**Requirement:** The application is in online mode.

When you execute this command, CODESYS performs a syntax check and generates the application code. This code is downloaded to the PLC. Furthermore, CODESYS generates the build log <project name>.<device name>.<application ID>.compile info in the project directory.

**NOTICE!**
During loading all variables are re-initialized with the exception of persistent variables.

**NOTICE!**
If a safety controller is inserted below a controller, then this command can interrupt the communication connections temporarily.

Connections of the safety controller to other safety controllers (via safety network variables), to field devices, and to the development system are affected. The safe field devices or other safety controller can enter the safe state as a reaction. The connection to the development system is affected only when a safety controller that is connected to the main controller via a fieldbus.

For more information, see the “Subordinate Safety Controllers” chapter.

The description of the “Login” command describes the possible situations when logging in and loading.

If you attempt to download an application when the same version of the application is already on the PLC, then you get the message: "Program is unchanged. Application was not downloaded". CODESYS downloads the application to the PLC.

During loading a record of the actions being executed (generation of code, execution of initialization, etc.) appears in the Message window in the message category “Compile”. Furthermore, information is displayed regarding the memory ranges, the size of the code, the global data and the allocated memory. For the purpose of clarity, as opposed to the online change, the modified function blocks are no longer listed.

See also
- [Chapter 1.3.1.25.2.6.2 “Command 'Login’” on page 923](#)
- [Chapter 1.3.1.10.5 “Subordinate safety controller” on page 294](#)
Command 'Online Change'

**Function:** This command is used for initiating an online change on the current application. When this is done, CODESYS re-downloads only the changed parts of an application that is already running on the PLC.

**Call:** Menu bar: “Online”; context menu of object: “Application”

**Requirement:** The application is in online mode.

The command is available in the context menu if an application is selected in the device tree. In this way, you can perform an online change just for one application, even if that application is not currently active.

An online change is not possible after the “Clean All” and “Clean” commands. The cleaning process deletes the compile information (build log) that was generated automatically at code generation. This log is the basis for an online change.

---

**CAUTION!**

An online change changes the running application program and requires a restart.

Make sure that the new application code still has the required effect on the controlled system.

Depending on the controlled plant, the plant and workpieces may be damaged or the health and life of persons could be endangered.

---

**NOTICE!**

If an online change is performed, then the application-specific initializations (example: homing) are not executed because the machine retains its status. For this reason, the new program code may not have the intended effect.

1. Pointer variables retain their value from the last cycle. If a pointer refers to a variable whose value was changed in an online change, then the variable no longer yields the correct value. Make sure that pointer variables are re-assigned in each cycle.

2. After the parent application has been changed, a child application is removed from the PLC when an online change is performed.

---

**NOTICE!**

For compiler version 3.5.0.0 and later, a fast online change is performed for minor changes. In this case, only the modified blocks are compiled and downloaded. In particular, no initialization code is generated. This means that also no code is generated when variables with the `init_on_onlchange` attribute are initialized. As a rule, this has no effect because the attribute is used primarily for initializing variables with addresses. However, it cannot happen that a variable changes its address during an online change.

To secure the effect of the `init_on_onlchange` attribute in the entire application code, you must deactivate the fast online change in general for the application by using the compiler definition `no fast online change`. To do this, insert the definition in the application “Properties” (“Build” tab).

At the time of download, CODESYS also lists the changed interfaces, affected variables, and all blocks with new generated code in the “Build” category of the message view. If memory locations change, a dialog will inform you of possible problems in conjunction with pointers.
In the view “Memory Reserve for Online Change”, memory reserves can be configured for the online change so that instance variables do not have to be moved in the memory when changing a function block in an online change.

See also

- Chapter 1.3.1.11.4 “Generating application code” on page 303
- Chapter 1.3.1.14.1 “Executing the online change” on page 355
- Chapter 1.3.1.24.6.2.17 “Attribute ‘init_on_onchange’” on page 616
- Chapter 1.3.1.9.4 “Configuring the memory reserve for an online change” on page 206

Command 'Source Download to Connected Device'

**Function:** This command loads the project source code (as project archive) to the controller currently connected.

**Call:** Main menu “Online”.

**Requirement:** The application is in online mode.

See also

- Chapter 1.3.1.25.2.1.10 “Command ‘Source Upload” on page 859
- Chapter 1.3.1.25.2.1.11 “Command ‘Source Download” on page 860

Command 'Multiple Download'

**Function:** The command causes the code generation of the applications contained in the project as well as the loading of the applications to the corresponding controllers.

**Call:** Menu bar: “Online”

The command opens a dialog with a list of the applications. In this dialog, select the applications that are to be loaded. Then, CODESYS performs the syntax check of these applications and generates the respective code. The code is then downloaded to the respective PLC. For each selected application, CODESYS generates a build log with the name <project name>.<device name>.<application ID>.compileinfo in the project directory.

**NOTICE!**

If a safety controller is inserted below a controller, then this command can interrupt the communication connections temporarily.

Connections of the safety controller to other safety controllers (via safety network variables), to field devices, and to the development system are affected. The safe field devices or other safety controller can enter the safe state as a reaction. The connection to the development system is affected only when a safety controller that is connected to the main controller via a fieldbus.

For more information, see the “Subordinate Safety Controllers” chapter.

Dialog 'Multiple Download'
“Please select the items to be downloaded”:
Selection of the applications. The applications are thereby also loaded to different controllers.

“Move Up”, “Move Down”:
Change of the order of download of the applications.
The applications are downloaded to the PLCs in the order of this list. By default, this list is alphabetically sorted. Parent-child relationships of applications are thereby taken into account.

“OK”:
Checks the syntax of all selected applications. Afterwards, the communication with the associated controller is verified for each application before the download takes place.

Table 146: “Online Change Options”

If an earlier version already exists on the PLC and is different from the current version, then the following options are provided:

“Try to perform an online change. If this is not possible, perform a full download.”
Activated by default. If an online change cannot be executed for one of the applications, then a download is performed.

“Force an online change. If this is not possible, cancel the operation.”
If an online change cannot be performed for (at least) one of the applications, then no download is performed and the online change is terminated (for example, if you have executed the command “Clean All” beforehand).

“Always perform a full download.”
Downloads all parts of the applications to the PLC, regardless of any existing versions.

For selected applications that do not exist on the PLC yet, CODESYS performs a download automatically to the PLC.

Table 147: “Other Options”

“Delete all applications on the PLC which are not part of the project.”
Corresponding applications are deleted

“Start all applications after download or online change”
The applications are started after the download or online change.

“Do not release forced variables”
If an application with forced variables exists on the controller, and if the implementation of this application has been changed, then no download is performed for this application.
The message “Error: Skipped because one or more variables have been forced” appears for this application in the window “Multiple Download - Result”.

Note that variables with the key attribute PERSISTENT RETAIN are not generally initialized. If you change the data layout, however, the persistent variables are automatically re-initialized.

After completion of the download a listing of all selected applications appears in the download order that you configured. In addition, you are shown information on the success of the download for each application.

See also
- Chapter 1.3.1.10.5 “Subordinate safety controller” on page 294
- Chapter 1.3.1.9.20 “Data persistence” on page 249
- Chapter 1.3.1.11.4 “Generating application code” on page 303
- Chapter 1.3.1.9.20 “Data persistence” on page 249
- Chapter 1.3.1.25.2.6.5 “Command ‘Load’” on page 927
Command 'Reset Cold'

Function: The command results in a cold start of the active application on the controller.

Call: Menu bar: "Online"

Requirement: The application is in online mode.

NOTICE!

If a safety controller is inserted below a controller, then this command can interrupt the communication connections temporarily.

Connections of the safety controller to other safety controllers (via safety network variables), to field devices, and to the development system are affected. The safe field devices or other safety controller can enter the safe state as a reaction. The connection to the development system is affected only when a safety controller that is connected to the main controller via a fieldbus. For more information, see the "Subordinate Safety Controllers" chapter.

After restarting with "Reset Cold", the following happens:

- Application code is retained on the controller.
- Variables are initialized (with the initialization value or the default initialization value 0), and the previous values are lost.
- Retain variables are initialized, and the previous values are lost.
- Persistent variables are retained with values.
- Breakpoints that were set in the code are retained with their status (for example, activated or deactivated).
- The application goes into the "STOP" state.

You can also select the command while debugging the application when it halts at a breakpoint in the "HALT ON BP" state. Then either the warm start is executed immediately, or the remaining statements of the current cycle are processed. Therefore, a message window opens for you to select the next action. However, the message window opens only if the runtime system is capable of restarting the cycle without terminating it first.

After the reset, you can run the application as usual and, for example, start the execution by clicking "Debug → Start".

See also

- Chapter 1.3.1.12.5 “Resetting applications” on page 318
- Chapter 1.3.1.9.20.1 “Preserving data with persistent variables” on page 252
- Chapter 1.3.1.12.2 “Using breakpoints” on page 310
- Chapter 1.3.1.24.2.12 “Persistent variable - PERSISTENT” on page 498
- Chapter 1.3.1.10.5 “Subordinate safety controller” on page 294
- Chapter 1.3.1.25.2.6.10 “Command ‘Reset Warm’” on page 931
- Chapter 1.3.1.25.2.6.11 “Command ‘Reset Origin’” on page 932

Command 'Reset Warm'

Function: The command results in a warm start of the active application on the controller.

Call: Menu bar: “Online”

Requirement: The application is in online mode.
NOTICE!

If a safety controller is inserted below a controller, then this command can interrupt the communication connections **temporarily**. Connections of the safety controller to other safety controllers (via safety network variables), to field devices, and to the development system are affected. The safe field devices or other safety controller can enter the safe state as a reaction. The connection to the development system is affected only when a safety controller that is connected to the main controller via a fieldbus.

For more information, see the "Subordinate Safety Controllers" chapter.

After restarting with "Reset Warm", the following happens:

- Application code remains loaded on the controller.
- Variables are initialized (with the initialization value or the default initialization value 0).
- Retain variables are retained with values.
- Persistent variables are retained with values.
- Breakpoints that were set in the code are retained with their status (for example, activated or deactivated).
- The application goes into the “STOP” state.

You can also select the command while debugging the application when it halts at a breakpoint in the “HALT ON BP” state. Then either the warm start is executed immediately, or the remaining statements of the current cycle are processed. Therefore, a message window opens for you to select the next action. However, the message window opens only if the runtime system is capable of restarting the cycle without terminating it first.

After the reset, you can run the application as usual and, for example, start the execution by clicking “Debug ➔ Start”.

See also

- Chapter 1.3.1.12.5 “Resetting applications” on page 318
- Chapter 1.3.1.20.1 “Preserving data with persistent variables” on page 252
- Chapter 1.3.1.12.2 “Using breakpoints” on page 310
- Chapter 1.3.1.24.2.12 “Persistent variable - PERSISTENT” on page 498
- Chapter 1.3.1.10.5 “Subordinate safety controller” on page 294
- Chapter 1.3.1.25.2.6.9 “Command 'Reset Cold'” on page 931
- Chapter 1.3.1.25.2.6.11 “Command 'Reset Origin'” on page 932

**Command 'Reset Origin'**

**Function:** The command results in a reset origin of the active application on the controller.

**Call:** Menu bar: “Online”

**Requirement:** The application is in online mode.
NOTICE!

If a safety controller is inserted below a controller, then this command can interrupt the communication connections permanently.

Connections of the safety controller to other safety controllers (via safety network variables), to field devices, and to the development system are affected. The safe field devices or other safety controller can enter the safe state as a reaction. The connection to the development system is affected only when a safety controller that is connected to the main controller via a fieldbus.

For more information, see the "Subordinate Safety Controllers" chapter.

After restarting with "Reset Origin", the following happens:

- The application code is deleted, and as a result the application has no state.
- Variables are deleted, and the values are lost.
- Retain variables are deleted, and the values are lost.
- Persistent variables are deleted, and the values are lost.
- Breakpoints that were set in the code are lost.

See also

- Chapter 1.3.1.12.5 “Resetting applications” on page 318
- Chapter 1.3.1.9.20.1 “Preserving data with persistent variables” on page 252
- Chapter 1.3.1.12.2 “Using breakpoints” on page 310
- Chapter 1.3.1.24.2.12 “Persistent variable - PERSISTENT” on page 498
- Chapter 1.3.1.10.5 “Subordinate safety controller” on page 294
- Chapter 1.3.1.25.2.6.10 “Command 'Reset Warm’” on page 931
- Chapter 1.3.1.25.2.6.9 “Command 'Reset Cold’” on page 931

Command 'Reset Origin Device'

**Function:** The command resets the device to its factory settings. All applications, boot applications, and remanent variables will be deleted from the device.

**Call:** Right-click a programmable device in the device tree.

**Requirement:** The application is in online mode.

NOTICE!

If a safety controller is inserted below a controller, then this command can interrupt the communication connections permanently.

Connections of the safety controller to other safety controllers (via safety network variables), to field devices, and to the development system are affected. The safe field devices or other safety controller can enter the safe state as a reaction. The connection to the development system is affected only when a safety controller that is connected to the main controller via a fieldbus.

For more information, see the "Subordinate Safety Controllers" chapter.

See also

- Chapter 1.3.1.25.2.6.11 “Command 'Reset Origin’” on page 932
- Chapter 1.3.1.25.2.6.10 “Command 'Reset Warm’” on page 931
- Chapter 1.3.1.25.2.6.9 “Command 'Reset Cold’” on page 931
- Chapter 1.3.1.10.5 “Subordinate safety controller” on page 294
Command 'Logoff Current Device User'

Symbol: 

Function: This command logs out the user currently logged in to the controller (device). If CODESYS still has a connection to the controller, then it will be disconnected.

Call: Main menu “Online”.

Requirement: The application is in online mode.

You can manage the device user management in the “Users and Groups” tab and “Access control” of the device editor. The commands in the “Online ➔ Security” menu provide another simple option for protecting access to the target device.

See also
- § Chapter 1.3.1.25.1.19.13 “Tab 'Users and Groups'” on page 760
- § Chapter 1.3.1.11.3 “Handling of device user management” on page 300

Command 'Add Device User'

Symbol: 

Function: This command configures a new device user and adds this user to the administrator group.

Call: Menu bar: “Online ➔ Security”

Requirement: The device supports a device user management. You are logged in to the device as a user.

You can manage the device user management in the “Users and Groups” tab and “Access control” of the device editor. The commands in the “Online ➔ Security” menu provide another simple option for protecting access to the target device.

See also
- § Chapter 1.3.1.25.1.19.13 “Tab 'Users and Groups'” on page 760
- § Chapter 1.3.1.11.3 “Handling of device user management” on page 300

This command opens the “Add Device User” dialog. Here you define the access data of the new user.

The dialog corresponds to the dialog in the “Users and Groups” tab of the device editor for adding a new user.

Please use a strong password as follows:
- Password length >= 8 characters (best >= 12)
- Use uppercase and lowercase
- Include numbers
- Use special characters
- Do not use existing names or sequence of characters that are easy to guess (for example, “123”, “abc”, “qwerty”)
CAUTION!

After performing this action, you can no longer use a blank user name and password to log in. You must remember your password.

See also
- Chapter 1.3.1.25.2.6.15 “Command ‘Remove Device User’” on page 935
- Chapter 1.3.1.25.2.6.16 “Command ‘Change Password Device User’” on page 935

Command 'Remove Device User'

Symbol: 🗝

Function: This command removes a user from the user management on the target system (device).

Call: Menu bar: “Online ➔ Security”

 Requirement: You are logged in to the device as a user.

You can manage the device user management in the “Users and Groups” tab and “Access control” of the device editor. The commands in the “Online ➔ Security” menu provide another simple option for protecting access to the target device.

See also
- Chapter 1.3.1.25.1.19.13 “Tab ‘Users and Groups’” on page 760
- Chapter 1.3.1.11.3 “Handling of device user management” on page 300

This command opens the “Remove Device User” dialog. Specify the user name and password of the user to be removed and click “OK” to confirm.

CAUTION!

After performing this action, you can no longer use this removed user account to log in. If this user is the only one on the target system, then a dialog prompt notifies you that this user cannot be removed.

See also
- Chapter 1.3.1.25.2.6.14 “Command ‘Add Device User’” on page 934
- Chapter 1.3.1.25.2.6.16 “Command ‘Change Password Device User’” on page 935

Command 'Change Password Device User'

Symbol: 🗝

Function: This command changes the password for the user who is currently logged on the PLC.

Call: Menu bar: “Online ➔ Security”

Requirement: You are logged in to the device as a user.

The command opens the “Change Password for Device User” dialog for defining a new password.
NOTICE!
After performing this action, you can no longer use the previous password to log in.

Please use a strong password as follows:

- Password length \(\geq 8\) characters (best \(\geq 12\))
- Use uppercase and lowercase
- Include numbers
- Use special characters
- Do not use existing names or sequence of characters that are easy to guess (for example, "123", "abc", "qwerty")

See also

- Chapter 1.3.1.25.2.6.14 “Command ‘Add Device User’” on page 934
- Chapter 1.3.1.25.2.6.15 “Command ‘Remove Device User’” on page 935
- Chapter 1.3.1.11.3 “Handling of device user management” on page 300

Command 'Stop Execution on Handled Exceptions'

**Function:** This command halts the application where the error is located despite a programmed exception handling.

**Call:** This command is not available by default, but it can be configured from the “Tools ➔ Customize”, “Add Command” dialog box (“Online” category).

**Requirement:** The application is in online mode and contains a programmed exception handling with the `__TRY` and `__CATCH` operators.

If you have configured this command from the “Online” menu and you call it from there, then the currently active application is affected. Furthermore, this command can help you to detect errors.

See also

- Chapter 1.3.1.24.3.61 “Operators ‘__TRY’, ‘__CATCH’, ‘__FINALLY’, ‘__ENDTRY’” on page 546
- “Adding commands” on page 103
- Chapter 1.3.1.25.2.8.9 “Command ‘Customize’” on page 959

Command 'Connect to Device'

**Function:** The command establishes a connection to the device currently selected in the device tree.

**Call:** Context menu of the device.

**Requirements:** A device is selected in the device tree. The communication settings are configured correctly.

See also

- Chapter 1.3.1.25.2.6.19 “Command ‘Disconnect from Device’” on page 936

Command 'Disconnect from Device'

**Function:** The command disconnects the connection from a device.

**Call:** Context menu of the device.

**Requirements:** A device is selected in the device tree.
See also

● § Chapter 1.3.1.25.2.6.18 “Command 'Connect to Device’” on page 936

**Command 'Wink'**

**Symbol:**

**Function:** The command causes a short blinking of an LED of the connected controller. Thus the hardware can be identified clearly.

**Call:** By default the command is not available in any menu. You can add it to a menu by using the “Tools → Customize” dialog, command category “Build”.

**Requirements:** The controller supports this function and the communication settings are configured correctly.

**Command 'Simulation'**

**Function:** This command switches the development system to simulation mode.

**Call:** Menu bar: “Online”

In simulation mode, you can start and debug the active application on a simulated target device. A physical target device is not necessary for testing the online behavior of an application. When logging in for the first time, you are prompted whether the application should be created or loaded. For a simulated device, you do not have to configure the communication settings. In CODESYS simulation mode, the controller entry in the device tree is shown in italics.

**NOTICE!**

No C code for simulation mode

- In simulation mode, C code is not generated and loaded to the runtime system. To simulate the code contained in the C modules anyway, you can implement it for this purpose in the respective IEC objects of the C code module.

To switch off simulation mode, log out of the controller and execute the “Simulation” command again.

This command affects the active application only. Via context menu the simulation mode can be activated for the selected application.

**Command 'Operating Mode'**

**Function:** These commands set the controller in a state that prevents accidental change to the project.

**Call:** Menu bar: “Online → Operating Mode”

You can use these commands, for example, to lock the state of a controller in order to prevent the controller from switching to another state while you program another controller.

When programming is complete, the controller should then be switched to a defined and externally visible state that is set exactly the same way after reboot.
If it supports the controller, then you can switch the controller to the following operating modes:

- **Debug**: No restrictions
- **Locked**: The current state of debugging is locked on the application. No more breakpoints can be set and no variables can be forced. Writing variables is still possible. Only the "RUN" state of an application is preserved in the "Locked" operating mode even if the controller is restarted. Otherwise the application remains in "STOP" state.
- **Operational**: Nothing can be changed. Breakpoints cannot be set. Writing variables is still possible.

The symbols ◼, ◽, and ◼ in the status bar indicate the current operating mode. Double-clicking one of these symbols opens a help window.

Conditions for activating the "Operational" mode

- A boot application for each application must be present on the controller.
- There must not be any active breakpoints set.
- All applications must be running.
- There must not be any forced values present.
- Moreover, the device can define more of its own restrictions.

You cannot switch the operating mode between "Locked" and "Operational".

---

**Menu 'Debug'**

<table>
<thead>
<tr>
<th>Command ID</th>
<th>Command Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.3.1.25.2.7.1</td>
<td>Command 'Start'</td>
<td>939</td>
</tr>
<tr>
<td>1.3.1.25.2.7.2</td>
<td>Command 'Stop'</td>
<td>939</td>
</tr>
<tr>
<td>1.3.1.25.2.7.3</td>
<td>Command 'Single Cycle'</td>
<td>939</td>
</tr>
<tr>
<td>1.3.1.25.2.7.4</td>
<td>Command 'New Breakpoint'</td>
<td>939</td>
</tr>
<tr>
<td>1.3.1.25.2.7.5</td>
<td>Command 'New Data Breakpoint'</td>
<td>939</td>
</tr>
<tr>
<td>1.3.1.25.2.7.6</td>
<td>Command 'Edit Breakpoint'</td>
<td>940</td>
</tr>
<tr>
<td>1.3.1.25.2.7.7</td>
<td>Command 'Enable Breakpoint'</td>
<td>940</td>
</tr>
<tr>
<td>1.3.1.25.2.7.8</td>
<td>Command 'Disable Breakpoint'</td>
<td>940</td>
</tr>
<tr>
<td>1.3.1.25.2.7.9</td>
<td>Command 'Toggle Breakpoint'</td>
<td>940</td>
</tr>
<tr>
<td>1.3.1.25.2.7.10</td>
<td>Command 'Step Over'</td>
<td>941</td>
</tr>
<tr>
<td>1.3.1.25.2.7.11</td>
<td>Command 'Step Into'</td>
<td>941</td>
</tr>
<tr>
<td>1.3.1.25.2.7.12</td>
<td>Command 'Step Out'</td>
<td>941</td>
</tr>
<tr>
<td>1.3.1.25.2.7.13</td>
<td>Command 'Run to Cursor'</td>
<td>942</td>
</tr>
<tr>
<td>1.3.1.25.2.7.14</td>
<td>Command 'Set Next Statement'</td>
<td>942</td>
</tr>
<tr>
<td>1.3.1.25.2.7.15</td>
<td>Command 'Show Next Statement'</td>
<td>942</td>
</tr>
<tr>
<td>1.3.1.25.2.7.16</td>
<td>Command 'Force Values'</td>
<td>943</td>
</tr>
<tr>
<td>1.3.1.25.2.7.17</td>
<td>Command 'Write Values'</td>
<td>944</td>
</tr>
<tr>
<td>1.3.1.25.2.7.18</td>
<td>Command 'Unforce Values'</td>
<td>944</td>
</tr>
<tr>
<td>1.3.1.25.2.7.19</td>
<td>Command 'Force All Values from &lt;Device.Application&gt;'</td>
<td>945</td>
</tr>
<tr>
<td>1.3.1.25.2.7.20</td>
<td>Command 'Write All Values from &lt;Device.Application&gt;'</td>
<td>946</td>
</tr>
<tr>
<td>1.3.1.25.2.7.21</td>
<td>Command 'Unforce All Values from &lt;Device.Application&gt;'</td>
<td>946</td>
</tr>
<tr>
<td>1.3.1.25.2.7.22</td>
<td>Command 'Flow Control'</td>
<td>947</td>
</tr>
<tr>
<td>1.3.1.25.2.7.23</td>
<td>Menu 'Core Dump'</td>
<td>947</td>
</tr>
<tr>
<td>1.3.1.25.2.7.24</td>
<td>Command 'Display Mode' - 'Binary', 'Decimal', 'Hexade-</td>
<td></td>
</tr>
<tr>
<td></td>
<td>decimal'</td>
<td>948</td>
</tr>
</tbody>
</table>
Command 'Start'

Symbol: \[;\] keyboard shortcut: [F5]

**Function:** This command starts the application (status: “RUN”).

**Call:** Menu bar: “Debug”; context menu of object: “Application”

**Requirement:** The application is in online mode and its status is “STOP”.

Executing this command from the “Debug” menu will affect the application that is currently in focus.

See also

- § Chapter 1.3.1.11.5 “Downloading the application code, logging in, and starting the PLC” on page 304

Command 'Stop'

Symbol: \[\]; keyboard shortcut: [Shift]+[F8]

**Function:** This command stops the application (status: “STOP”).

**Call:** Menu bar: “Debug”; context menu of object: “Application”

**Requirement:** The application is in offline mode and its status is “RUN”.

Executing this command from the “Debug” menu will affect the application that is currently in focus.

Command 'Single Cycle'

Keyboard shortcut [Ctrl]+[F5]

**Function:** This command executes the active application for one cycle.

**Call:** Main menu “Debug”.

**Requirement:** The application is in online mode and the program is halted at a program step.

Command 'New Breakpoint'

Symbol: \[\], keyboard shortcut [Alt]+[F7].

**Function:** This command opens the “Breakpoint Properties” dialog box.

**Call:** Main menu “Debug”.

**Requirement:** The application must be in online mode.

See also

- § Chapter 1.3.1.12.2 “Using breakpoints” on page 310

Command 'New Data Breakpoint'

Symbol: \[\\]

**Function:** This command opens the “New Breakpoint” dialog.
Call: Menu bar: “Debug”.

Requirement:

- The application is in online mode.
- The device description file of the target device contains the entries for the "data breakpoints" functionality.

See also

- Appendix A.3.1.25.3.8 “Dialog ‘New Breakpoint’” on page 1044
- Appendix A.3.1.12.2 “Using breakpoints” on page 310

Command 'Edit Breakpoint'

Symbol: 

Function: This command opens the “Breakpoint Properties” dialog box.

Call: Main menu “Debug”.

Requirement: The application is in online mode and the cursor is halted at a breakpoint.

See also

- Appendix A.3.1.12.2 “Using breakpoints” on page 310

Command 'Enable Breakpoint'

Function: This command enables a disabled breakpoint.

Call: Main menu “Debug”.

Requirement: The application is in online mode and the cursor is halted at a disabled breakpoint.

See also

- Appendix A.3.1.12.2 “Using breakpoints” on page 310

Command 'Disable Breakpoint'

Function: This command disables an enabled breakpoint.

Call: Main menu “Debug”.

Requirement: The application is in online mode and the cursor is halted at an enabled breakpoint.

See also

- Appendix A.3.1.12.2 “Using breakpoints” on page 310

Command 'Toggle Breakpoint'

Keyboard shortcut [F9]

Function: This command sets a breakpoint or clears an existing breakpoint.

Call: Main menu “Debug”.

Requirement: The application is in online mode. The cursor is positioned at a breakpoint.

See also

- Appendix A.3.1.12.2 “Using breakpoints” on page 310
Command 'Step Over'

Symbol ☑, shortcut [F10]

**Function:** The command executes the statement where the program is currently located and halts before the next statement in the POU.

**Call:** Menu bar: “Debug”

**Requirement:** The application is in online mode and the program is halted at the current break position (debug mode).

If the executed statement contains a call (from a program, function block instance, function, method, or action), then the subordinate POU is processed completely in one step and returned to the call. Then it halts before the next statement (in the next line of code).

> Click “Step Into” to jump to a subordinate POU and process it in single steps.

See also

- ⇨ Chapter 1.3.1.12.3 “Stepping through a program” on page 313
- ⇨ Chapter 1.3.1.25.2.7.11 “Command ‘Step Over’” on page 941

Command 'Step Into'

Symbol ☑, shortcut [F11]

**Function:** The command executes the statement where the program is currently located and halts before the next statement.

**Call:** Menu bar: “Debug”

**Requirement:** The application is in online mode and the program is halted at the current break position (debug mode).

If the executed statement contains a call (from a program, function block instance, function, method, or action), then the program execution jumps to this subordinate POU. Its code opens in a separate editor. The first statement there is executed and the program execution halts before the next statement. The new current breakpoint position is then in the called POU.

> Click “Step Over” to remain in the currently active POU and execute the call in one step.

See also

- ⇨ Chapter 1.3.1.12.3 “Stepping through a program” on page 313
- ⇨ Chapter 1.3.1.25.2.7.10 “Command ‘Step Over’” on page 941

Command 'Step Out'

Symbol ☑, shortcut [Ctrl]+[F11]

**Function:** The command executes the program until the next return and halts afterwards.

**Call:** Menu bar: “Debug”
**Requirement:** The application is in online mode and the program is halted at the current breakpoint position (debug mode).

If the current breakpoint position is in a subordinate POU, then this is run through to the end. Then the program execution jumps back to the calling point in the calling POU and halts there (in the line with the call).

If the current breakpoint position is in the main program, then the POU is run through to the end. Then the program execution jumps back to the beginning (to the program start at the first line of code in the POU) and halts there.

See also

- Chapter 1.3.1.12.3 “Stepping through a program” on page 313

**Command ‘Run to Cursor’**

Symbol: ⏯️

**Function:** The command executes a program until a specified position as marked by the cursor.

**Call:** Menu bar: “Debug”

**Requirement:** The application is in online mode and the program is halted at the current breakpoint position (debug mode). Moreover, you have marked any line of code in any POU with the cursor.

The statements between the current breakpoint position and the cursor position are executed in one step. Then the execution halts at the cursor position, which then becomes the next breakpoint position. Remember that the line of code where you placed the cursor is reached but not executed.

See also

- Chapter 1.3.1.12.3 “Stepping through a program” on page 313

**Command ‘Set Next Statement’**

Symbol: 🎉

**Function:** The command determines which statement is executed next.

**Call:** Menu bar: “Debug”

**Requirement:** The application is in online mode and the program is halted at the current breakpoint position (debug mode). Moreover, you have marked any line of code in any POU with the cursor.

The line of code marked with the cursor becomes the current breakpoint position without executing the statements in between or the statement that jumped to it.

See also

- Chapter 1.3.1.12.3 “Stepping through a program” on page 313

**Command ‘Show Next Statement’**

Symbol: ➤

**Function:** The command displays the program statement that is processed in the next step.

**Call:** Menu bar: “Debug”
**Requirement:** The application is in online mode and the program is halted at the current break position (debug mode). The break position is in a line of code that you cannot see.

The command makes the window with the current breakpoint position active (in the code highlighted in yellow and marked with the symbol) and makes the breakpoint position to become visible. This is useful if you have multiple editors open and the breakpoint position is hidden in an inactive editor.

See also
- © Chapter 1.3.1.12.3 “Stepping through a program” on page 313

---

**Command 'Force Values’**

Keyboard shortcut \[F7\]

**Function:** This command sets a permanent predefined value to a variable on the controller.

**Call:** Main menu “Debug”.

**Requirement:** The application is in online mode.

---

**CAUTION!**

Unusual changes to variable values in an application currently running on the controller can lead to undesired behavior of the controlled machinery.

Evaluate possible dangers before forcing variable values. Take the respective safety precautions. Depending on the controlled machinery, the result may lead to damage to machinery and equipment or injury to health and life of personnel.

With this command, CODESYS permanently sets one or more variables of the active application to defined values on the controller. This is done at the beginning and end of a processing cycle. Processing sequence: 1) read inputs, 2) force values, 3) process code, 4) force values, 5) write outputs.

You can prepare values as follows:
- Click in the “Prepared value” field in the declaration section and type in the value. For Boolean variables, you change the value by clicking the field.
- Click in the inline monitoring field in the implementation section of the FBD/LD/IL editor
- Click in the “Prepared value” field in the monitoring view and type in the value.

A forced value is marked with the forced symbol ( ).

CODESYS forces the value until you explicitly lift it by
- Clicking “Unforce Values”
- Lifting the force in the “Prepare Value” dialog box
- Logging out of the application

---

The command “Force Values [All Applications]” affects all application in the project and is not included in a menu by default.

See also
- © Chapter 1.3.1.25.3.7 “Dialog Box ‘Prepare Value” on page 1043
- © Chapter 1.3.1.25.2.7.18 “Command ‘Unforce Values” on page 944
- © Chapter 1.3.1.12.4 “Forcing and writing variables” on page 315
Command 'Write Values'

Keyboard shortcut [Ctrl]-[F7]

**Function:** This command sets a predefined value to a variable on the controller one time.

**Call:** Main menu “Debug”.

**Requirement:** The application is in online mode.

---

**CAUTION!**

Unusual changes to variable values in an application currently running on the controller can lead to undesired behavior of the controlled machinery. Evaluate possible dangers before forcing variable values. Take the respective safety precautions. Depending on the controlled machinery, the result may lead to damage to machinery and equipment or injury to health and life of personnel.

---

With this command, one or more variables of the active application are set to defined values on the controller one time. Writing is done one time at the beginning of the next cycle.

Values are prepared by

- Clicking in the field “Prepared value” in the declaration section
- Clicking in the inline monitoring field in the implementation section
- Clicking in the field “Prepared value” in the watch window

---

The command “Write Values [All Applications]” affects all application in the project and is not included in a menu by default.

---

See also

- ☞ Chapter 1.3.1.25.2.7.16 “Command ‘Force Values’” on page 943
- ☞ Chapter 1.3.1.12.4 “Forcing and writing variables” on page 315

---

Command 'Unforce Values'

Keyboard shortcut [Alt]+[F7]

**Function:** This command resets the forcing of all variables. The variables receive their current values from the PLC.

**Call:** “Debug”.

**Requirement:** The application is in online mode.

The “Remove Force List” command has the same functionality as this command with one difference. If the “Remove Force List” command cannot be executed for all forced values, then no message is displayed.

---

**CAUTION!**

Unusual changes to variable values in an application currently running on the PLC can lead to undesired behavior of the controlled machinery. Evaluate possible dangers before forcing variable values. Take the respective safety precautions. Depending on the controlled system, the result may lead to damage to machinery and equipment or injury to health and life of personnel.
The command “Force Values [All Applications]” affects all application in the project and is not included in a menu by default.

See also
- Chapter 1.3.1.25.2.7.16 “Command ‘Force Values’” on page 943
- Chapter 1.3.1.12.4 “Forcing and writing variables” on page 315

Command ‘Force All Values from <Device.Application>’

Function: This command resets all values of variables from the selection <Device.Application> to predefined values permanently.

Call:
- Context menu of the application in the device tree
- Context menu in the editor of a POU from the selected application

Requirement: The application is in online mode.

CAUTION!
Unusual changes to variable values in an application currently running on the controller can lead to undesired behavior of the controlled machinery.
Evaluate possible dangers before forcing variable values. Take the respective safety precautions. Depending on the controlled machinery, the result may lead to damage to machinery and equipment or injury to health and life of personnel.

With this command, CODESYS permanently sets one or more variables of the active application to defined values on the PLC. This is done at the beginning and end of a processing cycle. Order of processing: 1) read inputs, 2) force values, 3) process code, 4) force values, 5) write outputs.

You can prepare values as follows:
- Click in the “Prepared value” field in the declaration part and type in the value. For Boolean variables, you change the value by clicking the field.
- Click in the inline monitoring field in the implementation part of the FBD/LD/IL editor
- Click in the “Prepared value” field in the monitoring view and type in the value.

A forced value is marked with the forced symbol (F).
CODESYS forces the value until you explicitly lift it by
- Clicking “Unforce Values”
- Clicking “Unforce All Values from <Device.Application>”
- Lifting the force in the “Prepare Value” dialog
- Logging out of the application

The command “Force Values [All Applications]” affects all application in the project and is not included in a menu by default.

See also
- Chapter 1.3.1.25.3.7 “Dialog Box ‘Prepare Value’” on page 1043
- Chapter 1.3.1.25.2.7.18 “Command ‘Unforce Values’” on page 944
- Chapter 1.3.1.12.4 “Forcing and writing variables” on page 315
- Chapter 1.3.1.25.2.7.21 “Command ‘Unforce All Values from <Device.Application>’” on page 946
Command 'Write All Values from <Device.Application>'

**Function:** This command resets all values of variables from the selection <Device.Application> to predefined values one time.

**Call:**
- Context menu of the application in the device tree
- Context menu in the editor of a POU from the selected application

**Requirement:** The application is in online mode.

---

**CAUTION!**

Unusual changes to variable values in an application currently running on the controller can lead to undesired behavior of the controlled machinery. Evaluate possible dangers before forcing variable values. Take the respective safety precautions. Depending on the controlled machinery, the result may lead to damage to machinery and equipment or injury to health and life of personnel.

With this command, one or more variables of the selected <Device.Application> are set to defined values on the PLC one time. Writing is done one time at the beginning of the next cycle.

You can prepare values as follows:
- Click in the "Prepared value" field in the declaration part and type in the value. For Boolean variables, you change the value by clicking the field.
- Click in the inline monitoring field in the implementation part of the FBD/LD/IL editor
- Click in the "Prepared value" field in the monitoring view and type in the value.

See also
- ![Chapter 1.3.1.25.2.7.17 “Command 'Write Values’” on page 944](#)
- ![Chapter 1.3.1.25.2.7.19 “Command 'Force All Values from <Device.Application>’” on page 945](#)
- ![Chapter 1.3.1.12.4 “Forcing and writing variables” on page 315](#)

Command 'Unforce All Values from <Device.Application>'

**Function:** This command resets the forcing of all values of the variables from the selected <Device.Application>. The variables receive their current values from the PLC.

**Call:**
- Context menu of the application in the device tree
- Context menu in the editor of a POU from the selected application

**Requirement:** The application is in online mode.

---

**CAUTION!**

Unusual changes to variable values in an application currently running on the controller can lead to undesired behavior of the controlled machinery. Evaluate possible dangers before forcing variable values. Take the respective safety precautions. Depending on the controlled machinery, the result may lead to damage to machinery and equipment or injury to health and life of personnel.

See also
- ![Chapter 1.3.1.25.2.7.19 “Command 'Force All Values from <Device.Application>’” on page 945](#)
- ![Chapter 1.3.1.25.2.7.18 “Command 'Unforce Values’” on page 944](#)
Command 'Flow Control'

**Function:** This command activates and deactivates the flow control.

**Call:** Menu "Debug"

**Requirement:** The application is in online mode.

---

### NOTICE!

An active flow control extends application runtime.

When "Confirmed Online Mode" is activated in the communication settings, a dialog prompt appears when switching on the flow control to cancel the process.

When flow control is activated, it is not possible to use breakpoints or step through the program.

---

See also

- ↕ Chapter 1.3.1.12.6 “Flow control” on page 319

---

### Menu 'Core Dump'

1.3.1.25.2.7.23.1 Command 'Load Core Dump' .................... 947
1.3.1.25.2.7.23.2 Command 'Create Core Dump' .................... 947
1.3.1.25.2.7.23.3 Command 'Close Core Dump' .................... 948
1.3.1.25.2.7.23.4 Command 'Load Device Log from Core Dump' .... 948

---

### Command 'Load Core Dump'

**Function:** CODESYS scans the project directory for core dump files. When a new core dump is forced with the "Create Core Dump" command, the dump file is automatically loaded from the controller to the project directory. If multiple core dump files are available, then CODESYS prompts you to choose whether the latest file should be opened in the project. You can also select one of the other files.

When a file is loaded into the project, an online view of the application appears with state of the application at the time when the core dump was generated. You can then view the variable values afterwards. Finally, the call tree is also available.

**Call:** Main menu "Debug ➔ Core Dump".

**Requirement:** The application is in offline mode.

---

### NOTICE!

You can close the core dump view only by clicking "Close Core Dump". The "Logout" command has no effect in this view.

---

See also

- ↕ Chapter 1.3.1.13.7 “Analysing errors with core dump” on page 349
- ↕ Chapter 1.3.1.25.2.7.23.2 “Command ‘Create Core Dump’” on page 947
- ↕ Chapter 1.3.1.25.2.7.23.3 “Command ‘Close Core Dump’” on page 948

---

### Command 'Create Core Dump'

**Function:** This command causes CODESYS to check whether a core dump file is already available on the controller.

If a core dump file is available, then CODESYS prompts you to load this file to the project directory.
With the following requirements, CODESYS generates a new dump file with the current application data:
- A core dump file is still not available or CODESYS has rejected a core dump file from being loaded.
- The application is currently stopped at breakpoint or an exception has occurred.

The generated core dump file is saved directly to the project directory: `<project name>_<device name>_<application name>_<application Guid>.core`. You can cancel the file generation by clicking the button in the status bar.

The amount of detail in the dump depends on the support from the runtime system. Runtime systems that are appropriate for this purpose generate just one dump in the case of an exception error. The core dump output from clicking “Load Core Dump” can therefore be used for error analysis.

**Call:** Main menu “Debug ➔ Core Dump”.

**Requirement:** The application is in online mode.

**See also**
- Chapter 1.3.1.13.7 “Analysing errors with core dump” on page 349
- Chapter 1.3.1.25.2.7.23.1 “Command ‘Load Core Dump’” on page 947

**Command 'Close Core Dump’**

**Function:** This command closes the core dump view of the application that is open in the project.

**Call:** Main menu “Debug ➔ Core Dump”.

**Requirement:** The application is in offline mode and you have loaded a core dump file to the project from the controller.

**See also**
- Chapter 1.3.1.13.7 “Analysing errors with core dump” on page 349

**Command 'Load Device Log from Core Dump’**

**Function:** This command imports the controller log list that was saved with the last generated core dump. The log list is displayed in the same view as in online mode in the “Log” tab of the device editor.

**Call:** Main menu “Debug ➔ Core Dump”.

**Requirement:** The application is in offline mode and a core dump is open in the project.

**See also**
- Chapter 1.3.1.13.7 “Analysing errors with core dump” on page 349
- Chapter 1.3.1.25.1.19.8 “Tab ‘Log’” on page 750

**Command 'Display Mode’ - 'Binary', 'Decimal', 'Hexadecimal’**

**Function:** These commands in the “Display Mode” submenu are used for setting the format of values in the display mode when monitoring in online mode.

**Call:** Main menu “Debug”.

**Requirement:** The project is in either online or offline mode.
The "Binary" and "Hexadecimal" display modes are unsigned, and "Decimal" is signed.

See also
- § Chapter 1.3.1.13.1.1 “Calling of monitoring in programming objects ” on page 323

Menu 'Tools'

<table>
<thead>
<tr>
<th>Command</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.3.1.25.2.8.1 Command ‘Package Manager’</td>
<td>949</td>
</tr>
<tr>
<td>1.3.1.25.2.8.2 Command ‘Library Repository’</td>
<td>951</td>
</tr>
<tr>
<td>1.3.1.25.2.8.3 Command ‘License Manager’</td>
<td>953</td>
</tr>
<tr>
<td>1.3.1.25.2.8.4 Command ‘License Repository’</td>
<td>956</td>
</tr>
<tr>
<td>1.3.1.25.2.8.5 Command ‘Device Repository’</td>
<td>957</td>
</tr>
<tr>
<td>1.3.1.25.2.8.6 Command ‘Scripting’ - ‘Execute Script File’</td>
<td>958</td>
</tr>
<tr>
<td>1.3.1.25.2.8.7 Command ‘Scripting’ - ‘Enable Script Tracing’</td>
<td>958</td>
</tr>
<tr>
<td>1.3.1.25.2.8.8 Command ‘Scripting’ - ‘Scripts’</td>
<td>959</td>
</tr>
<tr>
<td>1.3.1.25.2.8.9 Command ‘Customize’</td>
<td>959</td>
</tr>
<tr>
<td>1.3.1.25.2.8.10 Command ‘Options’</td>
<td>959</td>
</tr>
<tr>
<td>1.3.1.25.2.8.11 Command ‘Import and Export Options’</td>
<td>959</td>
</tr>
<tr>
<td>1.3.1.25.2.8.12 Command ‘Device Reader’</td>
<td>960</td>
</tr>
</tbody>
</table>

Command ‘Package Manager’

Symbol: 📦

**Function:** The command opens the dialog box “Package Manager”, where you can install, uninstall and manage packages.

**Call:** Main menu “Tools”

You can also call the package manager as a stand-alone application with the command line.

**Table 148: “Currently Installed Packages”**

<table>
<thead>
<tr>
<th>List of the already installed packages with “Name”, “Version”, “Installation Date”, “Update Info”, “License Information”</th>
</tr>
</thead>
<tbody>
<tr>
<td>If a package originates from the CODESYS Store, CODESYS marks it with the red package symbol 📦 instead of the yellow symbol 🟢.</td>
</tr>
<tr>
<td>If an update is available, CODESYS indicates this with an entry in the column “Update info” and with the symbol 🔄.</td>
</tr>
<tr>
<td>“Refresh”                                                           Updates the list</td>
</tr>
</tbody>
</table>

2020/12/10  3ADR010583, 1, en_US  949
### Install

Opens the standard dialog box for browsing for a package in the file system. The standard file type is ".package.

You can also install two versions of a package.

Following the selection of the package the setup wizard appears with the dialog box:

- **"Installation - License Agreement"**
  
  CODESYS also displays the "Checksum" of the package in this dialog box. Appears only if the package contains a license agreement.

- **"Choose Setup Type"**
  
  The options are package-dependent.
  
  - **"Complete setup":** CODESYS installs all components.
  
  - **"Typical setup":** CODESYS installs a standard set from components defined in the package.

- **"Installation - target system versions"**: You select which of the already existing target system versions are to be updated by the package installation. You must select at least one version profile.

### Uninstall

Uninstalls the selected package

- If "Display Versions" is not activated, CODESYS uninstalls all versions of the selected package.

- If "Display Versions" is activated and you have selected a package node at the highest level, CODESYS uninstalls all versions of the selected package.

- If "Display Versions" is activated and you have selected a single package version, CODESYS uninstalls precisely this version.

### Details

Opens the dialog box "Details" for the selected package with the following tabs:

- **"Package Details"**
  
  - **"Name":** package name
  
  - **"Version":**
  
  - **"Checksum":** SHA-1 CRC of the package
  
  - **"Vendor":**
  
  - **"Copyright":**
  
  - **"Description":**
  
  - **"Installation date":**

- **"License agreement":**

- **"Installation log":**

### Search for updates in background

☑️: CODESYS automatically searches for updates after each start of the programming system and after that once per hour.

### Display versions

☑️: Displays all versions of the installed packages.

---

You can compare the "CRC" that CODESYS displays in the dialog box "Details" and in the dialog box "Installation - License Agreement" of the setup wizard with the package CRC of the package vendor. This way you can ensure that you have installed an original package.

If you install a newer version of the programming system in the same installation directory as the previous version, the license information for the packages already installed is retained and CODESYS displays the information in the dialog box "Package Manager".
**Table 149: “Updates”**

<table>
<thead>
<tr>
<th>“Search Updates”</th>
<th>Searches on your system and in the CODESYS Store for updates for the selected package. CODESYS displays updates found in the column “Update info” of the package list.</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Download”</td>
<td>Installs the update package with the help of the dialog box “Download Package”. To do this, click on the “Download and Installation” button in the dialog box “Download Package”.</td>
</tr>
</tbody>
</table>

**Table 150: “CODESYS Store”**

<table>
<thead>
<tr>
<th>“Rating”</th>
<th>Submit an evaluation of the package</th>
</tr>
</thead>
<tbody>
<tr>
<td>“CODESYS Store”</td>
<td>Link to the homepage of the store</td>
</tr>
</tbody>
</table>

See also

- Chapter 1.3.1.21.1 “Installing/Uninstalling a package” on page 377

**Command ‘Library Repository’**

Symbol:

**Function:** The command opens the “Library Repository” dialog. In this dialog you define which libraries are installed on the local system and are thus available for your application.

**Call:** Menu bar: “Tools”

**Dialog ‘Library Repository’**

**Table 151: “Location”**

Display of the directory on the local system in which the library files are located. The libraries in this “Location” are listed in the “Installed libraries” area.

<table>
<thead>
<tr>
<th>“Edit Locations”</th>
<th>Opens the “Edit Repository Locations” dialog.</th>
</tr>
</thead>
</table>

1. You can only use empty directories for new repositories.
2. You can also use existing repositories as locations.
3. The “System” repository is not editable; CODESYS indicates this by the italic lettering of the entry.

**Table 152: Dialog “Edit Repository Locations”**

List of the repositories with “Location” and “Name”

<table>
<thead>
<tr>
<th>“Add”</th>
<th>Creates a new repository. Opens the “Repository Location” dialog. The selected directory (“Location” input field) has to be either empty or an existing valid repository. “Name” is the input field for a symbolic repository name.</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Edit”</td>
<td>Opens the “Repository Location” dialog (see “Add”)</td>
</tr>
<tr>
<td>“Remove”</td>
<td>A dialog box open, asking whether only the entry should be removed from the list of repositories, or whether the directory with the library files should be deleted from the file system. If you want to delete the directory, you have to confirm this.</td>
</tr>
</tbody>
</table>
Table 153: “Installed libraries”

<table>
<thead>
<tr>
<th>“Company”</th>
<th>List box for filtering the displayed libraries.</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Install”</td>
<td>Opens the “Select Library” dialog. Possible filters:</td>
</tr>
<tr>
<td></td>
<td>● “Compiled CODESYS library files (*.compiled-library)”.</td>
</tr>
<tr>
<td></td>
<td>● “Compiled CODESYS library files (*.compiled-library-v3)” ab V3 SP15</td>
</tr>
<tr>
<td></td>
<td>● “Library files (*.library)” for still uncompiled library projects</td>
</tr>
<tr>
<td></td>
<td>● “All files (*)”</td>
</tr>
<tr>
<td>“Uninstall”</td>
<td>Uninstalls the selected library.</td>
</tr>
<tr>
<td>“Export”</td>
<td>Opens the default dialog for saving the library project to the local file system. The file type is Library files (<em>.library), Compiled library files (</em>.compiled-library), or Compiled library files (*.compiled-library-v3).</td>
</tr>
<tr>
<td>“Find”</td>
<td>Searches for libraries and function blocks.</td>
</tr>
<tr>
<td></td>
<td>Opens the “Find Library” dialog. When you enter a string in the input field, CODESYS displays the libraries that it finds with a corresponding string.</td>
</tr>
<tr>
<td>“Details”</td>
<td>Opens the “Details” dialog with details from the project information of the library for the selected version of a library. You find the following information by clicking “More” in the “Details” dialog:</td>
</tr>
<tr>
<td></td>
<td>● “Size”: Specified in bytes</td>
</tr>
<tr>
<td></td>
<td>● “Created”: Creation date</td>
</tr>
<tr>
<td></td>
<td>● “Changed”: Date of the last change</td>
</tr>
<tr>
<td></td>
<td>● “Last access”: Date</td>
</tr>
<tr>
<td></td>
<td>● “Attributes”</td>
</tr>
<tr>
<td></td>
<td>● “Properties”</td>
</tr>
<tr>
<td>“Dependencies”</td>
<td>For the selected library, the “Dependencies” dialog opens, showing the dependencies on other libraries. “Title”, “Version” and “Company” are shown for each library reference. References that function via placeholders are displayed according to the syntax: #&lt;placeholder name&gt;.</td>
</tr>
<tr>
<td>“Group by category”</td>
<td>● ✔: Grouping by library category</td>
</tr>
<tr>
<td></td>
<td>● ☐: Alphabetical sorting</td>
</tr>
<tr>
<td></td>
<td>The categories are defined by external description files '*.libcat.xml'.</td>
</tr>
</tbody>
</table>

Table 154: “Library Profiles”

A library profile defines the library version with which CODESYS resolves a library placeholder if a certain compiler version is set in the project.

| “Import”  | Imports a *.libraryprofile file. |
|           | If the import already contains existing placeholder entries, a query appears asking whether CODESYS should overwrite it. |
| “Export”  |Exports an xml file with the extension *.libraryprofile with the assignments of the selected placeholder entries; you can only select a single entry of a “Compiler version”. |

Placeholder resolutions can also be defined in the target device currently in use and even by a specific local specification in the Placeholders dialog in the Library Manager.
See also

- % Chapter 1.3.1.19.1 “Information for library developers” on page 372
- % Chapter 1.3.1.25.1.31 “Object ‘Project Information’” on page 811
- % Chapter 1.3.1.25.1.25 “Object ‘Library Manager’” on page 771
- % Chapter 1.3.1.25.3.2 “Dialog ‘Library Reference Conversion’” on page 1040
- % Chapter 1.3.1.19.4 “Exporting library files” on page 374

Command 'License Manager'

Symbol: ⚑

Function: This command opens the wizard for configuring licenses for CODESYS add-on products. The wizard starts with the “License Manager - Select target” dialog.

Call: Menu bar: “Tools”.

The License Manager can handle licenses for CODESYS add-on products on the local computer, as well as licenses for RTS add-on products on devices. It supports both the installation in a soft container and on a dongle.

Dialog 'License Manager - Select Target'

This is the start dialog of the License Manager wizard. Here you decide where the license will be installed.

| “Workstation” | Local computer |
| "Device"      | Controller. The connection to this device must be configured correctly in order to license (“Communication Settings” tab of the device editor). |

After clicking “Next”, you decide the container where you want to manage the licenses.

Dialog 'License Manager - Select container'

| “Dongle” | A corresponding dongle must be connected to the computer or device. Not all devices support dongles. |
| “Softcontainer” | CODESYS Security Key. A corresponding soft container must be registered in the CodeMeter Control Center. The CODESYS installation provides an existing soft container. |

If you are installing a product on your local computer (“Workstation”), then the “License Manager” opens immediately for the specific selection of the dongle or soft container, and the next actions. This happens after you choose the container type and click “Next >”.

If you are licensing the add-on product for a controller, then first the dialog opens for selecting the device in the network after you click “Next >”. This dialog corresponds to the classic view of the “Communication Settings” tab of the device editor.
| “Container” | Depending on whether “Dongle” or “Softcontainer” was selected: Drop-down list of all CODESYS dongles or soft containers that were found on the computer or device. |
| “Products” | List of all installed products that are subject to licensing. A prepended symbol indicates the existence and validity of the license. On the right side of the window, the following information is displayed for the selected product and corresponding licenses: |
| | “Name” |
| | “Company” |
| | “Unit counter” |
| | “License quantity” |
| | “Usage period” |
| | “Feature map” |
| | “Activation time” |
| | “Expiration time” |
| | “Firmcode” |
| | “Product code” |
| | “Description” |

**“Install Licenses”**

Opens the dialog “Install licenses on <computer> - Select Operation”:

- “Activate license”: Opens the dialog “Install licenses on <computer> - Activate License” (see more below)
- “Request license”: Opens the dialog “Install licenses on <computer> - Request License” (see more below)
- “Install license”: Opens the dialog “Install licenses on <computer> - Install License” (see more below)

**“Additional Functions”**

Opens the menu with the following actions:

- “Return license”: Opens the “Return Licenses” (see more below)
- “Restore license”: This function is available in the case of device licensing only. Opens the “Restore Licenses” dialog (see more below)

**Table 155: “Install Licenses on <computer> - Activate License”**

This is the recommended way to activate a license available via the License Server when you have an Internet connection. Requirement: The computer has an Internet connection.

**“Ticket ID”**

Input field for the ticket ID that you received from the software vendor. The ticket ID consists of five sets of five alphanumerical characters (for example: LYSQ3-ZU93K-24LWC-XGWJ8-5AY7H).

**“License server”**

Drop-down list of the license server that provides the license for activating the product. You receive the server URL from the software vendor.

**“Select Ticket from Repository”**

Opens the “License Repository” dialog.

**“Next”**

CODESYS connects to the license server.

- If the specified ticket contains only one license, then a dialog opens to confirm the successful activation after completion of the server action.
- If the specified ticket contains multiple licenses, then the dialog “Install licenses - Select Licenses” opens with a list of these licenses (see description below).
Table 156: “Install Licenses - Select Licenses”

<table>
<thead>
<tr>
<th>Selection of the licenses to be activated for the ticket which you specified in the dialog “Install Licenses - Activate License”.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>“Name”</strong></td>
</tr>
<tr>
<td><strong>“Available”</strong></td>
</tr>
<tr>
<td><strong>“Used”</strong></td>
</tr>
<tr>
<td><strong>Total</strong></td>
</tr>
<tr>
<td><strong>Next</strong></td>
</tr>
</tbody>
</table>

Table 157: “Install Licenses on <computer> - Request License”

If the computer does not have an Internet connection, then you can generate a context file from this dialog. The file “WibuCmRaC” is then transmitted to the license server via an Internet-enabled computer. When activation is complete, a license update file “WibuCmRaU” is provided for download.

| **“Software vendor”** | Input field for firmcodes from the software vendor that provided the license for activating the product. As an alternative, you can select the software vendor from the drop-down list. |
| **“Context file”** | Location and name |

Table 158: “Install Licenses on <computer> - Install License”

If you downloaded a license update file from the Internet during software activation, then you can use this dialog to install the license on your dongle. To do this, specify the path of the license update file in the input field.

Table 159: “Return License”

If the license permits, you can return it in order to reactivate it later on another computer.

| **“Ticket ID”** | Field for specifying the ticket ID that was used for licensing. |
| **“License server”** | Drop-down list for selecting the license server that provides the license for activating the product. You receive the server URL from the software vendor. |
| **“Load License(s)”** | Button for showing all current licenses installed for the given ticket ID on the server in the “Licenses” window. |
| **“Licenses”** | List of licenses available on the server for the given ticket ID. The following information for the selected license is displayed next to the window on the right: |
| | ● “Name” |
| | ● “Number of activations” |
| | ● “Return allowed” |
| | ● “Activation type” |
| | ● “Activation date” |
| | ● “Firmcode” |
| | ● “Comment” |
| **“Return License(s)”** | Button for returning the selected license(s). These can be reactivated later on another system. |
Table 160: "Install Licenses - Restore Licenses"

<table>
<thead>
<tr>
<th>“Ticket ID”</th>
<th>Field for specifying the ticket ID that was used for licensing that has already occurred.</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Restore”</td>
<td>If a corresponding license backup file is found, then the license is reactivated in the device.</td>
</tr>
</tbody>
</table>

See also

- Chapter 1.3.1.21.2 “Licensing of products” on page 377
- Chapter 1.3.1.25.1.19.2 “Tab ‘Communication Settings’” on page 741
- Chapter 1.3.1.25.2.8.4 “Command ‘License Repository’” on page 956

Command ‘License Repository’

Symbol: 

**Function:** This command opens the dialog box “License Repository” for viewing information about the individual licenses.

**Call:** Main menu “Tools”

**Requirements:** CODESYS is in offline or online mode.

In the license repository, after entering the ticket number, you can obtain information about the licenses concerned from the central license server.

To do this you can paste the ticket number(s) from the clipboard or import it/them from a text file.

Table 161: “Tickets”

<table>
<thead>
<tr>
<th>&lt;List of the ticket IDs imported into the repository for components requiring licenses&gt;</th>
<th>If you select an entry in the list of tickets, the name and the status of the licensed component are displayed here.</th>
</tr>
</thead>
</table>
| “Licenses” | ✔️: License available and valid  

  🎎: License found, but invalid  

  🐘: License not found  

  In the right-hand part of the dialog box you will then receive the following information about this license: |
| “Name”: name of the product to be licensed  

  “Item number”: item number in the license server.  

  “Return allowed”: It is possible to have this license deactivated so that it can be re-activated on another system.  

  “Can be activated”: you can have the license activated via the license manager.  

  “Activation quantity”: number of activations that have taken place so far.  

  “Activation date”: date of the current activation  

  “Container serial”  

  “Firm codes”  

  “Comment” |
**"Import Tickets"**

The standard dialog box for browsing the local file system appears. If you open a text file containing one or more "tickets", i.e. license numbers, these are imported into the repository. Alternatively you can also insert the numbers from the clipboard into the list.

See also
- § “Licensing in CODESYS” on page 375
- § “License manager” on page 376

**Command 'Device Repository'**

Symbol: ![Command](image)

**Function:** This command opens the “Device Repository” dialog. This dialog is used for managing the devices that are installed on the local system and can be integrated into CODESYS projects.

**Call:** Menu bar: “Tools”.

**Dialog 'Device Repository'**

<table>
<thead>
<tr>
<th>CAUTION!</th>
</tr>
</thead>
<tbody>
<tr>
<td>Do not change the internal device repository manually. Do not copy any files to or from the repository. Always use the device repository dialog to install or uninstall devices.</td>
</tr>
</tbody>
</table>

**"Location"**

Shows the device repository directory on the local system. The list box shows the currently set save locations. By default, CODESYS creates the system repository during installation. The devices of the selected location are listed in the “Installed device descriptions” field.

**"Edit Locations"**

Opens the “Edit Repository Locations” dialog.

**Table 162: Dialog 'Edit Repository Locations'**

<table>
<thead>
<tr>
<th>List of the repositories with “Location” and “Name”.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>&quot;Add&quot;</strong></td>
</tr>
<tr>
<td><strong>&quot;Edit&quot;</strong></td>
</tr>
<tr>
<td><strong>&quot;Remove&quot;</strong></td>
</tr>
</tbody>
</table>

**Table 163: “Installed device descriptions”**

<table>
<thead>
<tr>
<th>List of device descriptions in multilevel tree structure. Shows all device descriptions with “Name”, “Vendor”, and “Version”. The top nodes represent device categories, for example PLCs, fieldbuses, and logical devices.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>&quot;String for full-text search in all devices&quot;</strong></td>
</tr>
<tr>
<td><strong>&quot;Vendor&quot;</strong></td>
</tr>
</tbody>
</table>
“Install” Opens the “Install Device Description” dialog.
For the default devices with file type “*.devdesc.xml”. You can also select manufacturer-specific description files, such as “*.gsd” files for PROFIBUS DP modules, “*.eds” and “*.dcf” files for CAN devices.
When you click “OK” to confirm the selection, CODESYS inserts the new device into the device repository. If an error occurs during installation (for example, missing files that are referenced by the device description), then CODESYS reports the error to the lower part of the device repository dialog.

“Uninstall” Removes the selected device. If you delete the device from the device repository, then it is no longer available for use in the programming system.

“Renew Device Repository” Updates all devices in the device repository.
When new versions of import plug-ins are available, some device descriptions may be outdated. The affected devices are marked with a warning symbol (⚠️). This command opens a dialog to confirm the update.

“Download Missing Device Descriptions” Opens when you use devices in your project that are not available in the device repository. When you execute this command, a list of missing devices is displayed. There you can select the corresponding devices for download.

“Details” Opens the “Details” dialog for the selected device description. This dialog provides additional information from the device description file.

NOTICE!
During installation, CODESYS copies the device description files and all additional reference files to an internal location. Therefore, any changes to the original files no longer influence the installed devices. You must reinstall the devices to make any changes effective. We recommended that you change the internal version number of a device description after a modification.

See also
- Chapter 1.3.1.20.1 “Installing devices” on page 374
- Chapter 1.3.1.25.3.13.7 “Dialog ‘Options’ – ‘Device Description Download’” on page 1079

Command 'Scripting' - 'Execute Script File'
Symbol: 🌐
Function: This command opens a dialog for selecting and then executing the script file (*.py).
Call: Menu bar: “Tools ➔ Scripting”.
See also
- Chapter 1.3.1.22 “Using scripts” on page 388

Command 'Scripting' - 'Enable Script Tracing'
Symbol: 🌐
Function: This command makes CODESYS print all commands from the script file to the message view. Use this command for monitoring and debugging scripts. A blue frame around the symbol indicates that the option is active.
Call: Main menu “Tools ➔ Scripting”.
Command 'Scripting' - 'Scripts'

**Function:** This command executes a script that is stored in the ScriptDir folder.

**Call:** Menu bar: “Tools ➔ Scripting ➔ Scripts”.

**Requirement:** The ScriptDir folder exists in the CODESYS installation directory. Python scripts are stored in this folder with the file extension .py.

All scripts that are contained in the ScriptDir folder are executable as menu commands and are sorted alphabetically by file name.

Command 'Customize'

**Function:** This command opens the “Customize” dialog box, where you can customize the menus, toolbars, and keyboard shortcuts according to your individual requirements.

**Call:** Main menu “Tools”

Command 'Options'

**Function:** The command opens the dialog box “Options” for the configuration of the CODESYS options. These options define the behavior and appearance of the CODESYS user interface. CODESYS saves the settings in your current user profile on your local system. The current profile specifies the standard settings.

**Call:** “Tools” menu

See also

- Chapter 1.3.1.25.3.13.1 “Dialog ‘Options’ - ‘SFC Editor’” on page 1074
- Chapter 1.3.1.25.3.13.4 “Dialog ‘Options’ - ‘CFC Editor’” on page 1076
- Chapter 1.3.1.25.3.13.6 “Dialog ‘Options’ - ‘Declaration Editor’” on page 1078
- Chapter 1.3.1.25.3.13.9 “Dialog ‘Options’ - ‘Device Editor’” on page 1081
- Chapter 1.3.1.25.3.13.7 “Dialog ‘Options’ - ‘Device Description Download’” on page 1079
- Chapter 1.3.1.25.3.13.8 “Dialog ‘Options’ - ‘FBD, LD, and IL’” on page 1079
- Chapter 1.3.1.25.3.13.12 “Dialog ‘Options’ - ‘International Settings’” on page 1083
- Chapter 1.3.1.25.3.13.2 “Dialog ‘Options’ - ‘Libraries’” on page 1075
- Chapter 1.3.1.25.3.13.3 “Dialog ‘Options’ - ‘Library Download’” on page 1075
- Chapter 1.3.1.25.3.13.13 “Dialog ‘Options’ - ‘Load and Save’” on page 1083
- Chapter 1.3.1.25.3.13.15 “Dialog ‘Options’ - ‘PLCopenXML’” on page 1084
- Chapter 1.3.1.25.3.13.16 “Dialog ‘Options’ - ‘Proxy Settings’” on page 1084
- Chapter 1.3.1.25.3.13.17 “Dialog ‘Options’ - ‘Refactoring’” on page 1085
- Chapter 1.3.1.25.3.13.11 “Dialog ‘Options’ - ‘SmartCoding’” on page 1082
- Dialog ‘Options’ - ‘Store’
- Chapter 1.3.1.25.3.13.20 “Dialog ‘Options’ - ‘Text Editor’” on page 1086

Command 'Import and Export Options'

**Function:** This command opens the “Import and Export Options” dialog. Here you can configure the export and import of selected settings of the CODESYS options. The settings are saved to an XML file with the default extension (options.xml).

**Call:** Menu bar: “Tools”.

Dialog 'Import and Export Options'
In the table, you can select the categories of options, either user-specific or machine-specific (computer), whose current settings are to be exported to the XML file.

**“File”**: Path of the export file in the local file system. Example: D:\system1.options.xml.

Button Opened the default dialog to search for an existing file in the local file system, or to create one. The “File type” option export (*.options.xml) is preset.

**“Import selected options”**

**“File”**: Path of the options export file whose contents are to be imported.

Button Opens the default dialog to search for an existing file of type option export (*.options.xml) in the local file system.

After you click “OK” to close the dialog, the settings described in the file are applied to the project.

---

**Command 'Device Reader’**

**Function**: The command opens the standard “Select Device” dialog and reads the license and product information of the selected controller. This license and product information is displayed in the “Device Reader” dialog.

**Call**: Menu bar: “Tools”

**Requirement**: No applications exist on the controller.

---

---

Table 164: Dialog “Device Reader”

<table>
<thead>
<tr>
<th>“Status of available device features”</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>“Product”</strong></td>
</tr>
<tr>
<td><strong>“Feature”</strong></td>
</tr>
<tr>
<td>Example: CNC is a “Feature” of SoftMotion.</td>
</tr>
<tr>
<td><strong>“License Active/Count”</strong></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>

---

---

See also

- Chapter 1.3.1.1.1 “Setting CODESYS options” on page 102

---

---

---

---

---

---
Menu 'Window'

1.3.1.25.2.9.1 Command 'Next Editor' .............................................................. 961
1.3.1.25.2.9.2 Command 'Previous Editor' ....................................................... 961
1.3.1.25.2.9.3 Command 'Close All Editors' ..................................................... 961
1.3.1.25.2.9.4 Command 'Close All Editors of Inactive Applications' ................. 962
1.3.1.25.2.9.5 Command 'Reset Window Layout' ............................................. 962
1.3.1.25.2.9.6 Command 'New Horizontal Tab Group' ..................................... 962
1.3.1.25.2.9.7 Command 'New Vertical Tab Group' .......................................... 962
1.3.1.25.2.9.8 Command 'Float' ........................................................................ 963
1.3.1.25.2.9.9 Command 'Dock' ....................................................................... 963
1.3.1.25.2.9.10 Command 'Auto Hide' ............................................................ 963
1.3.1.25.2.9.11 Command 'Next Pane' ............................................................. 963
1.3.1.25.2.9.12 Command 'Previous Pane' ....................................................... 963
1.3.1.25.2.9.13 Command 'Toggle First Pane' .................................................. 964
1.3.1.25.2.9.14 Command 'Toggle Second Pane' ............................................ 964
1.3.1.25.2.9.15 Command 'Windows' .............................................................. 964
1.3.1.25.2.9.16 Command 'Close All Editors But This' ..................................... 965
1.3.1.25.2.9.17 Command 'Select Object in Navigator' .................................... 965
1.3.1.25.2.9.18 Command 'Select Parent Object in Navigator' ........................ 965
1.3.1.25.2.9.19 Commands of the Submenu 'Window' .................................... 965

Command 'Next Editor'

Keyboard shortcut: [Ctrl]+[F6]

Function: This command switches focus from the currently active view to the next view. The next view is identified by the tab to the right of the currently active tab.

Call: Main menu “Window”

Requirement: At least one object is open.

See also
● Chapter 1.3.1.25.2.9.2 “Command ‘Previous Editor’” on page 961

Command 'Previous Editor'

Keyboard shortcut: [Shift]+[Ctrl]+[F6]

Function: This command switches focus from the currently active view to the previous view. The previous view is identified by the tab to the left of the currently active tab.

Call: Main menu “Window”

Requirement: At least one object is open.

See also
● Chapter 1.3.1.25.2.9.1 “Command ‘Next Editor’” on page 961

Command 'Close All Editors'

Symbol: 🗑

Function: This command closes all currently open editor views.

Call: Main menu “Window”

Requirement: At least one editor is open.
Command 'Close All Editors of Inactive Applications'

**Function:** This command closes all editor views for objects that are located directly below a currently inactive application. Object editors in the POU view remain open.

**Call:** Main menu “Window”

**Requirement:** At least one object of an inactive application is open.

See also
- Chapter 1.3.1.25.2.9.3 “Command ‘Close All Editors’” on page 961
- Chapter 1.3.1.25.2.9.16 “Command ‘Close All Editors But This’” on page 965

Command 'Reset Window Layout'

**Function:** This command resets all currently open windows and views to their default positions. You are prompted for a confirmation before the command is executed.

**Call:** Main menu “Tools”

Command 'New Horizontal Tab Group'

**Symbol:** ⏴

**Function:** This command moves the currently active view to a new, separate tab group below the existing one.

**Call:** Main menu “Window” or context menu of the tab

**Requirement:** Several editor views are open as tabs next to each other. If you open another object in the editor, then this is automatically included in the tab group that is currently in focus.

See also
- Chapter 1.3.1.25.2.9.7 “Command ‘New Vertical Tab Group’” on page 962

Command 'New Vertical Tab Group'

**Symbol:** ▼

**Function:** This command moves the currently active view to a new, separate tab group to the right of the existing one.

**Call:** Main menu “Window” or context menu of the tab

**Requirement:** Several editor views are open as tabs next to each other. If you open another object in the editor, then this is automatically included in the tab group that is currently in focus.

See also
- Chapter 1.3.1.25.2.9.6 “Command ‘New Horizontal Tab Group’” on page 962
Command 'Float'

**Function:** This command releases a docked view from its frame in the user interface and reposi-
sitions it on the screen as a floating window.

**Call:** Main menu “Window”

**Requirement:** The application is in online mode.

This window can then be positioned outside of the user interface. Use the “Dock” command to
return a floating window to the frame of the user interface.

See also
- \* Chapter 1.3.1.25.2.9.9 “Command ‘Dock’” on page 963

Command 'Dock'

**Function:** This command returns a floating window, which was released by the “Float” com-
mand, to the frame of the user interface.

**Call:** Main menu “Window”

See also
- \* Chapter 1.3.1.25.2.9.8 “Command ‘Float’” on page 963

Command 'Auto Hide'

Keyboard shortcut: [F7]

**Function:** This command shows or hides a view.

**Call:** Main menu “Window”

Hide simply means that CODESYS minimizes the view to a tab at the bottom of the user inter-
face which is visible only when you move the mouse over the tab. The command functions like
a check box. When a window is hidden, the check box is selected in the menu. When you click
the command again, the checkbox is cleared and the window is shown.

Command 'Next Pane'

Keyboard shortcut: [F6]

**Function:** This command sets the focus on the next pane.

**Call:** Main menu “Window”

**Requirement:** An object is open that contains two or more panes.

Example: If an object is open in the ST editor and the cursor is currently in the declaration sec-
tion, then command sets the focus to implementation section.

See also
- \* Chapter 1.3.1.25.2.9.12 “Command ‘Previous Pane’” on page 963

Command 'Previous Pane'

Keyboard shortcut: [Shift]+[F6]

**Function:** This command sets the focus on the previous pane.

**Call:** Main menu “Window”

**Requirement:** An object is open that contains two or more panes.
Example: If an object is open in the ST editor and the cursor is currently in the declaration section, then command sets the focus to implementation section.

See also

- % Chapter 1.3.1.25.2.9.11 “Command ‘Next Pane’” on page 963

Command ‘Toggle First Pane’

Keyboard shortcut [Alt]+[F6]

Function: This command shows and hides the declaration view.

Call: “Window”.

Requirement: The cursor is positioned in the editor of one of the following objects:

- POU
- Transition
- Method
- Get accessor method of a property
- Set accessor method of a property
- Visualization

You can also toggle the subviews by means of the buttons.

See also

- % Chapter 1.3.1.25.2.9.14 “Command ‘Toggle Second Pane’” on page 964

Command ‘Toggle Second Pane’

Function: This command shows and hides the implementation view.

Call: “Window”.

Requirement: The cursor is positioned in the editor of one of the following objects:

- POU
- Transition
- Method
- Get accessor method of a property
- Set accessor method of a property
- Visualization

You can also toggle the subviews by means of the buttons.

See also

- % Chapter 1.3.1.25.2.9.13 “Command ‘Toggle First Pane’” on page 964

Command ‘Windows’

Function: This command opens the “Windows” dialog box, which lists all open objects. You can then activate or close any of the listed views.
Call: Main menu “Window”

Command 'Close All Editors But This'
Function: This command closes all editor views except the currently open one.
Call: Right-click the tab
Requirement: At least two objects are open.
See also
● § Chapter 1.3.1.25.2.9.3 “Command ’Close All Editors’” on page 961
● § Chapter 1.3.1.25.2.9.4 “Command ’Close All Editors of Inactive Applications’” on page 962

Command 'Select Object in Navigator'
Function: This command selects the object of the active editor in the device tree.
Call: Right-click the tab
Requirement: At least one object is open.
This command is executed automatically when you select the “Track active editor” option for the device tree.
See also
● § Chapter 1.3.1.25.2.9.18 “Command ’Select Parent Object in Navigator’” on page 965

Command 'Select Parent Object in Navigator'
Function: This command selects the parent object in the device tree.
Call: Right-click the tab
Requirement: At least one object is open.
See also
● § Chapter 1.3.1.25.2.9.17 “Command ’Select Object in Navigator’” on page 965

Commands of the Submenu 'Window'
Function: The command activates the selected window.
Call: Main menu “Window”
For each opened editor window the menu “Window” contains a command “<n><object name>”.
Choosing this command activates the corresponding window. In offline mode CODESYS adds the extension “(Offline)”. To differentiate between the implementation or the instances of a function block the extension “(Impl)” or “<instance path>” is added.
Menu 'Help'

1.3.1.25.2.10.1  Command 'Contents'............................................................... 966
1.3.1.25.2.10.2  Command 'Index'............................................................... 966
1.3.1.25.2.10.3  Command 'Find'............................................................... 966
1.3.1.25.2.10.4  Command 'About'............................................................... 967

Command 'Contents'

Symbol: ; keyboard shortcut: [Ctrl]+[Shift]+[F1]

Function: This command opens the CODESYS help.

Call: Menu bar: “Help”.

Command 'Index'

Symbol: ; keyboard shortcut: [Ctrl]+[Shift]+[F2]

Function: This command opens the CODESYS help.

Call: Menu bar: “Help”.

An index search is not possible in the online help. The “Index” tab opens in the offline help.

All index entries of the help are listed alphabetically in the index view.

<table>
<thead>
<tr>
<th>“Look for”</th>
<th>As you type letters into the input field, CODESYS searches automatically for matches in the index list.</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Display”</td>
<td>Opens the help page for the highlighted index entry in the list and displays the title of the help page and location of the help file (*.chm) in the “Index results for &lt;index entry&gt;” view. When several pages are found and then displayed in this view, then you view a specific help page by clicking its entry in the list. Clicking an entry in the index list achieves the same result.</td>
</tr>
</tbody>
</table>

Command 'Find'

Symbol: 💡

Function: This command opens the CODESYS help.

Call: Menu bar: “Help”.

In the online help, you can run a full-text search from the input field on the top right of the help page. The “Find” tab opens in the offline help.

Table 165: Tab ‘Search’

<table>
<thead>
<tr>
<th>“Search for”</th>
<th>Combo box for defining the search term or for selecting the 25 most recent search terms.</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Search in titles only”</td>
<td>The search is performed only in the titles of the help pages.</td>
</tr>
<tr>
<td>“Display partial matches”</td>
<td>Displays terms also as search results that include the search term.</td>
</tr>
<tr>
<td>“Limit to .... matches ”</td>
<td>Limits the number of search results. Maximum value: 1000</td>
</tr>
<tr>
<td>“Find”</td>
<td>Starts the full-text search.</td>
</tr>
</tbody>
</table>
Command 'About'

**Function:** This command opens a splash screen with information about the CODESYS version and copyright. In addition, buttons are available for detailed information about the version, license, and acknowledgments.

**Call:** Main menu “Help”.

<table>
<thead>
<tr>
<th>“Version Info”</th>
<th>Opens the “Detailed Version Information” dialog box with a list of CODESYS components and information about the operating system.</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Export”:</td>
<td>Exports the detailed version information as a *.txt file or in any other format.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>“License Info”</th>
<th>Opens the “License Information” dialog box.</th>
</tr>
</thead>
<tbody>
<tr>
<td>● “Plug-in”:</td>
<td>Drop-down list for the plug-in to display the license information</td>
</tr>
<tr>
<td>● “Software License”:</td>
<td>License information about selected “Plug-in”</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>“Acknowledgments”</th>
</tr>
</thead>
</table>

Menu 'SFC'

1.3.1.25.2.11.1 Command ‘Init Step’................................................................. 967
1.3.1.25.2.11.2 Command ‘Insert Step’.............................................................. 968
1.3.1.25.2.11.3 Command ‘Insert Step After’..................................................... 968
1.3.1.25.2.11.4 Command ‘Insert Transition After’............................................. 968
1.3.1.25.2.11.5 Command ‘Insert Transition’...................................................... 969
1.3.1.25.2.11.6 Command ‘Insert Step-Transition’............................................... 969
1.3.1.25.2.11.7 Command ‘Insert Step-Transition After’..................................... 969
1.3.1.25.2.11.8 Command ‘Add Entry Action’...................................................... 970
1.3.1.25.2.11.9 Command ‘Add Exit Action’........................................................ 970
1.3.1.25.2.11.10 Command ‘Parallel’.................................................................. 970
1.3.1.25.2.11.11 Command ‘Alternative’............................................................ 971
1.3.1.25.2.11.12 Command ‘Insert Branch’.......................................................... 971
1.3.1.25.2.11.13 Command ‘Insert Branch Right’.................................................. 971
1.3.1.25.2.11.14 Command ‘Insert Action Association’......................................... 972
1.3.1.25.2.11.15 Command ‘Insert Action Association After’.............................. 973
1.3.1.25.2.11.16 Command ‘Insert Jump’............................................................ 973
1.3.1.25.2.11.17 Command ‘Insert Jump After’.................................................... 973
1.3.1.25.2.11.18 Command ‘Insert Macro’.......................................................... 974
1.3.1.25.2.11.19 Command ‘Insert Macro After’.................................................. 974
1.3.1.25.2.11.20 Command ‘Zoom Into Macro’.................................................... 974
1.3.1.25.2.11.21 Command ‘Zoom Out of Macro’.................................................. 974
1.3.1.25.2.11.22 Command ‘Paste After’.............................................................. 975
1.3.1.25.2.11.23 Command ‘Change Duplication' - 'Set'........................................ 975
1.3.1.25.2.11.24 Command ‘Change Duplication' - 'Remove'.................................. 975
1.3.1.25.2.11.25 Command ‘Do Not Display Embedded Objects’............................ 976

Command 'Init Step'

**Symbol:** ![Symbol](image)

**Function:** This command converts the selected step into an initial step.

**Call:** Main menu “SFC”
After you choose this command, the borders of the step element change to a double line. The previous initial step is automatically displayed as a normal step with a single-line border.

You can also activate and deactivate the property “Init step” in the properties dialog of a step. However, CODESYS does not automatically adjust the settings of other steps.

This command is useful if you want to convert a diagram. When you create a new SFC object, it automatically includes an initial step followed by a transition (TRUE) and a jump back to the initial step.

Please note: In online mode, it is possible to reset the diagram to the initial step using the SFCInit and SFCReset flags.

See also

- § Chapter 1.3.1.24.1.4.6 “SFC flags” on page 447
- § Chapter 1.3.1.24.1.4.7.6 “SFC element properties” on page 459

**Command 'Insert Step'**

Symbol: 

**Function:** This command inserts a step before the selected point.

**Call:** Menu bar “SFC”; context menu in SFC editor

The new step is named Step<n> by default, where n is an incremental number starting at 0 for the first step that is inserted in addition to the initial step. The name can be edited by clicking on it.

See also

- § Chapter 1.3.1.25.2.11.7 “Command 'Insert Step-Transition After’” on page 969
- § Chapter 1.3.1.25.2.11.1 “Command 'Init Step’” on page 967
- § Chapter 1.3.1.24.1.4.7.1 “SFC elements 'Step' and 'Transition’” on page 452

**Command 'Insert Step After'**

Symbol: 

**Function:** This command inserts a step after the selected point.

**Call:** Menu bar “SFC”; context menu in SFC editor

The new step is named Step<n> by default, where n is an incremental number starting at 0 for the first step that is inserted in addition to the initial step. The name can be edited by clicking on it.

See also

- § Chapter 1.3.1.25.2.11.7 “Command 'Insert Step-Transition After’” on page 969
- § Chapter 1.3.1.25.2.11.1 “Command 'Init Step’” on page 967
- § Chapter 1.3.1.24.1.4.7.1 “SFC elements 'Step' and 'Transition’” on page 452

**Command 'Insert Transition After'**

Symbol: 

**Function:** This command inserts a transition after the selected point.

**Call:** Menu bar “SFC”; context menu in SFC editor
The new transition is named Trans<n> by default, where n is an incremental number beginning at 0 for the first transition. The name can be edited by clicking on it.

See also
- § Chapter 1.3.1.25.2.11.7 “Command 'Insert Step-Transition After’” on page 969
- § Chapter 1.3.1.24.1.4.7.1 “SFC elements 'Step' and 'Transition’” on page 452

Command 'Insert Transition'

Symbol: \( \uparrow \uparrow \)

Function: This command inserts a transition before the selected point.

Call: Menu bar “SFC”; context menu in SFC editor

The new transition is named Trans<n> by default, where n is an incremental number beginning at 0 for the first transition. The name can be edited by clicking on it.

See also
- § Chapter 1.3.1.25.2.11.7 “Command 'Insert Step-Transition After’” on page 969
- § Chapter 1.3.1.24.1.4.7.1 “SFC elements 'Step' and 'Transition’” on page 452

Command 'Insert Step-Transition'

Symbol: \( \uparrow \nabla \)

Function: This command inserts a step and a transition before the selected point.

Call: Main menu “SFC”

If you have selected a step, then CODESYS inserts a new step-transition combination. If you have selected a transition, then a new transition-step combination is inserted.

The new step is named Step<n> by default, where n is an incremental number beginning at 0 for the first step that was inserted in addition to the initial step. The new transition is named Trans<n> by default. You can edit the default names directly by clicking the names.

See also
- § Chapter 1.3.1.25.2.11.7 “Command 'Insert Step-Transition After’” on page 969
- § Chapter 1.3.1.25.2.11.1 “Command 'Init Step’” on page 967
- § Chapter 1.3.1.24.1.4.7.1 “SFC elements 'Step' and 'Transition’” on page 452

Command 'Insert Step-Transition After'

Symbol: \( \nabla \uparrow \)

Function: This command inserts a step and a transition after the selected point.

Call: Main menu “SFC”

If you have selected a step, then CODESYS inserts a new transition-step combination. If you have selected a transition, then a new step-transition combination is inserted.

The new step is named Step<n> by default, where n is an incremental number beginning at 0 for the first step that was inserted in addition to the initial step. The new transition is named Trans<n> by default. You can edit the default names directly by clicking the names.

See also
- § Chapter 1.3.1.25.2.11.6 “Command 'Insert Step-Transition’” on page 969
Command 'Add Entry Action'

Symbol: 

**Function:** This command opens the “Add Entry Action” dialog box where you define a new entry action. Depending on the SFC options, a dialog prompt may open for selecting the duplication mode for the new step action.

**Call:** Menu bar: “SFC”; context menu of the selected step element.

**Requirement:** A step element in SFC is selected.

The entry action is opened automatically in the ST editor. The step element contains an \( \equiv \) in the lower left corner.

**Options:**

- **“Copy reference: A new step will call the same actions”:** If the step is copied in SFC, the link to the step action(s) is also copied. The steps copied by each other will therefore call the same actions.
- **“Copy implementation: New action objects are created for a new step.”:** This means that the step actions for a copied step are embedded. By default, the generated action objects appear below an SFC box in the device tree or “POUs” view. These objects contain a copy of the original implementation code of the respective action.

The display of the embedded objects can be activated and deactivated in the tree by means of the “Show Embedded Objects” and “Hide Embedded Objects” commands in the context menu of an SFC object.

See also

- § Chapter 1.3.1.25.3.13.1 “Dialog ‘Options’ - ‘SFC Editor’” on page 1074
- § Chapter 1.3.1.9.3.4.1 “Programming the sequential function chart (SFC)” on page 202
- § Chapter 1.3.1.24.1.4.7.2 “SFC element ‘Action’” on page 454
- § Chapter 1.3.1.25.2.11.25 “Command ‘Do Not Display Embedded Objects’” on page 976

Command 'Add Exit Action'

Symbol: 

**Function:** This command opens the “Add Exit Action” dialog box where you define a new exit action. Depending on the SFC options, a dialog prompt may open for selecting the duplication mode for the new step action. For more information, refer to the help page for the “Add Exit Action” command.

**Call:** Menu bar: “SFC”; context menu of the selected step element.

**Requirement:** A step element in SFC is selected.

See also

- § Chapter 1.3.1.25.2.11.8 “Command ‘Add Entry Action’” on page 970
- § Chapter 1.3.1.25.3.13.1 “Dialog ‘Options’ - ‘SFC Editor’” on page 1074
- § Chapter 1.3.1.9.3.4.1 “Programming the sequential function chart (SFC)” on page 202
- § Chapter 1.3.1.24.1.4.7.2 “SFC element ‘Action’” on page 454
- § Chapter 1.3.1.25.2.11.25 “Command ‘Do Not Display Embedded Objects’” on page 976

Command 'Parallel'

Symbol: 

**Function:** This command converts the selected alternative branch into a parallel branch.

**Call:** Main menu “SFC”

**Requirement:** The horizontal connecting line of a branch is selected.
Please note that after you convert a branch, you must check and modify the layout of the steps and transitions before and after the branch.

See also
● % Chapter 1.3.1.25.2.11.11 “Command ‘Alternative’” on page 971

Command ‘Alternative’

Symbol: ☑

Function: This command converts the selected parallel branch into an alternative branch.

Call: Main menu “SFC”

Requirement: The horizontal connecting line of a branch is selected.

Please note that after you convert a branch, you must check and modify the layout of the steps and transitions before and after the branch.

See also
● % Chapter 1.3.1.25.2.11.10 “Command ‘Parallel’” on page 970

Command ‘Insert Branch’

Symbol: ☑

Function: This command inserts a branch to the left of the selected point.

Call: Main menu “SFC”

This command functions similar to the “Insert Branch Right” command.

See also
● % Chapter 1.3.1.24.1.4.7.3 “SFC element ‘Branch’” on page 457
● % Chapter 1.3.1.25.2.11.13 “Command ‘Insert Branch Right’” on page 971

Command ‘Insert Branch Right’

Symbol: ☑

Function: This command inserts a branch to the right of the selected point.

Call: Main menu “SFC”

The type of inserted branch depends on the selected element.

● If the uppermost element of the selected elements is a transition or an alternative branch, then CODESYS inserts an alternative branch.

● If the uppermost element of the selected elements is a step, a macro, a jump, or a parallel branch, then CODESYS inserts a parallel branch with the Branch<x> jump marker, where x is an incremental number. You can edit the default name of the jump marker or define the jump marker as a jump destination.

● If a common element of an existing branch (horizontal line) is selected, then CODESYS inserts the new branch line as a branch line on the far right. If an entire branch line of an existing branch is selected, then CODESYS inserts the new branch line directly to the right as a new branch line.

Please note: You can convert a branch into another type with the “Alternative” and “Parallel” commands.
The following image shows a new inserted parallel branch generated by the “Insert Branch Right” command while the Step11 step was selected. CODESYS automatically inserts a step (Step2 in the example).

Processing in online mode: If \( t_2 \) yields TRUE, then CODESYS executes Step2 immediately after step11 and before \( t_3 \) is passed.

Thus, CODESYS processes both branch lines as opposed to alternative branches.

Example of alternative branch

The following image shows a new inserted alternative branch generated by the “Insert Branch Right” command while the \( t_4 \) transition was selected. CODESYS automatically inserts a step (Step32 in the example), a preceding transition, and a subsequent transition (\( t_{41}, t_{42} \)).

Processing in online mode: If Step3 is active, then CODESYS passes the subsequent transitions (\( t_4, t_{41} \)) from left to right. The first branch line of the main branch with the first transition yielding TRUE is passed. Therefore, only one branch line is processed as opposed to with a parallel branch.

See also

- § Chapter 1.3.1.24.1.4.7.3 “SFC element ‘Branch’” on page 457
- § Chapter 1.3.1.25.2.11.12 “Command ‘Insert Branch’” on page 971
- § Chapter 1.3.1.25.2.11.10 “Command ‘Parallel’” on page 970
- § Chapter 1.3.1.25.2.11.11 “Command ‘Alternative’” on page 971

Command 'Insert Action Association'

Symbol: \( \text{inded} \)

Function: This command assigns an IEC action to a step.

Call: Main menu “SFC”
**Requirement:** A step is selected.

CODESYS inserts the action element to the right of the selected step element.

If you have already assigned one or more actions to the step, they are displayed in an action list. The new action is then inserted as follows:

- If you selected the step element, the action is inserted as the first action of the step at first position of the action list.
- If you selected one of the available actions in the action list, then the action is inserted directly above the selected action.

The left section of the action element includes the qualifier (N by default). You enter the action name in the right section. To set this value, click in the box to obtain an editing frame. You must have already created this action as a POU in the project.

You can also edit the qualifier. Valid qualifiers are described in the chapter “Qualifiers for Actions in SFC”.

See also

- Chapter 1.3.1.25.2.11.15 “Command ’Insert Action Association After’” on page 973
- Chapter 1.3.1.24.1.4.4 “Qualifiers for actions in SFC” on page 445

**Command 'Insert Action Association After’**

**Symbol:** :eq:

**Function:** This command assigns an IEC action to a step.

**Call:** Main menu “SFC”

**Requirement:** A step is selected.

This command functions similar to the “Insert Action Association” command. The difference between the two commands is that CODESYS inserts the new action in the last position of the action list, not the first position. If you select an action in the action list, then CODESYS inserts the new action at the bottom of the list, not at the top.

See also

- Chapter 1.3.1.25.2.11.14 “Command ’Insert Action Association’” on page 972
- Chapter 1.3.1.24.1.4.4 “Qualifiers for actions in SFC” on page 445

**Command 'Insert Jump'**

**Symbol:** :eq:

**Function:** This command inserts a jump element before the selected element.

**Call:** Main menu “SFC”

**Requirement:** A step is selected.

CODESYS automatically inserts the jump with the Step destination. Then, you still have to replace this jump destination with an actual destination by using the input assistant.

See also

- Chapter 1.3.1.24.1.4.7.4 “SFC element ‘Jump’” on page 458
- Chapter 1.3.1.25.2.11.17 “Command ’Insert Jump After’” on page 973

**Command 'Insert Jump After’**

**Symbol:** :eq:

**Function:** This command inserts a jump element after the selected element.
Call: Main menu “SFC”
CODESYS automatically inserts the jump with the Step destination. Then, you still have to replace this jump destination with an actual destination by using the input assistant.

See also
- Chapter 1.3.1.24.1.4.7.4 “SFC element ‘Jump’” on page 458
- Chapter 1.3.1.25.2.11.16 “Command ‘Insert Jump’” on page 973

Command 'Insert Macro'

Symbol: 
Function: This command inserts a macro element before the selected element.
Call: Main menu “SFC”
The new macro is named Macro<x> by default, where x is an incremental number beginning at 0 for the first macro. You can edit the default name directly by clicking the name.
To edit the macro, click “Zoom Into Macro” in the macro editor.

See also
- Chapter 1.3.1.25.2.11.20 “Command ‘Zoom Into Macro’” on page 974
- Chapter 1.3.1.25.2.11.19 “Command ‘Insert Macro After’” on page 974

Command 'Insert Macro After'

Symbol: 
Function: This command inserts a macro element after the selected element.
Call: Main menu “SFC”
This command functions similar to the “Insert Macro” command.

See also
- Chapter 1.3.1.25.2.11.20 “Command ‘Zoom Into Macro’” on page 974
- Chapter 1.3.1.25.2.11.18 “Command ‘Insert Macro’” on page 974

Command ‘Zoom Into Macro’

Symbol: ⬆
Function: This command opens a macro for editing in the macro editor.
Call: Main menu “SFC”
Requirement: A macro is selected.
By choosing this command, CODESYS closes the main view of the SFC editor and opens the macro editor. This is also an SFC editor for editing the section of the SFC diagram that is displayed as a macro box in the main view.
Click “Zoom Out of Macro” to return to the main view.

See also
- Chapter 1.3.1.25.2.11.21 “Command ‘Zoom Out of Macro’” on page 974

Command 'Zoom Out of Macro'

Symbol: ⬇
**Function:** This command closes the macro editor and returns to the main view of the SFC editor.

**Call:** Main menu “SFC”

**Requirement:** A macro is open in the macro editor.

See also

- § Chapter 1.3.1.25.2.11.20 “Command 'Zoom Into Macro’” on page 974

---

**Command 'Paste After'**

Symbol: 📋

**Function:** This command pastes the elements from the clipboard after the selected position.

**Call:** Main menu “SFC”.

---

**Command 'Change Duplication' - 'Set'**

**Function:** This command embeds every step action or transition, which is called by a step or transition in the SFC box, with the caller. In this way, the action or transition object can be called only from exactly this caller (pseudo-embedding). The result is that copying step and transition elements that call actions or transitions automatically creates new action or transition objects. The implementation code is also copied.

**Call:** Menu bar: “SFC”.

For more details about duplication mode, refer to the help page for the SFC element properties and the instructions for adding step actions.

> Pseudo-embedded objects can be hidden in the “Devices” or “POUs” view by means of a command.

See also

- § Chapter 1.3.1.24.1.4.7.6 “SFC element properties” on page 459
- § Chapter 1.3.1.9.3.4.1 “Programming the sequential function chart (SFC)” on page 202

---

**Command 'Change Duplication' - 'Remove'**

**Function:** This command removes the embedding of action, transition, and property objects by a step or transition that calls it for the entire SFC box. In this way, the pseudo-embedding of the action, transition, or property objects is removed. If step or transition elements are copied, which call actions, transitions, or properties, then the copying calls the same actions and transitions as the source.

**Call:** Menu bar: “SFC”.

For more details about duplication mode, refer to the help page for the SFC element properties and the instructions for adding step actions.

> Pseudo-embedded objects can be hidden in the “Devices” or “POUs” view by means of a command.
See also
- Chapter 1.3.1.24.1.4.7.6 “SFC element properties” on page 459
- Chapter 1.3.1.9.3.4.1 “Programming the sequential function chart (SFC)” on page 202
- Chapter 1.3.1.25.2.11.25 “Command 'Do Not Display Embedded Objects'” on page 976

Command 'Do Not Display Embedded Objects'

**Function:** This command causes action and transition objects, which are embedded in an SFC box by a step or transition, do not appear in the tree.

**Call:** Context menu of an SFC box in the “Devices” or “POUs” view.

See also
- Chapter 1.3.1.9.3.4.1 “Programming the sequential function chart (SFC)” on page 202
Menu 'CFC'

1.3.1.25.2.12.1 Command 'Edit Worksheet'..................................................... 977
1.3.1.25.2.12.2 Command 'Edit Page Size'...................................................... 978
1.3.1.25.2.12.3 Command 'Negate'................................................................. 978
1.3.1.25.2.12.4 Command 'EN/ENO'................................................................. 978
1.3.1.25.2.12.5 Command 'None'.................................................................. 979
1.3.1.25.2.12.6 Command 'R-Reset'................................................................ 979
1.3.1.25.2.12.7 Command 'S-SET'................................................................... 979
1.3.1.25.2.12.8 Command 'REF= (Reference Assignment)'............................ 979
1.3.1.25.2.12.9 Command 'Display Execution Order'....................................... 980
1.3.1.25.2.12.10 Command 'Set Start of Feedback'........................................ 980
1.3.1.25.2.12.11 Command 'Send to Front'.................................................... 981
1.3.1.25.2.12.12 Command 'Send to Back'..................................................... 981
1.3.1.25.2.12.13 Command 'Move Up'............................................................. 982
1.3.1.25.2.12.14 Command 'Move Down'......................................................... 982
1.3.1.25.2.12.15 Command 'Set Execution Order'........................................... 982
1.3.1.25.2.12.16 Command 'Order by Data Flow'........................................... 983
1.3.1.25.2.12.17 Command 'Order by Topology'............................................. 983
1.3.1.25.2.12.18 Command 'Edit Parameters'................................................... 984
1.3.1.25.2.12.19 Command 'Save Prepared Parameters to Project'.................. 985
1.3.1.25.2.12.20 Command 'Connect Selected Pins'......................................... 985
1.3.1.25.2.12.21 Command 'Unlock Connection'............................................ 986
1.3.1.25.2.12.22 Command 'Show Next Collision'......................................... 986
1.3.1.25.2.12.23 Command 'Select Connected Pins'......................................... 986
1.3.1.25.2.12.24 Command 'Reset Pins'......................................................... 986
1.3.1.25.2.12.25 Command 'Remove Unused Pins'......................................... 987
1.3.1.25.2.12.26 Command 'Add Input Pin'..................................................... 987
1.3.1.25.2.12.27 Command 'Add Output Pin'................................................... 987
1.3.1.25.2.12.28 Command 'Route All Connections'...................................... 987
1.3.1.25.2.12.29 Command 'Remove Control Point'....................................... 988
1.3.1.25.2.12.30 Command 'Create Control Point'......................................... 988
1.3.1.25.2.12.31 Command 'Connection Mark'............................................... 988
1.3.1.25.2.12.32 Command 'Create group'..................................................... 989
1.3.1.25.2.12.33 Command 'Ungroup'............................................................. 989
1.3.1.25.2.12.34 Command 'Prepare Function Block Element for Forcing'....... 989
1.3.1.25.2.12.35 Command 'Force FB Input'................................................... 989
1.3.1.25.2.12.36 Command 'Use Attributed Member as Input'....................... 990

Command 'Edit Worksheet'

Function: This command opens the “Edit Worksheet” dialog box in which you set the size of the worksheet.

Call: Main menu “CFC”

Requirements: A CFC editor is active.

Dialog box 'Edit worksheet'
| “Use following dimensions” | Here is where you set the size of the worksheet. Your change is only accepted if the size is sufficient for the existing program. |
| “Adapt the dimensions automatically” | Automatically adapts the size of the worksheet to the size of your program. |
| “Move the working sheet origin relatively” | Shifts the worksheet along the x or y axis. The input of negative numbers is permitted. |

**Command 'Edit Page Size’**

**Function:** This command opens the “Edit Page Size” dialog box, in which you change the size of the page-oriented CFC editor.

**Call:** Main menu “CFC”

**Requirements:** A page-oriented CFC editor is active.

**Dialog box 'Edit page size' dialog box**

| “Width” | Width of the page (minimum 24, maximum 1024). Elements outside of the working area are marked red. |
| “Height” | Height of the page (minimum 24, maximum 1024). Elements outside of the working area are marked red. |
| “Margin width” | Width of the margin (minimum 6, maximum 25% or page width). |
| “Set as standard for new CFC objects” | The current settings are selected as standard for new CFC objects. |

See also

- [Chapter 1.3.1.24.1.6.4.1 “CFC element 'Page’” on page 486](#)

**Command 'Negate’**

**Symbol:** -\( \triangleright \)

**Function:** This command negates the selected function block input or function block output.

**Call:** Main menu “CFC”, context menu

**Requirements:** A CFC editor is active. A function block input or function block output is selected.

**Command 'EN/ENO’**

**Symbol:** \( \triangleright EN \)

**Function:** This command adds a boolean input “\( \triangleright EN \)” (Enable) and a boolean output “\( \triangleright ENO \)” (Enable Out) to the selected function block.

**Call:** Main menu “CFC”, context menu

**Requirements:** A CFC editor is active. A function block is selected.

The added input “\( \triangleright EN \)” activates the function block. The function block is executed only if the input is \( \text{TRUE} \). The value of this signal is output at the “\( \triangleright ENO \)” output.
Command 'None'

Symbol: \[ \text{None} \]

**Function:** This command removes a Reset (R) or a Set (S) from the input of the “Output” element.

**Call:** Main menu “CFC \( \Rightarrow \) Set/Reset”, context menu “Set/Reset”

**Requirements:** A CFC editor is active. The input of an “Output” element is selected.

Command 'R-Reset'

Symbol: \[ \text{R-Reset} \]

**Function:** This command adds a reset to the input of a boolean element “Output”.

**Call:** Main menu “CFC \( \Rightarrow \) Set/Reset”, context menu “Set/Reset”

**Requirements:** A CFC editor is active. The input of an “Output” element is selected.

If an element “Output” has a reset input, the boolean output value is set to “FALSE” as soon as the value of the input is “TRUE”. The value “FALSE” remains at the output, even if the input value changes again.

See also

- Chapter 1.3.1.25.2.12.7 “Command ‘S-SET’” on page 979

Command 'S-SET'

Symbol: \[ \text{S-SET} \]

**Function:** This command adds a set (S) to the input of a boolean element “Output”.

**Call:** Main menu “CFC \( \Rightarrow \) Set/Reset”, context menu “Set/Reset”

**Requirements:** A CFC editor is active. The input of an “Output” element is selected.

If an element “Output” has a set input, the boolean output value is set to “TRUE” as soon as the value of the input is “TRUE”. The value “TRUE” remains at the output, even if the input value changes again.

See also

- Chapter 1.3.1.25.2.12.6 “Command ‘R-Reset’” on page 979

Command 'REF= (Reference Assignment)'

Symbol: \[ \text{REF=} \]

**Function:** This command assigns a reference to an element “Output”.

**Call:** Main menu “CFC \( \Rightarrow \) Set/Reset”, context menu “Set/Reset”

**Requirements:** A CFC editor is active. The input of an “Output” element is selected.
Example:

Declaration:

```plaintext
ref_int : REFERENCE TO INT;
a : INT;
```

CFC:

![Diagram of CFC example]

This corresponds to the ST-code `ref_int REF= a;`

Further information can be found in the description of the data type `REFERENCE TO`.

See also:
- Chapter 1.3.1.9.3.2.2 “Programming in the CFC editor” on page 193
- Chapter 1.3.1.24.5.13 “Reference” on page 579

Command 'Display Execution Order'

**Function:** The command temporarily shows a numbered tag for all CFC elements of the programming object.

**Call**:
- Menu bar: “CFC ➔ Execution Order”
- Context menu in the CFC editor

**Requirement:** A CFC editor is active and the “Auto Data Flow Mode” property is selected.

The numbers represent the automatically determined execution order. The execution order is determined by data flow. In the case of multiple networks, it is determined by their topological position in the editor.

The tags are hidden as soon as you click in the CFC editor.

See also:
- Chapter 1.3.1.9.3.2.1 “Automatic execution order by data flow” on page 189
- Chapter 1.3.1.25.2.12.10 “Command 'Set Start of Feedback’” on page 980
- Chapter 1.3.1.25.3.10.12 “Dialog ‘Properties’ - ‘CFC Execution Order’” on page 1054

Command 'Set Start of Feedback'

**Symbol:** ➔

**Function:** The command defines the selected element as the starting point within a feedback.

**Call**:
- Menu bar: “CFC ➔ Execution Order”
- Context menu: “Execution Order”

**Requirement:** A CFC editor is active and the “Auto Data Flow Mode” property is selected. Moreover, a network of the CFC POU contains a feedback, and an element within the feedback is selected.

In the CFC editor, the starting point within the feedbacks is decorated with the ➔ symbol. Then the element has the lowest number in the execution order within the feedbacks. At runtime, the processing of the feedback begins with this element.
Command 'Send to Front'

Symbol: 🎚

**Function:** The command numbers the elements so that the selected elements are located at the front of the execution order.

**Call:** Menu bar: “CFC ➔ Execution Order”, context menu: “Execution Order”

**Requirements:** A CFC editor is active and the “Explicit Execution Order Mode” property is selected. At least one element is selected.

The selected elements get the lowest numbers beginning at 0 while keeping the previous order. The remaining elements are numbered so that their execution order remains the same. The topological positions of the elements are retained anyway.

See also
- Chapter 1.3.1.9.3.2.1 “Automatic execution order by data flow” on page 189
- Chapter 1.3.1.25.3.10.12 “Dialog 'Properties' - 'CFC Execution Order'” on page 1054
- Chapter 1.3.1.25.2.12.12 “Command 'Send to Back'” on page 981
- Chapter 1.3.1.25.2.12.13 “Command 'Move Up'” on page 982
- Chapter 1.3.1.25.2.12.14 “Command 'Move Down'” on page 982
- Chapter 1.3.1.25.2.12.16 “Command 'Order by Data Flow'” on page 983
- Chapter 1.3.1.25.2.12.17 “Command 'Order by Topology'” on page 983

Command 'Send to Back'

Symbol: 🎚

**Function:** The command numbers the elements so that the selected elements are located at the end of the execution order.

**Call:** Menu bar: “CFC ➔ Execution Order”, context menu: “Execution Order”

**Requirements:** A CFC editor is active and the “Explicit Execution Order Mode” property is selected. At least one element is selected.

The selected elements get the highest numbers while keeping the previous order. The remaining elements are numbered so that their execution order remains the same. The topological positions of the elements are retained anyway.

See also
- Chapter 1.3.1.9.3.2.1 “Automatic execution order by data flow” on page 189
- Chapter 1.3.1.25.3.10.12 “Dialog 'Properties' - 'CFC Execution Order'” on page 1054
- Chapter 1.3.1.25.2.12.11 “Command 'Send to Front'” on page 981
- Chapter 1.3.1.25.2.12.13 “Command 'Move Up'” on page 982
- Chapter 1.3.1.25.2.12.14 “Command 'Move Down'” on page 982
- Chapter 1.3.1.25.2.12.16 “Command 'Order by Data Flow'” on page 983
- Chapter 1.3.1.25.2.12.17 “Command 'Order by Topology'” on page 983
Command 'Move Up'

Symbol: 

**Function:** The command numbers the elements so that the selected elements are located one position forward.

**Call:** Menu bar: “CFC ➔ Execution Order”; context menu: “Execution Order”

**Requirements:** A CFC editor is active and at least one element is selected. The “Explicit Execution Order Mode” property is selected.

The selected elements get a numbering decreased by one while keeping the previous order. The selected elements are processed one position earlier. The remaining elements are numbered so that their execution order remains the same. The topological positions of the elements are retained anyway.

See also

- Chapter 1.3.1.9.3.2.1 “Automatic execution order by data flow” on page 189
- Chapter 1.3.1.25.3.10.12 “Dialog ‘Properties’ - ‘CFC Execution Order’” on page 1054
- Chapter 1.3.1.25.2.12.11 “Command ‘Send to Front’” on page 981
- Chapter 1.3.1.25.2.12.12 “Command ‘Send to Back’” on page 981
- Chapter 1.3.1.25.2.12.14 “Command ‘Move Down’” on page 982
- Chapter 1.3.1.25.2.12.16 “Command ‘Order by Data Flow’” on page 983
- Chapter 1.3.1.25.2.12.17 “Command ‘Order by Topology’” on page 983

Command 'Move Down'

Symbol: 

**Function:** The command numbers the elements so that the selected elements are located one position backward.

**Call:** Menu bar: “CFC ➔ Execution Order”; context menu: “Execution Order”

**Requirements:** A CFC editor is active and at least one element is selected. The “Explicit Execution Order Mode” property is selected.

The selected elements get a numbering increased by one while keeping the previous order. The elements are processed one position later. The remaining elements are numbered so that their execution order remains the same. The topological positions of the elements are retained anyway.

See also

- Chapter 1.3.1.9.3.2.1 “Automatic execution order by data flow” on page 189
- Chapter 1.3.1.25.3.10.12 “Dialog ‘Properties’ - ‘CFC Execution Order’” on page 1054
- Chapter 1.3.1.25.2.12.11 “Command ‘Send to Front’” on page 981
- Chapter 1.3.1.25.2.12.12 “Command ‘Send to Back’” on page 981
- Chapter 1.3.1.25.2.12.13 “Command ‘Move Up’” on page 982
- Chapter 1.3.1.25.2.12.16 “Command ‘Order by Data Flow’” on page 983
- Chapter 1.3.1.25.2.12.17 “Command ‘Order by Topology’” on page 983

Command 'Set Execution Order'

**Function:** The command opens a dialog for setting the number of the selected element to any value.
Call: Menu bar: “CFC ➔ Execution Order”; context menu: “Execution Order”

Requirements: A CFC editor is active and the “Explicit Execution Order Mode” property is selected. Exactly one element is selected.

The selected element gets the number specified in the dialog. The remaining elements are numbered so that their execution order remains the same. The topological positions of the elements are retained anyway.

See also
- § Chapter 1.3.1.9.3.2.1 “Automatic execution order by data flow” on page 189
- § Chapter 1.3.1.25.3.10.12 “Dialog 'Properties' - 'CFC Execution Order'” on page 1054
- § Chapter 1.3.1.25.2.12.11 “Command 'Send to Front'” on page 981
- § Chapter 1.3.1.25.2.12.12 “Command 'Send to Back'” on page 981
- § Chapter 1.3.1.25.2.12.13 “Command 'Move Up'” on page 982
- § Chapter 1.3.1.25.2.12.14 “Command 'Move Down'” on page 982
- § Chapter 1.3.1.25.2.12.16 “Command 'Order by Data Flow'” on page 983
- § Chapter 1.3.1.25.2.12.17 “Command 'Order by Topology'” on page 983

Command 'Order by Data Flow'

Function: The command numbers the elements in the program by data flow, or in the case of multiple networks by their topological position in the editor.

Call: Menu bar: “CFC ➔ Execution Order”; context menu: “Execution Order”

Requirements: A CFC editor is active and the “Explicit Execution Order Mode” property is selected.

The command is also available when no element is selected.

The execution order is determined by data flow. In the case of multiple networks, it is determined by their topological position of the networks. All numbered elements of the POU are set accordingly. Afterwards, the execution order is identical to that in auto data flow mode. The topological positions of the elements are retained anyway.

See also
- § Chapter 1.3.1.9.3.2.1 “Automatic execution order by data flow” on page 189
- § Chapter 1.3.1.25.3.10.12 “Dialog 'Properties' - 'CFC Execution Order'” on page 1054
- § Chapter 1.3.1.25.2.12.11 “Command 'Send to Front'” on page 981
- § Chapter 1.3.1.25.2.12.12 “Command 'Send to Back'” on page 981
- § Chapter 1.3.1.25.2.12.13 “Command 'Move Up'” on page 982
- § Chapter 1.3.1.25.2.12.14 “Command 'Move Down'” on page 982
- § Chapter 1.3.1.25.2.12.17 “Command 'Order by Topology'” on page 983

Command 'Order by Topology'

Function: The command orders the execution order of the elements by their topological position from right to left and from top to bottom.

Call: Menu bar: “CFC ➔ Execution Order”; context menu: “Execution Order”

Requirements: A CFC editor is active and the “Explicit Execution Order Mode” property is selected. At least one element is selected.
The command applies to all elements in the program, even if not all elements are selected when the command is executed. The topological positions of the elements are retained anyway.

See also
● Chapter 1.3.1.9.3.2.1 “Automatic execution order by data flow” on page 189
● Chapter 1.3.1.25.3.10.12 “Dialog ‘Properties’ - ‘CFC Execution Order’” on page 1054
● Chapter 1.3.1.25.2.12.11 “Command ‘Send to Front’” on page 981
● Chapter 1.3.1.25.2.12.12 “Command ‘Send to Back’” on page 981
● Chapter 1.3.1.25.2.12.13 “Command ‘Move Up’” on page 982
● Chapter 1.3.1.25.2.12.14 “Command ‘Move Down’” on page 982
● Chapter 1.3.1.25.2.12.16 “Command ‘Order by Data Flow’” on page 983

Command 'Edit Parameters'

Function: This command opens the “Edit Parameters” dialog box, where you change the constant input parameters of a function block.

Call: Main menu “CFC ➔ Edit Parameters”, or “Right-Click ➔ Edit Parameters”, click the “Parameter” function block.

Requirements: A CFC editor is active. An instantiated function block has VAR_INPUT CONSTANT variables in its declaration.

This functionality applies only to blocks that are inserted in a CFC with CODESYS >= V3.5 SP4.

CODESYS displays blocks with VAR_INPUT CONSTANT variables by the word “Parameter” in the lower left corner of the block.

Dialog Box 'Edit Parameters'

<table>
<thead>
<tr>
<th>“Parameters”</th>
<th>Name of the variable</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Type”</td>
<td>Data type of the variables</td>
</tr>
<tr>
<td>“Value”</td>
<td>Click into the field to type a value.</td>
</tr>
<tr>
<td>“Initial Value”</td>
<td>Initialization Value</td>
</tr>
<tr>
<td>“Category”</td>
<td>Additional information about the parameters; these values are defined by attributes and cannot be changed in this dialog box.</td>
</tr>
<tr>
<td>“Unit”</td>
<td>● parameterCategory</td>
</tr>
<tr>
<td>“Min”</td>
<td>● parameterUnit</td>
</tr>
<tr>
<td>“Max”</td>
<td>● parameterMinValue</td>
</tr>
<tr>
<td></td>
<td>● parameterMaxValue</td>
</tr>
</tbody>
</table>

“Delete Prepared Parameters” This command is active when you write a prepared value (“Debug ➔ Write Value”).

When you exit the field and the dialog box by clicking “OK”, the value changes are applied to the project.
FUNCTION_BLOCK FB1
VAR_INPUT CONSTANT
  {attribute 'parameterCategory':='General'}
  {attribute 'parameterUnit':= 'm/s'}
  {attribute 'parameterMinValue':= '0'}
  {attribute 'parameterMaxValue':= '100'}
  fbin1:INT;
  fbin2:DWORD:=24354333;
  fbin3:STRING:='abc';
END_VAR

This functionality and the declaration of variables with keyword VAR_INPUT CONSTANT applies only to the CFC editor. In the FBD editor, CODESYS always shows all input parameters on the block, regardless of whether or not they are declared as VAR_INPUT or VAR_INPUT CONSTANT. CODESYS also does not make a distinction about this in text editors.

See also
- § Chapter 1.3.1.24.1.6.3 "CFC editor in online mode" on page 480
- § Chapter 1.3.1.25.2.12.19 "Command 'Save Prepared Parameters to Project'" on page 985

Command 'Save Prepared Parameters to Project'

**Function:** This command saves the prepared parameter values to the project.

**Call:** Main menu “CFC”.

**Requirements:** A CFC editor is active. Parameter values of function block instances are changed in online mode. You are in offline mode.

If the values of constants on the controller are different from the values in the application, then this is indicated by a red asterisk next to the parameter field. Clicking "Incur Prepared Parameters" saves the controller values to the application.

See also
- § "Changing of constant input parameters of function block instances" on page 482
- § Chapter 1.3.1.25.2.12.18 "Command 'Edit Parameters'" on page 984

Command 'Connect Selected Pins'

**Symbol:** 🔄

**Function:** The command establishes a connection between the selected pins.

**Call:** Main menu “CFC”, context menu

**Requirements:** A CFC editor is active. Precisely one output and several inputs are selected.

In order to select the pins you must keep the [CTRL] key pressed while clicking on the pins. Then you execute the command.

See also
- § Chapter 1.3.1.25.2.12.23 "Command 'Select Connected Pins'" on page 986
Command 'Unlock Connection'

Symbol: [Image]

**Function:** This command unlocks a disabled connection.

**Call:** Main menu “CFC ➔ Routing”, context menu “Routing”

**Requirements:** A CFC editor is active. A connection or a connection mark is selected.

You obtain a disabled connection if you change the connections of the automatic routing. If you wish to carry out automatic routing again, you must first unlock a disabled connection.

With a mouse-click on the [Image] icon of a disabled connection you can similarly unlock this connection.

See also
- Chapter 1.3.1.9.3.2.2 “Programming in the CFC editor” on page 193
- Chapter 1.3.1.24.1.6.4.12 “CFC element ‘Connection Mark - Source/Sink’” on page 488

Command 'Show Next Collision'

**Function:** This command displays the next collision in the editor and marks the place concerned.

**Call:** Menu menu “CFC ➔ Routing”, context menu “Routing”

**Requirements:** A CFC editor is active and at least one connection with a collision is present.

This function is very useful if you operate with large networks and see only one sub-area. A collision is additionally indicated to you by the red bordered symbol in the top right corner of the editor.

Command 'Select Connected Pins'

Symbol: [Image]; shortcut: [Ctrl]+[Left Arrow], or [Ctrl]+[Right Arrow]

**Function:** The command selects all pins that are connected to the currently selected line, or connected to the currently selected connection mark in page-oriented CFC.

**Call:** “CFC” menu; context menu

**Requirements:** A CFC editor or a page-oriented CFC editor is active. One line and therefore exactly one connection or exactly one connection mark is selected.

See also
- Chapter 1.3.1.24.1.6.1 “CFC editor” on page 477
- Chapter 1.3.1.24.1.6.2 “CFC editor, page-oriented” on page 479
- Chapter 1.3.1.24.1.6.4.12 “CFC element ‘Connection Mark - Source/Sink’” on page 488

Command 'Reset Pins'

Symbol: [Image]

**Function:** The command restores deleted pins of a function block.

**Call:** Main menu “CFC ➔ Pins”, context menu “Pins”

**Requirements:** A CFC editor is active and a function block is selected.
The command restores all inputs and outputs of the function block just as they are defined in its implementation.

See also
- Chapter 1.3.1.25.2.12.25 “Command ’Remove Unused Pins’” on page 987

**Command ’Remove Unused Pins’**

**Symbol:** 

**Function:** The command removes all unused pins of the selected element.

**Call:** Menu “CFC ➔ Pins”, context menu “Pins”

**Requirements:** A CFC editor is active. An element is selected.

See also
- Chapter 1.3.1.25.2.12.24 “Command ’Reset Pins’” on page 986

**Command ’Add Input Pin’**

**Symbol:** 

**Function:** The command adds a further input to the selected function block.

**Call:** Main menu “CFC ➔ Pins”, context menu “Pins”

**Requirements:** A CFC editor is active. A function block is selected.

See also
- Chapter 1.3.1.25.2.12.27 “Command ’Add Output Pin’” on page 987

**Command ’Add Output Pin’**

**Symbol:** 

**Function:** The command adds a further output to the selected function block.

**Call:** Main menu “CFC ➔ Pins”, context menu “Pins”

**Requirements:** A CFC editor is active. A suitable function block is selected.

See also
- Chapter 1.3.1.25.2.12.26 “Command ’Add Input Pin’” on page 987

**Command ’Route All Connections’**

**Symbol:** 

**Function:** This command cancels all manual changes to the connections in the program and re-establishes the original state.

**Call:** Main menu “CFC ➔ Routing”, context menu “Routing”

**Requirements:** A CFC editor is active.

CODESYS cannot automatically route connections that are fixed by control points. You must remove the control points before executing the command. Use the “Remove Control Point” command to do this. Furthermore you must disconnect connections that have been changed manually and are marked by the icon. Use the “Disconnect Connection” command to do this.
See also

- Chapter 1.3.1.25.2.12.29 “Command 'Remove Control Point'' on page 988
- Chapter 1.3.1.25.2.12.21 “Command 'Unlock Connection'' on page 986

Command 'Remove Control Point'

**Function:** This command removes a control point.

**Call:** Context menu “Routing”

**Requirements:** A CFC editor is active. You have selected a connecting line.

If you move the mouse pointer over a selected connecting line, the existing control points are displayed with yellow circle symbols. Set the cursor on the control point to be deleted and execute the command from the context menu.

See also

- Chapter 1.3.1.24.1.6.4.2 “CFC element 'Control Point'” on page 486
- Chapter 1.3.1.25.2.12.30 “Command 'Create Control Point'” on page 988

Command 'Create Control Point'

**Symbol:** 🔽

**Function:** The command creates a control point on a connecting line.

**Call:** Context menu “Routing”

**Requirements:** A CFC editor is active. The cursor is over a connection.

The control point is created in the position on the connection at which the cursor is located when calling the command. The command corresponds to the “Control Point” element in the “Tools” window.

See also

- Chapter 1.3.1.24.1.6.4.2 “CFC element 'Control Point'” on page 486
- Chapter 1.3.1.25.2.12.29 “Command 'Remove Control Point'” on page 988

Command 'Connection Mark'

**Symbol:** ☻

**Function:** This command switches the display of the connection between two elements back and forth between a connecting line and the use of connection marks.

**Call:** Main menu “CFC”, context menu

**Requirements:** A CFC editor is active. A connection or a connection mark is selected.

If you have selected a connecting line, the command removes this line and adds a “Connection Mark - Source” at the output of one element and a “Connection Mark - Sink” at the input of the other. Both are given the same name by default, “C-><n>”, where n is a sequential number.

If you select a pair of connection marks, the command converts these marks into a connecting line.

See also

- Chapter 1.3.1.24.1.6.4.12 “CFC element 'Connection Mark - Source/Sink'” on page 488
Command 'Create group'

Symbol: ☑️

**Function:** This command groups the selected elements.

**Call:** Main menu “CFC ➔ Group”, context menu “Group”

**Requirements:** A CFC editor is active. Several elements are selected.

Grouped elements can only be moved together. The position of the elements is not affected by the grouping.

See also

- Chapter 1.3.1.25.2.12.33 “Command 'Ungroup’” on page 989

Command 'Ungroup'

Symbol: ✗

**Function:** The command undoes a previous grouping.

**Call:** Main menu “CFC ➔ Group”, context menu “Group”

**Requirements:** A CFC editor is active. A grouping is selected.

See also

- Chapter 1.3.1.25.2.12.32 “Command 'Create group’” on page 989

Command 'Prepare Function Block Element for Forcing'

**Function:** This command activates and deactivates the forceability of the inputs for a function block element.

**Call:**

- CFC
- Context menu

**Requirements:** The CFC editor is in offline mode and a function block element is selected.

After executing the command, the command “Force FB Input” is available in online mode to open a dialog for forcing the POU input values.

See also

- Chapter 1.3.1.25.2.12.35 “Command 'Force FB Input’” on page 989
- Chapter 1.3.1.24.1.6.1 “CFC editor” on page 477
- Chapter 1.3.1.12.4 “Forcing and writing variables” on page 315

Command 'Force FB Input'

**NOTICE!**

This kind of forcing uses a data breakpoint internally and is therefore different from forcing with the “Force Values” command or [F7].

Values that were forced by the command “Force FB Input” do not respond to the commands “Show All Forces” or “Unforce Values”.

**Function:** This command opens the “Force Value” dialog to force the selected input of a function block. Forcing can be canceled with the same command and dialog.
Call:

- CFC
- Context menu

Requirements: The "forcability" of the function block is activated (by the command "Prepare Box for Forcing"). The CFC editor is in online mode and the input of the function block is selected in the CFC editor.

In the "Force Value" dialog, you can either specify a value that the input of the function block should be forced, or remove the currently forced value.

After forcing, the input is highlighted in green again. Boolean inputs get a small monitoring view with the forced value. The forced value is displayed in the "Value" column of monitoring views (in the declaration part of the POU or in a watch list).

Dialog 'Force Value'

| "Expression" | Name of the function block input. Example: TON_1.IN. |
| "Type"       | Data type of the input                               |

Table 166: "What do you want to do?"

| "Set a new value to force" | ⚠️ You can specify a new value in the input field. The format must correspond to the data type. |
| "Remove value"            | ⚠️ Forcing at the input is canceled.                                                             |

See also

- Chapter 1.3.1.24.1.6.1 “CFC editor” on page 477
- Chapter 1.3.1.12.4 “Forcing and writing variables” on page 315
- Chapter 1.3.1.25.2.12.34 “Command 'Prepare Function Block Element for Forcing'” on page 989

Command 'Use Attributed Member as Input'

Symbol: →

Function: This command allows for connecting a structure member to a scalar type input.

Call: Menu bar: "CFC ➔ Pins"; context menu: "Pins"

Requirements: A CFC editor is active and a function block input is selected.

The member of the structure that is connected to the input of the subsequent function block must be provided with the pragma \{attribute 'ProcessValue\}. The data type of the structure member has to be compatible with the data type of the subsequent input. Inputs connected in this way are flagged with the "V" symbol.
Example

```plaintext
TYPE QINT :
STRUCT
    Status : STRING;
    {attribute 'ProcessValue'}
    Value1 : INT;
    Value2 : INT;
END_STRUCT
END_TYPE

PROGRAM PLC_PRG
VAR
    input1: QINT;
    output1: QINT;
    intValue: INT;
END_VAR

If you do not execute the command “Use attributed member as input” for this link, then a compiler error is issued.

See also
- Chapter 1.3.1.24.6.2.40 “Attribute ‘ProcessValue’” on page 637
```
## Menu 'FBD/LD/IL'

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.3.1.25.2.13.1</td>
<td>Command 'Insert Network'</td>
</tr>
<tr>
<td>1.3.1.25.2.13.2</td>
<td>Command 'Insert Network (Below)'</td>
</tr>
<tr>
<td>1.3.1.25.2.13.3</td>
<td>Command 'Toggle Network Comment State'</td>
</tr>
<tr>
<td>1.3.1.25.2.13.4</td>
<td>Command 'Insert Assignment'</td>
</tr>
<tr>
<td>1.3.1.25.2.13.5</td>
<td>Command 'Insert Box'</td>
</tr>
<tr>
<td>1.3.1.25.2.13.6</td>
<td>Command 'Insert Box with EN/ENO'</td>
</tr>
<tr>
<td>1.3.1.25.2.13.7</td>
<td>Command 'Insert Empty Box'</td>
</tr>
<tr>
<td>1.3.1.25.2.13.8</td>
<td>Command 'Insert Empty Box with EN/ENO'</td>
</tr>
<tr>
<td>1.3.1.25.2.13.9</td>
<td>Command 'Insert Box Parallel (Below)'</td>
</tr>
<tr>
<td>1.3.1.25.2.13.10</td>
<td>Command 'Insert Jump'</td>
</tr>
<tr>
<td>1.3.1.25.2.13.11</td>
<td>Command 'Insert Label'</td>
</tr>
<tr>
<td>1.3.1.25.2.13.12</td>
<td>Command 'Insert Return'</td>
</tr>
<tr>
<td>1.3.1.25.2.13.13</td>
<td>Command 'Insert Input'</td>
</tr>
<tr>
<td>1.3.1.25.2.13.14</td>
<td>Command 'Insert Coil'</td>
</tr>
<tr>
<td>1.3.1.25.2.13.15</td>
<td>Command 'Insert Set Coil'</td>
</tr>
<tr>
<td>1.3.1.25.2.13.16</td>
<td>Command 'Insert Reset Coil'</td>
</tr>
<tr>
<td>1.3.1.25.2.13.17</td>
<td>Command 'Insert Contact'</td>
</tr>
<tr>
<td>1.3.1.25.2.13.18</td>
<td>Command 'Insert Contact (Right)'</td>
</tr>
<tr>
<td>1.3.1.25.2.13.19</td>
<td>Command 'Insert Contact in Parallel (Below)'</td>
</tr>
<tr>
<td>1.3.1.25.2.13.20</td>
<td>Command 'Insert Contact in Parallel (Above)'</td>
</tr>
<tr>
<td>1.3.1.25.2.13.21</td>
<td>Command 'Toggle Parallel Mode'</td>
</tr>
<tr>
<td>1.3.1.25.2.13.22</td>
<td>Command 'Insert Negated Contact'</td>
</tr>
<tr>
<td>1.3.1.25.2.13.23</td>
<td>Command 'Insert Negated Contact Parallel (Below)'</td>
</tr>
<tr>
<td>1.3.1.25.2.13.24</td>
<td>Command 'Paste Contacts: Paste Below'</td>
</tr>
<tr>
<td>1.3.1.25.2.13.25</td>
<td>Command 'Paste Contacts: Paste Above'</td>
</tr>
<tr>
<td>1.3.1.25.2.13.26</td>
<td>Command 'Paste Contacts: Paste Right (After)'</td>
</tr>
<tr>
<td>1.3.1.25.2.13.27</td>
<td>Command 'Insert IL Line Below'</td>
</tr>
<tr>
<td>1.3.1.25.2.13.28</td>
<td>Command 'Delete IL Line'</td>
</tr>
<tr>
<td>1.3.1.25.2.13.29</td>
<td>Command 'Negation'</td>
</tr>
<tr>
<td>1.3.1.25.2.13.30</td>
<td>Command 'Edge Detection'</td>
</tr>
<tr>
<td>1.3.1.25.2.13.31</td>
<td>Command 'Set/Reset'</td>
</tr>
<tr>
<td>1.3.1.25.2.13.32</td>
<td>Command 'Set Output Connection'</td>
</tr>
<tr>
<td>1.3.1.25.2.13.33</td>
<td>Command 'Insert Branch'</td>
</tr>
<tr>
<td>1.3.1.25.2.13.34</td>
<td>Command 'Insert Branch Above'</td>
</tr>
<tr>
<td>1.3.1.25.2.13.35</td>
<td>Command 'Insert Branch Below'</td>
</tr>
<tr>
<td>1.3.1.25.2.13.36</td>
<td>Command 'Set Branch Start Point'</td>
</tr>
<tr>
<td>1.3.1.25.2.13.37</td>
<td>Command 'Set Branch End Point'</td>
</tr>
<tr>
<td>1.3.1.25.2.13.38</td>
<td>Command 'Update Parameters'</td>
</tr>
<tr>
<td>1.3.1.25.2.13.39</td>
<td>Command 'Remove Unused FB Call Parameters'</td>
</tr>
<tr>
<td>1.3.1.25.2.13.40</td>
<td>Command 'Repair POU'</td>
</tr>
<tr>
<td>1.3.1.25.2.13.41</td>
<td>Command 'View as Function Block Diagram'</td>
</tr>
<tr>
<td>1.3.1.25.2.13.42</td>
<td>Command 'View as Ladder Logic'</td>
</tr>
<tr>
<td>1.3.1.25.2.13.43</td>
<td>Command 'View as Instruction List'</td>
</tr>
<tr>
<td>1.3.1.25.2.13.44</td>
<td>Command 'Go to'</td>
</tr>
</tbody>
</table>

---

### Command 'Insert Network'

**Symbol:** ![Insert Network](image)

**Shortcut:** `[Ctrl] + [I]`

**Function:** This command inserts a further network in the FBD/LD/IL editor.

**Call:** Main menu "FBD, LD, IL", context menu
**Requirements**: The FBD, LD or IL editor is active. No box is selected.

See also

- % Chapter 1.3.1.24.1.5.4.1 “FBD/LD/IL element 'Network'” on page 470
- % Chapter 1.3.1.24.1.5.1 “FBD/LD/IL editor” on page 461
- % Chapter 1.3.1.25.2.13.2 “Command 'Insert Network (Below)’” on page 993

---

**Command 'Insert Network (Below)'**

Symbol: [ ], shortcut: [Ctrl]+ [T]

**Function**: This command inserts a further network in the FBD/LD/IL editor below the selected network.

**Call**: Main menu “FBD, LD, IL”, context menu

**Requirements**: The FBD, LD or IL editor is active. A network is selected. No box is selected.

See also

- % Chapter 1.3.1.24.1.5.1 “FBD/LD/IL editor” on page 461

---

**Command 'Toggle Network Comment State'**

Symbol: [··· ], shortcut: [Ctrl]+ [O]

**Function**: The command comments the selected network in or out.

**Call**: Main menu “FBD, LD, IL”, context menu

**Requirements**: The FBD, LD or IL editor is active. A network is selected, but no box is selected.

See also

- % Chapter 1.3.1.24.1.5.1 “FBD/LD/IL editor” on page 461

---

**Command 'Insert Assignment'**

Symbol: [··· ], shortcut: [Ctrl]+ [A]

**Function**: This command inserts an assignment in the FBD or LD editor.

**Call**: Main menu “FBD/LD/IL”, context menu

**Requirements**: The FBD, LD or IL editor is active. A network is selected, but no box is selected.

**In IL an assignment is programmed via the operators LD and ST.**

See also

- % Chapter 1.3.1.24.1.5.1 “FBD/LD/IL editor” on page 461

---

**Command 'Insert Box'**

Symbol: [··· ], shortcut: [Ctrl]+ [B]

**Function**: This command inserts a box that is available in the project at the end of the selected network.

**Call**: Main menu “FBD, LD, IL”, context menu
**Requirements:** The FBD, LD or IL editor is active. A network is selected, but no box is selected.

If you select this command the input assistant opens, where you can select the desired box.

See also:
- % Chapter 1.3.1.24.1.5.4.2 “FBD/LD/IL element 'Box'” on page 471
- % Chapter 1.3.1.24.1.5.1 “FBD/LD/IL editor” on page 461
- % Chapter 1.3.1.9.6 “Using input assistance” on page 208

**Command 'Insert Box with EN/ENO'**

Symbol: ☉; shortcut: [Ctrl] + [Shift] + [E]

**Function:** This command inserts a box with a boolean input “Enable” and a boolean output “Enable Out” at the end of the selected network.

**Call:** Main menu “FBD, LD, IL”, context menu

**Requirements:** The FBD, LD or IL editor is active. A network is selected, but no box is selected.

See also:
- % Chapter 1.3.1.24.1.5.1 “FBD/LD/IL editor” on page 461
- % Chapter 1.3.1.9.6 “Using input assistance” on page 208

**Command 'Insert Empty Box'**

Symbol: ☐; shortcut: [Ctrl] + [Shift] + [B]

**Function:** This command inserts an empty function block at the end of the currently selected network.

**Call:** Main menu “FBD/LD/IL”, context menu

**Requirements:** The FBD, LD or IL editor is active. A network is selected, but no box is selected.

See also:
- % Chapter 1.3.1.24.1.5.1 “FBD/LD/IL editor” on page 461

**Command 'Insert Empty Box with EN/ENO'**

Symbol: ☒

**Function:** The command inserts an empty box with a Boolean input “Enable” and a Boolean output “Enable Out” at the end of the selected network.

**Call:** Main menu “FBD/LD/IL”, context menu

**Requirements:** The FBD editor, the IL editor or the LD editor is active. A network must be selected. No other box may be selected.

If “Enable” has the value FALSE at the time of the function block call, then the operations defined in the fb are not executed. Otherwise, if “Enable” has the value TRUE, these operations are executed. The ENO output acts as a repeater of the EN input.

See also:
- % Chapter 1.3.1.24.1.5.1 “FBD/LD/IL editor” on page 461

**Command 'Insert Box Parallel (Below)'**

**Function:** This command inserts an empty box parallel below the selected function block.

**Call:** Menu bar: “FBD/LD/IL”; context menu.
**Requirements:** A box is selected in the LD editor.
See also

- Chapter 1.3.1.24.1.5.1 “FBD/LD/IL editor” on page 461

### Command ‘Insert Jump’

**Symbol:** ➔
**Shortcut:** [Ctrl]+[L]

**Function:** This command inserts a jump element before the selected element.

**Call:** Main menu “FBD/LD/IL”, context menu

**Requirements:** The FBD, LD or IL editor is active. A connecting line is selected.

See also

- Chapter 1.3.1.24.1.5.1 “FBD/LD/IL editor” on page 461
- Chapter 1.3.1.24.1.5.4.7 “FBD/LD/IL element ’Jump’” on page 472

### Command ‘Insert Label’

**Symbol:** ⇌

**Function:** This command inserts a jump label into the currently selected network.

**Call:** Main menu “FBD, LD, IL”, context menu

**Requirements:** The FBD, LD or IL editor is active. A network is selected. No jump label is selected.

See also

- Chapter 1.3.1.24.1.5.1 “FBD/LD/IL editor” on page 461
- Chapter 1.3.1.24.1.5.4.6 “FBD/LD/IL element ’Label’” on page 472

### Command ‘Insert Return’

**Symbol:** ↓

**Function:** This command inserts an element “Return” in the selected place.

**Call:** Main menu “FBD/LD/IL”, context menu

**Requirements:** The FBD, LD or IL editor is active. A box output is selected.

If a box is selected, the command “Append Input” is available in the context menu. The input is inserted at the lower end of the box.

See also

- Chapter 1.3.1.24.1.5.1 “FBD/LD/IL editor” on page 461
- Chapter 1.3.1.24.1.5.4.8 “FBD/LD/IL element ’Return’” on page 472

### Command ‘Insert Input’

**Symbol:** +, shortcut: [Ctrl]+[Q]

**Function:** This command adds a further input to an extendable box (ADD, OR, ADD, MUL, SEL) above the selected input.

**Call:** “FBD/LD/IL” menu

**Requirements:** The FBD or LD editor is active. An input of a box is selected.

If a box is selected, the command “Append Input” is available in the context menu. The input is inserted at the lower end of the box.

See also

- Chapter 1.3.1.24.1.5.1 “FBD/LD/IL editor” on page 461
Command 'Insert Coil'

Symbol: ![coil] shortcut: [Ctrl] + [A]

**Function:** This command inserts a coil into the network.

**Call:** Main menu “FBD, LD, IL”, context menu

**Requirements:** The LD editor is active. A network, a coil or a connecting line is selected, but no box is selected.

See also

- Chapter 1.3.1.24.1.5.1 “FBD/LD/IL editor” on page 461

Command 'Insert Set Coil'

Symbol: ![set coil]

**Function:** This command inserts a set coil into the network.

**Call:** Main menu “FBD, LD, IL”, context menu

**Requirements:** The LD editor is active. A network, a coil or a line is selected, but no box is selected.

See also

- Chapter 1.3.1.24.1.5.1 “FBD/LD/IL editor” on page 461

Command 'Insert Reset Coil'

Symbol: ![reset coil]

**Function:** This command inserts a reset coil into the network.

**Call:** Main menu “FBD, LD, IL”, context menu

**Requirements:** The LD editor is active. A network, a coil or a line is selected, but no box is selected.

See also

- Chapter 1.3.1.24.1.5.1 “FBD/LD/IL editor” on page 461
- Chapter 1.3.1.24.1.5.4.12 “LD element ‘Coil’” on page 474
- “Ladder diagram (LD)” on page 183

Command 'Insert Contact'

Symbol: ![contact], shortcut: [Ctrl] + [K]

**Function:** This command inserts a contact to the left of the selected element.

**Call:** Main menu “FBD/LD/IL”, context menu

**Requirements:** The LD editor is active. A line or a contact is selected.

See also

- Chapter 1.3.1.24.1.5.1 “FBD/LD/IL editor” on page 461
- Chapter 1.3.1.24.1.5.4.11 “LD element ‘Contact’” on page 473
Command 'Insert Contact (Right)"

Symbol: 🗓️, shortcut: [Ctrl] + [D]

Function: This command inserts a contact to the right of the selected element.

Call: Main menu “FBD/LD/IL”, context menu

Requirements: The LD editor is active. A line, a contact or a box is selected.

See also
● Chapter 1.3.1.24.1.5.1 “FBD/LD/IL editor” on page 461
● Chapter 1.3.1.24.1.5.4.11 “LD element ‘Contact’” on page 473

Command 'Insert Contact in Parallel (Below)"

Symbol: 🕯️; keyboard shortcut: [Ctrl]+[R]

Function: This command inserts a contact with lines in parallel with and below the selected element.

Call: Menu bar: “FBD/LD/IL”; context menu.

Requirements: The LD editor is active. A line or a contact or a box is selected.

You can program closed parallel branches in a LD network as short circuit evaluation (SCE) or OR constructs. SCE branches are displayed with double vertical lines, and OR branches with single lines. Refer to the help page for "Closed branches".

See also
● Chapter 1.3.1.24.1.5.1 “FBD/LD/IL editor” on page 461
● Chapter 1.3.1.25.2.13.20 “Command 'Insert Contact in Parallel (Above)’” on page 997
● Chapter 1.3.1.24.1.5.4.14 “Closed branch” on page 475

Command 'Insert Contact in Parallel (Above)"

Symbol: 🕯️; keyboard shortcut: [Ctrl]+[P]

Function: This command inserts a contact with lines in parallel with and above the selected element.

Call: Menu bar: “FBD/LD/IL”; context menu.

Requirements: The LD editor is active. A line, a contact or a box is selected.

You can program closed parallel branches in a LD network as short circuit evaluation (SCE) or OR constructs. SCE branches are displayed with double vertical lines, and OR branches with single lines. Refer to the help page for "Closed branches".
Command 'Toggle Parallel Mode'

**Function:** This command toggles a parallel branch between an OR construct and the Short Circuit Evaluation (SCE).

**Call:** Menu bar: “FBD/LD/IL”; context menu.

**Requirements:** The LD editor is active. A vertical line of a parallel branch is selected.

You can program closed parallel branches in a LD network as short circuit evaluation (SCE) or OR constructs. SCE branches are displayed with double vertical lines, and OR branches with single lines. Refer to the help page for “Closed branches”.

Command 'Insert Negated Contact'

**Symbol:** ⼴, shortcut: \[Ctrl\] + \[K\]

**Function:** This command inserts a negated contact to the left of the selected element.

**Call:** Main menu “FBD/LD/IL”, context menu

**Requirements:** The LD editor is active. A line or a contact is selected.

See also
- Chapter 1.3.1.24.1.5.1 “FBD/LD/IL editor” on page 461
- Chapter 1.3.1.24.1.5.4.11 “LD element 'Contact’” on page 473

Command 'Insert Negated Contact Parallel (Below)'

**Symbol:** ⼴

**Function:** The command inserts a negated contact with lines in parallel with and below the selected element.

**Call:** Main menu “FBD/LD/IL”, context menu

**Requirements:** The LD editor is active. A line, a contact or a box is selected.

See also
- Chapter 1.3.1.24.1.5.1 “FBD/LD/IL editor” on page 461
- Chapter 1.3.1.9.3.1.2 “Programming ladder diagrams (LD)” on page 186
Command 'Paste Contacts: Paste Below'

Shortcut: [Ctrl] + [F]

Function: This command inserts a previously copied contact with lines below the selected element.

Call: Main menu “FBD/LD/IL ➔ Paste”, context menu

Requirements: the LD editor is active.

See also
- Chapter 1.3.1.24.1.5.1 “FBD/LD/IL editor” on page 461
- Chapter 1.3.1.24.1.5.4.11 “LD element ‘Contact’” on page 473

Command 'Paste Contacts: Paste Above'

Shortcut: [Ctrl] + [F]

Function: This command inserts a previously copied contact with lines above the selected element.

Call: Main menu “FBD/LD/IL ➔ Paste Contacts”, context menu

Requirements: the LD editor is active. A line or a contact is selected.

See also
- Chapter 1.3.1.24.1.5.1 “FBD/LD/IL editor” on page 461
- Chapter 1.3.1.24.1.5.4.11 “LD element ‘Contact’” on page 473

Command 'Paste Contacts: Paste Right (After)'

Shortcut: [Ctrl] + [G]

Function: this command inserts a previously copied contact to the right of the selected element.

Call: Main menu “FBD/LD/IL ➔ Paste Contacts”, context menu

Requirements: The LD editor is active. A line or a contact is selected.

See also
- Chapter 1.3.1.24.1.5.1 “FBD/LD/IL editor” on page 461
- Chapter 1.3.1.24.1.5.4.11 “LD element ‘Contact’” on page 473

Command 'Insert IL Line Below'

Symbol: 

Function: The command inserts an instruction line below the selected line.

Call: Main menu “FBD/LD/IL”, context menu

Requirements: The IL editor is active. A line is selected.

See also
- Chapter 1.3.1.24.1.5.1 “FBD/LD/IL editor” on page 461

Command 'Delete IL Line'

Symbol: , shortcut: [Ctrl]+[Del]

Function: This command deletes the selected instruction line.
Call: Main menu “FBD/LD/IL”, context menu

Requirements: The IL editor is active. A line is selected.

See also
● % Chapter 1.3.1.24.1.5.1 “FBD/LD/IL editor” on page 461

Command 'Negation'

Symbol: \[\text{_negation}\], shortcut: [Ctrl] + [N]

Function: This command negates the following elements:

● Input/output of a box
● Jump
● Return
● Coil

Call: Main menu “FBD/LD/IL”, context menu

Requirements: The FBD or LD editor is active. The corresponding element is selected.

See also
● % Chapter 1.3.1.24.1.5.1 “FBD/LD/IL editor” on page 461

Command 'Edge Detection'

Symbol FBD: \[\text{edge}\], symbol LD: \[\text{edge}\], shortcut: [Ctrl] + [N]

Function: This command inserts an edge detector before the selected box input or box output.

Call: Main menu “FBD/LD/IL”, context menu

Requirements: The FBD or LD editor is active. A box input or box output is selected.

See also
● % Chapter 1.3.1.24.1.5.1 “FBD/LD/IL editor” on page 461

Command 'Set/Reset'

Symbol: \[\text{set/reset}\], shortcut [Ctrl] + [M]

Function: In the case of an element with a boolean output, this command switches between reset, set and no mark.

Call: Main menu “FBD/LD/IL”, context menu

Requirements: The FBD or LD editor is active. An element with a boolean output is selected.

See also
● % Chapter 1.3.1.24.1.5.1 “FBD/LD/IL editor” on page 461

Command 'Set Output Connection'

Symbol: \[\text{set output}\], shortcut [Ctrl]+ [W]

Function: This command turns the selected box output into the forwarding box output.

Call: Main menu “FBD/LD/IL”, context menu

Requirements: The FBD or LD editor is active. One of several box outputs is selected.
See also
- Chapter 1.3.1.24.1.5.1 “FBD/LD/IL editor” on page 461

Command 'Insert Branch'
Symbol: $\text{\texttrademark}\text{\textregistered}$, shortcut [Ctrl] + [Shift] + [V]

Function: This command creates an open line branch on the selected line.

Call: Main menu “FBD/LD/IL”, context menu

Requirements: The FBD or LD editor is active. An input or an output of a box is selected.

See also
- Chapter 1.3.1.24.1.5.4.9 “FBD/LD/IL element 'Branch'” on page 472
- Chapter 1.3.1.24.1.5.1 “FBD/LD/IL editor” on page 461

Command 'Insert Branch Above'
Symbol: $\text{\texttrademark}\text{\textregistered}$

Function: This command inserts a line branch above the selected open line branch.

Call: Main menu “FBD/LD/IL”, context menu

Requirements: The FBD or LD editor is active. An open line branch is selected.

See also
- Chapter 1.3.1.24.1.5.1 “FBD/LD/IL editor” on page 461
- Chapter 1.3.1.24.1.5.4.9 “FBD/LD/IL element 'Branch'” on page 472

Command 'Insert Branch Below'
Symbol: $\text{\texttrademark}\text{\textregistered}$

Function: This command inserts a line branch below the selected open line branch.

Call: Main menu “FBD/LD/IL”, context menu

Requirements: The FBD or LD editor is active. An open line branch is selected.

See also
- Chapter 1.3.1.24.1.5.1 “FBD/LD/IL editor” on page 461
- Chapter 1.3.1.24.1.5.4.9 “FBD/LD/IL element 'Branch'” on page 472

Command 'Set Branch Start Point'
Symbol: $\text{\texttrademark}\text{\textregistered}$

Function: This command sets the starting point of a line branch on the selected line.

Call: Main menu “FBD/LD/IL”, context menu

Requirements: The LD editor is active. A line is selected.

See also
- Chapter 1.3.1.24.1.5.4.14 “Closed branch” on page 475
- Chapter 1.3.1.24.1.5.1 “FBD/LD/IL editor” on page 461
Command 'Set Branch End Point'

Symbol:  

**Function:** This command sets the end point of a line branch on the selected line.

**Call:** Main menu “FBD/LD/IL”, context menu

**Requirements:** The LD editor is active. A line is selected. A starting point of the line branch has been set.

See also

- Chapter 1.3.1.24.1.5.1 “FBD/LD/IL editor” on page 461
- Chapter 1.3.1.24.1.5.4.14 “Closed branch” on page 475

Command 'Update Parameters'

**Function:** This command enters changes to the declaration of the selected element in the diagram.

**Call:** Main menu “FBD/LD/IL”, context menu

**Requirements:** The FBD, LD or CFC editor is active. A box is selected. An extending change has been made to the declaration.

The command checks whether a box and its declaration in the declaration editor correspond. The change is accepted for the box only if the declaration was extended. Deletions and overwrites are not updated.

See also

- Chapter 1.3.1.24.1.5.1 “FBD/LD/IL editor” on page 461

Command 'Remove Unused FB Call Parameters'

Symbol:  

**Function:** This command deletes inputs and outputs of the selected box to which no variable and no value were assigned. However, the default inputs and outputs are always retained.

**Call:** Main menu “FBD/LD/IL”, context menu

**Requirements:** The FBD or LD editor is active. A box is selected. The box has interfaces to which no value is assigned.

See also

- Chapter 1.3.1.24.1.5.1 “FBD/LD/IL editor” on page 461

Command 'Repair POU'

Symbol:  

**Function:** This command repairs internal inconsistencies in the selected box.

**Call:** Main menu “FBD/LD/IL”, context menu

**Requirements:** The FBD or LD editor is active. The defective box is selected. The editor has found internal inconsistencies in the programming module that can possibly be resolved automatically. CODESYS reports the inconsistencies in the Message window.

This situation is conceivable when editing a project that was created with an older programming system version that did not yet handle the inconsistency concerned as an error.

See also

- Chapter 1.3.1.24.1.5.1 “FBD/LD/IL editor” on page 461
Command 'View as Function Block Diagram'

**CAUTION!**

*Loss of data!* An error-free conversion requires syntactically correct code. Otherwise parts of the implementation can be lost.

Shortcut: `[Ctrl] + [1]`

**Function:** This command converts the active instruction list or the active ladder diagram into the function block diagram.

**Call:** Menu “FBD/LD/IL → View”

**Requirements:** The LD or IL editor is active.

See also
- Chapter 1.3.1.24.1.5.1 “FBD/LD/IL editor” on page 461

Command 'View as Ladder Logic'

**CAUTION!**

*Loss of data!* An error-free conversion requires syntactically correct code. Otherwise parts of the implementation can be lost.

Shortcut: `[Ctrl] + [2]`

**Function:** This command converts the current function block code or the active instruction list into a ladder diagram.

**Call:** Menu “FBD/LD/IL → View”

**Requirements:** The FBD or IL editor is active.

See also
- Chapter 1.3.1.24.1.5.1 “FBD/LD/IL editor” on page 461

Command 'View as Instruction List'

*If necessary, IL can be activated in the CODESYS options.*

**CAUTION!**

*Loss of data!* An error-free conversion requires syntactically correct code. Otherwise parts of the implementation can be lost.

Shortcut: `[Ctrl] + [3]`

**Function:** This command converts the active function block code or the active ladder diagram into an instruction list.

**Call:** Menu “FBD/LD/IL → View”

**Requirements:** The LD or FBD editor is active.
Command 'Go to'

Symbol:

**Function:** This command allows you to jump to any network.

**Call:** Main menu “FBD/LD/IL”

**Requirements:** The LD, FBD or IL editor is active. A network is selected.

This command opens a dialog box with an input field. Enter the number of the desired network in the input field.

See also
- Chapter 1.3.1.24.1.5.1 “FBD/LD/IL editor” on page 461

Menu 'Library'

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.3.1.25.2.14.1</td>
<td>Command 'Add Library'................................. 1004</td>
</tr>
<tr>
<td>1.3.1.25.2.14.2</td>
<td>Command 'Try to Reload Library'............................ 1005</td>
</tr>
<tr>
<td>1.3.1.25.2.14.3</td>
<td>Command 'Properties'........................................ 1006</td>
</tr>
<tr>
<td>1.3.1.25.2.14.4</td>
<td>Command 'Placeholders'..................................... 1008</td>
</tr>
<tr>
<td>1.3.1.25.2.14.5</td>
<td>Command 'Export Library'.................................... 1008</td>
</tr>
</tbody>
</table>

Command 'Add Library'

**Function:** The command opens the “Add Library” dialog. In this dialog, you can add libraries to the Library Manager and then integrate them in your application.

**Call:** Menu bar: “Libraries”

**Requirement:** The Library Manager is open in the editor.

Dialog 'Add Library'

<table>
<thead>
<tr>
<th>“Library”</th>
<th>Suitable libraries that are installed in the library repository. For example, the selection of libraries is defined in the device description or by the OEM. By default, the displayed libraries are grouped into categories.</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Company”</td>
<td>Creator of the library</td>
</tr>
<tr>
<td>“Advanced”</td>
<td>Opens the advanced “Add Library” dialog</td>
</tr>
</tbody>
</table>

Specific libraries can be blacklisted in a device description. These libraries cannot be added below this device in the Library Manager.
See also

- % Chapter 1.3.1.25.2.8.2 “Command ‘Library Repository’” on page 951

Dialog ‘Add Library’ – ‘Advanced’

You should use this dialog only if you have expert knowledge of library referencing. Using this dialog, you can link special versions or change placeholder definitions.

We recommend seriously that you follow the guidelines for the creation of libraries when developing and referencing libraries.

Table 167: Tab ‘Library’

<table>
<thead>
<tr>
<th>“Company”</th>
<th>Filtering the list according to vendor</th>
</tr>
</thead>
</table>
| “Group by category” | ☑: Display of the libraries in a tree structure grouped in categories.  
☐: Display of the libraries in alphabetical order in a flat structure. |
| “Display all versions” | ☑: Display of all versions of the libraries. Version specification ‘*’ means the latest version available in the repository.  
☐: Display of the latest versions of the libraries only. A multiple selection of libraries is possible in this display. To do this, hold down the [Shift] key and select the entries. |
| “Details” | Opens a detailed view with the library modules. |
| “Library Repository” | Opens the “Library Repository” dialog. There you can install more libraries to your local system. |

Table 168: Tab ‘Placeholder’

| “Placeholder name” | The input field provides a combo box for entering the valid placeholder names that are read from the currently accessible device descriptions. You can also enter a new placeholder name in order to define a free placeholder, which is not resolved by the device or by the library profile. |
| “Default library” | CODESYS uses this library when for any reason no device is available that the resolution defines. In this way it is possible to compile the current project without errors. |

Note about placeholder resolution

For compiler version V3.5.8.0 and later, the following statement applies in the case of library placeholders with a resolution in the device description that are located in the Library Manager of the POU pool. This placeholder is always resolved automatically according to the description of the device that compiles the application.

See also

- % Chapter 1.3.1.19.1 “Information for library developers” on page 372
- % Chapter 1.3.1.25.2.14.4 “Command ‘Placeholders’” on page 1008

Command ‘Try to Reload Library’

**Function:** This command tries to reload the selected library.

**Call:** Main menu “Library”.

**Requirement:** A library is selected that failed to load.
If for any reason a library is not available in the defined repository location when a project is opened, CODESYS displays a corresponding error message. Once you have rectified the error, i.e. when the library is properly available again, you can reload the library with this command without having to leave the project.

See also

- Chapter 1.3.1.25.2.8.2 “Command 'Library Repository'” on page 951

Command 'Properties'

**Function:** This command opens the “Properties” dialog box for the library selected in the library manager.

**Call**

- “Library” menu
- Context menu of the selected library
- symbol in the toolbar of the library manager

**Requirement:** A library is selected.

![](notice)

This dialog is intended for library developers. Please use this only if you have profound knowledge of library referencing. In addition, please follow the guidelines for library developers.

See also

- Guidelines for Developing Libraries

Dialog box 'Properties'

**Table 169: “General”**

<table>
<thead>
<tr>
<th>“Namespace”</th>
<th>Namespace of the selected library. By default, this is identical to the library name, unless it was defined explicitly in the project information when the library was created. You can change the namespace for the open project.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Example:</strong> LA</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>“Standard”</th>
<th>Library that triggers the placeholder when no other trigger is defined or is possible.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Requirement:</strong> The selected library is a library placeholder, and therefore the setting is available.</td>
<td></td>
</tr>
<tr>
<td><strong>Note:</strong> Regarding compiler version 3.5.8.0 and later: In the case of library placeholders that have a trigger in the device description and that located in the library manager of the “POUs” view, the placeholder is always triggered automatically according to the description of the device for which the application is compiled.</td>
<td></td>
</tr>
</tbody>
</table>

If the selected library is developed in compliance with the "Guidelines for Developing Libraries", then we do not recommend that you change the following settings.
Table 170: “Version”

<table>
<thead>
<tr>
<th>Selection of version constraint</th>
<th>Note: Container and interface libraries are created automatically with library references with version constraint. As far as possible, do not create libraries that include library references with version constraint. Otherwise, you reference the libraries by placeholders. Edit a placeholder resolution in the “Placeholders” dialog box.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>“Specific version”</strong></td>
<td>☑️: (selected from drop-down list) Version is integrated into the project. Note: This option is strongly recommended for container libraries, and it is usually preset for this library type.</td>
</tr>
<tr>
<td><strong>“Newest version always”</strong></td>
<td>☑️: The library repository is scanned and the latest detected version is integrated. Note: If a newer library version is available, then the library modules that are actually used can change. This option is strongly recommended for interface libraries, and it is usually preset for this library type.</td>
</tr>
</tbody>
</table>

Table 171: “Visibility”

| “Only allow qualified access to all identifiers.” | ☑️: Library modules (and variables) are called in the project only with prepended namespace paths. |
| “If the current project is referenced as a library by another project.” | Note: Changing the following settings makes sense only if you created a library with your project and therefore opened a library project. In this way, the selected library is referenced in the new library. |
| “Publish all IEC symbols to that project as if this reference would have been included there directly.” | ☑️: As a container library, the selected library makes the content of the referenced library visible at the top level (later in a project). Requirement: A container project is created with a library project. A container library does not implement its own modules, but references other libraries exclusively. It bundles libraries. A container library can be employed sensibly to bundle multiple libraries (in a reference) in a project. This option must be activated for each library reference. Symbolic access to library modules: <namespace of container library>.<module name> ☐️: The contents of the referenced library is accessed uniquely by means of the namespace. The path name consists of the library name and the unique name (library reference), and it is prepended to the module name. Requirement: No container project is created with a library project. Unique symbolic access to the library module: <namespace of library>.<namespace of sublibrary>.<module name>. |
| “Do not show this reference in the dependency tree.” | ☑️: The selected library is not displayed in the library manager as a library reference (later in a project). The library is a hidden reference. Warning: If there are compile errors resulting from hidden library errors, then detecting the errors may be difficult. ☐️: The selected library is displayed as a library reference (later in a project). |
| “Optional (if the library is missing, no error will be reported).” | ☑️: The selected library is treated as optional. When downloading the project that references the library, no error is reported, even if the library is not available in the library repository. |

See also

- Chapter 1.3.1.25.2.14.4 “Command ‘Placeholders’” on page 1008
Command 'Placeholders'

**Function:** This command opens the “Placeholders” dialog box. The dialog shows information on the currently selected placeholder library and allows to assign a project-specific resolution.

**Call:**
- Menu “Libraries”
- Symbol placeholders in the symbol bar in the upper part of the Library Manager window.

**Requirement:** A placeholder library is selected in the Library Manager.

A placeholder library, which is included in the “Devices” view, will be resolved as follows:
- If you have assigned a specific resolution to the placeholder library via the dialog “Placeholder”, this will be applied.
- If no specific resolution is defined, it will be checked, whether there is one specified in the device description of the application.
- Afterwards the library profile will be checked for a resolution definition.
- The result is displayed in the Library Manager below the “Effective Version”.

A placeholder library, which is included in the “POUs” view, gets resolved as follows:
- A specific resolution defined in the dialog “Placeholder” will be ignored.
- For the application it will be checked whether there is a resolution defined in the device description.
- Afterwards the library profile will be checked.
- The result is displayed in the tooltip of the symbol.

See also:
- Chapter 1.3.1.19 “Using libraries” on page 371
- Chapter 1.3.1.25.1.25 “Object 'Library Manager'” on page 771
- Chapter 1.3.1.19.1 “Information for library developers” on page 372

**Dialog box 'Placeholders'**

<table>
<thead>
<tr>
<th>“Name”</th>
<th>Name of the placeholder.</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Library”</td>
<td>Current resolution, valid for the project</td>
</tr>
<tr>
<td></td>
<td>Double-click on the entry in order to edit the placeholder resolution. A selection list with the available library versions appears. Additionally the command “Other Library” is available.</td>
</tr>
<tr>
<td>Command “Other Library”</td>
<td>The command opens the dialog box “Bibliothek durchsuchen” for searching and installing libraries. Choose this command, if you do not want to redirect to another version, but on a specific library.</td>
</tr>
<tr>
<td>“Info”</td>
<td>Type of placeholder resolution:</td>
</tr>
<tr>
<td></td>
<td>• Resolved by device description</td>
</tr>
<tr>
<td></td>
<td>• Resolved by library profile</td>
</tr>
<tr>
<td></td>
<td>• Resolved by &lt;specific library&gt;</td>
</tr>
</tbody>
</table>

Command 'Export Library'

**Function:** This command is used for saving the library file to the hard disk.

**Call:** Context menu of the Library Manager

**Requirement:** A library is selected in the Library Manager.

The command opens the standard dialog for saving a file in the local file system. The library file can have the file type Library files (*.library), Compiled library files (*.compiled-library), or Compiled library files (*.compiled-library-v3).
Menu 'Image Pool'

1.3.1.25.2.15.1 Command 'Insert Image'.......................................................... 1009

Menu 'Declarations'

1.3.1.25.2.16.1 Command 'Insert'................................................................. 1009
1.3.1.25.2.16.2 Command 'Edit Declaration Header'..................................... 1009
1.3.1.25.2.16.3 Command 'Move Down'..................................................... 1010
1.3.1.25.2.16.4 Command 'Move Up'......................................................... 1010

Command 'Insert Image'

Symbol: 

Function: This command inserts a new line into an image pool.

Call: Main menu "Imagepool", or right-click.

Requirements: An image pool is active and a line is selected in the image pool.

See also

● Chapter 1.3.1.25.1.24 “Object 'Image Pool'” on page 770

Command 'Insert'

Symbol: 

Function: This command inserts a new line for a variable declaration in the declaration editor and the input field for the variable name opens.

Call: Context menu in the tabular declaration editor; button in the declaration heading.

To edit the other fields of the declaration lines, double-click the fields and select the data from the drop-down lists or by means of the respective dialogs.

See also

● Chapter 1.3.1.9.2.1 “Using the declaration editor” on page 173

Command 'Edit Declaration Header'

Function: The command opens the dialog "Edit Declaration Header", which serves in the declaration editor for the configuration of a POU header.

Call: Context menu of the tabular declaration editor

Requirements: The tabular declaration editor is the active editor.

See also

● “Declaring in the tabular declaration editor” on page 174

Dialog 'Edit Declaration Header'

Function: The dialog is for configuring the declaration part of a POU.

Call: Click on the header bar of the tabular declaration editor, or context menu in the tabular declaration editor.
"Declaration"

Selection list for changing the POU type

- "PROGRAM"
- "FUNCTION_BLOCK"
  - "EXTENDS": Input field for a basic function block
  - "IMPLEMENTES": Input field for an interface
- "FUNCTION"
  - "Return type"

Input field with current POU name: you can change the name of the POU

"Automatically adapt all references on rename"

☑: Dialog box “Refactoring” opens.
☐: Renaming is only effective in the declaration header of the POU.

"Attributes"

The dialog box “Attribute” opens for the input of attributes and pragmas.

See also

- ☞ Chapter 1.3.1.9.2.1 “Using the declaration editor” on page 173
- ☞ Chapter 1.3.1.24.6 “Pragmas” on page 599
- ☞ Chapter 1.3.1.9.16 “Refactoring” on page 236

Command 'Move Down'

Symbol: 

**Function:** This command shifts a variable declaration downwards by one row.

**Call:** Context menu

**Requirement:** A row with a variable declaration is selected in the tabular declaration editor.

See also

- ☞ Chapter 1.3.1.9.2.1 “Using the declaration editor” on page 173

Command 'Move Up'

Symbol: 

**Function:** This command shifts a variable declaration upwards by one row.

**Call:** Context menu

**Requirement:** A row with a variable declaration is selected in the tabular declaration editor.

See also

- ☞ Chapter 1.3.1.9.2.1 “Using the declaration editor” on page 173

Menu 'Declarations' (Persistence)

<table>
<thead>
<tr>
<th>Command</th>
<th>Persistence</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.3.1.25.2.17.1</td>
<td>Command 'Reorder List and Clean Gaps'</td>
</tr>
<tr>
<td>1.3.1.25.2.17.2</td>
<td>Command 'Save Current Values to Recipe'</td>
</tr>
<tr>
<td>1.3.1.25.2.17.3</td>
<td>Command 'Restore Values from Recipe'</td>
</tr>
<tr>
<td>1.3.1.25.2.17.4</td>
<td>Command 'Add all instance paths'</td>
</tr>
</tbody>
</table>
Command 'Reorder List and Clean Gaps'

**Function:** This command cleans the gaps that can result when you make changes to the declaration of persistent variables. The memory requirement is reduced by this cleaning. When the command is executed, CODESYS displays a warning informing the user about the possible loss of data.

**Call:** Main menu “Declarations”, context menu

**Requirement:** The persistence editor (persistent variable list) is active.

Before cleaning you should consider saving the current values of the persistent variables to a recipe (command “Save Current Values to Recipe”). Then you can load the values to the controller again after the next download.

See also

- § Chapter 1.3.1.9.20 “Data persistence” on page 249
- § Chapter 1.3.1.25.2.17.3 “Command ‘Restore Values from Recipe’” on page 1011
- § Chapter 1.3.1.25.2.17.2 “Command ‘Save Current Values to Recipe’” on page 1011

Command 'Save Current Values to Recipe'

**Function:** This command creates a new recipe definition in the recipe manager and stores the current values of the persistent variables in it. You should execute this command before the command “Reorder List and Clean Gaps” in order to avoid a possible loss of data. You can subsequently restore the data with the command “Restore Values from Recipe”.

**Call:** Main menu “Deklarationen”

**Requirement:** The application is in online mode and the persistence editor (persistent variable list) is active.

If a list already exists in the recipe manager with the corresponding names when saving a persistent variable list, then the current persistent variables are sorted into the list:

- New persistent variables are added to the list
- Variables, that are not in the list, will be deleted

Therefore, it is possible to add more recipes to the list in the recipe manager and these will be retained. However, if new variables are added to the list, then these are deleted the next time the command “Save Current Values to Recipe” is executed.

See also

- § Chapter 1.3.1.9.20 “Data persistence” on page 249
- § Chapter 1.3.1.25.2.17.1 “Command ‘Reorder List and Clean Gaps’” on page 1011
- § Chapter 1.3.1.25.2.17.3 “Command ‘Restore Values from Recipe’” on page 1011
- § Chapter 1.3.1.25.1.23 “Object ‘Persistent variable list’” on page 770

Command 'Restore Values from Recipe'

**Function:** This command restores the values of the persistent variables that you have stored in a recipe using the command “Save Current Values to Recipe”. You would normally select this command after executing the command “Reorder List and Clean Gaps”.

**Call:** Main menu “Declarations”

**Requirement:** The persistence editor (persistent variable list) is active, the application is in online mode
Command 'Add all instance paths'  

**Function:**  
- When you execute the command in the persistence editor, the application is searched for declarations of persistent variables with the `PERSISTENT` keyword which are outside of the persistence editor. For each declaration found, an instance path of this variable is added in the persistence editor.  
- When you execute the command in a variable configuration, an instance path is added for each variable with an incomplete address. All function blocks of the application are considered in this case.  

**Call:** Menu bar: “Declarations”, right-click.  

**Requirement**  
- the persistence editor (global persistent variable list) is active or a variable configuration (global variable list with `VAR_CONFIG` declarations) is opened.  
- The application was compiled successfully.  

See also  
- ☐ Chapter 1.3.1.24.2.12 “Persistent variable - PERSISTENT” on page 498  
- ☐ Chapter 1.3.1.24.2.13 “Retain variable - RETAIN” on page 500  

---  

Menu 'Device Communication', Gateway  

| 1.3.1.25.2.18.1 | Command 'Add gateway'........................................................................ 1012  
| 1.3.1.25.2.18.2 | Command 'Configure the local gateway'............................................... 1013  

Command 'Add gateway'  

**Function:** This command opens the “Gateway” dialog where you can define a gateway channel and add it to the current device configuration.  

**Call:** Menu bar: “Gateway” in the “Communication” dialog of the device editor.  

**Dialog 'Gateway'**  

| “Name” | Name of the gateway. |  
| “Drivers” | Driver type from a drop-down list. |  
| Driver-specific settings, for example: IP address, port | Editable after double-clicking the predefined value. A short description for each parameter is displayed in the lower part of the dialog. Note: You can also specify the address of a DNS domain. This has to begin with `dns:` (example: `dns:MyDynDNSAdress`). |  

The dialog is also used for later editing of the gateway entries of your project.  

See also  
- ☐ Chapter 1.3.1.25.1.19.2 “Tab 'Communication Settings'” on page 741
Command 'Configure the local gateway'

**Function:** The command opens the dialog “Gateway Configuration”, where you can set up the block driver configuration for the local gateway. This is an alternative to manually editing of the configuration file `Gateway.cfg`.

**Call:** Context menu of a gateway entry in the “Communication Settings” dialog.

---

**NOTICE!**

A correct gateway configuration requires detailed knowledge. In case of any doubts do not change the standard configuration settings!

---

**Dialog 'Gateway Configuration'**

The configuration tree shown in the dialog corresponds to the currently valid configuration file `gateway.cfg`. For the concerned interfaces it shows the parameters and current settings. Changes of the configuration in the dialog, confirmed with “OK” effect an direct update of the configuration file.

After having changed the gateway configuration file `gateway.cfg`, the gateway must be stopped and restarted in order to get the changes applied.

---

<table>
<thead>
<tr>
<th>“Add”</th>
<th>Menu with commands for adding interfaces and settings. The commands are also available in the context menu of the dialog. The command set depends on the currently selected entry and on the already entered settings:</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Add Top Level Interface”: Choose an interface for the communication via the gateway. It will be inserted on top level of the tree. See the table below for the possible block driver interfaces.</td>
<td></td>
</tr>
<tr>
<td>“Add Sub Level Interface”: Choose an interface for the communication via the gateway. It will be inserted in the tree below the currently selected interface. See the table below for the possible block driver interfaces.</td>
<td></td>
</tr>
<tr>
<td>“Add Configuration Setting”: Choose a setting for the selected interface. It will be inserted in the tree below that interface. To get an edit field for the setting, perform a double-click in column “Setting”. See the table below for the possible settings for the block driver interface.</td>
<td></td>
</tr>
</tbody>
</table>

| “Delete” | Deletes the selected configuration settings. |
| “Up”, “Down” | Moves the selected configuration setting one position up or down. |
Table 172: Possible block driver interfaces

<table>
<thead>
<tr>
<th>Interface Type</th>
<th>Description</th>
<th>Configuration Settings</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;COM Port&quot;</td>
<td>Serial port on the device, for example for data exchange according to standard RS232 on a COM port intended for this purpose.</td>
<td>Possible configuration settings:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&quot;Name&quot;: just symbolic</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&quot;Port&quot;: physical serial port used for this interface; for example COM 5 on a Windows computer</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&quot;Baudrate&quot;: 2400, 4800, 9600, 19200, 38400, 57600, 115200</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&quot;Enable auto addressing&quot;: [ ] (Standard = [ ]): The setting</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Local address is evaluated; then the two devices communicating via the serial interface, will negotiate their addresses autonomously before they start exchanging messages; in case the two devices have equal addresses, then they will be renegotiated; this setting will be useful if the local addresses cannot be set explicitly, for example at physically separated devices.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&quot;Local address&quot;: is evaluated only if &quot;Enable auto addressing&quot; is set! Standard = actual value for port</td>
</tr>
<tr>
<td>&quot;Shared Memory&quot;:</td>
<td>Shared Memory driver</td>
<td>Possible settings:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&quot;Name&quot;: just symbolic</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&quot;Forced address&quot;: Standard = -1 (= no forced address); Example: 42 means that the driver must use the fix address defined here, and that addresses are assigned freely in the range of 0-255; this setting may be useful when more than 1 Shared Memory driver is enabled within the configuration.</td>
</tr>
<tr>
<td>&quot;Ethernet UDP/IP&quot;:</td>
<td>Ethernet interface for data exchange matching the &quot;User Datagram Protocol&quot;.</td>
<td>Possible settings:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&quot;Name&quot;: just symbolic</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&quot;Port index&quot;: Port number for the communication; port indices are in the range of 0 - 3; they get mapped on the following Ethernet ports: 1740 - 1743</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&quot;IP address&quot;: Standard = 127.0.0.1; this setting may be useful for explicitly setting an interface if there are multiple network interfaces on the device; Example: 127.0.0.1 stands for any local network interface, also named localhost; each other address - e.g. 10.27.7.72 - represents a real IP address which must be available on the device.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&quot;Network mask&quot;: Standard = 255.255.255.0; Example: 255.255.252.0; this setting may be useful for explicitly setting an interface if there are multiple network interfaces on the device;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&quot;PPP remote address&quot;: Standard = 127.0.0.1; Example: 10.13.42.240; establishes a logical point-to-point connection between the UDP interface and the node which is named by the here-defined address; this effects that the UDP interface will communicate exclusively with this node, and that no broadcasts will be sent within the network.</td>
</tr>
<tr>
<td>&quot;Ethernet TCP/IP&quot;:</td>
<td>Ethernet interface for the data exchange matching the &quot;Transmission Control Protocol&quot;.</td>
<td>Possible settings:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&quot;Name&quot;: just symbolic</td>
</tr>
</tbody>
</table>
Possible settings:
"Name", "Port", "IP address": see above: Ethernet UDP/IP
"Inactivity timeout": Standard = 0; this setting defines the time in seconds, after which the TCP connections will be closed if no more data are exchanged

"CAN Client"  "Name", just symbolic
For information on the other settings please see the description field the lower part of the dialog.

"USB Port"  "Name", just symbolic
For information on the other settings please see the description field the lower part of the dialog.

See also
● § Chapter 1.3.1.25.1.19.2 “Tab ‘Communication Settings’” on page 741

Menu 'Recipes'

1.3.1.25.2.19.1 Command 'Insert Variable'..................................................... 1015
1.3.1.25.2.19.2 Command 'Add a New Recipe'.............................................. 1015
1.3.1.25.2.19.3 Command 'Remove Recipe'.................................................. 1016
1.3.1.25.2.19.4 Command 'Load Recipe'....................................................... 1016
1.3.1.25.2.19.5 Command 'Save Recipe'....................................................... 1016
1.3.1.25.2.19.6 Command 'Read Recipe'....................................................... 1017
1.3.1.25.2.19.7 Command 'Write Recipe'....................................................... 1017
1.3.1.25.2.19.8 Command 'Load and Write Recipe'.................................... 1017
1.3.1.25.2.19.9 Command 'Read and Save Recipe'....................................... 1018
1.3.1.25.2.19.10 Command 'Remove Variables'........................................... 1018
1.3.1.25.2.19.11 Command 'Load Recipes from Device'................................ 1019
1.3.1.25.2.19.12 Command 'Update Structured Variables'................................ 1019

Command 'Insert Variable'

Symbol: 

Function: This command inserts a variable into the currently opened recipe definition before the selected position.

Call: Main menu “Recipes”.

Requirement: You have opened a recipe definition in the editor and selected the normal view.

CODESYS inserts the default text "NewVariable" in the column “Variable”. You must replace this name with the respective variable name. To do this, open the input assistant by clicking ⨁ or enter the variable name directly into the table element.

See also
● § Chapter 1.3.1.13.2 “Changing values with recipes” on page 331
● § Chapter 1.3.1.25.2.19 “Menu ‘Recipes’” on page 1015

Command 'Add a New Recipe'

Symbol: 

Function: This command opens a dialog box for adding a new recipe (new column) to the recipe definition.
Call: Main menu “Recipes”.

Requirement: You have opened a recipe definition in the editor.

After choosing the command, a dialog box opens for you to define the name of the new recipe. The dialog box also provides the capability of copying existing recipes into the new recipe.

See also
- ≈ Chapter 1.3.1.13.2 “Changing values with recipes” on page 331
- ≈ Chapter 1.3.1.25.2.19 “Menu ‘Recipes’” on page 1015

Command ‘Remove Recipe’

Symbol: 🗑

Function: This command removes a recipe from the currently opened recipe definition.

Call: Main menu “Recipes”.

Requirement: You have selected a field in the recipe column of a recipe definition.

See also
- ≈ Chapter 1.3.1.13.2 “Changing values with recipes” on page 331
- ≈ Chapter 1.3.1.25.2.19 “Menu ‘Recipes’” on page 1015

Command ‘Load Recipe’

Symbol: 📄

Function: The command loads a recipe from a file.

Call: Menu bar: “Recipes”.

Requirement: You have selected a field in the recipe column of a recipe definition.

This command overwrites the values of the selected recipe of the recipe definition.

If you want to overwrite only individual recipe variables with new values, then remove the values for the other variables before loading to the recipe file. Entries without value definitions are not read, and therefore updating leaves these variables unchanged on the PLC and in the project.

For values of the data type REAL/LREAL, the hexadecimal value is also written to the recipe file in some cases. This is necessary so that the exact identical value is restored when converting back. In this case, change the decimal value and delete the hexadecimal value.

See also
- ≈ Chapter 1.3.1.13.2 “Changing values with recipes” on page 331
- ≈ Chapter 1.3.1.25.2.19 “Menu ‘Recipes’” on page 1015

Command 'Save Recipe'

Symbol: 🔄
**Function:** This command saves the variable values of a recipe to a file.

**Call:** Main menu “Recipes”.

**Requirement:** You have selected the value of a recipe in the recipe definition.

When you choose this command, CODESYS saves the values of the selected recipe to a file. You can define the format in the settings for the recipe manager in the tab “Storage”.

See also

- Chapter 1.3.1.13.2 “Changing values with recipes” on page 331
- Chapter 1.3.1.25.2.19 “Menu ‘Recipes’” on page 1015

---

**Command 'Read Recipe’**

**Function:** This command reads the variable values of a recipe from the controller.

**Call:** Main menu “Recipes”.

**Requirement:** The application is in online mode and you have selected the value of a recipe in the recipe definition.

When you choose this command, CODESYS overwrites the values of the selected recipe with the read values from the controller.

*If you have selected the option “Recipe Management in the PLC”, please note the following.*

*If you change recipes in the project by choosing the command “Load Recipe” or “Read Recipe”, then an online change is required when logging in again.*

See also

- Chapter 1.3.1.13.2 “Changing values with recipes” on page 331
- Chapter 1.3.1.25.2 “Menu Commands” on page 852

---

**Command 'Write Recipe’**

**Function:** This command writes the values of a recipe to the variables in the controller.

**Call:** Main menu “Recipes”.

**Requirement:** The application is in online mode and you have selected the value of a recipe in the recipe definition.

When you choose this command, CODESYS overwrites the values in the controller with the values of the selected recipe.

See also

- Chapter 1.3.1.13.2 “Changing values with recipes” on page 331
- Chapter 1.3.1.25.2.19 “Menu ‘Recipes’” on page 1015

---

**Command 'Load and Write Recipe’**

**Function:** This command loads a recipe from a file and writes the values to the variables in the PLC.

**Call:** Menu bar: “Recipes”.

---
**Requirement:** The application is in online mode. You have selected the value of a recipe in the recipe definition.

After choosing the command, you are prompted either to write the variable values also to the recipe in the project or only to write them to the PLC. Updating the values in the recipe could require an online change when logging in again.

When you choose this command, CODESYS overwrites the values of the selected recipe of the recipe definition. In addition, these recipe values overwrite the variable values in the PLC.

> **If you want to overwrite only individual recipe variables with new values, then remove the values for the other variables before loading to the recipe file. Entries without value definitions are not read, and therefore updating leaves these variables unchanged on the PLC and in the project.**

*For values of the data type REAL/LREAL, the hexadecimal value is also written to the recipe file in some cases. This is necessary so that the exact identical value is restored when converting back. In this case, change the decimal value and delete the hexadecimal value.*

See also
- § Chapter 1.3.1.13.2 “Changing values with recipes” on page 331
- § Chapter 1.3.1.25.2.19 “Menu ‘Recipes”’ on page 1015

**Command ‘Read and Save Recipe’**

**Symbol:** 

**Function:** This command reads the variable values of a recipe from the controller and saves them to a file.

**Call:** Main menu “Recipes”.

**Requirement:** The application is in online mode and you have selected the value of a recipe in the recipe definition.

After choosing the command, you are prompted either to read the variable values to the recipe or only to save them. Updating the values in the recipe could require an online change when logging in again.

The values are saved with the default name for recipe files according to the settings for the recipe manager (tab “Storage”).

See also
- § Chapter 1.3.1.13.2 “Changing values with recipes” on page 331
- § Chapter 1.3.1.25.2.19 “Menu ‘Recipes”’ on page 1015

**Command ‘Remove Variables’**

**Symbol:** 

**Function:** This command removes the selected variables from a “Recipe Definition”.

**Call:** The command is not in any menu by default. You can add it to a menu by using the dialog box from “Tools ➔ Customize” (command category “Recipe”).

See also
- § Chapter 1.3.1.13.2 “Changing values with recipes” on page 331
- § Chapter 1.3.1.25.2.19 “Menu ‘Recipes”’ on page 1015
Command 'Load Recipes from Device'

Symbol: 📁

Function: This command initiates the synchronization of the recipes from the open recipe definition in the project and the recipes located on the device in the form of recipe files.

Call: Menu bar: “Recipes”.

Requirement: The application is in online mode and a recipe definition is open in the editor.

In detail, the synchronization is described as follows:

● The current values for the recipe variables located in the project are overwritten by the values from the recipes on the controller. As a result, there is likely an online change at the next login.

● If recipe variables are defined in the recipe files on the controller, and the recipe variables are missing in the recipe definition of the project, then these variables are ignored when at the time of download. Before that, a message appears for each recipe file regarding the variables in question.

● If recipe variables are missing in the recipe files on the controller, and these recipe variables are included in the recipe definition of the project, then a message appears for each recipe file with the variables in question.

● If more recipes for these variables have been created on the controller, then they are added to the recipe definition in the project.

Command 'Update Structured Variables'

Symbol: ➗

Function: This command opens the “Update Structured Variables” dialog box.

Call: Main menu “Recipes”.

In this dialog box, you can update recipe definitions if the declaration of a structured variable or a block has changed. For example, if the dimension of an array is changed, then you can automatically add or remove the entries in the recipe definition.

Table 173: Dialog Box 'Update Structured Variables'

<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Remove not existing variables&quot;</td>
<td>Variables are removed from the recipe definition when they no longer exist in the project due to a change to a structured element.</td>
</tr>
<tr>
<td>&quot;Update instances of structures and function blocks&quot;</td>
<td>If the declaration of a structure or function block is extended and available in the recipe definition with an instance, then the respective variables are added to the recipe definition.</td>
</tr>
<tr>
<td>&quot;Update array dimensions of array instances&quot;</td>
<td>If the dimension of an array is extended and available in the recipe definition with an instance, then the respective variables are added to the recipe definition.</td>
</tr>
<tr>
<td>&quot;Update contained global variable lists&quot;</td>
<td>If the declaration of a global variable list is extended and available in the recipe definition with an instance, then the respective variables are added to the recipe definition.</td>
</tr>
<tr>
<td>&quot;Update contained programs&quot;</td>
<td>If the declaration of a program is extended and instanced in the recipe definition, then the respective variables are added to the recipe definition.</td>
</tr>
</tbody>
</table>

See also

● ❱ Chapter 1.3.1.13.2 “Changing values with recipes” on page 331
● ❱ Chapter 1.3.1.25.2.19 “Menu ‘Recipes’” on page 1015
Menu 'Text List'

1.3.1.25.2.20.1 Command 'Add Language'.................................................... 1020
1.3.1.25.2.20.2 Command 'Create Global Text List'........................................ 1020
1.3.1.25.2.20.3 Command 'Export Everything as Text'.................................... 1020
1.3.1.25.2.20.4 Command 'Export All Unicode .txt Text List Files'...................... 1021
1.3.1.25.2.20.5 Command 'Insert Text'........................................................... 1021
1.3.1.25.2.20.6 Command 'Import/Export Text Lists'........................................ 1021
1.3.1.25.2.20.7 Command 'Remove Language'................................................ 1022
1.3.1.25.2.20.8 Command 'Rename Language'................................................ 1022
1.3.1.25.2.20.9 Command 'Remove Unused Text List Entries'............................ 1023
1.3.1.25.2.20.10 Command 'Check Visualization Text IDs'.................................. 1023
1.3.1.25.2.20.11 Command 'Update Visualization Text IDs'................................ 1023
1.3.1.25.2.20.12 Command 'Add Text List Support'.......................................... 1024
1.3.1.25.2.20.13 Command 'Remove Text List Support'.................................... 1024

Command 'Add Language'

Symbol: 📑

Function: This command adds a further language column to the text list.
Call: Main menu “Textlist”, context menu
Requirement: A text list or a global text list is open and active.
In the dialog box “Enter Language”, enter a code for the new language, for example “en-US”. CODESYS inserts the code as column header.

Command 'Create Global Text List'

Symbol: 📑

Function: This command creates the global text list in the “POUs” view.
Call: “Visualization”, context menu.
Requirements: A visualization is open.
See also
● Chapter 1.3.1.25.1.20 “Object ‘GlobalTextList’” on page 768
● Chapter 1.3.1.9.9.1 “Managing static text in global text lists” on page 217

Command 'Export Everything as Text'

Symbol: 💾

Function: This command exports all the text lists of the project.
Call: Main menu “Textlist”, context menu
Requirement
● A text list or a global text list is open and active.
● The visualization does not code the characters of the texts in Unicode.
CODESYS creates a file as plain text in the format .txt for each text list. The name of the text list becomes the name of the file. The directory into which the files are exported is set in “Project ➤ Project Settings ➤ Visualization”, category “General” in “Text list files”.
A controller can read and use this format. For example, you can copy the file to a controller and, by means of a configuration in the visualization manager, prevent the text lists from being transmitted again when loading the application.
Command 'Export All Unicode .txt Text List Files'
Symbol:  

**Function:** This command exports all the text lists of the project.

**Call:** Main menu “Textlist”, context menu

**Requirement**
- A text list or a global text list is open and active.
- The visualization codes the characters of the texts in Unicode.
  - The option “Use Unicode strings” in the visualization manager is activated.
  - The compiler instruction VISU_USEWSTRING in the application is set. Check this by selecting the command “Properties” in the context menu of the application. Then select the “Compile” tab. VISU_USEWSTRING must be entered in the input field for “Compiler defines.”

CODESYS creates a file as plain text in the format .txt for each text list. The name of the text list becomes the name of the file. The directory into which the files are exported is set in “Project ➔ Project Settings ➔ Visualization”, category “General” in “Text list files”.

A controller can read and use this format. For example, you can copy the file to a controller and, by means of a configuration in the visualization manager, prevent the text lists from being transmitted again when loading the application.

See also
- Chapter 1.3.1.9.9 “Managing text in text lists” on page 213

Command 'Insert Text'
Symbol:  

**Function:** This command inserts a new line into the text list above the selected line. An input field under “Standard” opens, in which you input the source text.

**Call:** Main menu “Textlist ➔ Insert Text”, context menu

**Requirement:** A text list, not a “GlobalTextList”, is open and active. A field in the table is selected.

See also
- Chapter 1.3.1.9.9 “Managing text in text lists” on page 213

Command 'Import/Export Text Lists'
Symbol:  

**Function:** This command exports an active text list, imports a file, or matches a text list with a file. The file has the CSV format. The “Import/Export” dialog provides options for this.

**Call:** Menu bar: “Text List ➔ Import/Export Text Lists”; context menu

**Requirement:** A text list or global text list is active.

Dialog 'Import/Export'

<table>
<thead>
<tr>
<th>“Select File for Import”</th>
<th>File that CODESYS reads.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>opens the dialog “Select Text List File” for you to select a file.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>“Select export file”</th>
<th>File that CODESYS writes to.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>opens the dialog “Select Text List File” for you to select a file and directory.</td>
</tr>
</tbody>
</table>
Table 174: “Import/Export Type”

| “Import”          | Requirement: A file is selected in “Select file for compare or import”. The file can contain text list entries for both the global text list and text lists.  
|                  | Global text list  
|                  | • CODESYS reads the file, compares the text list entries for the same source text, and accepts differences in the translations. CODESYS overwrites any translations in the project.  
|                  | Text lists  
|                  | • CODESYS reads the file, compares the text list entries for the same ID, and accepts differences in the source text and translations into the project. CODESYS overwrites any text list entries in the project.  
|                  | • If the file contains a new ID, then the text list entry is imported into the text list of the project and the text list is added.  
| “Import replacement file” | Requirement: A replacement file is selected in “Select file for compare or import”. The replacement file contains replacements for the global text list. CODESYS processes the replacement file row by row and performs the specified replacements in the global text list. The structure of the replacement file is described in the section "Managing static text in a global text list".  
| “Export”         | Requirement: The file that CODESYS writes to is selected in “Select export file”. CODESYS exports all texts from all text lists of the current project. All languages available in the project are inserted as columns in the export file. The file can be used for the external translation of the language-dependent texts.  
| “Export text differences only” | Requirement: An import file is selected for the comparison in “Select file for compare or export”, and an export file that CODESYS writes to is selected in “Select export file”. CODESYS reads the import file and then uses that information to compare the rows of the active text list. CODESYS ignores the rows that match. If rows differ, then CODESYS writes the row to the export file and, if necessary, copies translations from the text list. CODESYS accepts the translations from the import file and overwrites them if necessary.  

See also
- Chapter 1.3.1.9.9 “Managing text in text lists” on page 213  
- “Updating the global text list with a replacement file” on page 218  

Command 'Remove Language'

Symbol: 🍀  
**Function**: Removes the selected language column from the text list.  
**Call**: Main menu “Textlist”, context menu  
**Requirement**: A text list or a global text list is open and active. A field is selected in the column of the language that you wish to remove.  

See also
- Chapter 1.3.1.9.9 “Managing text in text lists” on page 213  

Command 'Rename Language'

Symbol: 🍀
Function: Opens a dialog for specifying a new name for a language that is displayed in the text list as a column heading.

Call: Menu bar: “Text List”, context menu.

Requirement: A text list or global text list is active. A field in the language column to be renamed is selected.

See also
- © Chapter 1.3.1.9.9 “Managing text in text lists” on page 213

Command 'Remove Unused Text List Entries'

Symbol: 

Function: This command checks whether a text list entry in the project is used as static text. If not, CODESYS removes it from the text list.

Call: Main menu “Textlist”, context menu

Requirement: The “GlobalTextList” is open and active. A field in the table is selected.

See also
- © Chapter 1.3.1.9.9 “Managing text in text lists” on page 213

Command 'Check Visualization Text IDs'

Symbol: 

Function: This command checks whether the ID of a text list entry in the project is correct and reports the result.

Call: Main menu “Textlist”, context menu

Requirement: The “GlobalTextList” is open and active. A field in the table is selected.

If CODESYS finds during checking that the global text list and the static texts of the visualizations do not correspond, this could be because the global text list is or was write protected. The requirement for this is that you have set up a user management system in the project.

See also
- © Chapter 1.3.1.9.9 “Managing text in text lists” on page 213

Command 'Update Visualization Text IDs'

Symbol: 

Function: This command updates all inconsistent IDs in a static text list.

Call: Main menu “Textlist ➔ Paste Text”, context menu

Requirement: The “GlobalTextList” is open and active. A field in the table is selected. The object is write protected.

If CODESYS finds during checking that the global text list and the static texts of the visualizations do not correspond, this could be because the global text list is or was write protected. The requirement for this is that you have set up a user management system in the project.

See also
- © Chapter 1.3.1.9.9 “Managing text in text lists” on page 213
- © Chapter 1.3.1.6.5 “Protecting objects in the project by access rights” on page 146
**Command 'Add Text List Support'**

**Symbol:** 📑

**Function:** This command adds text list support to the selected DUT object (type: enumeration).

**Call:** Context menu of a standard DUT object (type: enumeration 📑).

Text list support allows the localization of the enumeration component identifier and the display of the symbolic component value in a text output of a visualization.

See also

- 📑 Chapter 1.3.1.25.1.14 “Object ‘DUT’” on page 735
- 📑 Chapter 1.3.1.25.2.20.13 “Command ‘Remove Text List Support’” on page 1024

**Command 'Remove Text List Support'**

**Symbol:** 📑

**Function:** This command removes text list support from the selected enumeration object.

**Call:** Context menu of an object of an enumeration with text list support 📑.

Text list support allows the localization of the enumeration component identifier and the display of the symbolic component value in a text output of a visualization.

47293

See also

- 📑 Chapter 1.3.1.25.1.14 “Object ‘DUT’” on page 735
- 📑 Chapter 1.3.1.25.2.20.12 “Command ‘Add Text List Support’” on page 1024

**Menu 'Trace'**

1.3.1.25.2.21.1  Command 'Add Variable'................................................................................. 1024
1.3.1.25.2.21.2  Command 'AutoFit'....................................................................................... 1025
1.3.1.25.2.21.3  Command 'Compress'.................................................................................... 1025
1.3.1.25.2.21.4  Command 'Configuration'.............................................................................. 1025
1.3.1.25.2.21.5  Command 'Cursor'......................................................................................... 1025
1.3.1.25.2.21.6  Command 'Download Trace'........................................................................ 1026
1.3.1.25.2.21.7  Command 'Export Symbolic Trace Config’.................................................. 1027
1.3.1.25.2.21.8  Command 'Load Trace’.................................................................................. 1029
1.3.1.25.2.21.9  Command 'Mouse Zooming’......................................................................... 1029
1.3.1.25.2.21.10 Command 'Convert to Multi-Channel’......................................................... 1029
1.3.1.25.2.21.11 Command 'Convert to Single-Channel’......................................................... 1030
1.3.1.25.2.21.12 Command 'Online List’................................................................................ 1031
1.3.1.25.2.21.13 Command 'Reset Trigger’........................................................................... 1032
1.3.1.25.2.21.14 Command 'Reset View’............................................................................... 1032
1.3.1.25.2.21.15 Command 'Save Trace’............................................................................... 1033
1.3.1.25.2.21.16 Command 'Start Trace’.............................................................................. 1033
1.3.1.25.2.21.17 Command 'Stop Trace’.............................................................................. 1033
1.3.1.25.2.21.18 Command 'Stretch’.................................................................................... 1034
1.3.1.25.2.21.19 Command 'Upload Trace’........................................................................... 1034
1.3.1.25.2.21.20 Command 'Statistics’............................................................................... 1034

**Command 'Add Variable'**

**Function:** This command adds a trace variable to the configuration.
**Call**: Main menu “Trace”.

A new variable is displayed in the tree view of the trace configuration. The associated variables configuration appears to the right in “Variable Settings”.

See also

- Chapter 1.3.1.25.3.15.2 “Dialog ‘Trace Configuration’” on page 1092
- Chapter 1.3.1.13.3.2 “Creating trace configuration” on page 338

**Command ’AutoFit’**

Symbol: 

**Function**: This command scales the y-axis of the trace diagram for optimum display of all graphs, making sure that the y-values fit in the visible region of the diagrams. The command works with both single-channel and multi-channel displays.

**Call**: Menu bar: “Trace”; context menu.

**Trace in a single-channel display**

When all trace variables are displayed in one diagram, the trace is in single-channel display.

**Trace in a multi-channel display**

When the trace variables are displayed in multiple diagrams, the trace is in multi-channel display.

See also

- Chapter 1.3.1.25.3.15.2 “Dialog ‘Trace Configuration’” on page 1092
- Chapter 1.3.1.13.3.5 “Navigating into trace data” on page 343

**Command ’Compress’**

Symbol: 

**Function**: This command compresses the trace graph by zooming into the displayed time range by a fixed percentage.

**Call**: Main menu “Trace”, or context menu.

See also

- Chapter 1.3.1.25.2.21.18 “Command ’Stretch’” on page 1034
- Chapter 1.3.1.13.3.5 “Navigating into trace data” on page 343

**Command ’Configuration’**

**Function**: This command opens the “Trace Configuration” dialog box for enabling the configuration of the data recording.

**Call**: Main menu “Trace”, or context menu.

See also

- Chapter 1.3.1.25.3.15.2 “Dialog ‘Trace Configuration’” on page 1092
- “Subdialog ‘Variable Settings’” on page 1095

**Command ’Cursor’**

Symbol: 

---

2020/12/10

3ADR010583, 1, en_US 1025
**Function:** This function
- inserts a trace cursor into the trace diagram when no trace cursor is available
- inserts a second trace cursor into the trace diagram when 1 trace cursor is available
- removes the trace cursors when 2 trace cursors are available

**Call:** Menu bar: “Trace”; context menu.

A trace cursor is a small black triangle with a vertical black line running parallel to the y-axis.

**Trace diagram without trace cursors**
In this mode, you can process the trace diagram with the mouse pointer. The x-value that focuses on with the cursor is displayed in the status bar with normal style. Example: “Time: 1m23s456ms; Value: 1”

**Trace diagram with one trace cursor**
In the status bar and y-value, CODESYS prints the time that was marked by the trace cursor. Example: “Time: 1m23s456ms”

**Trace diagram with two trace cursors**
In the status bar, CODESYS prints the two times and the time interval that are marked by the two trace cursors. Example: “Time: 1m23s456ms - Time: 1m24s456ms (Δ 1s)”.

**User input in the trace diagram**
If one or two trace cursors are available, then you can move them along the x-axis.

<table>
<thead>
<tr>
<th>Mouse Input</th>
<th>Symbol</th>
<th>Effect</th>
</tr>
</thead>
<tbody>
<tr>
<td>Drag the triangle of a trace cursor to another position.</td>
<td>![Cursor Symbol]</td>
<td>While the mouse button is pressed, the cursor can be moved without restriction. The current y-value is always displayed in the status bar. When the mouse button is released, the cursor jumps to the nearest measuring point</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Keyboard Shortcuts</th>
<th>Effect</th>
</tr>
</thead>
<tbody>
<tr>
<td>![Left arrow] ![Right arrow]</td>
<td>CODESYS moves the black trace cursor to the next measuring point.</td>
</tr>
<tr>
<td>![Shift]+![Left Arrow] ![Shift]+![Right Arrow]</td>
<td>CODESYS moves the gray trace cursor to the next measuring point.</td>
</tr>
</tbody>
</table>

See also
- *Chapter 1.3.1.13.3.5 “Navigating into trace data” on page 343*

**Command ’Download Trace’**

**Symbol:** ![Download Trace]

**Function:** This command transfers the trace configuration on the controller to the associated application, and starts the data recording. The recorded data is transferred back to the development system. The trace diagram shows the current samples and continues.

**Call:** Menu bar: “Trace”; context menu.

**Requirement:** The command is available when the assigned application is in online mode.

See also
- *Chapter 1.3.1.13.3.3 “Operating the data recording” on page 341*
Command 'Export Symbolic Trace Config'

**Function:** This command exports a trace configuration to a `traceconfig` file.

**Call:** Main menu “Trace”, or context menu.

**Requirement:** The origin application includes a symbol configuration that defines the configured trace variables as symbols. Access to the IEC variables where data was recorded is therefore symbolic. Then you can use the trace configuration for various similar applications.

Using the configuration file

You can transfer this file to any runtime system. At runtime, its `CmpTraceMgr` runtime system component can access and perform data recording. The configuration file also includes information about the application context in addition to the configuration data.

The configuration file defines the following context:

- Application name
- Trace name
- Task name

The application that is executed at runtime must fulfill the following conditions:

- The application has the same name as the origin application.
- The trace that is configured in the application has the same as the trace that is configured in the origin application.
- The task that is running in the data recording has the same name as the task that is configured in the origin application.

**NOTICE!**

The configuration is not loaded automatically. You must execute the command explicitly.

You can proceed as follows:

- Access the trace manager programmatically via IEC code by using library interfaces.
- Register the configuration file with the trace manager. Then the trace manager loads the configuration file when the application is started.

For more information about the functionality of the trace manager, refer to "Trace Manager Runtime System Component Description".
Sample configuration file

```
[key]; [value]
Version; 0x03050000
Name; Application.Trace_Trigger
ApplicationName; Application
ApplicationDataGuid; 00000000-0000-0000-0000-000000000000
IecTaskName; MainTask
Comment;
Trigger.Flags; 5
Trigger.Edge; 2
Trigger.Position; 0
Trigger.UpdatesAfterTrigger; 50
Trigger.Variable.Name; PLC_PRG.B.OUT
Trigger.Variable.AddrFlags; 0x00000101
Trigger.Variable.Class; 0
Trigger.Variable.Size; 1
Trigger.Level;
Condition.Name;
Condition.AddrFlags; 0x00000000
Condition.Class; 0
Condition.Size; 0
EveryNCycles; 1
BufferEntries; 100
Flags; 16
0.Variable; PLC_PRG.S5.OUT
0.Address.AddrFlags; 0x00000101
0.Class; 7
0.Size; 2
0.GraphColor; 4278190335
0.GraphType; 3
0.MinWarningColor; 4278190080
0.MaxWarningColor; 4294901760
0.CriticalLowerLimit; 0
0.CriticalUpperLimit; 0
0.ActivateMinWarning; 0
0.ActivateMaxWarning; 0
0.YAxis; 0
0.Data;
1.Variable; PLC_PRG.B.OUT
1.Address.AddrFlags; 0x00000101
1.Class; 0
1.Size; 1
1.GraphColor; 4278222848
1.GraphType; 1
1.MinWarningColor; 4278190080
1.MaxWarningColor; 4294901760
1.CriticalLowerLimit; 0
1.CriticalUpperLimit; 0
1.ActivateMinWarning; 0
1.ActivateMaxWarning; 0
1.YAxis; 0
1.Data;
```
Command 'Load Trace'

**Function:** This command makes it possible to load a file, which contains the configuration and data, and was saved to the file system of the development system. The "Load Trace" dialog box opens.

**Call:** Main menu “Trace”, or context menu.

### Dialog box “Load Trace”

<table>
<thead>
<tr>
<th><strong>“File name”</strong></th>
<th>Name of the file that is loaded</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>“File type”</strong></td>
<td>File format</td>
</tr>
<tr>
<td></td>
<td>● *.trace: “Trace file” that includes the trace configuration</td>
</tr>
<tr>
<td></td>
<td>● *.csv: Text file in CSV format that includes a trace configuration</td>
</tr>
</tbody>
</table>

See also

- Chapter 1.3.1.13.3.6 "Managing trace" on page 344
- Chapter 1.3.1.25.2.21.15 “Command 'Save Trace'" on page 1033
- Chapter 1.3.1.13.3.6 "Managing trace" on page 344

Command 'Mouse Zooming'

**Symbol:**  (command disabled),  (command enabled)

**Function:** This command enables and disables mouse zooming in the trace diagram.

**Call:** Menu bar: “Trace”; context menu.

**User input in the trace diagram**

If the command is enabled, then you can stretch a box with the mouse. When you release the mouse button, the display zooms in on the box and the data is enlarged.

See also

- Chapter 1.3.1.13.3.5 “Navigating into trace data” on page 343

Command 'Convert to Multi-Channel'

**Function:** This command switches the display in the trace editor from single-channel to multi-channel.

**Call:** Menu bar: “Trace”; context menu.

**Multi-channel display**

Multi-channel display means that the trace variables are displayed in multiple diagrams.
Command 'Convert to Single-Channel'

**Function:** This command switches the display in the trace editor from multi-channel to single-channel.

**Call:** Menu bar: “Trace”; context menu.

**Single-channel display**
If a trace is displayed as single-channel, then all trace variables are included in one diagram.
See also

- § Chapter 1.3.1.25.3.15.2 “Dialog 'Trace Configuration'” on page 1092
- § Chapter 1.3.1.13.3.5 “Navigating into trace data” on page 343

**Command 'Online List'**

**Function:** This command opens the “Online List” dialog. If the trace editor of a DeviceTrace object is active, then all traces that are running on the controller are displayed in a tree view. If the trace editor of an application-specific trace object is active, then only this trace is displayed if it is running on the controller.

**Call:** Menu bar: “Trace”; context menu of the trace editor.

**Requirement:** The runtime system uses the CmpTraceMgr components. An application belonging to the device is in online mode.
NOTICE!
Closing the DeviceTrace editor terminates the connection to the controller.

Please note that the connections to the controller is also terminated when the last open "DeviceTrace" editor is closed. In order for device traces to be displayed again in the project, you have to reload them into the "DeviceTrace" objects.

At this time, closing the editor is also the recommended procedure for deliberately terminating the connection to the controller. Logging out is not enough for this.

**Dialog 'Online List'**

<table>
<thead>
<tr>
<th>“Delete from runtime”</th>
<th>Stops and removes the selected trace from the running application.</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Upload”</td>
<td>This command is visible when a DeviceTrace is loaded in the trace editor. A DeviceTrace is a trace that runs on the controller: In the device tree, it can be represented with a DeviceTrace object directly below a device. When you execute this command, the trace that is selected in the tree view is loaded from the runtime system into the trace editor. Any existing configuration in the project is overwritten. For example, the device can provide traces for data of the processor load (cpuload, plcload), which then you can track in the trace editor in CODESYS. An individual &quot;DeviceTrace&quot; object is necessary in the device tree for each trace of the device that should be displayed in the project.</td>
</tr>
</tbody>
</table>

See also
- Chapter 1.3.1.13.3.4 “Accessing all traces on the controller” on page 342
- Chapter 1.3.1.25.1.40 “Object ’DeviceTrace’” on page 845
- Chapter 1.3.1.25.1.39 “Object ’Trace’” on page 842
- Chapter 1.3.1.25.2.21.19 “Command ’Upload Trace’” on page 1034

**Command 'Reset Trigger'**

Symbol: 

**Function:** This command resets the trace configuration after a triggered data recording. Then the application can record new data and react to a trigger again.

**Call:** Main menu “Trace”, or context menu.

**Requirement:** After triggering, the complete data is in the buffer of the development system.

See also
- Chapter 1.3.1.13.3.3 “Operating the data recording” on page 341

**Command 'Reset View'**

Symbol: 

**Function:** This command resets the trace diagram to the default view.

**Call:** Main menu “Trace”, or context menu.
**Requirement:** The display in the trace diagram has been changed by zooming, scrolling, or “AutoFit”.

See also
- % Chapter 1.3.1.25.2.21.2 “Command ‘AutoFit’” on page 1025
- % Chapter 1.3.1.13.3.5 “Navigating into trace data” on page 343

**Command ‘Save Trace’**

**Function:** This command saves the data to a file on the development system. Depending on the file format, the configuration may also be saved. The “Save Trace” dialog box opens.

**Call:** Main menu “Trace”, or right-click.

**Dialog Box ‘Save Trace’**

<table>
<thead>
<tr>
<th>“File name”</th>
<th>Name and location of the trace file</th>
</tr>
</thead>
<tbody>
<tr>
<td>“File type”</td>
<td>File format</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>● *.trace:</td>
</tr>
<tr>
<td></td>
<td>“Trace file” contains the data and configuration. You can run the “Load Trace” command to load the file to the trace editor when offline.</td>
</tr>
<tr>
<td></td>
<td>● *.txt:</td>
</tr>
<tr>
<td></td>
<td>“Text file” contains the recorded data. You can load this file type and edit it with tools that support CSV format. It cannot be loaded to the trace editor when offline because the trace editor cannot read this format.</td>
</tr>
<tr>
<td></td>
<td>● *.trace.csv</td>
</tr>
<tr>
<td></td>
<td>“Trace CSV file (data only)” contains the recorded data. Address information is provided for each trace variable. The created file can be read in the runtime system. The data is imported but the trace cannot be started because the variable addresses are not saved.</td>
</tr>
</tbody>
</table>

See also
- % Chapter 1.3.1.25.2.21.8 “Command ‘Load Trace’” on page 1029
- % Chapter 1.3.1.13.3.6 “Managing trace” on page 344

**Command ‘Start Trace’**

Symbol: ➤

**Function:** This command starts the data recording on the controller when it is stopped.

**Call:** Main menu “Trace”, or context menu.

**Requirement:** The assigned application on the runtime system is running and a trace configuration is loaded.

See also
- % Chapter 1.3.1.13.3.3 “Operating the data recording” on page 341

**Command ‘Stop Trace’**

Symbol: ⊹
**Function**: This command stops the data recording of a trace.

**Call**: Main menu “Trace”, or context menu.

**Requirement**: The assigned application on the runtime system is running and executes a trace.

See also

- Chapter 1.3.1.13.3.3 “Operating the data recording” on page 341

---

**Command 'Stretch’**

Symbol: \[\text{\un 11}\]

**Function**: This command stretches the trace graph by zooming out of the displayed time range by a fixed percentage.

**Call**: Main menu “Trace”, or context menu.

See also

- Chapter 1.3.1.25.2.21.3 “Command 'Compress'” on page 1025
- Chapter 1.3.1.13.3.5 “Navigating into trace data” on page 343

---

**Command 'Upload Trace’**

**Function**: This command establishes the connection to the PLC device, if not already connected. Then it opens the “Online List” dialog listing the traces running on the controller. Then the selected trace is loaded to the trace editor by means of the “Upload” command in the dialog.

**Call**: Menu bar: “Trace”; context menu of the trace editor.

**Requirement**: The editor of a “DeviceTrace” object is open. The runtime system uses the CmpTraceMgr components (trace manager). At least one application in the runtime system in running. The communication settings for the PLC are configured correctly in the CODESYS project.

**NOTICE!**

Closing the DeviceTrace editor terminates the connection to the controller.

Please note that the connections to the controller is also terminated when the last open “DeviceTrace” editor is closed. In order for device traces to be displayed again in the project, you have to reload them into the “DeviceTrace” objects.

At this time, closing the editor is also the recommended procedure for deliberately terminating the connection to the controller. Logging out is not enough for this.

See also

- “Dialog 'Online List’” on page 1032
- Chapter 1.3.1.25.1.40 “Object 'DeviceTrace’” on page 845
- Chapter 1.3.1.13.3.4 “Accessing all traces on the controller” on page 342

---

**Command 'Statistics’**

**Function**: This command opens the “Trace Statistics” dialog box, which shows statistics about each trace variable.

**Call**: Main menu “Trace”, or right-click.

**Requirement**: The trace editor contains samples.
Dialog Box 'Trace Statistics' 

The analyzed time range and duration are shown in the caption. The table contains one line per signal.

<table>
<thead>
<tr>
<th>“Signal”</th>
<th>Name pf the trace variable (for example, PLC_PRG.S1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Index”</td>
<td>0-based index of the signal order</td>
</tr>
<tr>
<td>“N”</td>
<td>Number of measurements for the calculations</td>
</tr>
<tr>
<td>“Min”</td>
<td>Smallest value</td>
</tr>
<tr>
<td>“Avg”</td>
<td>Average</td>
</tr>
<tr>
<td>“Median”</td>
<td>Middle value when the values are ordered by size</td>
</tr>
<tr>
<td>“RMS”</td>
<td>Root mean square</td>
</tr>
<tr>
<td>“StdDev”</td>
<td>Standard deviation</td>
</tr>
<tr>
<td>“Max”</td>
<td>Largest value</td>
</tr>
<tr>
<td>“Integral”</td>
<td>Integral</td>
</tr>
<tr>
<td>“Min Δt [s]”</td>
<td>Smallest change of time intervals for successive values</td>
</tr>
<tr>
<td>“Avg Δt [s]”</td>
<td>Average change of time intervals for successive values</td>
</tr>
<tr>
<td>“Median Δt [s]”</td>
<td>Median change of time intervals for successive values</td>
</tr>
<tr>
<td>“StdDev Δt [s]”</td>
<td>Standard deviation of change of time intervals for successive values</td>
</tr>
<tr>
<td>“Max Δt [s]”</td>
<td>Largest change of time intervals for successive values</td>
</tr>
</tbody>
</table>

Click a column head in the table. CODESYS sorts the table by that column, changing the order from ascending to descending and back. Default: The table is sorted ascending by the “Index” column. The signals are then sorted in the same order as in the signal tree.

Click in the line. The line is selected. You can select or clear other lines by pressing [Shift]+[arrow] up or down.

[Ctrl]+[C] CODESYS copies the selected lines as text to the clipboard. The values of the individual columns are tab-separated, and the lines are delimited with the control character [CR] or [LF]. Requirement: At least one line is selected.

See also
- % Chapter 1.3.1.13.3 “Data recording with trace” on page 335

Menu 'Device Application'

1.3.1.25.2.22.1 Command 'Login to All Applications' ........................................ 1036
1.3.1.25.2.22.2 Command 'Logout All Applications' ........................................... 1036
1.3.1.25.2.22.3 Command 'Delete Applications from Device' .............................. 1036
1.3.1.25.2.22.4 Command 'Reset Cold All Applications' .................................... 1036
1.3.1.25.2.22.5 Command 'Reset Warm All Applications' .................................. 1037
1.3.1.25.2.22.6 Command 'Start All Applications' ............................................ 1037
1.3.1.25.2.22.7 Command 'Stop All Applications' ............................................. 1037
Command 'Login to All Applications'

**Function:** This command has the same effect on all applications of the device as the command "Login" for a single application. Therefore, it calls the usual login dialog box, or if no changes were made to the applications, it logs into all applications of the device without any dialog prompt. The applications go into online mode.

**Call:** Context menu of the device object in the device tree.

**Requirement:** The device has a device application. At least one of the applications of the device is not in online mode.

> If you login with a single application of a device for which a device application is configured, then you also login automatically with the device application.

See also
- ☞ Chapter 1.3.1.16 “Using device applications” on page 358
- ☞ Chapter 1.3.1.25.2.6.2 “Command 'Login'” on page 923

Command 'Logout All Applications'

**Function:** This command has the same effect on all applications of the device as the command "Logout" for a single application. The applications go into offline mode.

**Call:** Context menu of the device object in the device tree.

**Requirement:** The device has a device application. At least one of the applications of the device is in online mode.

See also
- ☞ Chapter 1.3.1.16 “Using device applications” on page 358
- ☞ Chapter 1.3.1.25.2.6.3 “Command 'Logout'” on page 926

Command 'Delete Applications from Device'

**Function:** This command deletes the application of the device selected in the device tree.

**Call:** Context menu of the application object in the device tree.

See also
- ☞ Chapter 1.3.1.25.1.19.4 “Tab 'Applications'” on page 747
- ☞ Chapter 1.3.1.25.1.1 “Object 'Application'” on page 710

Command 'Reset Cold All Applications'

**Function:** This command resets all applications of the device. This behavior is as for the use of the “Reset Cold” command for a single application.

**Call:** Context menu of the device object in the device tree.

**Requirement:** The device has a device application. All applications are in online mode.

See also
- ☞ Chapter 1.3.1.16 “Using device applications” on page 358
- ☞ Chapter 1.3.1.25.2.6.9 “Command 'Reset Cold'” on page 931
Command 'Reset Warm All Applications'

**Function:** This command resets all applications of the device. This behavior is as for the use of the “Reset Warm” command for a single application.

**Call:** Context menu of the device object in the device tree.

**Requirement:** The device has a device application. All applications are in online mode.

See also
- Chapter 1.3.1.16 “Using device applications” on page 358
- Chapter 1.3.1.25.2.6.10 “Command ‘Reset Warm’” on page 931

Command 'Start All Applications'

**Function:** This command starts all applications that belong to the device configuration.

**Call:** Context menu of the device object in the device tree.

**Requirement:** The device has a device application and is in online mode.

See also
- Chapter 1.3.1.16 “Using device applications” on page 358

Command 'Stop All Applications'

**Function:** This command stops all applications that belong to the device configuration, but not the device application itself.

**Call:** Context menu of the device object in the device tree.

**Requirement:** The device has a device application and is in online mode.

See also
- Chapter 1.3.1.16 “Using device applications” on page 358

Other

<table>
<thead>
<tr>
<th>Command</th>
<th>Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.3.1.25.2.23.1</td>
<td>Command 'Add Watch'</td>
</tr>
<tr>
<td>1.3.1.25.2.23.2</td>
<td>Command 'Implement Interfaces'</td>
</tr>
<tr>
<td>1.3.1.25.2.23.3</td>
<td>Command 'Limit Results to Current Declaration'</td>
</tr>
</tbody>
</table>

Command 'Add Watch'

**Symbol:**  

**Function:** This command adds the variable of the current location of the cursor to a watchlist for the purpose of online monitoring.

**Call:** Right-click a variable in an editor when the application is in online mode.

This command adds the variable to the currently opened watchlist. If a watchlist is not open, then the variable is added to the “Watch 1” list and that view opens.

See also
- Chapter 1.3.1.13.1.2 “Using watch lists” on page 330
- Chapter 1.3.1.13.1 “Monitoring of values” on page 323
Command 'Implement Interfaces'

**Function:** This command updates the implemented interfaces for a function block.

**Call:** Context menu of the selected function block (FB) in the device tree.

**Requirement:** The function block implements an interface that you have modified. For example, an additional method was added to the interface.

In object-oriented programming, if you derive a function block (FB) from a base function block, which implements one or more interfaces, for the purpose of inheritance, then the following applies:

When you execute the “Implement Interfaces” command for the derived FB, all interface methods and interface attributes of the base FB are accepted into the derived FB in the form of stubs (without implementation). Then you are responsible for making sure that an "empty" method/attribute in the derived FB does not conflict with an implemented one in the base FB. The following actions are taken to support you in this case: If there es a base implementation for a method/attribute, then CODESYS adds a pragma attribute {error..} in the first line of the affected derived interface method or interface attribute that will generate the error message. If there is no base implementation for the method/attribute, then there is a pragma attribute entry for a warning. After editing the block, you must remove the error pragma attribute entry explicitly.

See also
- § Chapter 1.3.1.25.1.28.4 “Object 'Interface'” on page 783
- § Chapter 1.3.1.9.23.2 “Implementing interfaces” on page 265

Command 'Limit Results to Current Declaration'

**Function:** When multiple declarations have been found, this command collapses the display in the cross-reference list. It shows only the results for the declaration that you selected explicitly in the list.

**Call:** Right-click.

**Requirement:** The cross-reference list is active. Multiple declarations for the searched symbol are listed as cross-references.

See also
- § Chapter 1.3.1.9.14.1 “Using the cross-reference list to find occurrences” on page 232
The dialogs of the CODESYS user interface basically are described on the help pages for the CODESYS menu commands or CODESYS objects. The help book “Dialogs” contains only descriptions of dialogs, which

- appear only after multi-step calls after a certain menu command call or within an object editor,
- or which are not placed on a help page for a command or for an object because of their complexity (multiple subdialogs).

### Dialog 'Import Assistant'

**Function:** The dialog allows for the transfer of CODESYS options and package installations from an older CODESYS installation that was found in the local computer.

**Call:** The dialog opens when a recently installed CODESYS version is started for the first time and an older version is installed on the computer.

<table>
<thead>
<tr>
<th>“Program settings”</th>
<th>□: The user-specific CODESYS options are transferred from the older installation to the new installation.</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Packages”</td>
<td>□: The packages installed with the older CODESYS version are transferred to the Package Manager of the new version. See the list of discovered package installations with the “Name”, “Version”, and “Installation date”.</td>
</tr>
<tr>
<td>“Import”</td>
<td>The program settings and/or options are transferred to the current CODESYS version.</td>
</tr>
<tr>
<td>“Skip”</td>
<td>The program settings and/or options are not transferred to the current CODESYS version.</td>
</tr>
</tbody>
</table>

See also

- % Chapter 1.3.1.1.1 “Setting CODESYS options” on page 102
- % “Package manager” on page 376
Dialog 'Library Reference Conversion'

Function: The dialog defines how references to libraries that are no longer available are to be handled. Note: The undefined library references can be found in the Global Library Manager located in the "POUs" view.

Call: When opening a CoDeSys V2.3 project in V3, the dialog opens when the converter detects a library which cannot be used anymore in the current CODESYS version.

---

Table 175: “What do you want to do?”

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Convert and install the library as well.”</td>
<td>The converter also converts the library file into the new format. It remains referenced in the project. It is installed automatically in the library repository in the &quot;Other&quot; category. If the library does not provide the necessary project information for an installation, then the “Enter Project Information” dialog opens for the information to be added.</td>
</tr>
<tr>
<td>“Use the following library that has already been installed”</td>
<td>The previously used library is replaced by another library. The “Browse” button opens a dialog for selecting from the local library repository.</td>
</tr>
<tr>
<td>“Ignore the library. The reference will not appear in the converted project”</td>
<td>The library reference is removed from the project.</td>
</tr>
</tbody>
</table>

See also
- © Chapter 1.3.1.3.5 “Opening a V2.3 project” on page 128
- © Chapter 1.3.1.25.2.8.2 “Command ‘Library Repository’” on page 951

Dialog 'Select Function Block'

Function: The dialog is used for selecting a function block for I/O mapping. The function block should be mapped to the I/O channel selected in the “<device name> I/O Mapping” tab or to the object selected in the “<device name> IEC Objects” tab.

Call:
- Tab “<device name> I/O Mapping”, command button “Add FB for I/O channel”
- Tab “<device name> IEC Objects”, command button “Add”

The dialog provides all function blocks from the active application and the libraries included in the project which fulfill the following:
- The function block has the \{attribute 'io_function_block'\} attribute.
- The function block contains input or output parameters that match the channel type (input, output, data type) and has the \{attribute 'io_function_block_mapping'\} attribute.

When a function block is selected that provides multiple matching parameters, only the first one is mapped automatically to the channel. The others can only be assigned manually in the “<device name> I/O Mapping” tab.

After the function block is assigned, the parameter of the function block instance is entered in the “Variable” column of the mapping table. Then the path is composed as follows:
Example: App1.Out_4_Int_myScale_Output_Int_1.iOutput for the parameter iOutput of the first inserted instance of the function block myScale_Output.

<table>
<thead>
<tr>
<th>“Find”</th>
<th>Input field for searching for function block names</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Type”</td>
<td>Function blocks in the tree structure that match the channel type. Nodes: application, library name(s)</td>
</tr>
<tr>
<td>“Documentation”</td>
<td>Shows the available documentation for the library selected in the tree or the library block.</td>
</tr>
</tbody>
</table>

See also
- % Chapter 1.3.1.25.1.19.11 “Tab '<device name> I/O Mapping'” on page 755
- % Chapter 1.3.1.24.6.2.19 “Attribute 'io_function_block', 'io_function_block_mapping'” on page 617

Dialog 'Device Conversion'

Function: The dialog defines how references to devices that are no longer available are to be handled.

Call: When opening a CoDeSys V2.3 project in V3, the dialog opens when the converter detects a device reference which cannot be used anymore.

A CoDeSys V2.3 project can be converted into a CODESYS V3 project only if the CODESYS V2.3 Converter package is installed in CODESYS V3. The package is available in the CODESYS Store.

Table 176: “What do you want to do?”

<table>
<thead>
<tr>
<th>“Use the following device that has already been installed”</th>
<th>CODESYS replaces the previously used device in the device tree with another device. The “Browse” button opens a dialog for selecting from the local device repository.</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Ignore the device. All device specific objects will not be available in the new project”</td>
<td>The device entry with all objects inserted below it is removed from the device tree.</td>
</tr>
</tbody>
</table>

| “Remember this mapping for all future occurrences of that device” | The settings made here in the dialog are saved in the CODESYS Options, in the “CODESYS V2.3 Converter” category. As a result, they are also valid for future project conversions. |

See also
- % Chapter 1.3.1.3.5 “Opening a V2.3 project” on page 128
- % Chapter 1.3.1.25.3.13.5 “Dialog 'Options' - 'CoDeSys V2.3 Converter’” on page 1077
- % Chapter 1.3.1.25.2.8.5 “Command 'Device Repository'” on page 957

Dialog 'Breakpoint Properties'

Function: The dialog is used to display or change the properties of the selected breakpoint in the “Breakpoints” view.

Call:
● “Breakpoints” view, “Properties” button
● “Breakpoints” view, “New” button, “New Breakpoint” command or “New Data Breakpoint” command

**Requirement:** An entry is selected in the list of breakpoints.

The dialog is identical to the “New Breakpoint” dialog which is opened in the “Debug” menu by means of the respective commands. Therefore, see the description in the help for the “New Breakpoint” dialog.

See also
● Chapter 1.3.1.25.3.8 “Dialog 'New Breakpoint'” on page 1044

**Dialog 'Permissions'**

**Function:** Definition of the permissions for user groups to perform certain actions on certain objects in the project.

**Call:** Menu “Project ➔ User Management”.

Each modification in the dialog will be applied immediately.

**Actions** Below “Actions” you find all possible actions on objects of the project. The actions are divided in 4 categories and below each action you see the assignments to the current objects. For each assignment action->object you can define the permission for each existing user group.

**Action categories:**

<table>
<thead>
<tr>
<th>Categories</th>
<th>Actions</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Commands”</td>
<td>Actions concerning the execution of commands</td>
</tr>
<tr>
<td>“Users, groups and permissions”</td>
<td>Actions concerning the configuration of user accounts, user groups and their permissions</td>
</tr>
<tr>
<td>“Object types”</td>
<td>Actions concerning the creation of object types</td>
</tr>
<tr>
<td>“Project objects”</td>
<td>Actions concerning the viewing, the editing, the removing and the child object handling of project objects</td>
</tr>
</tbody>
</table>

**Actions in detail:**

<table>
<thead>
<tr>
<th>Action</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>“execute”</td>
<td>Executing a menu command</td>
</tr>
<tr>
<td>“create”</td>
<td>Creating a new object in the project</td>
</tr>
<tr>
<td>“add or remove children”</td>
<td>Adding or removing a child object below an existing object</td>
</tr>
<tr>
<td>“modify”</td>
<td>Modifying an object in the editor; modifying the users, groups or permission settings in the respective editor/dialog</td>
</tr>
<tr>
<td>“remove”</td>
<td>Deleting or cutting an object</td>
</tr>
<tr>
<td>“view”</td>
<td>Viewing the object in its editor</td>
</tr>
</tbody>
</table>

**Permissions** Below “Permissions” you find a list of all currently defined user groups (except group "Owner") and a symbol bar for configuring the permissions for a group.
The actions currently selected in the “Actions” window, on the currently selected targets are granted for the currently selected group.

The actions currently selected in the “Actions” window, on the currently selected targets are denied for the currently selected group.

The permission to perform the actions currently selected in the “Actions” window, on the currently selected targets, is not granted explicitly, however the actions are granted by default. For example because the respective permission is assigned to the "parent object". Example: The group has got the respective permission for object "mylec", and for this reason it has this permission also for object "mylec.pb_1".

The action currently expanded in the “Actions” window, on the currently selected targets, has not been denied explicitly, however the actions are denied by default. For example because the respective permission is assigned to the "parent object".

No symbol: Currently in the “Actions” window several actions are selected, for which the group has got different permissions.

Symbol bar:

**“Grant”**
The currently selected action on the currently selected target is explicitly granted for the currently selected group.

**“Deny”**
The currently selected action on the currently selected target is explicitly denied for the currently selected group.

**“Clear”**
The permission to perform the currently selected action on the currently selected target will be set back to the default for the currently selected group.

See also

- § Chapter 1.3.1.6.5 “Protecting objects in the project by access rights” on page 146

**Dialog Box 'Prepare Value’**

**Function:** This dialog box is used for preparing a value for a forced variable. CODESYS executes the prepared action with the next forcing.

CODESYS opens the dialog box in the following situations:

- When clicking in the field “Prepared value” of a forced variable in the declaration section
- When clicking in the inline monitoring field of a forced variable
- When clicking in the field “Prepared value” of a forced variable in the monitoring window

<table>
<thead>
<tr>
<th>“Prepare a new value for the next write or force operation”</th>
<th>Value that CODESYS writes to the variable with the next force operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Remove a preparation with a value”</td>
<td>CODESYS deletes the prepared value.</td>
</tr>
<tr>
<td>“Release the force, without modifying the value”</td>
<td>CODESYS retains the forced value and ends forcing. CODESYS marks the variable &lt;Unforce&gt;.</td>
</tr>
<tr>
<td>“Release the force and restore the variable to the value it had before forcing it”</td>
<td>CODESYS resets the forced value and ends forcing. The variable is marked with &lt;Unforce and restore&gt;.</td>
</tr>
</tbody>
</table>

See also

- § Chapter 1.3.1.25.2.7.16 “Command ‘Force Values’” on page 943
**Dialog 'New Breakpoint'**

**Function:** In the dialog, you define the settings for a new breakpoint or data breakpoint. It is identical to the "Breakpoint Properties" dialog which is used in the "Breakpoints" view.

**Call:**
- "Debug ➔ New Breakpoint"
- "Debug ➔ New Data Breakpoint"

**Requirement:** The application is in online mode.

**Tab 'Condition'**
The dialog defines the requirements for which program processing should halt at a breakpoint.

!NOTICE!
Using conditional breakpoints slows down code execution, even when the condition does not yield TRUE.

Conditional breakpoints required a CODESYS runtime >= V3.5.4.0.

---

**Table 177: "Tasks"**

<table>
<thead>
<tr>
<th>&quot;Only break if the breakpoint is hit in one of the following tasks&quot;</th>
</tr>
</thead>
<tbody>
<tr>
<td>CODESYS evaluates the breakpoint only if it is reached by specific tasks. The required tasks must be activated.</td>
</tr>
<tr>
<td>For example, you can define a single debug task and also prevent other tasks that use the same block from being affected when debugging.</td>
</tr>
</tbody>
</table>

---

**Table 178: "Hit Count"**

<table>
<thead>
<tr>
<th>&quot;Hit Count&quot;</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Break always&quot;: The program always halts at this breakpoint.</td>
</tr>
<tr>
<td>Alternative: The program halts at the breakpoint when the breakpoint has been hit as often as defined in the following (type in the required hit count or select it from the number list):</td>
</tr>
<tr>
<td>- &quot;Break when the hit count is equal to&quot;</td>
</tr>
<tr>
<td>- &quot;Break when the hit count is a multiple of&quot;</td>
</tr>
<tr>
<td>- &quot;Break when the hit count is greater than or equal to&quot;</td>
</tr>
</tbody>
</table>

---

**Table 179: "Condition"**

<table>
<thead>
<tr>
<th>&quot;Break, if true&quot;</th>
</tr>
</thead>
<tbody>
<tr>
<td>CODESYS evaluates the specified condition and halts the program at the breakpoint only when the result yields TRUE. You can define a condition as a valid Boolean expression. Examples: x&gt;100, x[y]=z, a AND b, boolVar.</td>
</tr>
</tbody>
</table>

---

**Tab 'Data'**

Requirement: This is used for the properties of a data breakpoint.
On the tab, the variable or memory address is specified for which the data breakpoint is set or will be set.

| “Break execution when the value of the variable or address changes” | ● Input of a qualified variable name  
| | ● Selection of a variable in the “Input Assistant” dialog, in the “Watch Variables” category  
| Examples: Variable: PLC_PRG.fb_DoSth.dwVariable, address: 16#12A, 0x12A, 129 |

| “Size” | Number of bytes of the specified variable or memory address above which should be monitored for changes. When a new variable or memory address is specified, a value that matches the data type or memory is set automatically at first. |
| Example: 4 for data type DWORD |

Tab 'Execution Point Settings'  Here, an existing breakpoint or data breakpoint can be converted into an execution point.

| “Execution point (Execution does not stop at breakpoint)” | [✓]: The breakpoint becomes an execution point. Processing does not halt at this point and the given code is executed.  
| | ● Execution point of a breakpoint: activated: [✓]; deactivated:  
| | ● Execution point of a data breakpoint: activated: [✓]; deactivated: |

| “Execute the following code” | Code that is executed when the execution point is reached.  
| Looping structures (For, While) and IF or CASE expressions are not possible. |

| “Print a message in the device log” | This option is available only when you select the “Enable logging in breakpoints” option in “Project Settings ➔ Compile Options”.  
| CODESYS can print variables with the placeholder {variable name} in the message text. |

Tab 'Location'  Requirement: The command “New breakpoint” was selected.

| “POU” | POU of the active application where the breakpoint is placed. |
| “Position” | Position of the breakpoint in the POU. Entry as row and column numbers (text editor) or as network or element numbers. |

| “Instances” | In the case of function blocks, you must define whether the breakpoint should be set in the implementation or in an instance.  
| [✓] CODESYS sets the breakpoint in the instance. For this option, select “Instance Path”.  
| [ ] CODESYS sets the breakpoint in the implementation. |

| “Enable breakpoint immediately” | [✓]: The breakpoint is activated.  
| [ ]: The breakpoint is not activated. To activate later, click the button in the “Breakpoints” view. |

See also  
- [Chapter 1.3.1.25.3.5 “Dialog ‘Breakpoint Properties’” on page 1041](#)  
- [Chapter 1.3.1.12.2 “Using breakpoints” on page 310](#)
Dialog 'Monitoring Range'

Function: This dialog restricts the range of array elements whose values are displayed during monitoring.

Call: Click in the column field “Data Type” that belongs to the array variable.

Requirement: A POU is in online mode and is being monitored. In addition, a variable of the POU has the data type “ARRAY”.

| “Valid range” | The validity range of the array elements that are monitored. Example of a three-dimensional array: [1..10][-3..3][-10..10] |
| “Maximum number of array elements” | Number of elements of the array variables Example: 1470 |

When you edit one of the settings “Start”, “End”, or “Scroll range of 1000 elements”, both of the other settings are adapted automatically.

“Start” | Index of the first array element whose value is displayed.
“End” | Index of the last array element whose value is displayed.
“Scroll range of 1000 elements” | Scrollbar for selecting a range from the set of array elements.

See also
- ☀ Chapter 1.3.1.13.1.1 “Calling of monitoring in programming objects” on page 323

Dialog 'Properties'

1.3.1.25.3.10.1 Dialog Box 'Properties' - 'Common'................................. 1047
1.3.1.25.3.10.2 Dialog 'Properties' - 'Boot Application'........................... 1047
1.3.1.25.3.10.3 Dialog 'Properties' - 'Encryption'................................. 1047
1.3.1.25.3.10.4 Dialog 'Properties' - 'Build'........................................ 1048
1.3.1.25.3.10.5 Dialog 'Properties' - 'Build' (C-integration)................... 1049
1.3.1.25.3.10.6 Dialog 'Properties' - 'Access Control'........................... 1050
1.3.1.25.3.10.7 Dialog 'Properties' - 'External file'.............................. 1050
1.3.1.25.3.10.8 Dialog Box 'Properties' - 'Bitmap'............................... 1051
1.3.1.25.3.10.9 Dialog 'Properties' - 'Application Build Options'.............. 1051
1.3.1.25.3.10.10 Dialog 'Properties' - 'Network Variables'.................... 1052
1.3.1.25.3.10.11 Dialog 'Properties' - 'Network Settings'...................... 1053
1.3.1.25.3.10.12 Dialog 'Properties' - 'CFC Execution Order'................ 1054
1.3.1.25.3.10.13 Dialog 'Properties' - 'SFC Settings'............................ 1055
1.3.1.25.3.10.14 Dialog 'Properties' - 'Link to File'............................ 1055
1.3.1.25.3.10.15 Dialog 'Properties' - 'Cam'..................................... 1055
1.3.1.25.3.10.16 Dialog 'Properties' - 'Image Pool'............................. 1057
1.3.1.25.3.10.17 Dialog 'Properties' - 'TextList'.................................. 1057
1.3.1.25.3.10.18 Dialog 'Properties' - 'Options'.................................. 1058
1.3.1.25.3.10.19 Dialog 'Properties' - 'Monitoring'.............................. 1058

This dialog box is for the configuration of the properties of an object in CODESYS. In addition, depending on the object, it contains different tabs that each handle a category of properties.

Call: Menu “View”, context menu of the object in the “Devices”, “POUs” or “Modules” view.
Dialog Box 'Properties' - 'Common'

**Function:** This dialog box shows common information about the selected object.

**Call:** Main menu “View ➔ Properties”, or context menu of the object (“Common”).

**Requirement:** An object is selected in the device tree or POUs view.

| “Name” | Object name as shown in the device tree or POUs view |
| “Object type” | Type of object (for example, POU, application, or interface) |
| “Open with” | Type of editor to display or edit the object |

Dialog 'Properties' - 'Boot Application'

**Function:** The settings in this tab define when and how a boot application is created from the application.

**Requirement:** The device supports the settings.

**Call:** Right-click application object and select “Properties”; main menu “View ➔ Properties” (“Boot Application” tab).

| “Create implicit boot application on download” | A boot application is created automatically when downloading the application. |
| “Create implicit boot application on Online Change” | A boot application is created automatically when for an online change. |
| “Remind boot application on project close” | Before closing the project, CODESYS prompts to create the boot application. |

Regardless of the presets defined here, you are always able to create a boot application explicitly when you log in.

See also

- ☰ Chapter 1.3.1.11.6 “Generating boot applications” on page 305
- ☰ Chapter 1.3.1.25.2.6.6 “Command 'Online Change’” on page 928

Dialog 'Properties' - 'Encryption'

**Function:** The dialog contains the properties of the application for encryption. If the CODESYS Security Agent is installed, then you can start a wizard for the encryption of downloads, online changes, and boot applications.

**Call:**

- Menu bar: “View ➔ Properties”
- Context menu of an application object
Table 182: “Encryption Technology”

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>“No Encryption”</td>
<td>You can download the boot application to the controller only when the defined dongle (license key) is connected to the computer. The dongle is provided by 3S-Smart Software Solutions GmbH or by the respective hardware manufacturer. The firmcode is displayed. Type in the delivered product code.</td>
</tr>
<tr>
<td>“Simple Encryption”</td>
<td>You can download the boot application to the controller only after you have specified the product code and firmcode, and the respective dongle is connected to both the development computer and the controller. You receive the codes from the vendor that manages the licenses.</td>
</tr>
<tr>
<td>“Encryption with license management”</td>
<td>You can download the boot application to the controller only when a valid certificate exists for it. The “Certificates” group is enabled. See the description below.</td>
</tr>
<tr>
<td>“Encryption with certificates”</td>
<td>The option is already selected if the “Enforce signing of downloads, online changes and boot applications” option is selected on the “User” tab of the “Security Screen” view.</td>
</tr>
</tbody>
</table>

Table 183: “Certificates”

Note: If the “Enforce encryption of downloads, online changes and boot applications” option is selected in the “Security Screen” view in the “Security level” group, then the encryption technology is set to “Encryption with certificates” and cannot be changed in the “Properties” dialog.

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Certificate Selection”</td>
<td>The “Certificate Selection” dialog opens. Here you can select previously installed certificates of devices for which the encryption of download, online change, and boot application is enabled. The list can contain several entries if several devices are authorized to run this application.</td>
</tr>
<tr>
<td>“Digitally sign application code”</td>
<td>The application is signed with a digital signature. The certificate for the digital signature is specified in the “Security Screen” view on the “User” tab.</td>
</tr>
</tbody>
</table>

Area for the display of the selected certificates with corresponding information

Information per certificate:
- “Issued for”
- “Issued by”
- “Valid from”
- “Valid until”
- “Thumbprint”

“Encryption Wizard” This button is available only if the CODESYS Security Agent is installed. It starts the wizard with the same name. See the help for CODESYS Security Agent in this case.

See also
- Chapter 1.3.1.9.18 “Encrypting an application” on page 242
- Chapter 1.3.1.21 “Managing packages and licenses” on page 375
- Chapter 1.3.1.25.2.3.18 “Command ‘Security Screen’” on page 891
- Help about CODESYS Security Agent

Dialog 'Properties' - 'Build'

Symbol: 📋

Function: This dialog includes options for compiling the object.

Call: Menu bar: “View ➤ Properties”; context menu of the object in the device tree.
**Name** | **Description**
---|---
"Exclude from build" | ☑: This object and inherently its child objects are not considered for the next compile process. The object entry is displayed in green fonts in the “Devices” or “POUs” view.

"External implementation"  
"(Late link in the runtime system) " | ☑: CODESYS does not generate any code for this object when compiling the project. The object is linked as soon as the project is running on the target system, provided it is available there (for example, in a library). The object name is postfixed with (EXT) in “Devices” or “POUs” view.

"Enable system call" | ☑: A system call (runtime system) for functions is possible. Background: As opposed to CoDeSys V2.3, the ADR operator in V3 can be used with function names, program names, function block names, and method names. It replaces the INSTANCE_OF operator. BUT: It is not possible to call function pointers from within CODESYS.

"Link Always" | ☑: The object is marked by the compiler and therefore always included in the compile information. This means that it is always compiled and downloaded to the PLC.

"Compiler defines" | Here you can specify defines or conditions for compiling the object (conditional compile). You can also type in the expression expr, which is used in these kinds of pragmas. Multiple entries are possible as a comma-separated list (see {define} statements). Example: hello, test:='1'

"Additional compiler definitions from the device description"  
"Defined in device" | List of compiler definitions that originate from the device description. These compiler definitions are used in the build if they are not listed in the “Ignored definitions” field.

"Ignored definitions" | List of compiler definitions from the device description that are not used in the build.

> | Copies the selected compiler definition from the “Defined in device” field to the “Ignored definitions” field.

< | Moves the selected compiler definition from the “Ignored definitions” field to the “Defined in device” field. The compiler definition is used in the build.

See also

- Chapter 1.3.1.24.6.3 “Conditional pragmas” on page 637

**Dialog 'Properties' – 'Build' (C-integration)**

**Function:** In this dialog, you configure the build environment and the necessary data for the integration of the C development environment.

**Call:** Main menu “View”, context menu of the object “C Code Module”

**Requirement:** The object “C Code Module” is selected in the device tree.

**NOTICE!**

The dialog in this form is valid only for CODESYS Control Win V3 and Visual Studio. For other environments, the dialog can look different or may not even be available at all.
### Visual Studio location
Installation path of Visual Studio on the hard disk
You can also select the path with the input assistant or search for it with the magnifying glass.

### Windows SDK location
Installation path of Windows SDK on the hard disk
You can also select the path with the input assistant or search for it with the magnifying glass.

### Temporary build folder Location
Path on the hard disk for the temporary build files

See also
- § Chapter 1.3.1.25.1.16 “Object 'C Code Module’” on page 737
- § Chapter 1.3.1.9.11 “Integrating C Modules’” on page 222

### Dialog 'Properties' - 'Access Control'

**Function:** The dialog defines the access rights of user groups for objects.  
**Call:** Main menu “View ➨ Properties”, context menu of an object in the view “Device” or “POUs”.

**Requirement:** An object is selected in the view “Device” or in the view “POUs”.

### Groups, actions and permissions
A table which displays the following user groups access rights on objects:
- “View”
- “Modify”
- “Remove”
- “Add/remove children”

Perform a double click on the access right symbol to open the drop down list with the available rights.

See also
- § Chapter 1.3.1.6.5 “Protecting objects in the project by access rights” on page 146
- § Chapter 1.3.1.25.3.6 “Dialog 'Permissions’” on page 1042

### Dialog 'Properties' - 'External file'

**Function:** The dialog is used to view and edit the properties of the external file. The properties were defined when the object was created. Changed properties are saved by pressing the “OK” button.

**Call:** “View ➨ Properties”, context menu of the object

**Requirement:** The object of the external file is selected in the “Devices” view or “POUs” view.

### What do you want to do with the external file?

<table>
<thead>
<tr>
<th>Remember the link</th>
<th>The file is available in the project only as long as it exists in the defined file path.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Remember the link and embed into project</td>
<td>CODESYS saves an internal copy of the file in the project, as well as the link to the defined file path. The update option selected below applies as long as the external file exists there. Otherwise CODESYS uses the version saved in the project.</td>
</tr>
<tr>
<td>Embed into project</td>
<td>CODESYS saves only one copy of the file in the project. There is no longer a link to the external file.</td>
</tr>
</tbody>
</table>
Table 184: “When the external file changes, then”

<table>
<thead>
<tr>
<th>Action</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>“reload the file automatically”</td>
<td>If the external file changes, then CODESYS updates the file in the project.</td>
</tr>
<tr>
<td>“prompt whether to reload the file”</td>
<td>If the external file changes, then a dialog prompt opens whether CODESYS should also update the file in the project.</td>
</tr>
<tr>
<td>“do nothing”</td>
<td>The file remains unchanged in the project, even if the external file changes.</td>
</tr>
</tbody>
</table>

Table 185: “Linked file”

- Requirement: Either the “Remember the link” option or “Remember the link and embed into project” option is selected.
- The following information about the linked file is shown: “Name”, “Location”, “Size”, “Changed”.
- “Display file properties” Clicking this button opens the default “Properties of <file name>” dialog, which you can also open in the Windows file system by right-clicking the file.

Table 186: “Embedded file”

- Shows the “Size” and “Changed” properties of the embedded file.

See also
- § Chapter 1.3.1.25.1.15 “Object 'External File’” on page 736

Dialog Box 'Properties' - 'Bitmap'

- **Function**: The dialog is for assigning a bitmap file to the object. The image will be used in the graphic view of the Library Manager and in the Toolbox view of the FBD/LD/IL editor.
- **Call**: Main menu “View ➔ Properties”, context menu of the object
- **Requirement**: The object is selected in the view “Devices” or in the view “POUs”

<table>
<thead>
<tr>
<th>Setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Render pixels of this color transparently: ”</td>
<td>The selected color will be displayed transparently.</td>
</tr>
</tbody>
</table>

Dialog 'Properties' - 'Application Build Options'

- **Function**: This dialog includes settings that CODESYS uses for creating a boot application for the controller.
- **Call**: Menu bar: “View ➔ Properties”, context menu of an application object.
**Download Application Info**

This feature requires compiler version >=3.5.0.0, runtime system version >= 3.5.0.0.

The information about the application contents is also downloaded to the PLC. We recommend that you keep this option activated because it enables a difference check between the current application and the application on the PLC. This compares the number of blocks, data, and memory locations.

To receive information about the differences, click “Details” in the “Applications” tab of the device editor. This is also in the message view that opens when you are downloading an application to the PLC when it is different from the one already on the PLC.

**Stop parent application in case of exception**

Available for applications with a parent application. The option is activated by default for device applications.

[✓] An exception in this application also leads to stopping the device application and therefore stopping all other applications.

**Dynamic memory settings**

Memory is allocated dynamically for the application, for example when using the __NEW operator. In this case, define the "Maximum size of memory (bytes)".

Caution: The entire memory is not available for creating objects dynamically. Instead, the system always uses part of it for management information.

See also

- ☰ Chapter 1.3.1.11.4 “Generating application code” on page 303
- ☰ Chapter 1.3.1.11.6 “Generating boot applications” on page 305
- ☰ Chapter 1.3.1.25.1.1 “Object ‘Application’” on page 710
- ☰ Chapter 1.3.1.16 “Using device applications” on page 358

**Dialog 'Properties' - 'Network Variables'**

**Symbol:** [ ]

**Function:** In this dialog, you define network properties for the variable list that is selected in the device tree. Furthermore, any variables in it that are declared as network variables are also available.

**Call:** “Context menu of variable list in device tree ➔ Properties”, “Network Variables” tab

<table>
<thead>
<tr>
<th><strong>“Network type”</strong></th>
<th>UDP</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>“Task”</strong></td>
<td>Task of the current application that controls the variables to be sent. CODESYS always sends the variables at the end of a task cycle.</td>
</tr>
<tr>
<td><strong>“List identifier”</strong></td>
<td>Used to identify the network variable list. Must be unique</td>
</tr>
<tr>
<td><strong>“Pack variables”</strong></td>
<td>The size of the packages (telegrams) that are transmitted depends on the network type. In the case of “UDP”, a package is 256 bytes.</td>
</tr>
<tr>
<td>[✓] CODESYS bundles the variables for sending in packages in order to reduce as much as possible the number of packages to send. In the case of variables of type array or structured data types, this can lead to the splitting of the variables into multiple telegrams. As a result, data inconsistencies are possible within these variables, even if the variable size is smaller than the package size.</td>
<td></td>
</tr>
<tr>
<td>[☐] CODESYS generates one package per variable.</td>
<td></td>
</tr>
</tbody>
</table>
“Transmit checksum”
- A checksum is provided for each variable package. The receiver checks the checksum to make sure that the variable definitions match from the sender and receiver. A package with non-matching checksums is not accepted.

“Acknowledgement”
- CODESYS sends an acknowledgement message for each received data package. If the sender does not receive a confirmation before it sends again, then an error is written to the diagnostic structure.

Note: For the NetVarUdp library version 3.5.7.0 and later, a receiver channel is no longer assigned when confirmed transfer is not selected. In this way, network variable exchange is also possible between two controllers on one hardware device.

“Cyclic transmission”, “Interval”
- CODESYS sends the variables within the defined interval. Example for time definition: "T#70ms".

“Transmit on change”, “Minimum gap”
- CODESYS sends the variables only if their values have changed. You can use "minimum gap" to define the least amount of time between two transmissions.

“Transmit on event”, “Variable”
- CODESYS sends the variables as soon as the defined variable yields \texttt{TRUE}.

“Settings”
- Protocol-specific settings; possible entries depend on the network library.
  - “Port”: Number of the port that CODESYS uses for data exchange with other network units. The "Default value" is "1202". You can change the current value in the "Value" field at any time. Select the field, press the \texttt{[Space Bar]}, and type the value.
  - Caution: The other nodes in the network must define the same port. If more than one UDP connection is defined in the project, then the port numbers in all configurations are adapted to this value.
  - “Broadcast Adr.”: The "Default value" is 255.255.255.255, which means that data exchange will take place with all network units. You can change the current value in the "Value": select the field, press the \texttt{[space bar]}, and type the address or address range of a subnet-work (for example, 197.200.100.x when communication should be with all nodes that have an IP address in the range 197.200.100.x.

See also
- Chapter 1.3.1.10.3 “Network variables” on page 277
- Chapter 1.3.1.10.3.1 “Configuring a network variable exchange” on page 278
- Chapter 1.3.1.25.1.27 “Object ‘Network Variable List (Receiver)’” on page 775
- Chapter 1.3.1.25.1.26 “Object ‘Network Variable List (Sender)’” on page 774

Dialog 'Properties' - 'Network Settings'
If the device supports the network functionality, then the current network settings for a GNVL (global network variable list) can be displayed and changed in the ‘Properties’ dialog of the object. These are the settings that were used when adding the GNVL in the “Add Network Variable List (Receiver)” dialog.

See also
- Dialog Box ‘Add Network Variable List (Receiver)” on page 775
- Chapter 1.3.1.10.3.1 “Configuring a network variable exchange” on page 278
Dialog 'Properties' - 'CFC Execution Order'

**Function:** The tab switches the mode of the execution order for CFC objects.

**Call:** Context menu: “Properties” of a CFC object in the “Devices” view or “POUs” view

### Tab 'CFC Execution Order'

<table>
<thead>
<tr>
<th><strong>“Execution order”</strong></th>
<th>In the CFC editor, you position the elements and therefore also the networks freely. Two modes are available to prevent the execution order in the CFC POU from being undefined.</th>
</tr>
</thead>
</table>
| **“Auto Data Flow Mode”** | In this mode, the execution order is determined automatically by data flow, or in case of ambiguity, by network topology. The POUs and the outputs are numbered internally. The networks are executed from top to bottom and left to right. Advantage: The automatically defined execution order is optimized by time and by cycle. You do not need any information about the internally managed execution order during the development process. The following commands are provided afterwards in the “CFC Execution Order” menu:  
  ● “Display Execution Order”  
  ● “Set Start of Feedback”  
  The elements in the CFC editor are displayed without markers and without numbering. It is not possible to change the execution order manually. For networks with feedback, you can also set a starting point. |
| **“Explicit Execution Order Mode”** | In this mode, you can define the execution order explicitly. To do this, the elements are displayed in the CFC editor with markers and numbering, and menu commands are provided for defining the order. The following commands are provided in the “CFC Execution Order” menu:  
  ● “Send to Front”  
  ● “Send to Back”  
  ● “Move Up”  
  ● “Move Down”  
  ● “Set Execution Order”  
  ● “Order by Data Flow”  
  ● “Order by Topology”  
  Note: Up to CODESYS V3.5 SP1, this was the usual behavior of CFC POUs. Pay attention that it is your responsibility to adapt the execution order and assess the consequences and impacts. This is another reason why the execution order is always displayed. |
| **“Apply to All CFCs”** | Changes the mode for all other CFC objects in the project to the mode selected in the list |

See also
- § Chapter 1.3.1.9.3.2.1 “Automatic execution order by data flow” on page 189
- § Chapter 1.3.1.25.2.12.10 “Command ‘Set Start of Feedback’” on page 980
- § Chapter 1.3.1.25.2.12.11 “Command ‘Send to Front’” on page 981
- § Chapter 1.3.1.25.2.12.12 “Command ‘Send to Back’” on page 981
- § Chapter 1.3.1.25.2.12.13 “Command ‘Move Up’” on page 982
- § Chapter 1.3.1.25.2.12.14 “Command ‘Move Down’” on page 982
- § Chapter 1.3.1.25.2.12.16 “Command ‘Order by Data Flow’” on page 983
- § Chapter 1.3.1.25.2.12.17 “Command ‘Order by Topology’” on page 983
Dialog 'Properties' - 'SFC Settings'

**Function:** The dialog defines the default settings for all POUs used in the project, which are programmed in SFC.

**Call:** Main menu “View ➔ Properties”, context menu of a SFC POU in the view “Device” or “POUs”.

**Flag**

<table>
<thead>
<tr>
<th>List of all possible SFC flags</th>
<th>“Use”✓: The SFC flag is activated and will be considered in the program execution.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>“Declare”✓: The SFC flag is declared automatically.</td>
</tr>
<tr>
<td></td>
<td>If “Declare” is activated, but “Use” is not activated, the variable will be declared but the flag has no effect in the program execution.</td>
</tr>
<tr>
<td></td>
<td>Hint: If you have manually declared a SFC variable you have to disable the declaration of this flag in the “SFC Settings”. Otherwise the automatically generated flag will overwrite the manually declared flag.</td>
</tr>
<tr>
<td></td>
<td>Hint: A automatically declared flag variable is only visible in the online mode in the declaration part of the SFC editor.</td>
</tr>
</tbody>
</table>

“Use defaults”✓: The settings of this dialog overwrites the “SFC settings” of the single POUs.

See also
- ⇧ Chapter 1.3.1.24.1.4.6 “SFC flags” on page 447

**Build**

**Table 187: “Code generation”**

| “Calculate active transitions only” | ✓: CODESYS generates code for the currently active transition only. |

Dialog 'Properties' - 'Link to File'

**Function:** The dialog defines the link of an external file with the content of the global variable list (GVL). You can either export the GVL to an external file or import it from an external file.

**Call:** Main menu “View ➔ Properties”, context menu of a GVL.

| “Import before compile” | ✓: With each compile of the project (for example with F11) CODESYS saves a file with extension gvl in the directory defined in “File name”. |
| “Export before compile” | ✓: An existing export file is read automatically with each project compile. Thus you can import a GVL exported from a different project, for example to setup a communication via network variables. |

See also
- ⇧ Chapter 1.3.1.25.1.21 “Object ‘GVL’ - Global Variable List” on page 769
- ⇧ Chapter 1.3.1.10.3.1 “Configuring a network variable exchange” on page 278

Dialog 'Properties' - 'Cam'

**Function:** Use this dialog to define the global variables of the cam.
### Table 188: “Dimensions”

<table>
<thead>
<tr>
<th>“Master start/end position”</th>
<th>The start and end positions of the master define the range of the master values and therefore the scale of the horizontal axis of the cam. The default settings are given in angular degrees with 0 and 360 as limiting values.</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Slave start/end position”</td>
<td>The associated slave positions are determined by the graph type that is defined for the cam. However, the segment depicted by the curves (this is also the scale of the vertical axis) can be defined by the slave start and end positions given here.</td>
</tr>
</tbody>
</table>

### Table 189: “Period”

| “Smooth transition” | The values for position, velocity, and acceleration are adjusted automatically. |
| “Slave period” | Indicates when the slave period is repeated mechanically. The slave position at the start and end of the master period may then be in an interval of a whole number multiple of this value. This value is effective only if the “Smooth transition” check box is selected. |

### Table 190: “Continuity Requirements”

Activation of these options for the continuity of the curve does not have any effect when editing the cam. It does, however, prompt a continuity check, which reports any violations to the message view (CAM). It is not possible to edit jumps in the position curve. The default setting also requires the continuity of velocity and acceleration. You can clear these options, for example in the special case of a curve that consists of only linear segments. However, this will lead to kinks in the position curve. By default, the jerk (3rd derivative) is not tested for jumps.

| “Position” | |
| “Velocity” | ✓: The entire curve is tested for jumps. |
| “Acceleration” | |
| “Jerk” | |

### Table 191: “Compile Format”

When compiling, `MC_CAM_REF` structure variables are generated. A cam is described according to the following options:

| “polynomial (XYVA)” | Polynomial description of the individual points consisting of the master position, slave position, slave velocity, and slave acceleration. |
| “one dimensional point array” | 1D table of slave positions |
| “two dimensional point array” | 2D table of composite master/slave positions |
| “Elements” | Number of elements in the arrays. This array has already been created in SM3 Basic for the default cases “128” and “256”. If you type in another value, you must create the structure in your application (see the following example). |
Example of an array with 720 elements

```plaintext
TYPE SMC_CAMTable_LREAL_720_2 : STRUCT
  Table: ARRAY[0..719] OF ARRAY[0..1] OF LREAL;
  fEditorMasterMin, fEditorMasterMax: REAL;
  fEditorSlaveMin, fEditorSlaveMax: REAL;
  fTableMasterMin, fTableMasterMax: REAL;
  fTableSlaveMin, fTableSlaveMax: REAL;
END_STRUCT
END_TYPE
```

Dialog 'Properties' - 'Image Pool'

**Function:** The dialog allows for setting the basic properties of the selected image pool.

**Call:** “View ➔ Properties” of an “Image Pool” object type; context menu of an “Image Pool” object type.

| “Download only used images” | Yes: Instead of loading all images from the image pool, CODESYS loads only the images that are actually used in the application on the PLC. |
| "Download by visualization" | Yes: The image pool is downloaded with the visualization to the controller. |
| “Internal” | Yes: CODESYS does not provide the image pool in the “ToolBox” view. You cannot drag these images to the visualization. |

**Table 192: “Symbol library settings”**

| “Mark library as symbol library” | Marks the image pool as a symbol library for use in a visualization. The symbol library receives the key VisuSymbolLibrary = TRUE as file property in the project information. The VisuElements library is inserted automatically as a placeholder library in the “POUs” pool of the Library Manager. |
| “Internal” | Requirement: A library project is open. CODESYS displays symbol libraries that are installed in the repository in the “Project Settings” (“Visualization” category, “Symbol Libraries” tab). |
| “Textlist for symbol translation” | Select the text list from the drop-down list that contains the translated texts for the image pool. |

See also

- % Chapter 1.3.1.25.1.24 “Object ‘Image Pool’” on page 770
- Displaying and Marking Visualization Elements
- % Chapter 1.3.1.9.10 “Using image pools” on page 221
- Using Text Lists
- Project Settings - Visualization % Chapter 1.3.1.25.3.11.9 “Dialog ‘Project settings’ - ‘Visualization’” on page 1068

Dialog 'Properties' - 'TextList'

**Function:** The dialog allows for setting the basic properties of the selected text list.

**Call:** “View ➔ Properties” of an “Text List” object type; context menu of an “Image List” object type.
"Download by visualization"

The text list is downloaded with the visualization to the controller.

"Internal"

The text list can be used only in a library. It is not available in an ordinary CODESYS project.

See also

- Chapter 1.3.1.25.1.35 “Object 'TextList’” on page 823
- Using Text Lists

Dialog 'Properties' - 'Options'

Options (Controller)

Function: This dialog provides the settings for monitoring an login for objects of type device. The availability of the options depends on the device description.

Call: Context menu of the device, or main menu "View ➔ Properties", if the device is selected.

| "Monitoring interval (ms)" | Interval of the monitoring (10 ms - 1000 ms) |

Table 193: “Interactive Login Mode”

This mode is used to prevent an accidentally login to a different controller.

| "None" | No interaction with the user during login. Corresponds to the behavior of previous versions. |
| "Enter ID" | During login CODESYS asks to enter an ID. The ID is stored in the controller. Without a valid ID no login is possible. When login a second time, CODESYS does not ask again for the ID if the computer name, the user name, the device name and the device address have not changed. The information is saved in the project options. |
| "Press key" | During login a dialog prompts and requests the user to press a key on the controller. The timeout for this action is saved in the device description. |
| "Wink (= blink an LED)" | During login a led blinks on the connected controller. |

Table 194: “Symbol Configuration”

| "Access variables in sync with IEC tasks" | Default setting, consistent access is not permitted |
| | Consistent access is permitted |

The setting only will take effect when all applications and boot applications are re-downloaded to the controller.

Note: If the option is activated, then the jitter for all IEC applications may increase on this device! The consistent access can disturb the real-time capability.

Siehe also

- Chapter 1.3.1.8.1 “Device tree and device editor” on page 155
- Chapter 1.3.1.25.1.19 “Object 'Device' and generic device editor” on page 740
- Chapter 1.3.1.25.1.36 “Object 'Symbol Configuration'” on page 824

Dialog 'Properties' - 'Monitoring'

Function: The tab contains options for the monitoring of transitions in SFC.

Call: Select transition object, click “Properties”; menu bar: “View ➔ Properties”.

Table 192: "Function"
**“Enable monitoring”**

☑️ An implicit variable is created for the transition, which is then always given the current property value when the application calls the Transition method. The value stored last in this variable is displayed in the monitoring.

**“Monitoring using call”**

☑️ The transition to be monitored is read by directly calling the transition.

Note: When you activate this option, you have to consider possible side effects. These kinds of side effects can occur if additional operations are implemented in the transition.

See also

- Chapter 1.3.1.25.1.28.10 “Object 'Transition'” on page 796
- Chapter 1.3.1.24.6.2.21 “Attribute ‘monitoring’” on page 619

---

**Dialog 'Project Settings'**

| 1.3.1.25.3.11.1 | Dialog ‘Project Settings’ - 'SFC' | 1059 |
| 1.3.1.25.3.11.2 | Dialog Box 'Project Settings' - 'Users and Groups' | 1060 |
| 1.3.1.25.3.11.3 | Dialog Box 'Project Settings' - ‘Compileoptions’ | 1062 |
| 1.3.1.25.3.11.4 | Dialog Box 'Project Settings' - 'Compiler Warnings' | 1062 |
| 1.3.1.25.3.11.5 | Dialog 'Project Settings' – 'Source Download' | 1063 |
| 1.3.1.25.3.11.6 | Dialog 'Project Settings' - 'Page Setup' | 1064 |
| 1.3.1.25.3.11.7 | Dialog 'Project Settings' - 'Security' | 1064 |
| 1.3.1.25.3.11.8 | Dialog 'Project Settings' - 'Static Analysis Light' | 1066 |
| 1.3.1.25.3.11.9 | Dialog 'Project settings' - 'Visualization' | 1068 |
| 1.3.1.25.3.11.10 | Dialog 'Project Settings' - 'Visualization Profile' | 1069 |

**Symbol:** 📜

**Function:** The object contains the basic configuration of the project. In the “Project Settings” dialogs the configuration can be adjusted.

**Call:** Double click on the “Project Settings” object in the device tree, or main menu “Project ➔ Project Settings”.

CODESYS saves the project settings directly in the project. If a project is transferred to another system for example the “Project Settings” object is transferred as well without the need of a project archive.

The project settings are valid project wide. Dependent on the installed packages the dialogs provide settings for several categories, as for example “SFC” or “User and Groups”.

See also

- Chapter 1.3.1.25.2.8.1 “Command ‘Package Manager’ ” on page 949

---

**Dialog 'Project Settings' - 'SFC'**

**Symbol:** 📜

**Function:** This dialog is used for configuring the settings of SFC objects. The properties of each new SFC object automatically have the configured settings.

**Call:** Menu bar: “Project ➔ Project Settings” (“SFC”).

**Requirement:** A project is open.

---

**Tab 'Flags'**
Implicitly generated variables for checking and monitoring the processing in an SFC diagram

| “Active” | ✅: The corresponding variable is used. |
| “Declare” | ✅: The corresponding variable is created automatically. Otherwise, you have to declare the variable explicitly if you intend to use it (“Use” is selected). |

| “Apply to all” | In this dialog, CODESYS applies changes to existing SFC objects. CODESYS selects the “Use defaults” check box in the properties of the SFC POUs. |

NOTICE!
Automatically declared variables are visible in the declaration part of the SFC editor only in online mode.

Tab 'Build'

Table 195: “Code Generation”

| “Calculate active transitions only” | ✅: CODESYS generates code only for currently active transitions. |

Table 196: “SFC Library”

This part of the dialog is available only for compiler versions < 3.4.1.0.

| “Company” | Defines the SFC library that CODESYS uses by default. |
| “Title” | |
| “Version” | |
| “Namespace” | Enables unique references to libraries. Required when various versions of the library are available on the system. Please make sure that there are no discrepancies between the namespace defined in the library manager and the namespace defined for the individual object. The SfcIec.library data is used for the default settings that CODESYS provides with the default profile. |

Each SFC block stores the information via the library version that applied when you added the block. This can cause you to use multiple library versions within the same project. In order to prevent this, you are prevented from defining specific versions of IecSfc.library (as of compiler version 3.4.1.0). The library version, which you use for all SFC blocks in the project, is defined with a placeholder. CODESYS resolves the placeholder depending on the compiler version in use. The allocation of the library version to the compiler version is defined in the library profile.

See also
● Chapter 1.3.1.24.1.4.6 “SFC flags” on page 447
● Chapter 1.3.1.19.1 “Information for library developers” on page 372

Dialog Box 'Project Settings' - 'Users and Groups'

Symbol: 🔄

Function: This dialog box is for the configuration of the user management for the current project.

Call: Menu “Project ➔ Project Settings”, category “Users and Groups”
Tab 'User'

Displays the users and their memberships in groups

<table>
<thead>
<tr>
<th>Action</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Add&quot;</td>
<td>Opens the dialog box “Add User”.</td>
</tr>
<tr>
<td>&quot;Edit&quot;</td>
<td>Opens the dialog box “Edit User”.</td>
</tr>
<tr>
<td>&quot;Delete&quot;</td>
<td>An error message appears if you attempt to delete the last user of a group, since a group must have at least one member.</td>
</tr>
</tbody>
</table>

Table 197: “Add User / Edit User”

Input fields for setting up a new user account or changing an existing one

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Active&quot;</td>
<td>☑: You may use the user account, default ☐: The user cannot log in. If he attempts to login again with incorrect login data, this can result in automatic deactivation of the account; see below: Settings.</td>
</tr>
<tr>
<td>&quot;Memberships&quot;</td>
<td>List of all user groups that you have defined in addition to the group “Everyone” (to which each new user automatically belongs). &lt;group name&gt;: the new user belongs to the group.</td>
</tr>
</tbody>
</table>

Table 198: “Export/Import”

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Export users and groups&quot;</td>
<td>The command opens the standard dialog for saving a file in the local file system. You can store the users and groups definitions of the project in a file *.users in xml format.</td>
</tr>
<tr>
<td>&quot;Import users and groups&quot;</td>
<td>Export users and groups opens the standard dialog for browsing the local file system for a file. Search for a file with extension *.users in order to import the users and groups definitions, stored in this file, into the project.</td>
</tr>
</tbody>
</table>

Tab 'Groups'

Display of the groups and their members. A group can also be a member of a group.

<table>
<thead>
<tr>
<th>Action</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Add&quot;</td>
<td>Opens the dialog box “Add Group”.</td>
</tr>
<tr>
<td>&quot;Edit&quot;</td>
<td>Opens the dialog box “Edit Group”.</td>
</tr>
<tr>
<td>&quot;Delete&quot;</td>
<td>If you delete a group, the user accounts of the members remain unchanged. You cannot delete the groups “Everyone” and “Owner”.</td>
</tr>
</tbody>
</table>

On button “Export/Import” please see above the “User” paragraph.

Tab 'Settings'

Display of the groups and their members in a tree structure. A group can also be a member of a group.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Maximum number of authentication trials&quot;</td>
<td>☑ (standard) : If the user has attempted to login with an incorrect password the number of times specified here, the user account is deactivated. ☐: The number of the unsuccessful attempts is unlimited</td>
</tr>
<tr>
<td>&quot;Automatic logoff after time of inactivity&quot;</td>
<td>☑: The user is automatically logged out if CODESYS does not register any user actions by mouse or keyboard during the time period (minutes) specified here.</td>
</tr>
</tbody>
</table>

See also

- Chapter 1.3.1.25.2.4.24 “Command 'User management' – 'Log in User'” on page 911
Dialog Box 'Project Settings' - 'Compileoptions'

Symbol: 

Function: This dialog box is for configuring the compiler options.

Call: Main menu “Project ➜ Project Settings” (“Compileoptions” category).

Requirement: A project is open.

Table 199: “Compilerversion”

| “Fix Version” | Defines the compiler version that CODESYS uses when compiling and downloading for compile (for example, “3.5.6.0” for version 3.5 SP6). |

Table 200: “Settings”

| “Allow unicode characters for identifiers” | Cleared by default because using Unicode characters in identifier names is not permitted in the IEC standard. May be required for some foreign languages (for example, Asian languages). |
| “Replace constants” | (default): CODESYS loads the value directly for every scalar constant (not for strings, arrays, and structures). In online mode, CODESYS marks the constants with a symbol that is prepended to the value in the declaration editor or monitoring view. In this case, access is not possible, for example by means of an ADR operator, forcing, and writing. |
| “Enable logging in breakpoints” | For breakpoints that are defined as execution points, you can create a message text in the “Execution point settings” dialog box. CODESYS prints this text to the device log when the application halts at the execution point. |

Table 201: “Compiler Warnings”

| “Maximum number of warnings” | Refers to the warnings that CODESYS prints to the messages view. You define the selection of displayed compiler warnings in the “Project Settings” dialog box in the “Compiler Warnings” category. |

See also

- Chapter 1.3.1.25.3.11.4 “Dialog Box 'Project Settings' - 'Compiler Warnings’” on page 1062
- Chapter 1.3.1.25.1.19.8 “Tab 'Log’” on page 750

Dialog Box 'Project Settings' - 'Compiler Warnings'

Symbol: 

Function: This dialog box is used for selecting the compiler warnings that CODESYS displays in the messages view during a compile process.

Call: Call: Main menu “Project ➜ Project Settings” (“Compiler Warnings” category).

Requirement: A project is open.

You define the maximum number of listed warnings in the “Compileoptions” dialog box.
Dialog 'Project Settings' – 'Source Download'

Symbol: 

Function: This dialog defines the compilation and the storage of the source code as a source-code download archive on one or more controllers.

Call: “Project ➔ Project settings” menu, “Download source code” category

A source-code download archive is a project archive with the name Archiv.prj.

Table 202: “Destination device”

<table>
<thead>
<tr>
<th>Options</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&lt;name of controller&gt;</code></td>
<td>Selected controller. CODESYS loads the project archive to this controller. Requirement: the project contains several controllers.</td>
</tr>
<tr>
<td><code>&lt;all devices in project&gt;</code></td>
<td>CODESYS loads the project archive to all controllers in the project.</td>
</tr>
</tbody>
</table>

Table 203: “Content”

<table>
<thead>
<tr>
<th>Options</th>
<th>Description</th>
</tr>
</thead>
</table>
| “Use compact download” | ☑: The project archive contains only that device in the project that contains the active application.  
  ☐: The project archive contains all the devices in the project |
| “Additional Files” | Opens the “Additional files” dialog where you can select additional files for downloading. |

Table 204: “Timing”

<table>
<thead>
<tr>
<th>Options</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Implicitly at program download and online change”</td>
<td>Each time an application is loaded or an online change is made, CODESYS additionally loads the project archive to the target device(s) with no further prompt.</td>
</tr>
<tr>
<td>“Implicitly at creating boot project”</td>
<td>Each time a boot application is created, CODESYS additionally loads the project archive to the target device(s) with no further prompt.</td>
</tr>
<tr>
<td>“Implicitly at creating boot project, download and online change”</td>
<td>Each time a boot application is created, an application is loaded or an online change is made, CODESYS additionally loads the project archive to the target device(s) with no further prompt.</td>
</tr>
</tbody>
</table>
Each time an application is loaded or an online change is made, CODESYS opens a prompt, where you can select whether CODESYS should load the project archive to the controller.

A prompt opens only if the command “Online ➔ Load source code to connected controller” is called. There you can select whether CODESYS should load the project archive to the controller.

See also
- Chapter 1.3.1.11.7 “Downloading source code to and from the PLC” on page 307
- Chapter 1.3.1.25.2.6.7 “Command ‘Source Download to Connected Device’” on page 929
- Chapter 1.3.1.25.2.1.11 “Command ‘Source Download’” on page 860

Dialog 'Project Settings' - 'Page Setup'

Symbol: ✎

Function: This dialog defines the layout for the print version of the project contents. This layout is used for the printout of the project information by clicking “File ➔ Print” and the printout of the project documentation by clicking “Project ➔ Document”.

Call: Main menu “Project ➔ Project Settings” (“Page Setup”)

You can change settings the following:
- “Paper”
- “Margins”
- “Header and Footer”
- “Document”
- “Title Page”

Table 205: “Edit Header, Edit Footer”

<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Row spanning”</td>
<td>Number of rows that CODESYS should merge into a single column.</td>
</tr>
<tr>
<td>“Column spanning”</td>
<td>Number of columns that CODESYS should merge into a single row.</td>
</tr>
<tr>
<td></td>
<td>Opens the list of available placeholders for the “Text” field. When printing the page, CODESYS provides the placeholders with the current values.</td>
</tr>
</tbody>
</table>

See also
- Chapter 1.3.1.25.2.1.14 “Command ‘Page Setup’” on page 860
- Chapter 1.3.1.25.2.4.17 “Command ‘Document’” on page 905
- Chapter 1.3.1.25.2.1.12 “Command ‘Print’” on page 860

Dialog 'Project Settings' - 'Security'

Symbol: ✎

Function: this dialog is for the configuration of the project protection by a password, a dongle, or a certificate.

Call: Menu bar: “Project ➔ Project Settings” (category “Security”).
NOTICE!

If the encryption password is lost you can no longer open the project. You can also no longer restore it.

"No protection"  ● The project file is not protected from unauthorized access and data manipulation.

Note: We strongly recommend that you use security functionality.

● The “Password”, “Dongle”, and “Certificates” options cannot be selected.

"Integrity check"  ● When you create a new project, this option is enabled by default.

● The project file is stored in a proprietary format and its integrity is checked each time the project is loaded. The file may be incompatible with older versions of the development system.

Please note that the project file is not encrypted. To better protect your data, activate one of the encryption functions.

"Encryption"  ● The “Password”, “Dongle”, and “Certificates” encryption functions can be selected.

"Password"  ● Entering, changing and confirming the encryption password.

If you save the project with these settings you must enter the password later in order to open the project again, even if it is to be loaded as a library reference.

"Dongle"  ● Requirement: you have connected the CODESYS security key (dongle) to the computer.

“Add”: The dialog “Add Registered Dongle” opens.

"Registered dongles"  ● Drop-down list of the registered dongles.

"Certificates"  ● Certificates are used for the encryption of contents of the open project file.

Requirement: The certificates for all users who share the project must be installed in the local memory.

● The “Certificate selection” dialog opens.

Table 206: Adding a registered dongle

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Dongle&quot;</td>
<td>Drop-down list of all connected dongles.</td>
</tr>
<tr>
<td>&quot;Update&quot;</td>
<td>CODESYS refreshes the drop-down list.</td>
</tr>
<tr>
<td>&quot;Flash&quot;</td>
<td>The LEDs of the currently selected dongle flash for two seconds (if it supports this function).</td>
</tr>
</tbody>
</table>

The dongle must be connected to the computer when CODESYS loads the project, even if it is loaded as a library reference.

See also

- § Chapter 1.3.1.6 “Protecting and saving a project” on page 139
- § Chapter 1.3.1.6.2 “Assigning passwords” on page 143
- § Chapter 1.3.1.6.3 “Protecting projects using a dongle” on page 144
- § Chapter 1.3.1.6.7 “Encrypting projects with certificates” on page 149
- § Chapter 1.3.1.25.3.18 “Dialog ‘Certificate Selection’” on page 1098
Dialog 'Project Settings' - 'Static Analysis Light'

Symbol: §

**Function:** This dialog activates the tests that the light version of CODESYS Static Analysis performs each time code is generated.

**Call:** Menu bar: “Project ➔ Project Settings” (“Static Analysis Light” category).

You can exclude lines of code from the static code analysis by marking the code with the pragma `{analysis ...}` or the pragma `{attribute 'analysis' := '...'}.`

### Additional compile tests

<table>
<thead>
<tr>
<th>Test Description</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>“SA0033: Unused variables”</strong></td>
<td>Finds variables that are declared, but not used within the compiled program code. For GVL variables: If there are multiple applications in one project, then only the objects under the currently active application are affected. If there is only one application, then the objects in the POU view are also affected.</td>
</tr>
<tr>
<td><strong>“SA0028: Overlapping memory areas”</strong></td>
<td>Detects the locations where two or more variables reserve the same storage space. For example, this occurs for the following declarations: <code>var1 AT %QB21: INT</code> and <code>var2 AT %QD5: DWORD</code>. In this case, both variables use byte 21, which means that the memory range of the variables overlap.</td>
</tr>
<tr>
<td><strong>“SA0006: Write access from multiple tasks”</strong></td>
<td>Detects variables that are written by more than one task.</td>
</tr>
<tr>
<td><strong>SA0004 “Multiple write access on output”</strong></td>
<td>Detects outputs that are written to more than one location. Note: No error is reported when an output variable (<code>VAR_IN_OUT</code>) is written in different branches of IF and CASE statements. Note: A pragma cannot deactivate this rule.</td>
</tr>
</tbody>
</table>
| **“SA0027: Multiple use of identifiers”** | Detects multiple uses of a name/identifier for a variable or an object (POU) within the scope of a project. The following cases are detected:  
  - The name of an enumeration constant is the same as in another enumeration in the application or used in an included library.  
  - The name of a variable is the same as an object in the application or an included library.  
  - The name of a variable is the same as for an enumeration constant in and enumeration in the application or an included library.  
  - The name of an object is the same as another object in the application.  
  - The name of a variable is the same as the name of a method.  
  - The name of an object is the same as the name of a superordinate object ("parent object"). |
| **“SA0167: Temporary function block instances”** | The test detects function block instances that are declared as temporary variables. This concerns instances that are declared in a method or in a function or as `VAR_TEMP`, and therefore are reinitialized in each processing cycle and for each POU call. |
Examples

SA0003: Empty statements

;  (* Comment *);
iVar;

SA0006: Concurrent access

FUNCTION_BLOCK ADD_FB
  g_iTemp1 := g_iTemp1 + INT#1;
PROGRAM PLC_PRG  //controlled by MainTask
  g_iTemp1 := g_iTemp1 + INT#2;
  g_xTemp2 := g_iTemp1 > INT#10;
PROGRAM PLC_PRG_1  //controlled by SubTask
  g_iTemp1 := g_iTemp1 - INT#3;
  g_xTemp2 := g_iTemp1 < INT#-10;

SA0004 Multiple write access on output

VAR_GLOBAL
  g_xVar AT %QX0.0 : BOOL;
  g_iTest AT %QW0 : INT;
END_VAR

PROGRAM PLC_PRG
  IF iCondition < INT#0 THEN
    g_xVar := TRUE;
    g_iTest := INT#12;
  END_IF
  CASE iCondition OF
    INT#1:
      g_xVar := FALSE;
    INT#2:
      g_iTest := INT#11;
    ELSE
      g_xVar := TRUE;
      g_iTest := INT#9;
  END_CASE

SA0006: Write access from multiple tasks

FUNCTION_BLOCK ADD_FB
  g_iTemp1 := g_iTemp1 + INT#1;
PROGRAM PLC_PRG  // Controlled by MainTask
  g_iTemp1 := g_iTemp1 + INT#2;
  g_xTemp2 := g_iTemp1 > INT#10;
PROGRAM PLC_PRG_1  //Controlled by SubTask
  g_iTemp1 := g_iTemp1 - INT#3;
  g_xTemp2 := g_iTemp1 < INT#-10;

SA0027: Multiple use of name

PROGRAM PLC_PRG
  VAR
    ton : INT;  // error SA0027
  END_VAR

SA0029: Different notation in implementation and declaration

The PLC_PRG POU and a fnc function POU are in the device tree.
PROGRAM PLC_PRG
VAR
    iVar: INT;
    _123test_var_: INT;
END_VAR
ivar := iVar + 1; // notation different to that in the declaration part -> SA0029
_123TEST_var_ := _123test_var_INT; // notation different to that in the declaration part -> SA0029
Fnc(); // notation different to that in the devices tree -> SA0029
END_VAR

SA0167: Temporary function block instances

PROGRAM PLC_PRG
VAR
END_VAR
VAR_TEMP
    yafb: AFB;
END_VAR

FUNCTION Fun : INT
VAR_INPUT
END_VAR
VAR
    funafb: AFB;
END_VAR

METHOD METH: INT
VAR_INPUT
END_VAR
VAR
    methafb: AFB;
END_VAR

See also
● § Chapter 1.3.1.9.13.2 “Analyzing code statically” on page 230

Dialog 'Project settings' - 'Visualization'

Symbol:

Function: This dialog is for the configuration of the project-wide settings for objects of the type "Visualization".

Call: Main menu “Project ➤ Project Settings”, category “Visualization”

Requirement: A project is open.
### Tab 'General'

#### Table 207: “Visualization Directories”

<table>
<thead>
<tr>
<th>Directory</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>“Text list files”</strong></td>
<td>Directory containing text lists that are available in the project for configuring texts for different languages. CODESYS uses the directory, for example, when exporting or importing text lists. After clicking on the “Select Directory” dialog opens to enable the selection of a directory in the file system.</td>
</tr>
<tr>
<td><strong>“Image Files”</strong></td>
<td>Directory containing image files that are available in the project. Multiple folders are separated with a semicolon. CODESYS uses the directory, for example, when exporting or importing image files. After clicking on the “Select Directory” dialog opens to enable the selection of a directory in the file system.</td>
</tr>
</tbody>
</table>

#### Table 208: “Advanced”

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>“Activate property handling in all element properties”</strong></td>
<td>You can configure a visualization element with a property instead of with an IEC variable in those of its properties in which you select a variable. CODESYS then creates additional code for the property handling when compiling a visualization. Requirement: Your IEC code contains at least one object of the type “Interface property”, i.e. a property.</td>
</tr>
</tbody>
</table>

**Requirement:** “Visible” is activated.

See also:
- ‘Property object’ (Chapter 1.3.1.25.1.28.8 “Object ‘Property’” on page 791)

### Tab ‘Symbol Libraries’

#### Table 209: “Visualization Symbol Libraries”

<table>
<thead>
<tr>
<th>Symbol libraries</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>“Assigned”</strong></td>
<td>Symbol library is assigned to the project and CODESYS makes it available in the “ToolBox” view of a visualization.</td>
</tr>
</tbody>
</table>

See also:
- CODESYS Visualization
- “Dialog ‘Add visualization’” (Chapter 1.3.5.19.3.2 “Dialog ‘Add Visualization’” on page 1583)

### Dialog 'Project Settings' - 'Visualization Profile'

**Symbol:**

**Function:** The dialog enables the setting of the visualization profile.

**Call:** Menu “Project ➔ Project Settings”, category “Visualization Profile”
**Requirement:** A project is open.

**Table 210: “Visualization Profile”**

<table>
<thead>
<tr>
<th>“Certain profile”</th>
<th>Profile that CODESYS uses in the project and that determines the visualization elements that are available in the project.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>The selection list contains all the profiles installed so far.</td>
</tr>
</tbody>
</table>

**Dialog 'Project Environment’**

<table>
<thead>
<tr>
<th>Version</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.3.1.25.3.12.1</td>
<td>Dialog 'Project Environment’ – ‘Library Versions’.................................................................. 1070</td>
</tr>
<tr>
<td>1.3.1.25.3.12.2</td>
<td>Dialog 'Project Environment' - ‘Compiler Version’.................................................................... 1070</td>
</tr>
<tr>
<td>1.3.1.25.3.12.3</td>
<td>Dialog 'Project Environment' - ‘Device Versions’...................................................................... 1071</td>
</tr>
<tr>
<td>1.3.1.25.3.12.4</td>
<td>Dialog 'Project Environment’ – ‘Visualization Profile’............................................................ 1071</td>
</tr>
<tr>
<td>1.3.1.25.3.12.5</td>
<td>Dialog 'Project Environment’ – ‘Visualization Styles’................................................................. 1072</td>
</tr>
<tr>
<td>1.3.1.25.3.12.6</td>
<td>Dialog 'Project Environment’ – ‘C Code Modules’......................................................................... 1072</td>
</tr>
<tr>
<td>1.3.1.25.3.12.7</td>
<td>Dialog 'Project Environment' – ‘Visualization Symbols’................................................................ 1073</td>
</tr>
</tbody>
</table>

**Function:** You use this dialog for checking the actuality of the software and of the files, which are included in the project. CODESYS checks for example the selected compiler and finds out if there is a newer version. In such a case you can update the affected components.

**Call:** Main menu “Project”

**Dialog 'Project Environment’ – ‘Library Versions’**

**Function:** This dialog displays the libraries of the opened project for which newer versions are available.

**Call:** Main menu “Project ➔ Project Environment”, tab “Library Version”

This dialog opens automatically when you open a project containing outdated libraries.

**Table 211**

<table>
<thead>
<tr>
<th>“Action”</th>
<th>Double-click inside the field to select the desired actions.</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Check for updates when loading this project”</td>
<td>☑: Checking takes place each time the project is opened.</td>
</tr>
<tr>
<td></td>
<td>☐: Checking takes place once only.</td>
</tr>
<tr>
<td>“Set all to newest”</td>
<td>CODESYS uses the newest available version of the library.</td>
</tr>
<tr>
<td>“OK”</td>
<td>CODESYS performs the selected action(s).</td>
</tr>
</tbody>
</table>

**Dialog 'Project Environment’ - ‘Compiler Version’**

**Function:** This dialog shows the current compiler version of the project and provides the capability of updating.

**Call:** Main menu “Project ➔ Project Environment” (“Compiler Version” tab).
Table 212

<table>
<thead>
<tr>
<th>“Current compiler version in project”</th>
<th>Shows the set compiler version for the open project.</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Recommended, newest version”</td>
<td>Shows the latest version.</td>
</tr>
</tbody>
</table>
| “Action”                              | • “Do not update”: The compiler version of the project remains the same.  
                                           • “Update to x.x.x.x”: The selected compiler version is set for the project. |
| “Check for updates when loading this project” | ✓: CODESYS checks for new versions each time the project is opened. If there is a new version, then the respective update dialog opens automatically.  
                                           □: The compiler version is not checked. The update dialogs do not open automatically. |
| “Set all to newest”                   | The compiler version is set to the latest version. |

Dialog 'Project Environment' - 'Device Versions'

**Function:** This dialog shows the devices of the open project in which there are new versions available.

**Call:** Main menu “Project ➔ Project Environment” (“Device Versions” tab)

This dialog opens automatically when you open a project that contains an outdated device.

Table 213

<table>
<thead>
<tr>
<th>Names of the outdated devices and their versions, as well as the current version and the planned action.</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Action”</td>
</tr>
</tbody>
</table>
| “Check for updates when loading this project” | ✓: The check is performed when the project is opened.  
                                           □: The check is performed one time only. |
| “Set all to newest”                     | CODESYS uses the latest library version. |
| “OK”                                    | CODESYS executes the selected actions. |

Dialog 'Project Environment' – 'Visualization Profile'

**Function:** This dialog shows the current visualization profile of the project. The profile can be updated here.

**Call:** Menu bar: “Project ➔ Project Environment” (“Visualization Profile” tab).

<table>
<thead>
<tr>
<th>“Current visualization profile in the project”</th>
<th>The set visualization profile of the open project.</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Recommended, newest profile”</td>
<td>The newest version.</td>
</tr>
</tbody>
</table>
| “Action”                                      | • “Do not update”: The visualization profile of the project remains unchanged.  
                                           • “Update to x.x.x.x”: CODESYS updates the project to the selected visualization profile. |
| “Check for updates when loading this project” | ✓: CODESYS checks for new profiles each time the project is opened. If there is a new version, then the respective update dialog opens automatically.  
                                           □: Not test of the profile when opening the project. The update dialogs do not open automatically. |
| “Set all to newest”                          | CODESYS updates the . |
See also
  ● Help about visualization

Dialog 'Project Environment' – 'Visualization Styles'

**Function:** This dialog shows the current visualization style of the project and provides the capability of updating it.

**Call:** Menu bar: “Project ➔ Project Environment” (“Visualization Styles” tab).

<table>
<thead>
<tr>
<th>“Visualization style”</th>
<th>Version of the set visualization style in the open project</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Current”</td>
<td>Current version of the visualization style (example: 3.5.6.0)</td>
</tr>
<tr>
<td>“Recommended”</td>
<td>Recommended version of the visualization style (example: 3.5.7.0)</td>
</tr>
</tbody>
</table>
| “Action”                    | ● "Do not update": The visualization style of the project remains unchanged.  
                              | ● "Update to x.x.x.x": CODESYS updates the project to the version of the selected visualization style. |
| “Check for updates when loading this project” | ☑: CODESYS checks for new versions each time the project is opened. If there is a new version, then the respective update dialog opens automatically.  
                              | ☐: The version is not checked. The update dialogs do not open automatically. |
| “Set all to newest”         | CODESYS updates the version. |

See also
  ● Help for visualization, section "Visualization style"

Dialog 'Project Environment' – 'C Code Modules'

**Function:** This dialog lists all C-code modules and their C-code files that have changed in the source directory on the disk. You can update individual C-code modules here.

**Call:** Menu “Project ➔ Project Environment”, tab “C Code modules”

<table>
<thead>
<tr>
<th>“Project”</th>
<th>Display of the C-code module with its changed C-code files in the project.</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Action”</td>
<td>Selection option for the C-code module of the “Project” field</td>
</tr>
<tr>
<td></td>
<td>A double-click on the field displays all selection options:</td>
</tr>
<tr>
<td></td>
<td>● “Update”</td>
</tr>
<tr>
<td></td>
<td>● “Do not update”</td>
</tr>
<tr>
<td></td>
<td>For each C-code file this indicates what action is executed if you select the action “Update” for the corresponding C-code module (“Project”).</td>
</tr>
<tr>
<td>“Delete IEC interfaces”</td>
<td>Deletes the created IEC interface if the headers in the project have changed. In this case you must create the IEC interface again.</td>
</tr>
<tr>
<td>“Check for updates when loading this project”</td>
<td>☑: checking takes place each time the project is opened.</td>
</tr>
<tr>
<td>“Set all to ’newest’”</td>
<td>CODESYS refreshes all C-code modules.</td>
</tr>
<tr>
<td>“OK”</td>
<td>CODESYS executes the selected actions in the project.</td>
</tr>
</tbody>
</table>
See also

- Chapter 1.3.1.25.1.16 “Object ‘C Code Module’” on page 737
- Chapter 1.3.1.25.2.5.14 “Command ‘Create IEC Interface’” on page 921
- Chapter 1.3.1.9.11 “Integrating C Modules” on page 222

Dialog 'Project Environment' – 'Visualization Symbols'

**Function:** This dialog lists installed symbol libraries and allows you to assign symbol libraries to a project.

**Call:** Menu bar: “Project ➔ Project Environment” (“Visualization Symbols” tab).

**Requirement:** The open project contains a visualization and has been saved with a compiler version < 3.5.7.0. CODESYS recognizes symbol libraries in compiler version 3.5.7.0 and later.

<table>
<thead>
<tr>
<th>“Symbol library”</th>
<th>List of all installed symbol libraries</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Active&quot;</td>
<td>☑: Symbol library is selected for the project. CODESYS provide its symbols in the “Tools” view. □: Symbol library has been previously installed only in the library repository.</td>
</tr>
</tbody>
</table>

See also

- Help for visualization, section "Using the symbol library in the visualization"

Dialog 'Options'

1.3.1.25.3.13.1 Dialog 'Options' - 'SFC Editor'…………………………………… 1074
1.3.1.25.3.13.2 Dialog 'Options' – 'Libraries'……………………………………… 1075
1.3.1.25.3.13.3 Dialog 'Options' – 'Library Download'………………………… 1075
1.3.1.25.3.13.4 Dialog 'Options' - 'CFC Editor'…………………………………… 1076
1.3.1.25.3.13.5 Dialog 'Options' - 'CoDeSys V2.3 Converter'…………………… 1077
1.3.1.25.3.13.6 Dialog 'Options' – ‘Declaration Editor’…………………………… 1078
1.3.1.25.3.13.7 Dialog 'Options' – 'Device Description Download'………………… 1079
1.3.1.25.3.13.8 Dialog 'Options' - 'FBD, LD, and IL'……………………………… 1079
1.3.1.25.3.13.9 Dialog 'Options' - 'Device Editor'………………………………… 1081
1.3.1.25.3.13.10 Dialog 'Options' - 'Help'………………………………………… 1081
1.3.1.25.3.13.11 Dialog 'Options' - 'SmartCoding'………………………………… 1082
1.3.1.25.3.13.12 Dialog 'Options' – 'International Settings'………………………… 1083
1.3.1.25.3.13.13 Dialog 'Options' – 'Load and Save'……………………………… 1083
1.3.1.25.3.13.14 Dialog 'Options' - 'Monitoring'……………………………………… 1083
1.3.1.25.3.13.15 Dialog 'Options' - 'PLCopenXML'………………………………… 1084
1.3.1.25.3.13.16 Dialog 'Options' - 'Proxy Settings'………………………………… 1084
1.3.1.25.3.13.17 Dialog 'Options' - 'Refactoring'…………………………………… 1085
1.3.1.25.3.13.18 Dialog Box 'Options' - 'Sequence Editor'…………………………… 1086
1.3.1.25.3.13.19 Dialog Box 'Options' - 'Syntax Highlighting'……………………… 1086
1.3.1.25.3.13.20 Dialog 'Options' - 'Text Editor'……………………………………… 1086

**Function:** You use the dialog box for selecting the CODESYS options. With these options you configure the appearance and the behavior of the user interface. CODESYS saves the current configuration as standard settings in the local system.

**Call:** Main menu “Tools ➔ Options”
Dialog 'Options' - 'SFC Editor'

**Symbol:**

**Function:** This dialog is used for configuring the settings for the SFC editor.

**Call:** Menu bar: “Tools ➔ Options” (“SFC Editor” category).

See also

- Chapter 1.3.1.9.3.4.1 “Programming the sequential function chart (SFC)” on page 202
- Chapter 1.3.1.25.2.11 “Menu ‘SFC’” on page 967
- Chapter 1.3.1.24.1.4.1 “SFC editor” on page 442

---

Tab 'Layout'

**Table 216: “Elements”**

This defines the dimensions of the SFC elements: step, action, qualifier, property. The values are given in matrix units, where one matrix unit equals the font size that you set in the text editor options (text area / font). The settings are always active immediately in all open SFC editor views.

<table>
<thead>
<tr>
<th>Setting</th>
<th>Possible values</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Step height”</td>
<td>1-100</td>
</tr>
<tr>
<td>“Step width”</td>
<td>2-100</td>
</tr>
<tr>
<td>“Action width”</td>
<td>2-100</td>
</tr>
<tr>
<td>“Qualifier width”</td>
<td>2-100</td>
</tr>
<tr>
<td>“Property width”</td>
<td>2-100</td>
</tr>
</tbody>
</table>

---

**Table 217: “Font”**

The example text shows the current font. Click it to change the font.

---

**Table 218: “Step Actions”**

- “Copy reference”: The reference to the action objects that call the step are also copied when the step is copied. The copied step and new step call the same action.
- “Duplicate implementation”: The reference to the action objects that call the step are linked to this step. When copying the step element, new action objects are created for the new step, and the implementation is duplicated.
- “Always ask”: When inserting a step action, you are always prompted whether the actions of a step element should be duplicated when it is copied, or whether the reference to the existing action should be applied.

Note: If a step already contains an embedded action, then new inserted actions of this step are also embedded. Likewise, new inserted actions are not embedded when the step already contains a non-embedded action. In these cases, you are no longer prompted for a duplication mode.

---

**Table 219: “Embedded Objects”**

- “Show action and transition objects in the navigator”

  - : Action and transition objects that are embedded in the SFC box by a step are displayed in the “Devices” or “POUs” tree view.

---

Tab 'View'

**Table 220: “Property Visibility”**

List of element properties for the categories “Common” and “Specific” with definitions of the display options.

<table>
<thead>
<tr>
<th>Property</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Property”</td>
<td>Defines the element properties displayed next to the element in the SFC diagram.</td>
</tr>
</tbody>
</table>
Dialog 'Options' – 'Libraries'

Symbol: 📚

**Function:** This dialog helps you to manage the mappings of library references that CODESYS uses during the conversion of an old project. If you have not yet stored any mapping for a certain library, you must redefine the mapping each time when opening an old project in which this library is integrated.

**Call:** Menu bar: “Tools ➔ Options”; category: “Libraries”.

A mapping defines what a library reference looks like following the conversion of the project to the current format. There are three possibilities:

- You retain the reference. This means that CODESYS similarly converts the library into the current format (*.library) and installs it in the local library repository.
- You replace a reference with another reference. This means that one of the installed libraries replaces the library that was integrated until now.
- You delete the reference. This means that the converted project no longer integrates the library.

CODESYS applies all the listed mappings to the library references of an old project the next time it is converted. Hence, you must repeat the mapping definition if the same library is integrated again in a project that is to be converted. You can enter a new mapping in the last line.

<table>
<thead>
<tr>
<th>“Source Library”</th>
<th>Path of the library that is integrated in the project before the conversion. A double-click an entry makes the field editable and the button for the input assistance appears.</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Target Library”</td>
<td>Name and location of the library that is to be integrated in the project after the conversion. A double-click an entry opens the dialog “Set target system library”.</td>
</tr>
</tbody>
</table>

**Table 222: “Set target system library”**

<table>
<thead>
<tr>
<th>“Scan”</th>
<th>The “Select Library” dialog opens. You can select a library from the library repository here. The dialog corresponds to the dialog in the library repository.</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Ignore”</td>
<td>When CODESYS converts the project, CODESYS always removes the existing source library from the project.</td>
</tr>
</tbody>
</table>

Dialog 'Options' – 'Library Download'

Symbol 🌐

**Function:** This dialog is for the setting of download servers.

**Call:** menu “Tools ➔ Options”, “Library Download” category
If you click on the button “Download Missing Libraries” in the library manager, CODESYS browses these download servers for libraries marked as missing in the library manager and uses the set login details for the proxy server.

<table>
<thead>
<tr>
<th>“Download servers”</th>
<th>URL of a server containing library files</th>
</tr>
</thead>
<tbody>
<tr>
<td>Double-click on “(Enter new download server here)”</td>
<td>An input field opens in which you can enter the URL address of a server.</td>
</tr>
</tbody>
</table>

See also
- § Chapter 1.3.1.25.1.25 “Object 'Library Manager’” on page 771
- § Chapter 1.3.1.25.3.13.16 “Dialog ‘Options’ - ‘Proxy Settings’” on page 1084

Dialog ‘Options’ - ‘CFC Editor’

Symbol: 🗞

Function: This dialog is for the configuration of the settings for editing and printing in the CFC editor.

Call: menu “Tools ➔ Options”, category “CFC Editor”

Tab 'General'

| “Enable AutoConnect” | ☑: If you drag a CFC element onto the work area of the editor and insert it, CODESYS automatically connects together unconnected pins that ‘touch’ one another. Make sure that you do not create unwanted connections when shifting elements! |
| “Prepare values in implementation part” | ☑: In online mode you can also prepare variable values for writing and forcing in the implementation part of the CFC module. In addition, CODESYS displays the values you have just prepared in the inline monitoring box of the variable in angle brackets. |

Tab 'View'

| “Display grid points” | ☑: Grid points at which you can position the elements are visible in the editor. |
| “Show box icon” | ☑: Existing function blocks that are linked with a bitmap are displayed by CODESYS in the CFC editor as symbols. 
Requirement: You have either created the link for a function block or a function in the object properties or loaded it via a library. |
| “Edit Line Colors” | Opens the “Edit Line Colors” dialog for the definition of the colors of the connecting lines, depending on the data type applied. The lines appear in offline and online mode in these colors, unless CODESYS paints over them with the thick black and blue lines used to display the boolean data flow.
- “Add Type:” Adds a data type to the list.
- “Delete Type” |
| “Font” | Display of the font and button for changing the font. |

See also
- § Chapter 1.3.1.25.3.10.8 “Dialog Box ‘Properties’ - ‘Bitmap’” on page 1051

Tab 'Print'
### Setting the “Layout Options”

<table>
<thead>
<tr>
<th>“Fit method”</th>
<th>“Page” or “Poster”</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Scale”</td>
<td>Possible values: 20 % - 200 %</td>
</tr>
</tbody>
</table>

See also
- Chapter 1.3.1.9.3.2.2 “Programming in the CFC editor” on page 193
- Chapter 1.3.1.24.1 “Programming languages and editors” on page 428

### Dialog 'Options' - 'CoDeSys V2.3 Converter'

**Symbol:** 🌐

**Function:** The dialog is used to define the mapping when converting a CoDeSys V2.3 project.

**Call:** Menu bar: “Tools ➔ Options”, “CoDeSys V2.3 Converter” category

> A CoDeSys V2.3 project can be converted into a CODESYS V3 project only if the CODESYS V2.3 Converter package is installed in CODESYS V3. The package is available in the CODESYS Store.

### Table 223: “Devices”

List and definition of the device mappings for a conversion.

All device mappings of this list are also in effect the next time a CoDeSys V2.3 project is converted. This eliminates the need to recreate the image definition when the same device is used in multiple projects.

<table>
<thead>
<tr>
<th>“Source Device”</th>
<th>Double-clicking an entry or an empty field opens the “Select Source Device” dialog. CODESYS lists all devices that converter can process.</th>
</tr>
</thead>
</table>
| “Target Device” | Double-clicking an entry or an empty field opens the “Select Target Device” dialog.  
  - “Select a target device from the list of installed devices below”: Confirm the selection by clicking “OK”.  
  - “None”: The source device will not be available in the new project. This also means that application-specific objects, such as the task configuration, are not imported. |
Table 224: “Visualizations”

<table>
<thead>
<tr>
<th>“Import of Targetvisualizations”</th>
<th>“Import of Webvisualizations”</th>
</tr>
</thead>
<tbody>
<tr>
<td>Defines the handling of the CODESYS TargetVisu and CODESYS WebVisu display variants when a CoDeSys V2.3 project is opened.</td>
<td></td>
</tr>
</tbody>
</table>

List box

- **“Try to imitate old behaviour”**: The option is available if the currently used device supports the option. The effect of this option is that the new target or web visualization settings correspond as accurately as possible to the settings in the old project.
- **“Ask the user once”**: CODESYS asks the user to define the settings only when an "old" visualization project is imported for the first time. After that, CODESYS applies the automatically settings in all other visualization imports.
- **“Ask the user”**: CODESYS asks the user to define the settings every time an "old" visualization project is opened.
- **“Setting from Device Description”**: CODESYS sets the settings according to the entries in the device description of the installed target device.
- **“Try to imitate old behaviour”**: The option is available only if the currently used device supports the option. The effect of this option is that the new display variant settings for the target or web visualization correspond as accurately as possible to the settings in the old project.

**NOTICE!**

CODESYS TargetVisu does not support the access to VAR_IN_OUT variables, because the compiler does not support this. This behavior corresponds to the behavior of CoDeSys V2.3 target visualizations.

See also

- “Chapter 1.3.1.3.4 “Opening a V3 project” on page 127
- “Chapter 1.3.1.25.2.8.1 “Command ‘Package Manager’ “ on page 949

**Dialog ‘Options’ – ‘Declaration Editor’**

**Symbol:** 📦

**Function:** This dialog is for the configuration of the display settings for the declaration editor.

**Call:** Main menu “Tools ➔ Options”, category “Declaration Editor”

<table>
<thead>
<tr>
<th>“Textual only”</th>
<th>Textual view of the declaration editor</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Tabular only”</td>
<td>Tabular view of the declaration editor</td>
</tr>
<tr>
<td>“Switchable between textual and tabular”</td>
<td>The declaration editor offers two buttons for switching between the textual and tabular views:</td>
</tr>
<tr>
<td></td>
<td>📝: textual view</td>
</tr>
<tr>
<td></td>
<td>📑: tabular view</td>
</tr>
<tr>
<td></td>
<td>The following option defines the view that appears by default when opening a programming object:</td>
</tr>
<tr>
<td></td>
<td>“Always textual”</td>
</tr>
<tr>
<td></td>
<td>“Always tabular”</td>
</tr>
<tr>
<td></td>
<td>“Remember recent setting (per object)”</td>
</tr>
<tr>
<td></td>
<td>“Remember recent setting (global)”</td>
</tr>
</tbody>
</table>

See also

- “Chapter 1.3.1.9.2.1 “Using the declaration editor” on page 173
Dialog 'Options' – 'Device Description Download'

Symbol: ⏀

**Function:** This dialog is for the configuration of addresses of download servers for device descriptions.

**Call:** Menu “Tools ➔ Options”, category “Download the device descriptions”.

See also
- Chapter 1.3.1.20 “Managing devices” on page 374

<table>
<thead>
<tr>
<th>Table 225: “Download server”</th>
</tr>
</thead>
<tbody>
<tr>
<td>List of download servers containing device descriptions. By default '<a href="https://store.codesys.com/">https://store.codesys.com/</a> CODESYSDevs' is entered as the download server.</td>
</tr>
</tbody>
</table>

If you select the button “Download Missing Device Descriptions” in the “Device Repository” dialog, CODESYS uses the servers entered here and uses the set login data for the proxy server.

<table>
<thead>
<tr>
<th>Double-click on “(Enter new download server here...)”</th>
<th>An input field opens in which you can enter the URL address of a server.</th>
</tr>
</thead>
<tbody>
<tr>
<td>[ Del ]</td>
<td>Deletes the selected download server.</td>
</tr>
</tbody>
</table>

See also
- Chapter 1.3.1.25.2.8.5 “Command ‘Device Repository’” on page 957
- Chapter 1.3.1.25.3.13.16 “Dialog ‘Options’ - ‘Proxy Settings’” on page 1084

Dialog 'Options' - 'FBD, LD, and IL'

Symbol: ⏀

**Function:** This dialog is used for configuring the display options for the FBD/LD/IL editor.

**Call:** “Tools ➔ Options” (category “FBD, LD, and IL”).

Tab 'General'

<table>
<thead>
<tr>
<th>Table 226: “View”</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>“Show network title”</strong></td>
</tr>
<tr>
<td><strong>“Show network comment”</strong></td>
</tr>
<tr>
<td><strong>“Show box icon”</strong></td>
</tr>
<tr>
<td><strong>“Show operand comment”</strong></td>
</tr>
<tr>
<td><strong>“Show symbol comment”</strong></td>
</tr>
<tr>
<td><strong>“Show symbol address”</strong></td>
</tr>
<tr>
<td><strong>“Show network separators”</strong></td>
</tr>
</tbody>
</table>
Table 227: “Behavior”

| “Placeholder for new operands” | The operand field of pins for the new function block is left blank (instead of “???”). |
| “Empty operands for function block pins” | Adds blank operands instead of ???. |

Table 228: “Font”

Click the input field to open the “Font” dialog.

| “Fixed size for operand fields” | ☑: “Edit operand sizes” can be enabled. |
| “Edit operand sizes” | The “Operand Sizes” dialog opens for setting the number of characters and lines. |

Tab ‘FBD’

Table 229: “View”

| “Networks with line breaks” | ☑: Display of the network with line breaks so that CODESYS can show as many blocks as possible in the current width of the window. |
| “Connect boxes with straight line” | ☑: The length of the lines between the elements are fixed and short. |

Table 230: “Behavior”

| “Default network content” | Drop-down list: Contents of a new network |
| “After insertion select” | Drop-down list: Element that CODESYS selects after inserting a new network |

Tab ‘LD’

Table 231: “View”

| “Networks with line breaks” | ☑: Display of the network with line breaks so that CODESYS can show as many blocks as possible in the current width of the window. |

Table 232: “Behavior”

| “Default network content” | Drop-down list: Contents of a new network |
| “After insertion select” | Drop-down list: Element that CODESYS selects after inserting a new network |

Tab ‘IL’

Table 233: “View”

| “Enable IL” | The IL implementation language is available in the development system. |

Table 234: “Behavior”

| “Default network content” | Drop-down list: Contents of a new network |
| “After insertion select” | Drop-down list: Element that CODESYS selects after inserting a new network |

Tab ‘Print’
Table 235: “Layout Options”

<table>
<thead>
<tr>
<th>“Fit method”</th>
<th>Drop-down list for resizing.</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Avoid cutting of elements”</td>
<td>Elements that do not fit on the page are printed on the next page.</td>
</tr>
<tr>
<td>“Mark connections on adjacent pages”</td>
<td>Enabled for selection when “Avoid cutting of elements” is selected.</td>
</tr>
</tbody>
</table>

See also

- Chapter 1.3.1.24.1 “Programming languages and editors” on page 428

Dialog 'Options' - 'Device Editor'

Symbol: 

**Function:** This dialog includes settings for displaying the device editor.

**Call:** Menu bar: “Tools ➔ Options”; category: “Device Editor”.

Tab 'View'

<table>
<thead>
<tr>
<th>“Show generic device configuration views”</th>
<th>![ ] This tab with the list of device parameters is available in the device editors of parameterizable devices.</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Create cross references for IEC addresses (clean necessary)“</td>
<td>![ ] CODESYS creates the cross-references for unmapped I/Os.</td>
</tr>
</tbody>
</table>
| “Communication page“                       | ![ ] “Classic mode”: The “Communication” tab of the device editors appears as a split window with the left side showing the current configured gateway channels in a tree structure and the right side showing the associated data and information.  
|                                       | ![ ] “Simple mode”: The “Communication” tab appears as described in the corresponding section in the help. |
| Additional modes may also be available from OEM extensions. |

| “Show implicit files for application download on the editor of a PLC” | ![ ] The tab for synchronized files is available in the device editors. Synchronized files are downloaded to the PLC at the time of application download. These can be external files that were added to the application, or implicit files such as a source code archive. |
| “Show access rights page“ | ![ ] The “Access Rights” tab is available in the device editors.  
| Note: Depending on the device, the device description may overwrite this setting. |

See also

- Chapter 1.3.1.25.1.19.2 “Tab 'Communication Settings'” on page 741
- Chapter 1.3.1.25.1.19.6 “Tab 'Synchronized Files'” on page 749

Dialog 'Options' - 'Help'

Symbol: 

**Function:** This dialog defines whether CODESYS Online Help or CODESYS Offline Help opens when help is called.

**Call:** Menu bar: “Tools ➔ Options”; category: “Help”.

| “Use CODESYS Online Help, if available“ | ![ ] CODESYS Online Help opens when CODESYS Help is called. This is the default setting.  
| ![ ] CODESYS Offline Help opens when CODESYS Help is called. |
See also

● “Using CODESYS help” on page 99

Dialog 'Options' - 'SmartCoding'

**Symbol:** 🛠

**Function:** This dialog is for configuring the settings for easier coding.

**Call:** Menu bar: “Tools ➤ Options”, “SmartCoding” category

<table>
<thead>
<tr>
<th>Setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Declare unknown variables automatically (AutoDeclare)</strong></td>
<td>The “Declare Variable” dialog opens when you type an undeclared identifier into an implementation language editor and then click away from the input line. In order for the AutoDeclare function to be available in the ST editor as well, the “Enable for ST editor” option also has to be selected.</td>
</tr>
</tbody>
</table>
| **Enable for ST editor** | Requirement: The “Declare unknown variables automatically (AutoDeclare)” option is selected.  
☑: The AutoDeclare function is also available in the ST editor.  
☐: The AutoDeclare function is not available in the ST editor. |
| **Show all instance variables in Input Assistant** | ☑: The "List components" function also lets you select the local variables of a function block instance.  
☐: The "List components" function lets select only the input variables and output variables of a function block instance. |
| **Show symbols from system libraries in Input Assistant** | System libraries are inserted in the library manager automatically and displayed in light gray.  
☑: Symbols, such as global variables, data types, and function blocks, are offered in the Input Assistant.  
☐: The symbols of the system libraries are not available in the Input Assistant. |
| **List components after typing a dot (.)** | ☑: Activates the "List components" function. When you type a dot (.) at a location where CODESYS expects an identifier, a list box appears with possible code. |
| **List components immediately when typing** | Requirement: The “List components after typing a dot (.)” check box is selected.  
☑: While you type code, a list box appears with possible identifiers and operators. |
| **Insert with namespace** | ☑: CODESYS adds the namespace before the identifier. |
| **Convert keywords to uppercase automatically (AutoFormat)** | ☑: CODESYS displays all keywords in uppercase. |
| **Automatically list selection in cross-reference view** | ☑: The cross-reference list automatically shows the references of variables, POU, and DUTs that are currently selected or where the cursor is waiting. |
| **Underline errors in the editor** | ☑: Incorrect or unknown program code is underlined. |
| **Highlight symbols** | ☑: All occurrences of a symbol where the cursor is positioned are highlighted in color within the editor. In this way, cross-references within the editor are quickly detected. |
| **Max. degree of parallelism** | List box for the number of parallel threads that can be used for the precompile processing.  
CODESYS detects the displayed number of threads from the number of CPU cores. This default number should be changed only in exceptional cases. |
See also
- Chapter 1.3.1.24.1 “Programming languages and editors” on page 428
- “Smart tag functions” on page 211
- Chapter 1.3.1.9.14.1 “Using the cross-reference list to find occurrences” on page 232
- Chapter 1.3.1.24.1.3.1 “ST editor” on page 430

Dialog 'Options' – 'International Settings'

Symbol: 🌍

Function: This dialog is for the setting of the language in the user interface and in the help.

Call: Menu bar: “Tools ➔ Options”, category “International Settings”.

Dialog 'Options' – 'Load and Save'

Symbol: 📄

Function: This dialog contains settings for the behavior of CODESYS when loading and saving a project.

Call: Menu “Tools ➔ Options”, category “Load and Save ”

| “Create backup copy”     | : Each time the project is saved, CODESYS also saves the project as the file <projectname>.backup in addition to the file <projectname>.project. You can rename the backup file and open it in the programming system. |
| “Automatically save every … minutes” | : CODESYS automatically saves the project at the specified time intervals in a file <projectname>.autosave, which you can reload following non-regular closing of the programming system. In case of regular closing or saving of the project, CODESYS deletes the .autosave file. In the case of a non-regular abort, CODESYS keeps the .autosave file. If you open a project for which there is an associated auto-save file, the “Auto Save Backup” dialog opens. In this dialog, you select whether the .autosave file or the version of the project last saved by the user should be opened. |
| “Save before build”     | CODESYS saves the project automatically before any build run. |
| “Advanced Settings”     | The “Advanced Settings” dialog opens. “Loading Behavior”: The loading of libraries and compilation information takes place in the background while you are already editing the project. |

See also
- Chapter 1.3.1.6 “Protecting and saving a project” on page 139

Dialog 'Options' - 'Monitoring'

Symbol: 📈

Function: This dialog includes settings for displaying the variable values in monitoring.

Call: Menu bar: “Tools ➔ Options”; category: “Monitoring”.

2020/12/10
Table 236: “Display Mode for Integer Variables”

<table>
<thead>
<tr>
<th>Display Mode</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Binary</td>
<td>The value of the variable is displayed in the corresponding format in online mode.</td>
</tr>
<tr>
<td>Decimal</td>
<td>This option corresponds to the setting of the command “Debug Display Mode”.</td>
</tr>
<tr>
<td>Hexadecimal</td>
<td></td>
</tr>
</tbody>
</table>

Table 237: “Floating Point Variables”

<table>
<thead>
<tr>
<th>Number of displayed digits</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Decimal places that are represented in online mode when REAL values are displayed.</td>
<td></td>
</tr>
<tr>
<td>Note: The settings apply to the watch list, the monitoring of the declaration editor and the trace editor. The configuration for inline monitoring of the editor is set in the text editor options.</td>
<td></td>
</tr>
</tbody>
</table>

See also

- “Tab ’Monitoring’” on page 1088

Dialog 'Options' - 'PLCopenXML'

Symbol: 🔍

Function: This dialog contains settings for the behavior of CODESYS when exporting or importing PLCopenXM.

Call: Main menu “Tools ➔ Options”, category “PLCopenXML”

Table 238: “PLCopenXML Export Settings”

<table>
<thead>
<tr>
<th>Setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Additionally export declarations as plain text</td>
<td>By default, CODESYS splits the declaration parts in accordance with the PLCopenXML scheme into individual variables and thus loses the formatting and some comment information.</td>
</tr>
<tr>
<td>☑</td>
<td>Formatting and comments are retained. CODESYS additionally writes the plain text of the exported declaration part into the PLCopenXML file and thus extends the PLCopenXML scheme.</td>
</tr>
<tr>
<td>Export Folder Structure</td>
<td>CODESYS also exports the folders if they contain one of the selected objects. That is a CODESYS-specific extension to the PLCopenXML scheme.</td>
</tr>
</tbody>
</table>

Table 239: “PLCopenXML Import Settings”

<table>
<thead>
<tr>
<th>Setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Import folder structure</td>
<td>If the import file contains information about the folder structure of the objects, CODESYS also imports this structure.</td>
</tr>
<tr>
<td>☑</td>
<td>CODESYS imports objects without structure.</td>
</tr>
</tbody>
</table>

See also

- Chapter 1.3.1.4.1 “Exporting and importing projects” on page 135
- Chapter 1.3.1.25.2.4.22 “Command ‘Export PLCopenXML’” on page 911
- Chapter 1.3.1.25.2.4.23 “Command ‘Import PLCopenXML’” on page 911

Dialog 'Options' - 'Proxy Settings'

Symbol: 💻

Function: You use this dialog for storing the authentication data for the proxy server which is currently used for accessing the internet from CODESYS.

Call: Main menu “Tools ➔ Options”, category “Proxy Settings”
Requirement: Internet access of the network via proxy server

|“Enter proxy credentials”| A double click opens the input request for the user name and the password for the proxy server.
CODESYS uses the access data for the establishment of the connection to the download server for libraries and the device description, for the establishment of the connection to the CODESYS Store and for the command “View ➔ Start Page”.
Requirement: If the internet access of your computer or of the network takes place via a proxy server, then the button is available.

- ☰ Chapter 1.3.1.25.1.25 “Object 'Library Manager” on page 771
- ☰ Chapter 1.3.1.25.2.3.20 “Command ‘Start Page”’ on page 895
- Store

Dialog 'Options' - 'Refactoring'
Symbol: 🇧 🇪 🇪 🇫 🇫

Function: This dialog is used for defining the operations in the project for which the automatic refactoring is suggested. The refactoring functionality helps you in your improvement endeavors.

Call: Menu bar: “Tools ➔ Options” (“Refactoring”).

'Suggest Refactoring for the Following Operations’

|“Auto declare”| When you change the name of a variable in a declaration by calling auto-declare ([(Shift]+[F2]), the activated option “Apply changes by means of refactoring” appears. Then the “Refactoring” dialog opens and you can change the variable throughout the project.

☑: “On renaming variables” ☐: You specify the names in the auto-declare (dialog “Declare Variable”) and click “OK” to close the dialog. Then the “Refactoring” dialog opens for renaming the variable throughout the project.
See the section: “Refactoring”, “Changing a variable declaration and applying refactoring automatically”.

☑: “On adding or removing variables, or for changing the namespace” ☐: You delete the names in the auto-declare (dialog “Declare Variable”) and click “OK” to close the dialog. Then the “Refactoring” dialog opens for removing the variable throughout the project.

“Unit conversion editor” ☑: When you change the name of a conversion in the unit conversion editor, you are prompted whether CODESYS should perform “Automatic Refactoring” when renaming.

“Mapping editor” ☑: When you change a variable name in the device editor (“I/O Mapping” tab), you are prompted whether CODESYS should perform "Automatic Refactoring" when renaming.

“Navigator” ☑: When you change the name of an object in the device tree or in the POUs view, you are prompted whether CODESYS should perform "Automatic Refactoring” when renaming.

“Tabular declaration editor” ☑: When you change the name of a variable in the tabular declaration editor, you are prompted whether CODESYS should perform "Automatic Refactoring" when renaming.
See also

- § Chapter 1.3.1.9.16 “Refactoring” on page 236
- § Chapter 1.3.1.9 “Programming of applications” on page 171
- § Chapter 1.3.1.25.2.2.39 “Command ‘Refactoring’ - ‘Rename <...>’” on page 876
- § Chapter 1.3.1.25.2.2.31 “Command ‘Auto Declare’” on page 872
- § Chapter 1.3.1.25.1.44 “Object ‘Unit Conversion’” on page 849

Dialog Box 'Options' - 'Sequence Editor'

Symbol: 🛠

Function: This dialog box contains the settings for the sequence editor.

Call: Main menu “Tools ➔ Options”, category “Sequence Editor”

<table>
<thead>
<tr>
<th>Table 240: “General”</th>
</tr>
</thead>
<tbody>
<tr>
<td>Selection of the elements which are displayed in the sequence step of the sequence editor</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Table 241: “Appearance”</th>
</tr>
</thead>
<tbody>
<tr>
<td>Defines the appearance of the sequence steps and the connecting lines between the sequence steps and the fonts for the “Text”, “Parameters”, “References” and “I/O-Channels”.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>“A Connection”</th>
<th>Selection of the color of the connecting line after a branching step</th>
</tr>
</thead>
<tbody>
<tr>
<td>“B Connection”</td>
<td>Selection of the color of the connecting line after a branching step (alternative branch)</td>
</tr>
</tbody>
</table>

See also

- Application Composer - Sequence Editor

Dialog Box 'Options' - 'Syntax Highlighting'

Symbol: 📜

Function: This dialog box is used for configuring the color and font settings for the text elements of an editor (for example, operands and pragmas).

Call: Main menu “Tools ➔ Options”, category “Syntax Highlighting”

Dialog 'Options' - 'Text Editor'

Symbol: 🖋

Function: The dialog contains settings for working in a text editor.


Tab 'Theme'

In this tab, you set the theme for the ST editor.

| “Theme” | Color theme for the text editor. The set theme is shown in the “Preview” window. The available themes are stored in the Themes folder of the installation directory. |
Tab 'Editing'

<table>
<thead>
<tr>
<th><strong>“Number of undos”</strong></th>
<th>Maximum number of editing steps to which you can apply the command “Edit → Undo”.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>“Folding”</strong></td>
<td>Defines the structuring of the code by means of indentation. When you select an indentation, you can expand or collapse the indentation section with the help of a plus or a minus sign respectively before the first line of the respective section.</td>
</tr>
<tr>
<td></td>
<td>● “Indent”: CODESYS collects all lines that are indented in comparison with the preceding line in an indentation unit.</td>
</tr>
<tr>
<td></td>
<td>● “Explicit”: You mark explicitly the code section to be summarized in an indentation unit with comments. Before the section, there must be a comment containing 3 opening braces “{{{”. After the section, there must be a comment containing 3 closing braces “}}}}”. The comments can contain additional text. Example:</td>
</tr>
<tr>
<td></td>
<td><img src="image" alt="Example Code" /></td>
</tr>
<tr>
<td><strong>“Word wrap”</strong></td>
<td>● “Soft”: The word wrap takes place at the edge of the editor window if 0 is entered for “Wrap margin”.</td>
</tr>
<tr>
<td></td>
<td>● “Hard”: The word wrap takes place after the number of characters specified for the “Wrap margin”.</td>
</tr>
<tr>
<td><strong>“Tab width”</strong></td>
<td>Number of characters</td>
</tr>
<tr>
<td><strong>“Keep tabs”</strong></td>
<td>The empty space that you inserted with the [Tab] key is not resolved into spaces afterwards by CODESYS.</td>
</tr>
<tr>
<td><strong>“Indent width”</strong></td>
<td>If you have selected the “AutoIndent” option “Smart” or “Smart with code completion”, then CODESYS inserts the number of spaces at the beginning of the line.</td>
</tr>
<tr>
<td><strong>“AutoIndent”</strong></td>
<td>● “None”</td>
</tr>
<tr>
<td></td>
<td>● “Block”: A new line automatically applies the indentation of the preceding line.</td>
</tr>
<tr>
<td></td>
<td>● “Smart”: Lines that follow a line containing a keyword (for example <code>VAR</code>) are automatically indented by the specified indentation size.</td>
</tr>
<tr>
<td></td>
<td>● “Smart with code completion”: Indentation as with the “Smart” option, but CODESYS additionally inserts the concluding keyword (for example <code>END_VAR</code>).</td>
</tr>
</tbody>
</table>

Tab 'Text Area'
### “Matching brackets”
If the cursor is positioned before or after a bracket within a code line, CODESYS marks the matching closing or opening bracket with a frame in the set color.

### “End of line markers”
CODESYS marks the end of each editor line with a small dash in the set color behind the last character (including spaces) of the line.

### “Wrap guide”
If a soft or hard word wrap is activated, the defined word wrap point is indicated by a vertical line in the selected color.

### “Caret color”
Color of the cursor character

### “Selection color”
Color of the selected text area

### “Inactive”
Color of a selection when the corresponding window is not active (focus on another window).

### “Foreground reduced line”
Color of the header of a closed, indented section in the code

### “Background reduced line”
The header of a closed, indented section in the code is given a colored background.

### “Font”
Clicking the field opens the common dialog for the configuration of the font.

---

#### Tab 'Margin'

Settings for the left margin of the text editor window, which is separated from the input area by a vertical line:

<table>
<thead>
<tr>
<th>Setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Line numbering”</td>
<td>Display of the line numbers in the declaration and implementation part, in each case starting with 1</td>
</tr>
<tr>
<td>“Foreground color”</td>
<td>Color of the line numbers</td>
</tr>
<tr>
<td>“Background color”</td>
<td>Color of the margin</td>
</tr>
<tr>
<td>“Bracketing”</td>
<td>Bracketing encompasses the lines between the keywords that open and close a construct, for example <strong>IF</strong> and <strong>END_IF</strong>.</td>
</tr>
<tr>
<td></td>
<td>If this option is enabled and the cursor is located before, after or in one of the keywords of a construct, the bracketed area is indicated by a square bracket in the border. You can select the color of the bracket from the drop-down list of this option.</td>
</tr>
<tr>
<td>“Active border color”</td>
<td>Color of the dividing line between the border and input area</td>
</tr>
<tr>
<td>“Inactive border color”</td>
<td>Color of the dividing line between the border and input area of the part of the window that is currently inactive</td>
</tr>
</tbody>
</table>

| “Mouse actions”                | |<br>You can assign one of the following actions to each of the specified mouse actions or mouse button combinations. CODESYS executes the actions when you execute the mouse action on the plus or minus sign before the header of a parenthesized area: |
| “Select fold”                  | CODESYS selects all lines of the bracketed area.                           |
| “Toggle fold”                  | CODESYS opens or closes the bracketed area or, if there are nested brackets, the first level of the bracketed area. |
| “Toggle fold fully”            | CODESYS opens or closes all levels of a nested bracketed area.             |

---

#### Tab 'Monitoring'

Settings for the display of the monitoring fields:

<table>
<thead>
<tr>
<th>Setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Enable inline monitoring”</td>
<td>Display of the monitoring fields behind the variables in online mode</td>
</tr>
<tr>
<td>“Monitoring foreground color”</td>
<td>Display of the value in the monitoring field</td>
</tr>
<tr>
<td>“Monitoring background color”</td>
<td>Display of the background in the monitoring field</td>
</tr>
<tr>
<td>“Flow control foreground color”</td>
<td>Display of the value in the monitoring fields at the flow control positions</td>
</tr>
</tbody>
</table>
"Flow control background color"  Display of the background in the monitoring fields at the flow control positions

"Number of displayed digits"  Number of decimal places in the monitoring field

"String length"  Maximum length of string variable values in the monitoring field

See also

- ¶ Chapter 1.3.1.9.3.1 “Programming structured text (ST)” on page 201

Dialog 'Customize'

1.3.1.25.3.14.1  Dialog ‘Customize’ - 'Menu'................................................... 1089
1.3.1.25.3.14.2  Dialog ‘Customize’ - 'Command Icons' ................................. 1090
1.3.1.25.3.14.3  Dialog ‘Customize’ - 'Toolbars'............................................... 1090
1.3.1.25.3.14.4  Dialog Box ‘Customize’ - 'Keyboard' .................................... 1091

The dialog contains the tabs to configure the user interface.

You can reset the CODESYS settings to default by use of the “Reset” button.

Dialog 'Customize' - 'Menu'

Function: With this dialog, you define the structure and contents of the user interface.

Call: Main menu “Tools → Customize” (“Menu”).

When you click “OK” to close the dialog, the changes are visible in the menu bar of the CODESYS user interface.

Table 242: “Menu”

Display of currently defined menus, submenus, and included commands. In CODESYS, a menu or submenu caption is identified by the caption symbol (≡). The layout from top to bottom corresponds to the layout displayed later in the CODESYS menu.

| “Add Command” | Enabled when a command is selected.
|               | Adds a command above the selected command. Opens the “Add Command” dialog. Use the “Add Command” dialog for selecting one or more commands. Left part: List of categories. Right part: List of commands in the selected category. |
| “Add Separator” | Adds a separator above the selected command. |
| “Add Popup Menu” | Adds a popup menu above the selected menu, submenu, or command. Opens the “Add Popup Menu” dialog. |
| “Edit Popup Menu” | Opens the “Edit Popup Menu” dialog. |
| “Reset” | Resets the default settings of the entire menu. |
| “Load” | Loads the settings from a stored file (<file name>.opt.menu). |

Table 243: “Add Popup Menu”

In CODESYS, a new menu is shown in the menu bar only when the menu contains at least one command.

| “Default text” | Select this check box when localization is available. |
Localized Texts

List: Languages and localized texts.

Add Language

Opens a drop-down list of available languages.
In CODESYS, the selected language is displayed in the area “Localized Texts”.
Use the “Text” column for typing the localized texts.

See also

- Chapter 1.3.1.2.1 “Customizing menus” on page 103
- Chapter 1.3.1.25.3.14.3 “Dialog ‘Customize’ - ‘Toolbars’” on page 1090

Dialog ‘Customize’ - ‘Command Icons’

Function: This dialog defines the icons of the menu commands.

Table 244: “Command icon”

<table>
<thead>
<tr>
<th>“Assign”</th>
<th>Opens a dialog for selecting the new icon (*.ico).</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Remove”</td>
<td>Removes the user-defined icon. The default icon is active again.</td>
</tr>
<tr>
<td>“Reset”</td>
<td>Resets all default settings of the command icons.</td>
</tr>
<tr>
<td>“Load”</td>
<td>Loads the settings from a stored file (&lt;file name&gt;.opt.keyb).</td>
</tr>
<tr>
<td>“Save”</td>
<td>Saves the current settings to a file (&lt;file name&gt;.opt.keyb).</td>
</tr>
</tbody>
</table>

See also

- Chapter 1.3.1.1.2.3 “Customize command icon” on page 105

Dialog ‘Customize’ - ‘Toolbars’

Function: Use this dialog for generating new toolbars or customizing existing toolbars.
Call: Main menu “Tools ➔ Customize” (“Toolbars”).
When you click “OK” to close the dialog, the changes are visible in the menu bar of the CODESYS user interface.

Table 245: “Toolbars”

Display of currently defined toolbars. In CODESYS, the associated commands are listed below each toolbar in the order they will appear in the toolbar.
Double-clicking a toolbar in the list switches to editing mode.

<table>
<thead>
<tr>
<th>“Add Toolbar”</th>
<th>Enabled when a toolbar is selected. In CODESYS, this adds a toolbar above the selected toolbar and places the cursor in the name field of the new toolbar.</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Add Command”</td>
<td>Enabled when you select a command or blank command entry below a toolbar. Adds a command above the selected command. Opens the “Add Command” dialog. Use the “Add Command” dialog to select one or more commands. Left part: List of categories. Right part: List of commands in the selected category.</td>
</tr>
<tr>
<td>“Add Separator”</td>
<td>Adds a separator above the selected command.</td>
</tr>
<tr>
<td>“Hide”</td>
<td>Hide the selected toolbar from the user interface.</td>
</tr>
<tr>
<td>“Show”</td>
<td>Shows the selected hidden toolbar in the CODESYS user interface.</td>
</tr>
</tbody>
</table>
Dialog Box 'Customize' - 'Keyboard'

**Function:** This dialog box is used for defining keyboard shortcuts (quick access keys or keyboard combinations) for commands.

**Call:** Main menu "Tools ➔ Customize" ("Keyboard").

<table>
<thead>
<tr>
<th>Table 246: “Keyboard”</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>“Shortcuts for selected command”</strong></td>
</tr>
<tr>
<td><strong>“Press shortcut keys”</strong></td>
</tr>
<tr>
<td><strong>“Shortcut keys currently used by”</strong></td>
</tr>
<tr>
<td><strong>“Reset”</strong></td>
</tr>
<tr>
<td><strong>“Load”</strong></td>
</tr>
</tbody>
</table>

Dialog 'Trace Configuration'

Function: This dialog provides extended settings for recording data.

Call: "Advanced" button in "Trace Configuration" dialog, "Record Settings" subdialog

Requirement: The trace editor is open and active. The dialog "Trace configuration" is open and the top node of the trace record tree is selected so that the subdialog "Record settings" is available.

For the calculation of the values, you have to select a task in the "Trace Configuration" dialog.

The buffer size is defined as "number of samples". CODESYS calculates the time intervals that corresponds to this number and displays the result in normal fonts on the right outside the table (for example, "1h1m1s1ms"). The calculation is possible only with the help of the task configuration settings and when the task cycle time is known.

<table>
<thead>
<tr>
<th>&quot;Measurement in every nth cycle&quot;</th>
<th>Data recording in every n task cycle</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Preset:</strong> 1; then the application performs the data recording in each task cycle.</td>
<td></td>
</tr>
<tr>
<td><strong>Scanning interval of the data recording</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Example:</strong> 100ms</td>
<td></td>
</tr>
</tbody>
</table>
"Recommended runtime buffer size (samples)"

Requirement: "Override runtime buffer size" is deactivated.
The maximum number of samples that CODESYS calculates and recommends, which the application stores at runtime per trace variable. CODESYS calculates the number in the task cycle time from the value in "Measure in every n-th cycle" and the value in Measure in every n-th cycle.

Maximum length of the time interval during which the application collects data on the runtime system.

Example: 2s

"Override runtime buffer size"

Maximum number of samples per trace variable that saves the application per trace variable in runtime mode.

Example: 100

Value range: starting at 10

☑: The application uses this value, not the value calculated by CODESYS from "Recommended runtime buffer size (samples)".

Maximum length of the time interval during which the application collects data on the runtime system.

Example: 6s

"Trace editor buffer size per variable (samples)"

Number of values that can be stored per variable in the trace editor.

Example: 10000

The maximum time period for the display in the trace editor results from the maximum number and the scanning interval of the data recording. You can scroll back a maximum of this time in the trace editor.

See also

- ☞ Chapter 1.3.1.25.1.39 “Object ‘Trace’” on page 842
- ☞ Chapter 1.3.1.13.3.2 “Creating trace configuration” on page 338

Dialog 'Trace Configuration'

Symbol: 🎥

Function: This dialog includes the trace configuration for the data recording.

Call

- Menu bar: "Trace ➔ Configuration"; context menu.
- Link “Configuration” in the trace editor
- Link “Add Variable” in the trace editor

Requirement: The editor of a trace object is open and active.

See also

- ☞ Chapter 1.3.1.13.3.2 “Creating trace configuration” on page 338
- ☞ Chapter 1.3.1.13.3 “Data recording with trace” on page 335
- ☞ Chapter 1.3.1.24.6.2.21 “Attribute ‘monitoring’” on page 619

Tree view 'Trace record'

The tree view lists the variables that are traced and allows for access to the variable settings.

<table>
<thead>
<tr>
<th>Selected trace name</th>
<th>The subdialog “Record Settings” is displayed on the right.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Selected trace variable</td>
<td>The subdialog “Variable Settings” is displayed on the right.</td>
</tr>
</tbody>
</table>
**Table 247: Context menu commands**

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>+ “Add Variable”</td>
<td>Adds a new trace variable. The subdialog “Variable Settings” opens on the right and it is partially configured. Select a variable in the input field of the “Variable” setting to trace its value graph.</td>
</tr>
<tr>
<td>“Assign to Diagram”</td>
<td>Lists the diagrams (in the submenu on the right) where the selected variable is not currently displayed. Select a diagram to display the variable there. The command is available when a variable is selected in the tree view. Tip: When the command is deactivated, the variable is already displayed in all diagrams.</td>
</tr>
<tr>
<td>“Activated”</td>
<td>Activated by default Deactivated variables are displayed as disabled. They are neither displayed nor recorded.</td>
</tr>
</tbody>
</table>

**Tree view 'Presentation (Diagrams)'** The tree view lists the diagrams that are displayed in the trace editor and allows for access to their display mode.

<table>
<thead>
<tr>
<th>Selected node “Time axis”</th>
<th>The subdialog “Display Mode” for the time axis is displayed on the right. You can specify the time axis display. See below.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Selected diagram name</td>
<td>The settings for the coordinate system of the diagram and a preview are displayed on the right. See below.</td>
</tr>
<tr>
<td>Selected node “Y-axis”</td>
<td>The subdialog “Display Mode” is displayed on the right. You can specify the axis display. See below.</td>
</tr>
<tr>
<td>“Show variables”</td>
<td></td>
</tr>
<tr>
<td>Selected trace variable</td>
<td>The subdialog “Variable Settings” is displayed on the right. You can configure the trace variable. See below. Note: These are the same settings that can be accessed in the tree view “Trace Record”.</td>
</tr>
</tbody>
</table>

**Table 248: Context menu commands**

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>+ “Add Diagram”</td>
<td>Adds a new diagram below and displays it in the tree view “Presentation (Diagrams)”.</td>
</tr>
<tr>
<td>+ “Add New Variable”</td>
<td>Adds a new trace variable. The subdialog “Variable Settings” opens on the right and it is partially configured. Select a variable in the input field of the “Variable” setting to trace its value graph. Specify its display. In addition, the variable is assigned to the selected diagram.</td>
</tr>
<tr>
<td>“Add Existing Variable”</td>
<td>Lists all trace variables (in the submenu on the right) where the selected diagram is not currently displayed. Select a variable in order to display it in the selected diagram. Tip: When the command is deactivated, all trace variables are already displayed in the selected diagram.</td>
</tr>
</tbody>
</table>

**Subdialog 'Record Settings'** Requirement: The top node is selected in the “Trace record” tree view.
**Enable trigger**

- Triggers are activated. The trace data is buffered in runtime mode only when a trigger signal has been sent. You determine how the trigger signal is sent in the settings “Trigger variable”, “Trigger parameter”, “Trigger edge”, “Post trigger (Samples)”, and “Trigger level”.

- Continuous display of current records

**Trigger variable**

- Signal that is used as a trigger. A complete instance path is required.
- A valid trigger signal is an IEC variable, a property, a reference, an array element of the application, or an expression. Permitted types are all IEC-based types except STRING, WSTRING, and ARRAY. Enumerations are permitted when the base type is not STRING, WSTRING, or ARRAY. The contents of a pointer are not a valid signal.
- When the runtime system uses the CmpTraceMgr component, a property that is linked to the 'monitoring' attribute can then be recorded as a variable.

**Trigger parameter**

- System parameter that is used as a trigger.
- The “Input Assistant” dialog lists all valid system parameters in the “Parameters” category of the “Categories” tab.

**Trigger edge**

- Defined the edge detection for triggering:
  - **“positive”**
    - For Boolean trigger variables, triggering occurs when the values change from FALSE to TRUE.
    - For analog trigger variables, triggering occurs when the value as defined in “Trigger level” is reached from below.
  - **“negative”**
    - For Boolean trigger variables, triggering occurs when the values change from TRUE to FALSE.
    - For analog trigger variables, triggering occurs when the value as defined in “Trigger level” is reached from above.
  - **“both”**
    - For Boolean trigger variables, triggering occurs when the values change.
    - For analog trigger variables, triggering occurs when the value as defined in “Trigger level” is reached.

**Post trigger (samples)**

- Number of records per trace variable that are buffered after triggering. Preset: 50. Value range: 0 to \((2^{32} - 1)\)

**Trigger level**

- Value that is reached for triggering

**Task**

- Task where data was recorded

**Recording condition**

- In runtime mode, the application checks the recording condition. If it is fulfilled, then the trace data is buffered.
- Record condition for data recording with CmpTraceMgr runtime system component:
  - As an expression that includes only permitted operators and operands.
    - Permitted operators that can also be nested: (logical) AND, NOT, OR, comparison operators <, <=, >, >=, =, <>
    - Permitted operands: Variables that are valid for trace.
  - As a variable.
    - Permitted type: BOOL, bit access, property. The condition is fulfilled for TRUE or 1. The contents of a pointer are not permitted.
- Recording condition for a data recording with IEC code.
  - As an expression that returns a Boolean value.

**Comment**

- Comment (for example, from the recording condition)
**“Resolution”**  Measure for the time stamp that is recorded per data set  
- **“ms”**: Time stamp (in milliseconds).  
- **“µs”**: Time stamp (in microseconds) for a task cycle time of 1 ms or less

**“Automatic restart”**  Persistently saves the trace configuration and the last contents of the RTS buffer to the target device. After the device is restarted, the trace is started automatically if the trigger has not occurred yet.

**“Advanced”**  Opens the “Advanced Trace Settings” dialog.

See also  
- % Chapter 1.3.1.25.3.15.1 “Dialog ‘Advanced Trace Settings’” on page 1091

**Subdialog ‘Variable Settings’**  Requirement: A trace variable is selected in the tree view of “Trace Record” or “Display (Diagrams)”.

<table>
<thead>
<tr>
<th><strong>“Variable”</strong></th>
<th>Valid variable Variable; value recorded with full instance path.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Valid:</td>
</tr>
<tr>
<td></td>
<td>- IEC variable</td>
</tr>
<tr>
<td></td>
<td>- Property</td>
</tr>
<tr>
<td></td>
<td>- Reference</td>
</tr>
<tr>
<td></td>
<td>- Contents of the pointer</td>
</tr>
<tr>
<td></td>
<td>- Array element</td>
</tr>
<tr>
<td></td>
<td>Permitted data type</td>
</tr>
<tr>
<td></td>
<td>- IEC-based type except STRING, WSTRING, or ARRAY</td>
</tr>
<tr>
<td></td>
<td>- Enumeration when the base type is not STRING, WSTRING, or ARRAY</td>
</tr>
</tbody>
</table>

When the runtime system uses the CmpTraceMgr component, a property that is linked to the ’monitoring’ attribute can then be recorded as a variable.

<table>
<thead>
<tr>
<th><strong>“Parameter”</strong></th>
<th>Parameter whose data is recorded.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Requirement:</td>
<td>Runtime system with CmpTraceMgr component</td>
</tr>
<tr>
<td></td>
<td>The “Input Assistant” dialog lists all valid system parameters in the “Parameters” category of the “Categories” tab.</td>
</tr>
</tbody>
</table>

|
| Enables toggling between “Variable” and “Parameter” |

<table>
<thead>
<tr>
<th><strong>“Color”</strong></th>
<th>Color of the variable in the trace diagram</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th><strong>“Line type”</strong></th>
<th>Display as line chart</th>
</tr>
</thead>
<tbody>
<tr>
<td>- <strong>“Line”</strong>: Values are linked to form a line.</td>
<td></td>
</tr>
<tr>
<td>- <strong>“Step”</strong>: Values are linked in the form of steps</td>
<td></td>
</tr>
<tr>
<td>- <strong>“None”</strong>: Values are not linked</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>“Point type”</strong></th>
<th>Display as scatter chart</th>
</tr>
</thead>
<tbody>
<tr>
<td>- <strong>“Dot”</strong>: Value is displayed as a dot</td>
<td></td>
</tr>
<tr>
<td>- <strong>“Cross”</strong>: Value is displayed as a cross.</td>
<td></td>
</tr>
<tr>
<td>- <strong>“None”</strong>: value is not displayed</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>“Activate minimum warning”</strong></th>
<th>Warning when less than the lower limit</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th><strong>“Critical lower limit”</strong></th>
<th>If the value of the trace variable falls below the limit, the variable is displayed in the warning color.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th><strong>“Color”</strong></th>
<th>Warning color on falling below the limit</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th><strong>“Activate maximum warning”</strong></th>
<th>Warning when exceeding the upper limit</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>“Critical upper Limit”</strong></td>
<td>If the value of the trace variable exceeds the upper limit, the variable is displayed in the warning color.</td>
</tr>
<tr>
<td>--------------------------</td>
<td>---------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>“Color”</strong></td>
<td>Warning color on exceeding the limit</td>
</tr>
</tbody>
</table>

**Subdialog 'Display mode'**  
Requirement: An axis is selected in the tree view “Presentation (Diagrams)”

<table>
<thead>
<tr>
<th><strong>“Display mode”</strong></th>
<th>Scaling</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>“Auto”: Automatically scaled time axis</td>
</tr>
<tr>
<td></td>
<td>“Fixed length”: Time axis segment with a constant “Length”</td>
</tr>
<tr>
<td></td>
<td>“Fixed”: Time axis segment from “Minimum” to “Maximum”</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>“Minimum”</strong></th>
<th>Start value of the time axis segment. Requirement: The “Display Mode” is “Fixed”.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>“Maximum”</strong></td>
<td>End value of the time axis segment. Requirement: The “Display Mode” is “Fixed”.</td>
</tr>
<tr>
<td><strong>“Length”</strong></td>
<td>Constant segment length; the initial value is adapted automatically.</td>
</tr>
<tr>
<td><strong>“Grid”</strong></td>
<td>☑: Diagram with grid line in the x-direction. Select the grid line color from the dropdown list of colors.</td>
</tr>
</tbody>
</table>

**Table 249: “Tick Marks”**

<table>
<thead>
<tr>
<th><strong>“Fixed spacing”</strong></th>
<th>☑: Display of tick marks with “Distance” and “Subdivisions”.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>“Distance”</strong></td>
<td>Distance between tick marks</td>
</tr>
<tr>
<td><strong>“Subdivisions”</strong></td>
<td>Number of subdivisions between two tick marks</td>
</tr>
</tbody>
</table>

| **“Font”**          | Font for the time axis.                                                                     |

Link “Preview”  
Displays the preview of the diagram.

**Diagram preview**  
Requirement: A diagram is selected in the tree view “Presentation (Diagrams)”
**Dialog Box 'Trend storage'

**Function:** This dialog box includes the configuration for buffering the trend data of a trend recording.

**Call:** “Trend Storage” button in the editor of a trend recording.

<table>
<thead>
<tr>
<th><strong>Maximum number of variables</strong></th>
<th>Maximum number of trend variables that can be managed in the database. If you increase this value afterwards, then will CODESYS perform a download and reconfigure the database.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Store every N milliseconds</strong></td>
<td>Time interval (in ms) when the CmpTraceMgr runtime system component buffers the recorded data before storing it persistently in the database. The application calculates internally the number of task cycles from the time interval. The duration of a task cycle is defined in the task configuration.</td>
</tr>
<tr>
<td></td>
<td>A high value results in better runtime performance. The disadvantage is the increased risk of losing data if the controller crashes or shuts down. A low value reduces this risk. The disadvantage is the slower control over a trend visualization with large amounts of data.</td>
</tr>
</tbody>
</table>
| **Limit**                       | ☰: Limit the recording  
  ◆ “No Limit”: Unlimited number of data records (not recommended)  
  ◆ “Maximum number of records”: Maximum number of data records that are stored in the database. A data record consists of time stamp and the values of the trend variables at this time.  
  ◆ “Maximum storage size”: Maximum size of the trend storage. The application calculates internally the number of data records. Clicking the “down” symbol (▼) of the drop-down list will set the units to kilobytes (KB), megabytes (MB), or gigabytes (GB). |

See also

- ☰ Chapter 1.3.1.25.1.42 “Object 'Trend Recording” on page 846
- ☰ Chapter 1.3.1.13.4.1 “Getting started with trend recording” on page 345

**Dialog Box 'Advanced Trend Settings'

**Function:** This dialog box provides more settings for configuring trend recording.
Call: Click “Advanced” in the editor of a “TrendRecording” object.

| “Measure in every n-th cycle” | Frequency that the runtime system records data, depending on the number of processed task cycles. Select a value from the drop-down list or type a value into the input field. Using the settings from the task configuration, CODESYS calculates the time interval according to the frequency. Therefore, the calculation is possible only if at least the task cycle time is set. The result is shown on the right of the input field in normal syntax (for example, “1h1m1s1ms”). Default: 1 means that data is recorded in each task cycle. |
| “Additional Runtime Buffer for” | Length of the time interval when the runtime system can record more data (for example, 1000 ms). If a delay occurs when writing data in the runtime system component, then there is a risk of data loss due to overwriting. In this case, the runtime system uses the addition buffer. |

See also
- Chapter 1.3.1.25.1.42 “Object ‘Trend Recording’” on page 846
- Chapter 1.3.1.13.4 “Data recording with trend” on page 344
- Chapter 1.3.1.13.4.2 “Configuring trend recording” on page 346

Dialog ‘Certificate Selection’

Symbol: 

Function: This dialog is used for selecting the certificates for encryption, decryption, and digital signatures.

Call:
- “Security Screen” view, “User” tab
- Main menu: “View ➔ Properties”, “Encryption” tab when the “Application” is selected in the device tree.
- Main menu: “Project ➔ Project Settings”, category “Security”
The purpose of the certificate that is selected in the dialog depends on the call location:

- Call location: “Security screen” view, “User” tab
  - Certificate for digital signatures
  - Certificate for the decryption of project files
  - Certificate for encrypted communication
- Call: “View ➔ Properties” of the application
  - “Certificates of devices that share the encrypted download and the boot application”
- Call location: “Project ➔ Project settings”, category “Security”
  - “Certificate for project encryption”

<table>
<thead>
<tr>
<th>Listing of the selected certificates in a table</th>
<th>The following properties are displayed for each selected X.509 certificate:</th>
</tr>
</thead>
<tbody>
<tr>
<td>• “Created for”</td>
<td>• “Created by ”</td>
</tr>
<tr>
<td>• “Valid as of”</td>
<td>• “Valid until”</td>
</tr>
<tr>
<td>• “Thumbprint”: SHA1 fingerprint</td>
<td></td>
</tr>
</tbody>
</table>

Double-clicking an entry opens the “Certificate” dialog with the “General” tab, “Details” tab, and “Certification Path” tab. In that tab, you will find a reference to Windows help with more information about the dialog.

- Adds the selected available certificate to the list of selected certificates.
- Deletes the certificate selected in the list.

“Available certificates in the local Windows Certificate Store” Double-clicking an entry opens the “Certificate” dialog with the “General” tab, “Details” tab, and “Certification Path” tab. In that tab, you will find a reference to Windows help with more information about the dialog.

<table>
<thead>
<tr>
<th>Certificate icons</th>
<th>● ![Certificate with private key] Certificate with private key</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>● ![Untrusted certificate] Untrusted certificate</td>
</tr>
</tbody>
</table>

See also

- ☰ Chapter 1.3.1.25.2.3.18 “Command ‘Security Screen’” on page 891

1.3.2 Fieldbus Support

1.3.2.1 Device diagnostics

CODESYS supports the linking to the common fieldbus types.

For general information about using devices in a project, see chapter “�� Chapter 1.3.1.8.1 “Device tree and device editor” on page 155”.

For specific information about using devices for a certain fieldbus type, see the corresponding chapter here.

1.3.2.1 Device diagnostics

CODESYS provides general and fieldbus-specific function blocks for performing diagnostics on connected devices.

General diagnostics You can perform diagnostics on devices regardless of the fieldbus. The function blocks from the CAA Device Diagnosis library are provided for this purpose.
Before you can work with these function blocks, you must activate the option "Enable diagnosis for devices" in the PLC settings. This causes CODESYS to create instances of the diagnostics functions blocks automatically. These function blocks can be used for your diagnostics.

Please work exclusively with the automatically generated instances of the diagnostics function blocks. Do not create your own instances.

See also
- Chapter 1.3.1.25.1.19.9 “Tab ‘PLC Settings’” on page 751
- Library CAA DeviceDiagnosis

1.3.3 Runtime Systems, OPC UA Server

1.3.3.1 OPC UA server

The standard installation of CODESYS includes an OPC UA server. You can use it to access the variable interface of the controller via a client. The OPC UA server communicates with connected OPC UA clients over a separate TCP connection. Therefore, these connections have to be examined again separately with regard to security.

The OPC UA server can now be safeguarded by using encrypted communication to the client and OPC UA user management. See the following sections for these settings.

The CODESYS OPC UA server supports the following features:
- Browsing of data types and variables
- Standard read/write services
- Notification for value changes: subscription and monitored item services
- Encrypted communication according to "OPC UA standard (profile: Basic256SHA256)"
- Imaging of the IEC application according to "OPC UA Information Model for IEC 61131-3"
- Supported profile: Micro Embedded Device Server Profile
- By default, there is no restriction in the number of sessions, monitored items, and subscriptions. The number depends on the performance of the respective platform.
- Sending of Events according to the OPC UA standard.

Creating a project for OPC UA access

1. Create a new project with a CODESYS Control Win V3 controller.
2. Declare some variables of different types in the PLC_PRG program.
3. Add a “Symbol Configuration” object below the application.
4. In the “Add Symbol Configuration” dialog, select the “Support OPC UA Features” option.
5. Open the symbol configuration in the editor.
6. Click “Build”.
   ⇒ The variables are shown in a tree structure.
7. Select the variables that you want to change with an OPC UA client. Specify the access rights.
8. Download the project to the controller.
In order to encrypt data and exchange it with the client safely, the server needs a certificate that the client must classify as trusted when a connection is established for the first time. Requirement: The active path to the controller is set.

1. Install the CODESYS Security Agent add-on.
2. Click “View ➔ Security Screen”.
3. Select the “Devices” tab.
4. Select the controller in the left view.
   ⇒ All services of the controller that require a certificate are displayed in the right view.
5. Select the service “OPC UA Server”.
6. Create a new certificate for the device. Click the icon 🌐.
   ⇒ The “Certificate Settings” dialog opens.
7. Define the certificate parameters and click “OK” to close the dialog.
   ⇒ The certificate is created on the controller.
8. Restart the runtime system.

The OPC UA client "UaExpert" is freely accessible software that you can download from the Internet. Using this client, you can connect to the CODESYS OPC UA server. The following description refers to this program. Other OPC UA clients work in a similar way.

1. Start the "UaExpert" program.
2. Click “Server ➔ Add”.
   ⇒ The “Add Server” dialog opens.
3. Expand “Local ➔ OPCUAServer@...” in the tree view.
4. Select the connection type "Basic256Sha256 - Sign & Encrypt (uatcp-uasc-uabinary)" and click "OK" to close the dialog.

5. Click "Server ➔ Connect".
   - The "Certificate Validation" dialog opens with an error message.

6. Activate the option "Accept the server certificate temporarily for this session" and click "Continue".

7. In CODESYS Development System, click the symbol.
   - The view is refreshed.

8. Select the certificate folder "Quarantined Certificates".
   - The client certificate "UaExpert@..." is displayed in the right view.

9. Drag the certificate to the certificate folder "Trusted Certificates".
   - Now the client certificate is classified by the server as trusted.

    - The "Certificate Validation" dialog opens with an error message.

11. Activate the option "Accept the server certificate temporarily for this session" and click "Continue".
    - The connection is established and objects are displayed in the "Address Space" view.

**User management in OPC UA**

The CODESYS OPC UA server supports the CODESYS user management. You set the access rights on the server from the "Access Rights" tab on the controller. To do this, select the object "RuntimeSystemObjects ➔ RemoteConnections ➔ OPCUAServer".

Access rights can be checked at both the service and objects levels. This means that a variable cannot be written by a user, although this user is generally allowed to write to the OPC UA server.
### OPC UA service

<table>
<thead>
<tr>
<th>OPC UA service</th>
<th>Access Rights</th>
</tr>
</thead>
<tbody>
<tr>
<td>AttributeRead</td>
<td>View</td>
</tr>
<tr>
<td>AttributeRead</td>
<td>Change</td>
</tr>
<tr>
<td>CreateMonitoredItem</td>
<td>View</td>
</tr>
<tr>
<td>ModifyMonitoredItem</td>
<td>View</td>
</tr>
<tr>
<td>SetMonitoringMode</td>
<td>View</td>
</tr>
<tr>
<td>DeleteMonitoredItem</td>
<td>View</td>
</tr>
<tr>
<td>CloseSession</td>
<td>View</td>
</tr>
<tr>
<td>CreateSubscription</td>
<td>View</td>
</tr>
<tr>
<td>ModifySubscription</td>
<td>View</td>
</tr>
<tr>
<td>SetPublishingMode</td>
<td>View</td>
</tr>
<tr>
<td>DeleteSubscriptions</td>
<td>View</td>
</tr>
<tr>
<td>Publish</td>
<td>View</td>
</tr>
<tr>
<td>Republish</td>
<td>View</td>
</tr>
<tr>
<td>Browse</td>
<td>View</td>
</tr>
<tr>
<td>BrowseNext</td>
<td>View</td>
</tr>
<tr>
<td>TranslateBrowsePathsToNodeIds</td>
<td>View</td>
</tr>
<tr>
<td>RegisterNodes</td>
<td>View</td>
</tr>
<tr>
<td>UnregisterNodes</td>
<td>View</td>
</tr>
</tbody>
</table>

See also
- § Chapter 1.3.1.11.3 “Handling of device user management” on page 300
- § Chapter 1.3.1.25.1.19.14 “Tab 'Access Rights'” on page 762

### Changing a variable via the OPC UA client

1. Expand the object “Objects ➔ DeviceSet ➔ CODESYS OPC UA ➔ Application ➔ Global Vars ➔ GVL” in the ”UaExpert“ client of the view “Address Space”.  
   - The variables of the global variable list are visible.
2. Select the variables and drag them to the “Data Access View”.  
   - The variables and their current values are shown.
3. Change the variable values by double-clicking the “Value” field.
Creating events in the CODESYS project

The CODESYS OPC UA server provides the capability of sending standard OPC UA events.

1. Create a new project with a CODESYS Control Win V3 controller.
2. Add an “Alarm configuration” object below the application.
3. Add an “Alarm class” object below the “Alarm Configuration”. Specify a name, for example Event. 
   ⇨ The new alarm group opens in the editor.
4. Select the acknowledgement method “REP”.
5. Add an “Alarm group” object below the “Alarm Configuration”. Specify a name, for example ApplicationEvent. 
   ⇨ The new alarm group opens in the editor.
6. Change the following parameters:
   ● “Observation type”: “Event”
   ● “Class”: “Event”
   ● “Message”: ”Message 1”
7. Add an “Visualization” object below the “Application”.
8. Add a “Symbol configuration” object below the application.
9. In the Program (for example, POU PLC_PRG), add a program call for triggering the event alarm.

   ```
   AlarmManager.AlarmGlobals.g_AlarmHandler.RaiseEvent(Alm_AlarmConfiguration_Alarmgroup_IDs.ID_ApplicationEvent, 
   Alm_ApplicationEvent_Alarm_IDs.ID_0);
   ```
10. Add the library CmpOPCUAProviderAlarmConfiguration to the Library Manager. 
    When the library is added, it connects automatically as a client to the alarm configuration 
    and sends the events to the OPC UA server.
11. Send the project to the controller and start it.

See also
   ● § Chapter 1.3.1.25.1.7 “Object ‘Alarm Group’” on page 718
   ● § Chapter 1.3.1.9.21.3 “Calling event alarms in the program ” on page 261

Monitoring an event via the OPC UA client "UaExpert"

1. Start the "UaExpert" program.
2. Click “Server ➔ Add”. 
   ⇨ The “Add Server” dialog opens.
3. Expand “Local ➔ OPCUAServer@...” in the tree view.
4. Select the connection type “None” and click “OK” to close the dialog.
5. Click “Server ➔ Connect”. 
   ⇨ An object tree is shown in the “Address Space” view.
6. Click “Documents ➔ Add”. 
   ⇨ The “Add Document” dialog opens.
7. Select the "Document Type" “Event View”.
  ⇒ The “Event View” tab opens.
8. Expand the object “Objects ➔ DeviceSet ➔ CODESYS Control Win V3” in the “Address Space” view.
9. Select the object CODESYS Control Win V3 in “Address Space” and drag it to the “Event View”.
  ⇒ The events are displayed.

1.3.4 Libraries

Libraries are used for preparing POUs and functions for use in CODESYS applications. In addition to the descriptions presented here in the help, always see the documentation included in the library as well.

For using libraries in your CODESYS project, see the "Managing Libraries" chapter.
To create your own CODESYS libraries, follow the guidelines for library developers.

See also
●  Chapter 1.3.1.19 “Using libraries” on page 371

1.3.4.1 Guidelines for creating libraries

Libraries must be created according to specific rules to avoid compatibility issues.
The main items include the following:
● Select a meaningful library name (required)
● Use templates to ensure consistency (optional)
● Use a familiar and uniform project structure, when possible (optional)
● Register a unique library namespace (required)
● Enter all project information (required)
● Apply the correct method for referencing other libraries correctly (required)
● Design smart external and internal interfaces (required)
● Implement a user-friendly error handling (required)
● Apply the correct method (protection) for deployment (required)
● Apply a consistent naming convention to get clean code (optional)
● When revising an existing library, consider the interface compatibility with previous versions.
Please follow these guidelines when developing libraries in CODESYS: "Library Development Summary". You will find this document as a CHM file (LibDevSummary.chm) in the installation directory of CODESYS, or in the online help.

See also
- Chapter 1.3.1.19 “Using libraries” on page 371

1.3.5 CODESYS Visualization

Everything in one project
In the same CODESYS project, you use CODESYS Visualization to create the suitable user interface for your application. You link the visualization to the application variables and in this way they can animate and display data. When creating a visualization and an application, you use common functions, for example, as library and source code management or find/replace throughout the project.

Overview of functionality
- Display variant depending on the target platform
  You can execute the same visualization on various target platforms. Possible display variants are CODESYS WebVisu, CODESYS TargetVisu. In addition, there is a display integrated in the development system.
- Visualization editor
  In the graphic editor you design the desired user interface from visualization elements. The visualization elements are provided via libraries in a "ToolBox". You drag them into the editor area and adapt them with the help of a property configurator.
- Referenceable visualizations
  A visualization can be referenced in other visualizations. This enables the creation of user interfaces with a complex structure. For this purpose CODESYS Visualization also provides predefined visualizations, e.g. for dialogs.
- Simple design change
  The simple change of the look & feel of a visualization is possible in one place by creating a different visualization style.
- Multilingualism
  You can conveniently prepare visualization texts in several languages with the help of text lists. You can configure a user input element for switching to a different language in online mode.
- User management
  You can set up the visualization's own user management for access control up to individual element level.
- Other useful features
  Function block instances of visualizations, array accesses to the visualization, real-time data logging, extendability of the pool of visualization elements, provision of graphic objects via symbol libraries, calls of PLC functions from the visualization, reusability of visualizations by depositing them in libraries.
| **Table 250: Overview of the objects, editors, repositories, etc. relevant for the visualization in the CODESYS Development System** |
|--------------------|---------------------------------------------------------------------------------------------------------|
| **Visualization** | Object below an application in the device tree or in the POUs pool that contains a visualization image. A visualization can reference other visualizations. |
| **Visualization editor and additional views** | In this IEC 61131-3-compliant editor you can create the desired graphical user interfaces, panels, dialogs, etc. from visualization elements. The editor is made up of the following components:  
  - Graphic editor area for arranging the elements  
  - “Interface Editor”: for the parameterization of the visualization  
  - “Hotkey Configuration”: editor for defining keys for online operation  
  - “Elementlist”: overview of all visualization elements used, editor for the position of the elements on the z-axis  
  The following views are also available:  
  - “ToolBox”: view for the provision of visualization elements  
  - “Properties”: view with editor for the configuration of the element that currently has the focus in the graphic editor |
| **Visualization element** | Ready-to-use elements from the visualization libraries are available in the Tools view of the visualization editor for insertion. |
| **Visualization profile** | The profile defines which visualization elements are available. Each project that contains a visualization is based on such a profile (project settings). |
| **Visualization Styles** | The selected style determines the "look & feel" of the elements. It is set application-wide in the visualization manager. Ready-to-use styles are provided and you can also create your own. |
| **Visualization Manager** | Each application has a visualization manager of its own for its visualizations with various settings such as user management, style, language, input type, etc. The “Visualization Manager” object is suspended in the device tree below the application. |
| **Display variant** | A visualization can be displayed in online mode in the following variants, which are created as objects under the visualization manager:  
  - CODESYS TargetVisu (target visualization and remote target visualization on PLC devices)  
  - CODESYS WebVisu (web visualization via a web browser)  
  - Visualization integrated in the development system |
| **Visualization library** | Collection of visualization elements that are provided in the toolbox. |
| **Symbol library** | Collection of images and graphics that you can use in visualizations. When inserting a visualization object you can choose whether the installed system libraries should be available in the project. |
| **Visualization Element Repository** | Repository for the management of the visualization profiles and the visualization element libraries. |
The user interfaces created in CODESYS can be used in different display variants, depending on which ones the controller employed supports.

The display variants

- **Visualization ("diagnostic visualization") integrated in the CODESYS Development System:** The integrated visualization in the development system is ideal for application tests, for service or diagnostic purposes and for the commissioning of a system. As soon as a connection to the controller has been established, the visualization editor switches over and animates the elements displayed. This variant is part of the free CODESYS Development System and can always be used, irrespective of the controller employed.

- **CODESYS WebVisu:**
  This variant means web-based display of the user interface in a standard browser (PCs, tablets, smartphones), enabling remote access, remote monitoring and service and diagnosis of a system via the Internet. A standard web browser communicates by Java Script (optionally with SSL encryption) with the web server in the controller and displays the visualization by means of HTML5. This technology is supported by virtually all browsers and is thus also available on terminal devices with iOS or Android.

- **CODESYS TargetVisu:**
  This variant runs independent of the platform on control systems with an integrated display. Logic application and user interface run on the same device; the user interface is displayed directly on the controller. This variant is suitable for the operation and monitoring of machines and plants. An optional extension of the runtime system is required for the use of CODESYS TargetVisu.

### 1.3.5.1 Preparing CODESYS and projects

The following provides details of the presets that exist for visualizations and the steps that are necessary for creating a visualization in a project.

#### Presets

When you create a visualization in a project, you should know that the following presets apply:
### Scope

**Location**

- Throughout the application

**Setting**

- Unicode, CurrentVisu variable, multitouch, semi-transparency, memory size, data transmission, number of clients
- Visualization styles
- Language setting, language-specific font
- Default keyboard configuration
- Visualizations and visualization references
- Font for each language
- User management

<table>
<thead>
<tr>
<th>Single visualization</th>
<th>“Properties” of the visualization object</th>
</tr>
</thead>
<tbody>
<tr>
<td>Category</td>
<td>“Visualization”</td>
</tr>
<tr>
<td></td>
<td>• Purpose and scope of use</td>
</tr>
<tr>
<td></td>
<td>• Size definition</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Display variant of a single visualization</th>
<th>Editor of the WebVisu or TargetVisu object</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Start visualization, refresh rate, buffer size, html file name</td>
<td></td>
</tr>
<tr>
<td>• Scaling options</td>
<td></td>
</tr>
<tr>
<td>• Display options</td>
<td></td>
</tr>
<tr>
<td>• Default text input</td>
<td></td>
</tr>
</tbody>
</table>

### Creating visualization objects in the project

For each visualization, you insert a “Visualization” object into your project like any other object. This also applies to visualizations that should be used later only within other visualizations. You can insert the new visualization object directly below an application, or below the root node of the “Devices” view (for availability throughout the entire project).

The required base libraries and other objects, such as the Visualization Manager, are inserted automatically. When you insert the visualization object below an application, the subordinate objects for the display variants supported by the device are also displayed.

Every visualization object can be edited separately in the visualization editor.

The following steps describe a simply example for creating an object for an application-specific visualization.

**Requirement:** A project is open. An application is created in the device tree.

1. Select the application in the device tree. Click “Add object ➔ Visualization” in the context menu.

   The “Add visualization” dialog box opens. In the “Symbol libraries” table, there is at least the standard entry “VisuSymbols Vx.x.x. (System)”, and possibly other installed symbol libraries.

---

### Project-specific updates

Project-specific updates of the visualization profile, the visualization styles, and the visualization symbol libraries are possible in “Project ➔ Project environment” of the respective tabs.

Customization of the visualization menu is performed in “Tools ➔ Customize”.

See also

- ☞ Chapter 1.3.5.19.3.7 “Dialog ‘Options’ - ‘Visualization Styles’” on page 1597
- ☞ Chapter 1.3.5.19.3.13 “Dialog ‘Project Settings’ - ‘Visualization’” on page 1602
- ☞ Chapter 1.3.5.19.3.14 “Dialog ‘Project Settings’ - ‘Visualization Profile’” on page 1603
- ☞ Chapter 1.3.5.19.3.10 “Dialog ‘Project Environment’ - ‘Visualization Profile’” on page 1601
- ☞ Chapter 1.3.5.19.3.11 “Dialog ‘Project Environment’ - ‘Visualization Styles’” on page 1601
- ☞ Chapter 1.3.5.19.3.12 “Dialog ‘Project environment’ - ‘Visualization Symbols’” on page 1601
- ☞ Chapter 1.3.5.19.3.15 “Dialog ‘Properties’ of Visualization Objects” on page 1603
2. Accept the default name Visualization. Activate the “VisuSymbols” option. Then the visualization symbols (graphical objects) are contained in the library in the visualization project. Click “Add” to close the dialog box.

In the device tree, the “Visualization manager” and “Visualization” objects are inserted below the application. Depending on the device in use, the “TargetVisu” and/or “WebVisu” objects are also created below the visualization manager.

If a “TargetVisu” object or “WebVisu” object is created, then a “VISU_TASK” object is also created below the task configuration with an implicit program call.

The required visualization libraries are added automatically in the “Library Manager” of the application.

The visualization editor opens with the “Visualization” editor window and the “ToolBox” and “Properties” views.

In the “ToolBox” view, there is a “Symbols” button for viewing the symbols from the library VisuSymbols.library.

3. Now you can create the required visualization in the visualization editor.

4. Note: You can create structured visualizations by using a frame element to reference one visualization in another visualization. Dialog visualizations are a special option for this. In this case, the input configuration of a visualization element is used for referencing.

---

For creating an application-dependent visualization, insert the visualization object directly below the root node of the device tree. This corresponds to insertion in the “POUs” view. In this case, the visualization manager is not created with objects for the display variants.

See also

- § Chapter 1.3.5.19.3.7 “Dialog ’Options’ - ’Visualization Styles’” on page 1597
- § Chapter 1.3.5.19.3.13 “Dialog ’Project Settings’ - ’Visualization’” on page 1602
- § Chapter 1.3.5.19.3.15 “Dialog ’Properties’ of Visualization Objects” on page 1603

1.3.5.2 Limitation of the number of usable web pages on AC500 V3 PLCs

Automation Builder will get all the available visualizations in the project and count those reachable from the visualization client objects (WebVisu and RemoteTargetVisualization).

In case the predefined number of visualizations is exceeded an error is shown in the message window, preventing the user from compiling the project.

The error will be shown under “Build” category when the user executes the build command. The PLC program won’t download to the PLC until this error is solved (like with any other build errors). In the image below, there are 5 visualizations being used (3 of them added directly into the Automation Builder project and the other 2 referenced from a library that was added to the project).
The error will look like this when build command is executed:

```
Messages - Total 3 error(s), 2 warning(s), 2 message(s)
Build       - 1 error(s)  1 warning(s)  0 message(s)
```

If the visualizations are in the project but not being referenced (e.g. not reachable from the “Start Visualization” in the WebVisu) they are not taken into account for this limitation. If the error condition is solved, the error will disappear when the user executes the build command again.

1.3.5.3 Designing a visualization with elements

The visualization editor provides the visualization elements for the design of a user interface in the view “Tools”.

Drag the desired element into the editor window and adapt it in the view “Properties”: purely visual design, labelling, display of data, reaction to user inputs, possibility to input values, etc. Static or dynamic configuration of the properties is possible, i.e. the assignment of fixed values or application variables. A dynamic configuration allows an animation that is executed at runtime.

See also

- Chapter 1.3.5.19.4.1.1 “Visualization editor” on page 1608
1.3.5.3.1 Selecting an element

The view “Toolbox” provides the following elements for selection:

- All visualization elements that define the set visualization profile.
- Image elements for all images in the project from the integrated libraries or the symbol libraries.
- Frame elements for all visualizations in the project or from the libraries.

The currently set visualization profile and the symbol libraries currently in use can be seen in the “Project Settings”.

The elements are summarized into certain categories, for each of which there is a button in “Toolbox”. These categories are created by providing each element with a certain "identifier", which appears as the button label.

The elements of all "pressed" identifiers appear in “Toolbox” as thumbnails for selection. It is also possible to search by element name.

Simply drag the element thumbnail with the mouse to the desired position in the editor window. The configurable properties of the element then automatically appear in the view “Properties” of the visualization editor.

See also

- Chapter 1.3.5.19.4.1.1 “Visualization editor” on page 1608
- Chapter 1.3.5.19.4.1.2 “View ’Toolbox’” on page 1609
- Chapter 1.3.5.18.1.5 “Visualization element 'Image'” on page 1272
- Chapter 1.3.5.18.1.6 “Visualization element 'Frame'” on page 1287

Creating or removing an identifier for element categories

Requirement: the visualization editor is opened.

1. Click on the button in the view “Toolbox”.

   - The dialog “Configure Tags and Items” appears.

2. In the dialog, open the “Add Tag” dialog by clicking on . Please note: you can delete the definition of an identifier with – or [Del].

3. Enter a name in the “Name” input field, for example tagA, and exit the dialog with “OK”.

   - The new user-defined identifier tagA is inserted at the bottom in the tree view in the dialog “Configure Tags and Items”. It is provided with the symbol .

4. Click on the option “Activate” for the new identifier and exit the dialog with “OK”.

   - CODESYS adds a button "tagA" to the view "Toolbox". By clicking on the button you can filter according to the new identifier.

See also

- Chapter 1.3.5.19.3.4 “Dialog ‘Configure Tags and Items’” on page 1585

Marking a visualization element

Requirement: the visualization editor is opened. You have already created a user-defined identifier tagA. A button labelled with tagA is visible in the view “Tools”.

1. Click on an element in the view “Toolbox” and open the context menu.

   - A context menu appears. It contains the commands “Add Entry to Identifier 'tagA'” and “Add to Identifier".
2. Select the command “Add Entry to Identifier "tagA"” and close the dialog with “OK”.
   ⇒ The element is marked with 'tagA'.
3. Click on the button “tagA”.
   ⇒ All elements assigned to this identifier appear, including the element that has just been assigned.

See also
   ● ❝ Chapter 1.3.5.19.4.1.2 “View 'Toolbox'” on page 1609

1.3.5.3.2 Positioning elements, adapting size and layer

A visualization is a raster image in pixels. The pixel position is specified in x/y-coordinates. The origin (0,0) is located at the upper left corner of the window. The positive x-axis runs to the right, and the positive y-axis runs downwards. The position of an element on the z-axis of the visualization is controlled by the position in the element list (see below).

Configuring the size and position in the editor

The size and position of an element are specified as pixel coordinates in the “Properties” view. These settings are displayed graphically in the editor view at the same time.

When you drag a visualization element from the “Toolbox” view to the editor view, it is shown as selected, as in the following example of a rectangle element:

The possible positions depend on the set grid. You can change its settings CODESYS options. Commands in the context menu are available for alignment and grouping.

You can now move or resize the element directly in the editor. As an alternative, you configure the “Position” property in the properties editor, which opens automatically for the selected element. Refer to the description for this, for example in the help page for the button element. The changes are also updated in the other editor.

1. Focus the element so that the shape of the mouse pointer indicates movement (example: 🔄).
2. Drag the element to any position.
   ⇒ The position of the element is also updated in the properties “Position ➔ X” and “Position ➔ Y”.
3. Focus on a blue box.
   ⇒ The shape of the mouse pointer is a double arrow that indicates the direction you can drag the box in order to resize the box 🔄.
4. Drag the blue box to resize the element.
   ⇒ The position of the element is also updated in the properties “Position ➔ X” and “Position ➔ Y”.

In addition, you can rotate the rectangle, line, polygon, and pie elements.
1. Select the element for static rotation. Example: Rectangle
   ⇒ The rectangle is displayed with a handle next to the movable position boxes.

   (1) Handle

2. Drag the mouse pointer over the handle.
   ⇒ The cursor is displayed as a rotating arrow 🔄.

3. Rotate the element to any position.
   ⇒ In the property “Position ➔ Angle”, the set angle is displayed in degrees.

See also

●

Moving the visualization element forward and back

Each visualization element is in its own layer of the visualization (z-axis). It can be hidden by other elements in the foreground and hide other elements in the background. The order of layers is visible in the “Element List” tab above the editor view. The order of elements from front to back specifies the order of visualization layers from back to front.

Use the commands from the “Order” context menu to move a selected element.

Example of an element list (1):
1.3.5.3.3 Assigning a color

You configure the color of a visualization element either statically by means of the “Color” property, or dynamically by assigning an application variable by means of the “Color variables” property. Depending on the element, color assignments are also available in other properties. For example, for the font color, this is provided in the “Text” property of a labeled element.

For the static assignment of a color value, you can always use the color dialog in the properties editor, which provides color palettes to choose from.

You can specify the color as a style color. Style colors are color names for color definitions from the actively applied style. When configuring an corresponding property, you are provided with a list of available style colors. **We recommend that you use style colors** because then you can change colors centrally by means of a style selection or a style customization. You can also open the “Color” dialog to select a value from color palettes.

In addition, you can define the fill color of an element as a “Gradient”. Then the color changes linearly, radially, or axially from the initial color to the final color. You configure the “Gradient setting” in the “Gradient Editor” dialog.
Designing a visualization element with a style color or a fixed color value

- Requirement: The visualization editor is open.
  1. Insert some Rectangle elements.
  2. Select an element.
     - The “Properties” view is active.
  3. Click in the “Colors ➔ Normal state ➔ Fill color” property.
     - A list box and the button appear.
  4. Assign a style color to the rectangle. For example, select “Elementfillcolor” from the list box.
  5. Define the degree of transparency in the “Colors ➔ Normal state ➔ Fill color ➔ Transparency” property. Use the slider to select the value “136”.
  6. Select another rectangle. Click in the “Colors ➔ Normal state ➔ Fill color” property.
     - A list box and the button appear.
  7. Assign a fixed color value to the rectangle. Click to do this.
     - The “Color” dialog opens.
  8. Select a standard color or “Define Custom Colors” to fine-tune your selection. Then click “OK”.
     - The color is set as a fixed value. The color is displayed as a small rectangle. The RGB values are also indicated next to it.
  9. Click in the “Colors ➔ Normal state ➔ Fill color ➔ Transparency” property.
  10. Use the slider to select the value “136”.
     - The color is semitransparent.

See also
- “Chapter 1.3.5.17 “Applying visualization styles” on page 1216

Designing a visualization element with a color gradient

- Requirement: The visualization editor is open.
  1. Drag a “Rectangle” element to the visualization.
  2. Select the “Colors ➔ Use gradient color” property.
  3. Click in the “Colors ➔ Gradient setting” property.
     - The “Gradient Editor” dialog opens.
4. Define the color gradient for the element:
   - “Gradient type”: “Radial”
   - “Standard radial”: “Center”
   ⇐ The fill color of the element changes radially from white to black.

See also
- ⇐ Chapter 1.3.5.19.3.5 “Dialog ‘Gradient Editor’” on page 1586

**Configuring a visualization element for color animation**

The “Color variables” property, which certain elements may have, is used for the color animation of the element. If you assign a variable there, then you can program color changes in the application code or configure a user input that results in a color change.

You can see an example in the "Animating Visualization Elements" chapter.

See also
- -
- ⇐ Chapter 1.3.5.17 “Applying visualization styles” on page 1216
- ⇐ Chapter 1.3.5.8.3 “Animating a color display” on page 1152

### 1.3.5.3.4 Using texts

You can get displayed text in an element by assigning a string in the element property “Texts ➔ Text”. For example all base elements have this property. Also, you can get displayed a text as a tooltip (element property “Texts ➔ Tooltip”). Texts assigned in this way are static. They are managed in the object “GlobalTextList” in view “POUs” and they cannot be modified during runtime, neither programmatically nor via an user input.

However, you can extend a static text by (exactly) 1 placeholder containing a formatting specification, in order to output the content of a variable at this place. At runtime the current value of the variable, which you have assigned to the element via property “Textvariable”, will be output.

```
On the possible formatting specifications please see: ⇐ Chapter 1.3.5.18.2
“Placeholders with format definition in the output text” on page 1546
```

By dynamic configuration you can animate the optical representation of the text.

You can localize the static texts, if you have set up multilingualism in your project.

See some examples for the text configuration of visualization elements in the following chapters.

See also
- ⇐ Chapter 1.3.5.18.2 “Placeholders with format definition in the output text” on page 1546
- ⇐ Chapter 1.3.5.6 “Setting up multiple languages” on page 1143
Labeling an image element with a static text

Precondition: A project containing a visualization is opened. You have an image file representing a stop symbol.

1. Below the Application object insert an object “Image Pool” named ImagePool_A.
2. In the image pool ImagePool_A add your stop symbol image file with ID Stop.
3. Open the visualization and from the ToolBox draw an element “Image” into the editor.
   ➔ The input assistant opens. In tab Category you see the image pool ImagePool_A.
4. Select the image Stop and close the dialog with “OK”.
5. Configure the property “Text” of the image: ImagePool_A, Stop
6. Configure the property “Text properties ➔ Horizontal alignment”: Left.
7. Configure the property “Text properties ➔ Vertical alignment”: Bottom.

Text output: Element outputs the result of ST code which is executed on a mouse-click

Precondition: A project containing a visualization is opened.

1. Open the visualization and insert a “Button” element.
   ➔ The “Properties” view opens for the new element.
2. Configure property “Text”: Number of clicks: %I
   ➔ The string contains the placeholder %I.
3. In POU PLC_PRG of the application declare a type-conform variable: iClicks : INT;
4. Configure the property “Text variable” of the button element with PLC_PRG.iClicks.
   ➔ At runtime the variable value will be output instead of the placeholder.
5. Below property “Inputconfiguration”, in the cell containing the input event OnMouseClick, click on “Configure”.
6. From the list of possible actions choose Execute ST-Code.
7. Enter the code for the action in the editor “Execute ST-Code”:
   PLC_PRG.iClicks := PLC_PRG.iClicks + 1;
8. Close the dialog with “OK”.
   ➔ The user input is configured.
9. Build, download and start the application.
   ➔ The application is running. The visualization opens. The element is labeled and the number of clicks will be output. If you as user click on the button, the number will be increased.

   ![Image of a button with the label 'Number of clicks: 4']

**Text output:**

**Dynamic output using a textlist**

Using the “Text field” element you can produce a dynamic text output. The text output can be effected via an user input or via the application program.

☐ Precondition: A project containing a visualization is opened.
1. Open the visualization and insert a “Text field” element.
   ➔ The “Properties” view shows the configuration of the element.
2. Below the application add a Text List with the following entries: Textlist_A.

   ![Image of a text list with entries]

3. In POU PLC_PRG of the application declare the text variable: strTextID : STRING := '0';
4. Also declare the variable strTooltipID : STRING := '0';
5. Also declare the variable iText : INT;
6. Configure the property “Dynamic texts ➔ Text List” with 'Textlist_A'.
7. Configure the property “Dynamic texts ➔ Text index” with PLC_PRG.strTextID.
8. Configure the property “Dynamic texts ➔ Tooltip index” with PLC_PRG.strTooltipID.
9. In POU PLC_PRG implement the CASE instruction as shown below.
   ➔ The variables in property “Dynamic Texts” are programmed.
10. Configure the property “Inputconfiguration ➔ OnMouseclick” for Execute ST-Code with PLC_PRG.iText := (PLC_PRG.iText + 1) MOD 4;

⇒ For element “Text field” an user input is configured.

11. Build, download and start the application.

⇒ The application is running. The visualization opens. In the text field the text None is output. When you as user click on the element, the text changes to Dynamic_File_A. And the matching tooltip is available: Information A. With each click the text changes according to the CASE instruction.

```plaintext
CASE iText OF
  0:    strTextID := '0';
        strToolTipID := '0';

  1:    strTextID := '1';
        strToolTipID := '4';

  2:    strTextID := '2';
        strToolTipID := '5';

  3:    strTextID := '3';
        strToolTipID := '6';
ELSE
  strTextID := '0';
  strToolTipID := '0';
END_CASE;
```

In property “Text ➔ Text” you can define a text in order to get a static text output. A text in “Text ➔ Tooltip” will be displayed as tooltip. You can configure the text in a way, that the content of a variable is additionally output.

You can extend a static text by (exactly) 1 placeholder including a formatting definition, in order to output the content of a variable at this place at runtime. The variable must be assigned in property “Text variable”. When the variable value changes in the application code, then at the same time the output in the visualization changes.

☑ Precondition: A project containing a visualization is opened.

1. Open the visualization and insert an element “Text field”.

⇒ The “Properties” view shows the element configuration.

2. Configure the property “Texte ➔ Text”: File name: %s

⇒ The text contains the placeholder %s.

3. In POU PLC_PRG of the application declare a type-conform variable strFileName : STRING := 'File_A';

4. Configure the property “Text variable” of the text field with PLC_PRG.strFileName.

⇒ At runtime the variable value will be output instead of the placeholder.

5. Build, download and start the application.

⇒ The application is running. The visualization opens. The text field element displays the text: File name: File_A
Configuring text input in a text field

You can use the “Text field” element in order to output the text given by a variable, or to provide a place, where the user can give input on the variable.

Additionally you can configure a text input. In this case on an user input an input field in the element “Text field” will appear. As a precondition you must have configured an user input action in the property “Inputconfiguration”.

See also

- Chapter 1.3.5.6 “Setting up multiple languages” on page 1143

Showing text as a tooltip

1. Open the visualization and insert an element “Text field”.
   - The “Properties” view shows the element configuration.
2. Configure the property “Texts ➔ Text”: File name: %s
   - The text contains the placeholder %s
3. Configure the property “Texts ➔ Tooltip”: Storage location: %s
4. In POU PLC_PRG of the application declare a type-conform variable strFileName : STRING := 'File_A';
5. In POU PLC_PRG of the application declare also the variable strFileDir : STRING := 'D:/Data';
6. Configure the property “Text variable” of the text field with PLC_PRG.strFileName.
   - At runtime the variable value will be output instead of the placeholder.
7. Configure the property “Tooltip” of the text field with PLC_PRG.strFileDir.
8. Build, download and start the application.
   - The application is running. The visualization opens. The text field element shows the text File name: File_A. When the mouse cursor is moved above the text field, the tooltip will be displayed: Storage location : D:/Data.

Configuring element “Text field” for text input

The user should be able to enter text in a text field. For this configure an input of type “Write variable” on a text output variable. This text output variable will store the text input of the user and will display this text instead of the placeholder (this is %s in the example below). You specify the text output variable in the property “Text variables ➔ Text variable”.

See also

- Chapter 1.3.5.18.2 “Placeholders with format definition in the output text” on page 1546
Precondition: A project containing a visualization is opened.

1. In POU PLC_PRG of the application declare a string variable: strInput : STRING;
2. Open the visualization and insert an element “Text field”.
    ☀ The “Properties” view shows the configuration of the element.
3. In property “Text” enter Input: %s.
4. In property “Inputconfiguration” for mouse action “OnMouseClicked” click on “Configure” to open the “Input Configuration” dialog box. There choose action “Write a Variable” and activate option “Use text output variable”. Close with “OK”.
5. In the element property “Text variables ➔ Text variable” assign the text output variable PLC_PRG.strInput.
6. Build, download and start the application.
    ☀ The application is running. The visualization opens. The element outputs the text: Input:. Click in the element to open an input field, where you can enter a string. After having terminated the input by [Enter], the text will be adopted.

Animating the text display
Configure the property “Font variables” in order to animate the display of a text. All base elements have this property, additionally the table, scrollbar and text field element.

See also
● ¶ Chapter 1.3.5.8.2 “Animating a text display” on page 1152

Configuring the ’Label’ element
Precondition: A project containing a visualization is opened.
1. Open a visualization and insert an element “Label”.
    ☀ The “Properties” view with the pre-set property configuration opens.
2. Configure the property “Text ➔ Text” Visualization A.

Making an element invisible
You can configure the property “State variables ➔ Invisible” in order to hide an element in the visualization.

Precondition: In the visualization you have configured a text field, which gets visible only, if a certain application variable gets TRUE. For example in order to show certain instructions or descriptions only in a certain state of the machine.

1. For the text field element configure the property “Text ➔ Text” with Error detected: Do the following....
    Configure the property “Text properties ➔ Font color” with “dark red”.
2. In PLC_PRG declare the variable bIsInvisible : BOOL : TRUE; (this is the initialization for the current example; normally the variable should be set to TRUE by the application program under certain conditions.
3. Configure the property “State variables ➔ Invisible” with PLC_PRG.bIsInvisible.
4. Build, download and start the application.
    ☀ The application is running. The visualization opens and the text field is not visible. When you set bIsInvisible to TRUE, the textfield will be displayed.
1.3.5.3.5 How to display variable values in the visualization

There are simple to very specialized visualization elements for displaying data from a running application.

Examples:

- Simple output of variable values: For example, you can configure a purely formatting specification for a “Rectangle” element in the “Text” property and the variable whose value is to be displayed in the element in the “Text variable” property.
- Display of structured variable values (structure, array, function block): You use the “Table” element and specify an array variable in its configuration in the “Data array” property whose values are to be displayed in the table. One-dimensional arrays can also be displayed in a “Histogram”.
- Display of values by image switching. Example: A specific screen is displayed depending on the error message that occurs. You do this by configuring an “Image” element with a variable for the “Bitmap ID variable” property.
- Display a variable value as a bar or with a pointer on a scale: You specify a variable in the “Value” property of the “Bar Display” element or “Meter” element to display its value as a bar on a horizontal or circular scale.
- Display of alarms: The alarms configured in the alarm management of the application can be made visible by means of the “Alarm Table” and “Alarm Banner” elements in the user interface.
- “Trace” and “Trend”: For graphical recording of variable values over a period of time.

For details, see the descriptions of the element properties.

See also

- Chapter 1.3.5.18.1 “Visualization elements” on page 1223
- Chapter 1.3.5.21.2 “Displaying array data in histograms” on page 1958

1.3.5.3.6 How to change variable values via the visualization

In addition to displaying values from the controller, a user interface is also used to enter and change values.

In general, you can configure user input for each element in its “Input configuration” properties. Moreover, elements have been developed especially for specific input.

Examples:

- A “Button” element (or “Rectangle” element, and so on) that is clicked to open a predefined dialog visualization for easily specifying a value.
- A “Slider” element for changing the value of a variable by moving visual element parts, for example with the mouse. In the case of the slider: The element adjusts the value of a variable, depending on the position of the slider within the slider. You define the value range of the slider bar by means of the scale start and scale end.
- A switch element (example; “Power Switch”) for setting a Boolean value.
- A “SpinControl” element for incrementing or decrementing the value of a variable in defined intervals.
- A “Button” element for writing a recipe, executing a specific ST code, writing a specific variable, and so on (definition in the input configuration).

See also

- Chapter 1.3.5.18.1 “Visualization elements” on page 1223
1.3.5.3.7 Designing a background

You can design the background of your visualization in color or with an image. To do this, use the command “Visualization ➔ Background”.

See also
- Chapter 1.3.5.19.2.10 “Command 'Background'” on page 1565
- Chapter 1.3.5.19.3.15 “Dialog 'Properties' of Visualization Objects” on page 1603

In addition, you can use the property “Integrate background” in the dialog “Properties” of a visualization object to specify whether the background image should always be displayed in its entirety or whether it should be truncated.

Requirement: A project with a visualization is open.

1. Open the visualization and select the command “Visualization ➔ Background”.
2. Activate the option “Image” and open the input assistant.
3. Select an image in the dialog “Input Assistant”.
   - The image serves as a background image.

See also
- Chapter 1.3.5.19.3.15 “Dialog 'Properties' of Visualization Objects” on page 1603

1.3.5.4 Configuring user inputs

User inputs for a visualization are configured in order to operate the visualization.

For this purpose, you configure input events on visualization elements where follow-up actions are triggered. The combination of user inputs and follow-up actions are defined in the “Input configuration” of an element. For example, you can select a mouse click on an element as the input event and opening a dialog box as the input action.

Keyboard events can also be configured that trigger actions in a specific visualization window when the events occur. You program this kind of input configuration for a visualization in its “Keyboard configuration” editor.

In addition, keyboard events can be configured that occur in all visualizations programmed in the application. You configure this kind of input configuration per application below the visualization manager in the “Standard keyboard shortcuts” tab.

Input is usually performed with the mouse and keyboard as controlling device. You can also configure a user operation by means of gestures.

If a visualization device is not equipped with a mouse, then you can activate default keyboard usage. Then a user can operate the visualization with the keyboard only by navigating with the arrow keys and triggering events by pressing the [Enter] key.

Configuring an image as a background

☐ Requirement: A project with a visualization is open.
1. Open the visualization and select the command “Visualization ➔ Background”.
2. Activate the option “Image” and open the input assistant.
3. Select an image in the dialog “Input Assistant”.
   - The image serves as a background image.

See also
- Chapter 1.3.5.19.3.15 “Dialog 'Properties' of Visualization Objects” on page 1603

Configuring a colored background

☐ Requirement: A project with a visualization is open.
1. Open the visualization and select the command “Visualization ➔ Background”.
2. Activate the option “Color”.
3. Select a style color such as “Element background color” from the selection list.
   - The background of the visualization is colored.
If a visualization device is not equipped with a keyboard, then you can call a virtual keyboard or a virtual numeric keypad.

**NOTICE!**
Configure keyboard events only for keys that the visualization device supports.

See also
- Chapter 1.3.5.19.2.2 “Command ‘Keyboard Configuration’” on page 1558
- Chapter 1.3.5.19.3.6 “Dialog ‘Input Configuration’” on page 1586
- Chapter 1.3.5.19.4.3 “Tab ‘Visualization Manager’ - ‘Default Hotkeys’” on page 1617

**Processing order of keyboard events**

1. Event handler of the application. Requirement: The event handler is activated.
2. Events of the default keyboard usage
3. Events of the keyboard usage are configured in the tab “Visualization manager” - “Default hotkeys”.
4. Events of keyboard usage are configured in the tab “Keyboard configuration” for the currently visible visualization.

See also
- Chapter 1.3.5.19.1 “Keyboard Shortcuts for Default Keyboard Action” on page 1555
- “Tab ‘Keyboard configuration’” on page 1558 ➔ Chapter 1.3.5.19.2.2 “Command ‘Keyboard Configuration’” on page 1558
- Chapter 1.3.5.19.4.3 “Tab ‘Visualization Manager’ - ‘Default Hotkeys’” on page 1617
- Chapter 1.3.5.19.4.3 “Tab ‘Visualization Manager’ - ‘Default Hotkeys’” on page 1617

**1.3.5.4.1 Configuring user inputs for visualization elements**

All base elements and some common control elements have the “Input configuration” property. This is where you can configure a user input for an element. For this purpose, you select an input event and an input action.

**Configuring user inputs**

- **Requirement:** A project is open with a visualization.
- 1. Open the visualization and added a “Button” element.
   ➔ The “Properties” view opens for the new button.
- 2. Configure the property “Text” with **Number of clicks: %i**.
- 3. Declare a variable **iClicks : INT;** in the application in the PLC_PRG POU.
- 4. Configure the “Text variable” property of the button as **PLC_PRG.iClicks**.
   ➔ At runtime, its variable value will replace the placeholder in the “Text” property.
- 5. In the “Input configuration” property, click the “Configure” button in the **OnMouseClick** line.
6. Select the **Execute ST code** action from the list of possible actions and click the symbol.

   ⇨ The action appears in the list of actions to be executed. The blank implementation of the action appears in the window area to the right of the list.

7. Program the action in the editor at “**Execute ST code**”:

   ```
   PLC_PRG.iClicks := PLC_PRG.iClicks + 1;
   ```

8. Click “**OK**” to close the dialog box.

   ⇨ The user input is configured.

9. Compile, download, and start the application.

   ⇨ The application runs. The visualization opens. If the user clicks the button, then the action is executed, the variable `PLC_PRG.iClicks` is incremented, and the number of clicks is printed.

See also

- ° Chapter 1.3.5.19.3.6 “Dialog ‘Input Configuration’” on page 1586

---

**1.3.5.4.2 Configuring gesture recognition**

You can execute a visualization on a device that is operated by means of gestures. The visualization retains its user input configuration for mouse and keyboard operation and also recognizes gestures and multi-touch events. Gesture events are recognized and interpreted as mouse events.

For this purpose, activate the “**Activate multi-touch**” setting in the visualization manager.

Elements of the type “**Frame**” or “**Tab control element**” display contents that a user should be able to move. Therefore, configure their “**Scaling type**” property with “**Fixed and scrollable**”. 
Gesture recognition for:

- **Tapping**  
  A quick tap on the element is interpreted as a mouse click.

- **Panning**  
  Pressing, moving, and releasing with one finger in a frame or with a tab control element (in the window area of the element) will move the contents.

- **Multi-finger touch detection**  
  Touching several elements at the same time will input for all elements. These touch events are interpreted as the respective mouse events.

  **Example:**  
  Two-hand operation in order to trigger an action with two simultaneous inputs on two different elements.
  Virtual mixing console where multiple sliders can be operated at the same time.

In addition, the `IGestureEventHandler` interface is available in the `VisuElems.VisuElemBase` library. You can use this to implement application code that recognizes gestures and executes follow-up actions.

The following display variants can execute a visualization on a multi-touch device

- CODESYS WebVisu

See also

- “Implementing event handling with multi-touch” on page 1128

### Using gestures to control visualizations

1. Requirement: A project is open with a visualization and a user input configuration. It contains one button. The visualization device is a display with multi-touch support.
   1. Double-click the “Visualization manager” object.
      🔗 The editor opens.
   2. Click the “Settings” tab.
   3. In the “Additional settings” group, activate the “Activate multi-touch” option.
   4. Compile, download, and start the application.
      🔗 The application runs. The visualization opens. When a user touches the display of the visualization device, the visualization responds. Elements that respond to mouse events also respond to touch events. Several buttons can be pressed at the same time. Scrollable frames or tab control elements are displayed without scrollbars and can be moved by panning.

Note: The “Scaling type” property of elements type “Frame” or “Tab control element” must be set to “Fixed and scrollable”.

See also

- “Chapter 1.3.5.18.1.6 “Visualization element ‘Frame’” on page 1287
- “Chapter 1.3.5.18.1.10 “Visualization element ‘Tab control’” on page 1316
Implementing event handling with multi-touch

- Requirement: The device is multi-touch capable
  1. Implement and register a function block that receives the gesture events.
     ⇒ FUNCTION_BLOCK GesturesHandler IMPLEMENTS VisuElems.VisuElemBase.IGestureEventHandler2
        VisuElems.g_VisuEventManager.SetGestureEventHandler(THIS^);
  2. Implement and register a function block that sets the touch areas.
     ⇒ FUNCTION_BLOCK RectProvider IMPLEMENTS VisuElems.VisuElemBase.IApplicationRectangleProvider
        VisuElems.g_VisuRectangleProvider := THIS^;
  3. Implement actions as application code that are executed when a gesture event occurs

1.3.5.4.3 Configuring text input with the virtual keyboard

A visualization is usually configured so that it calls a virtual keyboard for a text input event when an input device is not available. For this purpose, the follow-up action “Write variable” is preset accordingly in the user input: The value “Standard” is selected for the “Input type” setting.

However, you can also configure especially how text is input. For this purpose, more input types are available in the user input, such as Text input or the listed visualizations. These visualizations have the visualization type “Numpad/Keypad” and display virtual keyboards or numeric keypads.

In the “Settings for default text input” setting of the visualization manager you can preset a keyboard visualization that is called from all visualizations in the application when required. This is possible without having to customize the user inputs of the visualizations.

See also
- Chapter 1.3.5.19.3.6 “Dialog ‘Input Configuration’” on page 1586

Configuring text input especially for virtual keyboards

- Requirement: A project is open with a visualization.
  1. Declare an input variable in the PLC_PRG program.
     ⇒ VAR_INPUT stInput : STRING; VAR_END
  2. Add a button to the visualization and select the element.
  3. Configure the property “Texts ➤ Text” with Text input: %s.
  4. Configure the property “Text variables ➤ Text variable” with PLC_PRG.stInput.
  5. Click auf “Configure” in the property “Input configuration ➤ OnMouseClick”.
     ⇒ The “Input Configuration” dialog box opens. The selected input event is printed below the caption.
  6. Select the “Write variable” action.
  7. Select the visualization Visudialogs.Keypad in “Input type” of the implementation of the action.
     ⇒ The virtual keyboard Visudialogs.Keypad is selected as the input device.
8. Compile, download, and start the application.
   ⇒ The visualization opens.
9. Click the button as a visualization user.
   ⇒ The virtual keyboard appears and allows text input by means of the mouse.

Configuring numeric input especially for virtual numeric keypads

- Requirement: A project is open with a visualization.
- Declare an input variable in the PLC_PRG program.
  ⇒ VAR_INPUT iInput : INT; VAR_END
- Open the visualization and added a “Rectangle” element.
- Select the element in the editor.
  ⇒ The properties are visible in the “Properties” view.
- Configure the property “Texts ➔ Text” with Number input: %i.
- Configure the property “Text variables ➔ Text variable” with PLC_PRG.iInput.
- In the “Input configuration” property, click the “Configure” button in the OnMouseClick line.
  ⇒ The “Input Configuration” dialog box opens. The selected input event is printed below the caption.
- Select the Write variable action from the list of possible actions and click the symbol.
  ⇒ The action appears in the list of actions to be executed. The blank implementation of the action appears in the window area to the right of the list.
8. Select the following settings:
   “Input type” set to VisuDialogs.Numpad.
   “Choose variable to edit” set to “Use text output variable”.
   “Dialog title” set to 'My virtual numpad'.
9. Click “OK” to close the dialog box.
   ⇒ The user input is configured.
10. Compile, download, and start the application.
    ⇒ The application runs. The visualization opens. When a user clicks the rectangle, the numeric keypad opens.
Defining standard text input

- Requirement: A project is open with a visualization and a user input configuration. For all "Write variable" follow-up actions, the value "Default" is selected for the "Input type" setting.
  1. Double-click the visualization manager.
  2. Click in the default text input in the "Settings" tab ("Default text input" group) and assign visualizations.
     - These visualizations are defined as default text input. If a display variant does not have a keyboard, then these visualizations are called without you having to adapt the user input.

1.3.5.4.4 Configuring keyboard shortcuts

You can define keyboard shortcuts and assign specific actions to them. At runtime, a visualization detects the keyboard input event and executes the action.

There are different locations where you can configure a keyboard input event.

The options include the following:
- Configure keyboard input for a specific element.
- Configure keyboard input for a specific visualization.
- Configure keyboard input that is valid for all visualizations.
- Activate the preset default keyboard shortcuts.

If the visualization integrated in CODESYS is executed, then you can deactivate the keyboard input of the visualization in order to use the keyboard shortcut from CODESYS in this state.

Configuring keyboard shortcuts for elements

You can define a keyboard shortcut that triggers an action for an element. The element must be visible and operable. For this purpose, the property "Input configuration ➔ Keyboard shortcuts" is available in the "Properties" view of the visualization editor.

- Requirement: A CODESYS project is open with the existing visualizations visEllipse and visRectangle.
  1. Select the application in the device tree and add a visualization named visMain.
     - The visualization editor opens.
  2. Select the "Frame" element from the "Toolbox" view and drag it to the editor.
     - The "Configuration of Frame Visualizations" dialog box opens.
  3. Double-click in succession the visEllipse and visRectangle visualizations in "Available Visualizations".
     - The visualizations appear in "Selected Visualizations".
  4. Click "OK" to close the dialog box.
     - The visualization contains a new element type "Frame". The two selected visualizations appear in the "Referenced visualizations" property.
       In the editor, the frame shows the visualization with the index 0.
5. Add a button and configure its properties:
   Select Rectangle in the property “Texts ➔ Text”.
   In the “Input configuration ➔ OnMouseDown” property, select “Switch frame visualization” for the visualization visRectangle.
   Specify the value R in the property “Input configuration ➔ Keyboard shortcuts ➔ Key”.
   ✳️ The button has a user input and a keyboard shortcut.

6. Add a button and configure its properties:
   Select Ellipse in the property “Texts ➔ Text”.
   In the “Input configuration ➔ OnMouseDown” property, select “Switch frame visualization” for the visualization visEllipse.
   Specify the value E in the property “Input configuration ➔ Keyboard shortcuts ➔ Key”.
   ✳️ The button has a user input and a keyboard shortcut.

7. Click “Online ➔ Login” for the device and start the application.
   ✳️ The visualization starts. It has a frame where one of the referenced visualizations runs. Focus on the visEllipse visualization and press [E]. The visualization switches the contents in the frame to the visEllipse visualization. When you press [R], the visualization switches the contents in the frame to the visRectangle visualization.

See also

- ☢️ Chapter 1.3.5.18.1.6 “Visualization element ‘Frame’” on page 1287
- ☢️ “Input action ‘Switch frame visualization’” on page 1593

Configuring keyboard shortcuts for a specific visualization

You can define keyboard shortcuts that trigger an input action on a specific visualization. The “Keyboard configuration” tab in the editor of the visualization is used for this purpose.

☐ Requirement: A CODESYS project is open with the visualizations visEllipse and visRectangle.
   1. Open the CODESYS TargetVisu object and select visEllipse as the start visualization.
   2. Open the visEllipse visualization and click the “Keyboard Configuration” tab.
3. Click “Visualizations ➔ Keyboard configuration”.
   - The “Keyboard configuration” opens.
4. Select the value C in the “Key” column.
5. Activate the “Press key” option.
6. Select the value “Change shown visualization” in the “Action type” column.
7. Select visRectangle in the “Action” column.
   - The user input is configured for [C].
8. Open the visRectangle visualization and click the “Keyboard Configuration” tab.
9. Select the value C in the “Key” column.
10. Activate the “Press key” option.
11. Select the value “Change shown visualization” in the “Action Type” column.
12. Select visEllipse in the “Action” column.
   - The user input for [C] is also configured for this visualization.
13. Build the application.
14. Click “Online ➔ Login” for the device and start the application.
   - The visualization starts and displays an ellipse. Focus on the visEllipse visualization and press [C]. The visRectangle visualization is displayed. Focus on the visualization and press [C] again. Now the visualization is switched again to visRectangle.

See also
- “Tab ‘Keyboard configuration’” on page 1558
- “Input action ‘Change shown visualization’” on page 1589

Configuring keyboard shortcuts for all visualizations in the application

You can define keyboard shortcuts that trigger the same input action for all visualizations of the application. The “Default Keyboard Shortcuts” tab in the visualization manager is available for this purpose.

- ✔ Requirement: A project is open with a visualization.
- 1. Open the visualization.
- 2. Add a rectangle.
- 3. Configure the property “Texts ➔ Text” with Keyboard shortcut.
- 4. Double-click the “GlobalTextList” object.
- 5. Click in the table, “Add Language”, and then specify de.
   - The language de is configured.
- 6. Click in the table, “Add Language”, and then specify en.
   - The language en is configured.
7. Configure translations for **de** and **en** for the text *Keyboard shortcut.*
   
    🔄 Hotkey Tastaturkürzel Hotkey

8. Open the visualization manager and select the “Default Keyboard Shortcuts” tab.

9. Specify **D** in the “Key” column.

10. Activate the “Press key” option.

11. Select the value “Change language” in the “Action Type” column.

12. Select the language **de** in the “Action” column.
   
    🔄 The keyboard event for [D] is configured.

13. Specify **D** in the “Key” column.


15. Activate the “Alt” option.

16. Select the value “Change language” in the “Action Type” column.

17. Select the language **en** in the “Action” column.
   
    🔄 The keyboard event for [Alt]+[D] is configured.

18. Compile, download, and start the application.
   
    🔄 The visualization opens.

19. As visualization user, press [D].
   
    🔄 The text is displayed in the language **de**.

See also

- ☢️ Chapter 1.3.5.19.4.3 “Tab ‘Visualization Manager’ - ‘Default Hotkeys’” on page 1617
- ☢️ “Input action ‘Change the language’” on page 1589

---

**Activating default keyboard usage**

When you activate the universal keyboard shortcuts for default keyboard usage, the user can operate the visualization without a mouse. Elements that respond to user input can process a keyboard event instead of a mouse event without you adapting its input configuration.

- ✔️ Requirement: A project is open with a visualization.
- 1. Click the “Visualization Manager” object.
- 2. Activate the “Activate default keyboard usage” option.
   
    🔄 The universal keyboard shortcuts are activated.
- 3. Load the application to the device and start the application.
   
    🔄 The visualization starts. Now operation can continue without the mouse. You can navigate in the window by means of the [arrow] and [tab] keys and press [Enter] instead of the mouse button.

See also

- ✔️
-  - ☢️ Chapter 1.3.5.19.1 “Keyboard Shortcuts for Default Keyboard Action” on page 1555
**Activating and deactivating keyboard shortcuts for integrated visualizations**

If you execute the visualization as an integrated visualization, then the command “Visualization → Activate Keyboard Usage” is available in order to deactivate the capturing of keyboard events. It is actually possible for the same keyboard shortcuts to be defined in the visualization and in CODESYS.

When you activate the command, the visualization executes the configured keyboard events. When you deactivate the command, CODESYS executes the keyboard events. Capturing keyboard events is then deactivated for the visualization.

See also

- Chapter 1.3.5.19.2.4 “Command ’Activate Keyboard Usage’” on page 1560

**1.3.5.4.5 Capturing user input events**

You can capture user input events in the application. For this purpose, you can implement a function block that is executed when user events occur.

**Capturing the writing of variables**

When the user completes the input of a value (in an input field), an edit control event is closed. You can capture this event in the application as follows.

1. Create a function block that implements the VisuElems.IEditBoxInputHandler interface from the VisuElemBase library.

2. Pass the instance to the global event manager VisuElems.Visu_Globals.g_VisuEventManager by calling the SetEditBoxEventHandler method.
A visualization has two input fields for \texttt{iInput\_A} and \texttt{rInput\_B} and one text output element. The input fields are rectangles that the user is prompted to click in order to input text. The text output element is a rectangle where the contents of the text variable \texttt{PLC\_PRG.stInfo} are printed. The text variable contains the last input by a user in one of the input fields and the additional information that was added.

\begin{verbatim}
PROGRAM PLC\_PRG
VAR_INPUT
  iInput\_A:INT; (* Used in the visualization as user input variable*)
  rInput\_B:REAL; (* Used in the visualization as user input variable*)
  stInfo : STRING; (* Informs about the user input via the edit
\end{verbatim}
control field;
   String gets composed by method 'VariableWritten;
   Result is displayed in the lower rectangle of the
visualization *)
END_VAR
VAR
   inst : POU;
   bFirst : BOOL := TRUE;
END_VAR

IF bFirst THEN
   bFirst := FALSE;
   VisuElems.Visu_Globals.g_VisuEventManager.SetEditBoxEventHandler(ins
   t);
   (* Call of method VariableWritten *)
END_IF

FUNCTION_BLOCK POU IMPLEMENTS VisuElems.IEditBoxInputHandler
  (* no further declarations, no implementation code *)
METHOD VariableWritten : BOOL
  (* provides some information always when an edit control field is
  closed in the visualization, that is a variable gets written by
  user input in one of the upper rectangles *)
  VAR_INPUT
     pVar : POINTER TO BYTE;
     varType : VisuElems.Visu_Types;
     iMaxSize : INT;
     pClient : POINTER TO VisuElems.VisuStructClientData;
  END_VAR

  // String stInfo, which will be displayed in the lower rectangle,
  is composed here
  PLC_PRG.stInfo := 'Variable written; type: ';
  PLC_PRG.stInfo := CONCAT(PLC_PRG.stInfo, INT_TO_STRING(varType));
  PLC_PRG.stInfo := CONCAT(PLC_PRG.stInfo, ', adr: ');
  PLC_PRG.stInfo := CONCAT(PLC_PRG.stInfo, DWORD_TO_STRING(pVar));
  PLC_PRG.stInfo := CONCAT(PLC_PRG.stInfo, ', by: ');
  PLC_PRG.stInfo := CONCAT(PLC_PRG.stInfo,
      SEL(pClient^.globaldata.clienttype =
      VisuElems.Visu_ClientType.Targetvisualization,'other visu',
      'targetvisu'));
END_VAR

Capturing keyboard events

When the user presses and releases the key, a keyboard event is triggered in the visualization.
You can capture this event in the application as follows.

1. Create a function block that implements VisuElems.IVisuUserEventManager from the VisuElemBase library.

2. Pass the instance to the global event manager VisuElems.Visu_Globals.g_VisuEventManager by calling the SetKeyEventHandler method.
Example

A visualization has one text output element. The text output element is a rectangle where the contents of the text variable `PLC_PRG.stInfo` are printed. The text variable contains information about the last key pressed by the user.

<table>
<thead>
<tr>
<th>Properties of the rectangle for the text output</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Texts ➔ Text&quot;</td>
</tr>
<tr>
<td>&quot;Text variables ➔ Text variable&quot;</td>
</tr>
<tr>
<td><code>%s</code></td>
</tr>
<tr>
<td>PLC_PRG.stInfo</td>
</tr>
</tbody>
</table>

Implementation of the PLC_PRG program

```plaintext
PROGRAM PLC_PRG
VAR_INPUT
  stInfo : STRING;
END_VAR
VAR
  inst : POU;
  bFirst : BOOL := TRUE;
END_VAR
IF bFirst THEN
  bFirst := FALSE;
  VisuElems.VisuGlobals.g_VisuEventManager.SetKeyEventHandler(inst);
END_IF
```

Implementation of the POU function block

```plaintext
FUNCTION_BLOCK POU IMPLEMENTS VisuElems.IKeyEventHandler
(* no further declarations, no implementation code *)
/// This method will be called after a key event is released.
/// RETURN:
/// TRUE - When the handler has handled this event and it should not be handled by someone else
/// FALSE - When the event is not handled by this handler
METHOD HandleKeyEvent : BOOL
VAR_INPUT
  /// Event type. The value is true if a key-up event was released.
  bKeyUpEvent : BOOL;
  /// Key code
  dwKey : DWORD;
  /// Modifier. Possible values:
  /// VISU_KEYMOD_SHIFT : DWORD := 1;
  /// VISU_KEYMOD_ALT : DWORD := 2;
  /// VISU_KEYMOD_CTRL : DWORD := 4;
  dwModifiers : DWORD;
  /// Pointer to the client structure were the event was released
  pClient : POINTER TO VisuStructClientData;
END_VAR
VAR
END_VAR
PLC_PRG.stInfo := 'KeyEvent up: ';
PLC_PRG.stInfo := CONCAT(PLC_PRG.stInfo, BOOL_TO_STRING(bKeyUpEvent));
PLC_PRG.stInfo := CONCAT(PLC_PRG.stInfo, ', key: ');
PLC_PRG.stInfo := CONCAT(PLC_PRG.stInfo, DWORD_TO_STRING(dwKey));
PLC_PRG.stInfo := CONCAT(PLC_PRG.stInfo, ', modifier: ');
PLC_PRG.stInfo := CONCAT(PLC_PRG.stInfo, DWORD_TO_STRING(dwModifiers));
PLC_PRG.stInfo := CONCAT(PLC_PRG.stInfo, ', by: ');
PLC_PRG.stInfo := CONCAT(PLC_PRG.stInfo, ');
```
Recording variable value changes triggered by input events

All visualization elements that change the value of a variable by user input call the IValueChangedListener interface. With this interface, the value changes can be recorded and then processed programmatically.

1. Implement a function block (example: POU) that implements the IValueChangedListener interface.

   ```
   FUNCTION_BLOCK POU IMPLEMENTS VisuElems.IValueChangedListener
   In the device tree, the “ValueChanged” method is inserted below the function block.
   ```

2. In a program (example: “PLC_PRG”), implement the IEC code that registers the interface.

   ```
   VisuElems.g_itfValueChangedListenerManager.AddValueChangedListener(itfValueChangedListener)
   “PLC_PRG” receives all value changes by means of the “ValueChanged” method.
   Now you can record and process the value changes.
   ```

1.3.5.5 Setting up user management

1.3.5.5.1 Setting up user management for visualizations

   Setting up user management for visualizations.................................. 1139

   1.3.5.5.2 Configuring users and groups............................................. 1140

   1.3.5.5.3 Editing and selecting user management dialogs....................... 1141

   1.3.5.5.4 Configuring permissions for groups...................................... 1142

In the visualization user management, you define users and user groups and assign access rights to user groups for individual visualization elements. In the user management dialogs, users can be registered and unregistered in runtime mode and passwords and user management can be changed.

In a project with several applications, you can configure user management for each application.

**NOTICE!**

When a visualization user management exists, an unregistered user automatically receives the access rights from the None group.

1.3.5.5.1 Setting up user management for visualizations

When you set up user management for your visualization, the following variants are possible:

- **Empty user management**
  An empty user management contains the None user group. You configure all users and groups yourself.

- **User management with default users and groups**
  This user management contains the Admin, Service, Operator, and None groups. The first three groups each contain one user with the same name as the group.
Create new user management

☐ Requirement: A user management does not exist yet for your visualization.

1. Click the “Visualization Manager” object in the device tree.
2. Select the “User Management” tab.
3. Click “Create Empty User Management” or “Create User Management with Default Groups and Users” depending on which variant you need.
   ⇒ The “Groups”, “Users”, and “Settings” tabs are displayed.
   The “Group” tab opens.
4. If you have created an empty user management, then you configure the new groups and users.
   If you have created a user management with default groups and users, then you can grant permissions for elements in your visualization. You can also select the user management dialogs and assign them to the buttons of the visualization.

See also

● ☰ Chapter 1.3.5.5.2 “Configuring users and groups” on page 1140
● ☰ Chapter 1.3.5.5.4 “Configuring permissions for groups” on page 1142
● ☰ Chapter 1.3.5.5.3 “Editing and selecting user management dialogs” on page 1141

1.3.5.5.2 Configuring users and groups

Groups and their users for the basis for user management. A group has one or more users; a user can belong to multiple groups. The permissions of the visualization elements are always assigned to a group.

Adding groups

☐ Requirement: You have already created a user management by clicking “Create Empty User Management” or “Create User Management with Default Groups and Users” in the “Visualization Manager” (“User Management” tab).

1. Click the “Visualization Manager” object in the device tree.
2. Select the “User Management” tab.
3. Click in the last line of the list.
   In this line, the field of the “Group Name” column is still empty.
4. Click in the field of the “Group Game” column and specify the name for the new group.
5. If necessary, activate the options “Automatic Logout” and “Permission to Change User Data”.

Adding users and assigning groups

☐ Requirement: A user management exists with at least on group. The “Visualization Manager” is open.

1. Select the tab “User Management ➔ User”.
2. Click in the last empty line of the list.
3. Specify the "Login Name".
   CODESYS applies the "Login Name" as "Password".

4. If you want to change the password, click in the "Password" field of the user.
   The "Change Password" dialog box opens.

5. Click the "User Group" field.
   The "User Groups the User Belongs to" dialog box opens.

6. Activate the "Assigned" option for the groups that the user should be long to and click "OK" to confirm.
   In the "User Group" field, all groups are listed that the new user belongs to.

See also
- Chapter 1.3.5.5.1 “Setting up user management for visualizations” on page 1139
- 1.3.5.5.3 Editing and selecting user management dialogs

1.3.5.5.3 Editing and selecting user management dialogs

In the user management dialogs, you define the login, logout, changing of the user password, and editing of the user management in the visualization at runtime.

**NOTICE!**
If you create your own dialog as a user management dialog, then you should use the visualizations from the included library project VisuUserMgmtDialogs.library as the basis, because it uses the required interfaces. Your own user management dialog is listed then in “Visualization Manager ➔ Settings”, “Settings for User Management dialogs”.

Editing user management dialogs

Requirement: The library project VisuUserMgmtDialogs.library exists in the installation directory.

1. Click “File ➔ Open Project”.
2. Select the project VisuUserMgmtDialogs.library from the Projects folder of the installation directory.
3. Click “View ➔ POU’s”.
   In the "POUs" view, the project is displayed with the visualizations "UserMgmtChangePassword", "UserMgmtConfig", and "UserMgmtChangePassword".
4. Double-click a visualization (example: "UserMgmtLogin").
5. Change the visualization as you like and save the project.
6. Then, reinstall the library and add it to the “Library Manager” of your application.

Selecting user management dialogs

A user management already exists in your application in the “Visualization Manager” object (“User Management” tab).
The “VisuUserManagement” library is in the library manager.

1. Click the “Visualization Manager” object in the device tree.
2. Click the “Settings” tab.
3. In “Settings for User Management Dialogs”, select the dialogs for “Login dialog”, “Change password dialog”, and “Change configuration dialog”.

If no entries can be seen in “Settings for User Management Dialogs” in the dialog lists, then close the “Visualization Manager” and reopen it.

Configuring visualization buttons for the login, logout, change password, and user management dialogs

Requirement: A visualization is open.

1. Drag a “Button” element from the “Toolbox” view (“Common controls”) to the visualization.
2. Click the “Input configuration” node in the “Properties” view.
3. Click “Configure” in the property “Input configuration ➔ OnMouseClick”.
4. Click “User management” and ➔ in the “Input Configuration” dialog.
   ➔ The following “Dialogs and actions” are listed on the right: “Login”, “Logout”, “Change user password”, and “Open user management”.
5. Select the dialog or action to assign to the button and click “OK”.
   ➔ When the button is clicked at runtime, the selected dialog opens or the selected action is executed.

If you want to open and edit the user management in the visualization at runtime, you must be a member of a group that has the “Permission to Change User Data”.

See also

●
● Chapter 1.3.5.4.1 “Configuring user inputs for visualization elements” on page 1125
● Chapter 1.3.5.19.3.6 “Dialog ‘Input Configuration’” on page 1586

1.3.5.4 Configuring permissions for groups

Configuring permissions for an element of the visualization

Permissions for visualization-elements are not granted to individual users, but to groups with assigned users.
Requirement: A “Visualization” object is open with at least one inserted visualization element.

1. Click a visualization element in the editor.
2. Click the “Value” field of the “Permissions” element property in the “Properties” view.
   - The “Permissions” dialog box opens.
3. Select the permissions that the respective user group should have for the visualization element.
   
   Note: If the option “Group hierarchy is used” is activated, the groups lower in the hierarchy cannot be granted more permissions than groups higher in the hierarchy.

In the “Element List” of the visualization, the “Permissions” column shows the element permissions granted to groups.

See also
- § Chapter 1.3.5.19.3.1 “Dialog ‘Access Rights’” on page 1583
- § Chapter 1.3.5.19.4.1.1 “Visualization editor” on page 1608

1.3.5.6 Setting up multiple languages

Texts and tooltip texts for visualizations are managed in text lists and can be displayed in different languages. To switch a visualization between the available languages, configure a visualization element with the corresponding input configuration for changing the language.

There are static texts that are managed in “GlobalTextList” (generated automatically) and dynamic texts from created text lists. A dynamic text can be changed at runtime with a variable that defines the index of the text list entry. Static texts are fixed labels within a visualization; dynamic texts are often used for displaying variable values or error messages.

For creating and using text lists, see: Managing Text in a Text list.

You can modify the appearance and formatting of texts and tooltips with the element properties “Text properties” and “Font variables”.

See also
- § Chapter 1.3.5.18.2 “Placeholders with format definition in the output text” on page 1546
- § Chapter 1.3.5.19.3.6 “Dialog ‘Input Configuration’” on page 1586

Configuring the language switcher for texts from text lists

Requirement: An empty visualization object is inserted into the project and it is open for editing in the visualization editor. There is also a “Visualization Manager” object. User management is not created for the visualization.

The following instructions provide a simplified example:
By means of two buttons, the user should be able to toggle the visualization texts between English and German.

Static texts in the visualization include the labels "State, Machine 01", "State, Machine 02", "English", and "German". These texts are located in the “GlobalTextList” in English and German.

Dynamic texts will describe the state of both machines. The texts are provided in the text list “Status_Texts” in English (en) and German (de).

1. Drag a “Text field” from the “Toolbox” view (category “Common Controls”) to the editor view. Specify the value State, Machine M01 in the properties editor for the element property “Texts ➔ Text”.
2. Copy the element and change the copy label to State, Machine M02.
3. See also the figure in step 14 for the following steps.

Insert two elements of type “Button” from the “Toolbox” view (category “Common Controls”) in the visualization editor. With these elements, the user should be able to toggle the language of the visualization. Specify the text German or English in the properties editor for element property “Texts ➔ Text” (4).

4. Double-click and open “GlobalTextList” in the “POUs” view.
   
   - The texts are entered in the “Standard” (1) column, and the “ID”’s 0 and 1 are assigned automatically as additional information.

5. Add the languages “de” and “en” with the texts shown in the following figure.

![GlobalTextList](image)

6. Close the “GlobalTextList”.
7. Add two elements of type “Rectangle” from the “Toolbox” view (category “Basic”) in the visualization editor. The current state of each machine should be displayed.
8. For managing the texts for describing the states, add an object of type “Text List” below the application. Name the list Status_Texts.
9. Specify the texts shown in the figure for the standard language (1) and the target languages “en” and “de” in the editor of “Status_Texts”.

![Status_Texts](image)

10. Close the text list “Status_Texts”.

11. Select the rectangle element for displaying the state of machine M01. Select the text list Status_Texts from the combo box in the properties editor (2) for the element property “Dynamic texts” (5). Specify an application variable for “Text index” that shows the appropriate text index for the state of the machine at runtime. Example:
    
    PLC_PRG.ivar_status_m01.
12. Now configure the user input for both buttons for toggling the language in the visualization.

Select the button "German". Double-click “Configure” of the property “Input configuration” (6), “OnMouseClick”.

⇒ The “Input configuration / OnMouseClick” dialog opens.

13. Select “Change the language” on the left. Click the arrows to accept the setting to the right. Select “de” in “Language” to the right of the dialog in the input assistance. Click “OK” to confirm.


⇒ The following figure shows the performed properties configurations for the four visualization elements.

15. When the application is compiled without errors, you can test the visualization in simulation mode. Activate the option “Online ➔ Simulation”. Click “Online ➔ Login”.

⇒ The visualization appears in the visualization editor view in online mode:
16. Click the “German” button.

⇒ The language changes to German:

See also
●  Chapter 1.3.5.19.3.6 “Dialog ‘Input Configuration’” on page 1586

Setting up fonts for a language

The font for a visualization element is defined in the properties editor. If a language switch is provided, you can overwrite this basic font with another font for each language in the visualization manager.

Requirement: A visualization is set up with at least one language in addition to the default language. For an example, see “Configuring the language switcher for texts from text lists” on page 1143

1. Double-click and open the “Visualization Manager” object and select the “Font” tab.
2. Double-click the field in the “Font” line for a particular language. Select a font from the combo box.
3. In the “Font size” line, replace the value 1 with a value greater than 1 (example: 2) in order to increase the size of the font as defined by the visualization style; or replace it with a value less than 1 in order to decrease it (example: 0.5).

⇒ In online mode, the font changes depending on the set language.

See also
●  Chapter 1.3.5.19.4.6 “Tab ‘Visualization Manager’ - ‘Font’” on page 1622

1.3.5.7 Visualizing alarm management

In CODESYS, the alarm management is a powerful object for creating and managing alarms. You can group alarms and set the acknowledgement behavior individually. The alarm display in the visualization can also be customized.

The “Alarm Table” and “Alarm Banner” visualization elements are available for displaying and processing alarms. The alarm table lists the alarm texts. The alarm banner is a simplified version of the alarm table. It visualizes a single alarm only. However, by adding scroll elements you can allow for switching the display from one active alarm to another active alarm.

See also
●  Alarm Management

Creating an alarm table

Requirement: In your project, alarms are defined in alarm groups and they are assigned to an alarm class. The following instructions are based on the example that is described in the "Configuring alarm management" chapter.
1. Open the visualization editor.
2. Drag the “Alarm Table” element from the “Alarm Manager” group to the visualization editor.
   ⇐ The “Alarm Table” visualization element is visible in the editor.
3. In the “Alarm configuration” / “Alarm groups” property, define the alarm groups that you want to visualize. Click into the value field.
   ⇐ The “Select Alarm Group” dialog opens.
4. Clear the “All” check box and select the “PartsDeficit” alarm group. Add the group to the selected alarm groups by clicking the button.
5. In the “Alarm configuration” / “Alarm classes” property, define the alarm classes that you want to visualize. Click into the value field.
   ⇐ The “Select Alarm Class” dialog opens.
6. Clear the “All” check box and select the “PartsDeficit” alarm class. Add the alarm class to the selected alarm classes by clicking the button.
7. Add an additional column. Click the “Columns” / “Create New” button.
   ⇐ CODESYS adds the column “[2]” to the properties. The “Symbol” column is added to the table.
8. Select data type “State” for column [2].
   ⇐ The default column heading “State” is shown in the table.
9. Name the “Column heading” column “Status”.
10. Specify the appearance of the selected table cell. Set the “Selection” / “Selection color” to “Green”.
11. In the “Control variables” / “Confirm selection” property, specify the variable bQuitAlarm for confirming messages.
12. Adjust the other properties to your requirements. See the “Alarm table” visualization element for a complete description of the properties.

See also
- Configuring Alarm Management
- Chapter 1.3.5.18.1.22 “Visualization element ‘Alarm table’” on page 1395

Inserting elements for acknowledging alarms

In CODESYS, predefined buttons are available for controlling the alarms in an alarm table.

Requirement: An “Alarm table” element exists in the visualization.

1. Select the visualization element in the editor.
2. Click “Visualization ➔ Insert elements for acknowledging alarms”.
   ⇐ The “Alarm Table Wizard” dialog opens.
3. Click “OK” to accept all settings.
   - Four buttons are added for controlling the alarm table.

![Alarm table]

See also
- Chapter 1.3.5.18.1.22 “Visualization element 'Alarm table'” on page 1395

Creating an alarm banner

Requirement: In your project, alarms are defined in alarm groups and they are assigned to an alarm class. The following statement is based on the example that is described in the "Configuring alarm management" chapter.

The alarm banner displays an active alarm in online mode. If there are multiple active alarms, filtering takes place by means of the filter criteria set in the alarm banner (newest for filter criterion "Priority" and most important for filter criterion "Newest"). See the instructions below for adding scroll elements in order to switch the display between multiple alarms.

1. Open the visualization editor.
2. Drag the “Alarm banner” element from the “Alarm manager” group to the visualization editor.
   - The “Alarm banner” visualization element is visible in the editor.
3. In the “Alarm configuration” / “Alarm groups” property, define the alarm groups that you want to visualize. Click into the value field.
   - The “Select Alarm Group” dialog opens.
4. Clear the “All” check box and select the “PartsDeficit” alarm group. Add the group to the selected alarm groups by clicking the button.
5. In the “Alarm configuration” / “Alarm classes” property, define the alarm classes that you want to visualize. Click into the value field.
   - The “Select Alarm Class” dialog opens.
6. Clear the “All” check box and select the “PartsDeficit” alarm class. Add the alarm class to the selected alarm classes by clicking the button.
7. Set the “Alarm configuration” / “Filter criterion” property to “Newest”.
   - In online mode, the newest alarm message is always shown.
8. Add an additional column. Click the “Columns” / “Create new” button.
   - CODESYS adds the column “[2]” to the properties. The “Symbol” column is added to the table.
9. Select data type “State” for column [2].
   - The default column heading “State” is shown in the table.
10. In the “Confirmation variable” property, specify the variable bQuitAlarm for confirming messages.
Adding elements for scrolling the active alarms

Elements can be added to an alarm banner for switching the display between the individual active alarms. You can control the scrolling with visu-local variables or application variables.

1. Select the added "Alarm banner" visualization element. Click "Insert Elements for Scrolling Alarms" in the context menu.
   ⇒ The “Alarm Banner Wizard” opens.
2. Select the element type for the scroll elements: “Button” or “Rectangle”.
3. Activate the action(s) for which a control should be inserted: “Scroll to next alarm”, “Scroll to previous alarm”.
4. Specify a Boolean variable that gets the value TRUE when multiple active alarms are present. If you have already configured a project variable in the element properties, then it is also specified here in the wizard. Otherwise CODESYS automatically creates the visu-local variable “xMultipleAlarmsActive”.
5. In the next step, check the configuration of the element properties of the extended alarm banner.
6. Select the alarm banner element and look at the section “Handling of multiple active alarms” in the “Properties” view. You have two options:
   7. Option 1: The display should switch automatically. Activate the “Switch automatically” property.
      ⇒ Now, in “Every N seconds” you define the time interval after which the display in the alarm banner in online mode should switch to the next alarm.
   8. Option 2: The display should be controlled by means of the application. Deactivate the “Switch automatically” property.
      ⇒ Switching between the active alarms can be controlled by two variables. By default, xNext and xPrev are created for scrolling to the next or previous alarm. You can replace these variables with custom your own defined application variables.

Filtering by the contents of a latch variable can be useful when there are a lot of alarm events displayed. If the latch variable assigned to an alarm in the alarm group definition contains, for example, the error number or the name of a device instance, then you can filter the alarms in the visualization by it.

For this purpose, you configure an input option in the alarm visualization for the contents of the latch variable to be filtered by. For example, insert an input field which writes to the variable that is specified in the “Alarm configuration” - “Filter by latch 1” - “Filter variable” property of the configuration of the “Alarm table” element or “Alarm banner” element.

In addition, you configure an input option for the type of filtering. The type determines whether a numeric value (typed literal, LINT literal) or the string value of the latch variable is used for filtering. Filtering can also be switched off by means of type setting 0. For example, in the visualization, insert another input field which writes to the variable that is specified in the “Filter type” property of the configuration of the alarm table or alarm banner.

For more information, see the "Alarm Filter Latch Example" sample project in the CODESYS Store.

See also

- Configuring Alarm Management
The animation of a visualization element at runtime can serve to visualize value curves in addition to serving purely visual purposes. Animation is possible through a dynamic configuration of certain element properties, i.e. by controlling these properties with a variable. See the following examples of possible animations.

### 1.3.5.8.1 Configuring rotations and offsets

You can animate a visualization element and have it shifted or rotated at runtime. To do this you assign variables in its property “Absolute movement” and then program the animation in the application code.

**Configuring an offset**

- **Requirement:** A project with a visualization is open.
- 1. Open the visualization and add an element “Rectangle”.
  - The view “Properties” displays the configuration of the element.
- 2. In the application in the POU PLC_PRG, declare type-compliant variables: diOffsetX : DINT; and diOffsetY : DINT;
- 3. Configure the property “Absolute movement ➔ Movement ➔ X” with PLC_PRG.diOffsetX and “Y” with PLC_PRG.diOffsetY.
- 4. Implement a shift of the element, for example by means of a modulo division of the value:
  diOffsetX := diOffsetX MOD 100;
  diOffsetY := diOffsetY MOD 100;
- 5. Compile, load and start the application.
  - The application runs. The visualization opens. The rectangle moves.

**Configuring a rotating element**

When an element rotates, then the center point of the element rotates precisely around its center. The center is defined in the property “Center”. The center point of an element is calculated internally. If the center point and center coincide, then there is no rotation.

You can configure a clockwise rotation of the element by increasing the value of the variable “Absolute movement ➔ Rotation”.

- **Requirement:** A project with a visualization is open.
- 1. Open the visualization and add an element “Rectangle”.
  - The view “Properties” displays the configuration of the element.
- 2. In the application in the POU PLC_PRG, declare a type-compliant variable: rValue : REAL;
- 3. Configure the property “Absolute movement ➔ Rotation” with PLC_PRG.rValue.
- 4. Implement the clockwise rotation of the element by increasing the value of the variable:
  rValue := rValue + 0.1;
5. Compile, load and start the application.
   ⇒ The application runs. The visualization opens. The rectangle rotates about the center. The alignment of the element with respect to the coordinate system is fixed.

Configuring a rotating element

When an element performs an inner rotation and rotates, then the center point of the element rotates precisely around its center. This is the point defined in the property "Center". The alignment of the element also rotates relative to the coordinate system. If the center point of the element and the center coincide, this produces a rotation on the spot.

You can configure a clockwise rotation of the element by increasing the value of the variable "Absolute movement ➔ Inner rotation".

If the visualization is In runtime, you can see that the element rotates (also relative to the coordinate system of the visualization).

☐ Requirement: A project with a visualization is open.
1. Open the visualization and add an element “Polygon”, which you form into a pointer.
   ⇒ The view “Properties” displays the configuration of the element.
2. Drag the center point of the element to the base of the pointer.
3. In the application in the POU PLC_PRG, declare a type-compliant variable:

   rValue : REAL;

4. Configure the property “Absolute movement ➔ Inner rotation” with PLC_PRG.rValue.
5. Implement the clockwise rotation of the element by increasing the value of the variable:

   rValue := rValue + 0.1;
6. Compile, load and start the application.
   ⇒ The application runs. The visualization opens. The pointer rotates about its base.
1.3.5.8.2 Animating a text display

An animation of the text display can be configured in the property “Font variables”. All basic elements have this property as well as tables, scrollbars and text fields.

Example: animating the font size

☐ Requirement: A project with a visualization is open.
1. Open the visualization and add an element “Rectangle”.
   ⇒ The view “Properties” displays the configuration of the element.
2. Configure the property “Texts ➔ Text” with Important:
3. In the application in the POU PLC_PRG, declare a type-compliant variable:
   ⇒ iFontHeight : INT;
4. Configure the property “Font variables ➔ Size” with PLC_PRG.iFontHeight.
5. Implement a change of the font size.
   ⇒ iFontHeight := iFontHeight + 1) MOD 20;
6. Compile, load and start the application.
   ⇒ The application runs. The visualization opens. The rectangle is labelled with Important. The font size grows from 1 to 20.

See also
- Chapter 1.3.5.18.1.1 “Visualization element ‘Rectangle’, ‘Round rectangle’, ‘Ellipse’” on page 1224

1.3.5.8.3 Animating a color display

The colors of an element are specified in the “Colors” properties of the element properties. There you can select either a predefined style color from the selection list or a color in the color dialog.

The “Color variables” element property is used for the color animation of the element. If you pass variables to the properties, then you can program color changes in the application code or configure a user input that results in a color change. A color constant or color variable in the code has the data type DWORD and is encoded according to the RGB color space or RGBA extension.

NOTICE!

The “Activate semi-transparent drawing” option is provided in the Visualization Manager. This option is enabled by default so that the “Transparency” property is available for all color definitions. With programmatic color definition, the leading byte is interpreted as an alpha channel and therefore used as the transparency value of the color. When the option is cleared, the “Transparency” property is not available and the leading byte is ignored in color literals.

Color information in the code is specified as DWORD literals. The value is in the RGBA color space and is usually shown as a hexadecimal number. The value is coded with additive portions of red, green, and blue. It is appended with the alpha channel which determines the transparency of the color.
Byte order of a color literal

16#<TT><RR><GG><BB>

<TT> : 00 - FF     // Transparency in 256 levels
<RR> : 00 - FF     // Red in 256 levels
<GG> : 00 - FF     // Green in 256 levels
<BB> : 00 - FF     // Blue in 256 levels

The graduation value for transparency is 16#FF for opaque and 16#00 for transparent. For each color portion, one byte is reserved for 256 color graduations (16#FF to 16#00). 16#FF means 100% color portion and 16#00 means 0% color portion.

| <TT> | Byte for the transparent graduation of 00-FF |
| <RR> | Byte for the red portion of 00-FF          |
| <GG> | Byte for the green portion of 00-FF        |
| <BB> | Byte for the blue portion of 00-FF         |

Table 251: Color literal

<table>
<thead>
<tr>
<th>Color Literal</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>16#FF0000FF</td>
<td>Blue, opaque</td>
</tr>
<tr>
<td>16#FF00FF00</td>
<td>Green, opaque</td>
</tr>
<tr>
<td>16#FFFFFF00</td>
<td>Yellow, opaque</td>
</tr>
<tr>
<td>16#88888888</td>
<td>Gray, semitransparent</td>
</tr>
<tr>
<td>16#88000000</td>
<td>Black, semitransparent</td>
</tr>
<tr>
<td>16#FFFF0000</td>
<td>Red, opaque</td>
</tr>
</tbody>
</table>

Example

Global declaration of color constants

```plaintext
VAR_GLOBAL CONSTANT
c_dwBLUE : DWORD := 16#FF0000FF;         // Highly opaque
c_dwGREEN : DWORD := 16#FF00FF00;         // Highly opaque
c_dwYELLOW : DWORD := 16#FFFFFF00;        // Highly opaque
c_dwGREY : DWORD := 16#88888888;          // Semitransparent
c_dwBLACK : DWORD := 16#88000000;         // Semitransparent
c_dwRED : DWORD := 16#FFFF0000;           // Highly opaque
END_VAR
```

Example

Global declaration of color constants

```plaintext
VAR_GLOBAL CONSTANT
c_dwRED : DWORD := 16#FFFF0000;           // Highly opaque
c_dwGREEN : DWORD := 16#FF00FF00;         // Highly opaque
c_dwYELLOW : DWORD := 16#FFFFFF00;        // Highly opaque
c_dwGREY : DWORD := 16#88888888;          // Semitransparent
c_dwBLACK : DWORD := 16#88000000;         // Semitransparent
END_VAR
```

Animating a visualization element in color

1. Create a standard project in CODESYS.
2. Declare global color constants in the POU tree.

```plaintext
{attribute 'qualified_only'}
VAR_GLOBAL CONSTANT
  gc_dwRed : DWORD := 16#FFFF0000;           // Highly opaque
  gc_dwGreen : DWORD := 16#FF00FF00;         // Highly opaque
  gc_dwYellow : DWORD := 16#FFFFFF00;        // Highly opaque
  gc_dwGrey : DWORD := 16#88888888;          // Semitransparent
  gc_dwBlack : DWORD := 16#88000000;         // Semitransparent
END_VAR
```
3. In the device tree, declare local color variables in PLC_PRG.

```
VAR
  dwFillColor: DWORD := GVL.gc_dwGreen;
  dwFrameColor : DWORD := GVL.gc_dwBlack;
  dwAlarmColor : DWORD := GVL.gc_dwRed;
END_VAR
```

4. Declare a control variable.

```
  bChangeColor : BOOL;
```

5. Declare an input variable in PLC_PRG.

```
  bInput : BOOL;
```

6. Enable the visualization editor.

7. Drag a “Rectangle” element to the visualization editor.

```
The “Properties” view of the element opens.
```

8. Configure the properties of the rectangle as follows:

- Property “Color variables”, “Normal state”, “Filling color”: PLC_PRG.dwFillColor
- Property “Color variables”, “Normal state”, “Frame color”: PLC_PRG.dwFrameColor
- Property “Color variables”, “Alarm state”, “Filling color”: PLC_PRG.dwAlarmColor
- Property “Color variables”, “Toggle color”: <toggle/tap variable>
- Property “Input configuration”, “Toggle”, “Variable”: PLC_PRG.bInput

9. Program the variables as follows:

```
PROGRAM PLC_PRG
VAR
  dwFillColor: DWORD := GVL.gc_dwGreen;
  dwFrameColor : DWORD := GVL.gc_dwBlack;
  dwAlarmColor : DWORD := GVL.gc_dwRed;
  bChangeColor : BOOL;
  bInput : BOOL;
END_VAR

IF bChangeColor = TRUE THEN
  dwFillColor := GVL.gc_dwYellow;
  dwFrameColor := GVL.gc_dwBlue;
ELSE
  dwFillColor:= GVL.gc_dwGreen;
  dwFrameColor := GVL.gc_dwBlack;
END_IF
```

```
The colors are initialized at runtime. If the variable bChangeColor is then forced to
TRUE, the color display of the rectangle changes. When the rectangle is clicked in the
visualization, the rectangle is displayed in alarm colors.
```

See also

- 
- 
- § Chapter 1.3.5.17 “Applying visualization styles” on page 1216
- § Chapter 1.3.5.8.2 “Animating a text display” on page 1152
A frequently required function of a user interface is the display of data arrays. CODESYS Visualization provides the element "Table" for this.

In the configuration of the element "Table", enter an array variable in the property "Data array". The array components are displayed in the rows and columns of the table.

A table for displaying data arrays can also be created in the following way. You duplicate a single element having at least one property that is described by a structured variable. The single element is configured as a "template" for this and duplicated with a command.

1.3.5.9.1 Displaying structured variable values in tables

Subsequent instructions describe an example of how an array of a structure is displayed in a table. As a preparation, create the MyStruct DUT and the declarations in the PLC_PRG program.

```plaintext
TYPE MyStruct :
    STRUCT
        iNo : INT;
        bOnStock : BOOL;
        strPartNumber : STRING;
    END_STRUCT
END_TYPE

PROGRAM PLC_PRG
VAR
    arrStruct : ARRAY[0..6] OF MyStruct;
    iSelectedColumn : INT;
END_VAR
```

1. Drag the "Table" visualization element to the visualization editor.
2. Assign the array variable arrStruct to the "Data array" property.
   ⇒ The structure members are displayed as column headings and the array index as row headings.
3. Change the "Columns ➔ Column ➔ [0] ➔ Column header" property to an informative heading (example: Number).
5. Assign a color to the "Selection ➔ Selection color" property.
6. Define the "Selection ➔ Selection type" property as Row selection.
7. In the “Selection ➔ Variable for selected row” property, define the PLC_PRG.iSelectedColumn variable. The following display results in online mode:

<table>
<thead>
<tr>
<th>Number</th>
<th>in Stock</th>
<th>Part Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>FALSE</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>FALSE</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>FALSE</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>FALSE</td>
<td></td>
</tr>
</tbody>
</table>

See also
● Chapter 1.3.5.18.1.13 “Visualization element 'Table’” on page 1337

1.3.5.9.2 Configuring and multiplying visualization elements as templates

You can use the “Visualization ➔ Multiply Visu Element” command to display array data. The command multiplies a template element to create an element of the same type for each array component. The layout of the new elements in the visualization is one-dimensional as a row or column, or two-dimensional as a table.

To do this, drag an applicable element into the visualization editor. Then configure the properties of the element with array variables and specify the index access placeholder $FIRSTDIM$ as component access. If you have declared a multidimensional array, then you can use the second index access placeholder $SECONDDIM$ for the additional dimension. Configure the remaining properties as usual with the typical values. The purpose is to create a valid template element. Then execute the “Multiply Visu Element” command on the template element. Now the dialog with the same name opens. There you define in detail how many elements should be created and where they should be located.

After multiplying, the visualization contains as many of the same elements as are indexed using placeholders. In doing so, the settings in the “Multiply Visu Element” dialog are taken into consideration. All new elements in the properties that were preset with placeholders have these replaced with precise indexes. The remaining properties have been applied and copied without changes.

For example, you can have a layout of nine buttons as 3x3 tables, which are all the same size or the same color, but vary in the labeling. The labels are declared as a string array (nine components) and are passed as a value to the “Texts”->”Text” property.

Valid template element:
● Declaration of array variables
  Example: asText: ARRAY[1..3, 1..3] OF STRING;
● Element with applicable element type
● Configuration of at least one property of the applicable element with array variables with index access placeholders
  Example: “Texts”, “Text” property = PLC_PRG.asText[$FIRSTDIM$, $SECONDDIM$]
  This is possible for all properties that permit a variable as a value (for example, also properties from the "Animation" or "Input" categories. To configure multiple properties for an element with arrays and index access placeholders, all arrays must have the same structure with the same dimension. The declarations have to be compatible.
● Configuration of properties that do not vary (and are therefore the same for all generated elements) with the usual values without index access placeholders
  Example:
  sButtonTip : STRING := 'This element is created by multiplication'
  “Texts”, “Tooltip” property = %s
  “Text variables”, “Tooltip variable” property = sButtonTip
Applicable visualization elements

Visualization elements that can be multiplied:

- Rectangle
- Round rectangle
- Ellipse
- Line
- Polygon
- Polyline
- Bézier curve
- Image
- Frame
- Button
- Pie
- SpinControl
- Text field
- Checkbox
- Image switcher
- Lamp
- Dip switch
- Power switch
- Push switch
- Push switch LED
- Rocker switch
- Rotary switch

Configuring and multiplying lamps and buttons as templates

1. Create a new standard project.
   ⇒ A CODESYS Control Win V3 is configured as the device. The MainTask calls PLC_PRG. The implementation language is ST.

2. In PLC_PRG in the program code, declare array variables with basic data type STRING.
   ⇒
   
   ```
   PROGRAM PLC_PRG
   VAR
   axLampIsOn: ARRAY[1..2,1..3] OF BOOL;  // For lamp, property 'variable' and button, user input
   asButtonText: ARRAY[1..2,1..3] OF STRING := // Output text for button, property 'text variables''text variable'
   [
   '1A Lamp', '2A Lamp',
   '1B Lamp', '2B Lamp',
   '1C Lamp', '2C Lamp'
   ];
   END_VAR
   ```

3. Select the application in the device tree and click “Add Object ➤ Visualization”.

You can still use the placeholder % as usual for the text display of variable values in the properties in “Texts”.
4. In the “Add Visualization” dialog, specify the name VisuMain and click “Add” to close the dialog.
5. Drag a “Lamp” element from the “Toolbox” view to the visualization.
6. Configure the fixed property values.
7. Double-click the value field of the “Variable” property.
   ▶ The line editor opens.
8. Click .
   ▶ The Input Assistant opens.
9. Select the array variable PLC_PRG.axLampIsOn from the variable tree.
10. Extend the string at the end, for example with “[i].”
   ▶ If you have activated SmartCoding (“Options” dialog, “SmartCoding” category, “List components immediately when typing” option), then the current variable list appears with the placeholders:

   ![Variable List Screen](image)

11. Select the placeholder $FIRSTDIM$ for the first dimension and confirm the selection.
12. Extend the string at the end, for example with “, s”.
   ▶ The variable list appears again.
13. Select the placeholder $SECONDDIM$ for the second dimension and confirm the selection.
14. Complete the string with a closing bracket.
   ▶ PLC_PRG.axLampIsOn[$FIRSTDIM$, $SECONDDIM$]
   The lamp is configured as a template.
15. Click “Visualization ➔ Multiply Visu Element”.
   ⇒ The “Multiply Visu Element” dialog opens. The default values are derived from the array declarations.
   “Total number of elements”, “Horizontal” = 2
   “Total number of elements”, “Vertical” = 3
16. Declare the distance between the new elements.
   ⇒ “Offset between elements”, “Horizontal” = 3
   “Offset between elements”, “Vertical” = 3
17. Check the advanced settings.
18. Click “OK” to confirm the selection.
   ⇒ The new elements appear in the visualization editor. All properties are configured with a precise index and the array variables are indexed.
19. In the “Visualization Toolbox”, in the “Common Controls” category, select and drag the “Button” element to the visualization editor.
   ⇒ The “Properties” view of the element opens.
20. Configure the fixed property values.
   \( \Rightarrow \) PLC_PRG.asButtonText[$FIRSTDIM$, $SECONDDIM$]

   \( \Rightarrow \) PLC_PRG.axLampIsOn[$FIRSTDIM$, $SECONDDIM$]

   The button is configured as a template.

23. Click “Visualization ➔ Multiply Visu Element”.
   \( \Rightarrow \) The “Multiply visu element” dialog opens. The default values are derived from the array declarations.
   “Total number of elements”, “Horizontal” = 2
   “Total number of elements”, “Vertical” = 3

24. Declare the distance between the new elements.
   \( \Rightarrow \) “Offset between elements”, “Horizontal” = 3
   “Offset between elements”, “Vertical” = 3

25. Check the advanced settings.

26. Click “OK” to confirm the selection.
   \( \Rightarrow \) The new elements appear in the visualization editor. All properties are configured with a precise index and the array variables are indexed.

![Visualization editor with configured elements]
27. Build, start, and download the application.

Visualization at runtime:

You can also configure the template element with array variables that have more than two dimensions, but you can only assign placeholders to a maximum of two of the dimensions. In the additional dimensions, the indexes are fixed.

**Example**

Declaration

```
PROGRAM PLC_PRG
VAR
  asText: ARRAY[1..2, 1..3, 1..6, 1..2] OF STRING;
END_VAR
```

Configure the “Text variables”, “Tooltip variable” property for the template element:

```
PLC_PRG.asText[2, $FIRSTDIM$, $SECONDDIM$, 2]
```
You can configure the template element with a one-dimensional array by means of the index access placeholder $FIRSTDIM$. If the number of new elements to be created is greater than five, then a tabular layout is preset in the "Multiply Visu Element" dialog. The layout of the new elements is as quadratic as possible.

Example

```plaintext
PROGRAM PLC_PRG
VAR
  asText: ARRAY[1..100] OF STRING;
END_VAR
```

The default setting in the "Multiply Visu Element" dialog allows for a layout of 100 new elements in a 10x10 field.

See also

- § Chapter 1.3.5.19.2.11 “Command ‘Multiply Visu Element’” on page 1566
- § Chapter 1.3.5.18.2 “Placeholders with format definition in the output text” on page 1546
- Optionen für Intelligentes Kodieren

1.3.5.10 Displaying data curve with trace

With this element, you can integrate a trace graph in the visualization that monitors and displays variable values permanently. You configure the displayed trace graph in the element properties. In addition, you can add control elements that control the trace functionality. This is done manually or by using the "Insert Elements for Controlling Trace" command.

Configurations for the 'Trace' visualization element can be taken from the 'Trace' object.
1.3.5.10.1 Getting started with trace

Create a project with the following program PLC_PRG:

PROGRAM PLC_PRG
VAR
   iVar : INT;
   rSin : REAL;
   rVar : REAL;
END_VAR

   iVar := iVar + 1;
   iVar := iVar MOD 33;
   rVar := rVar + 0.1;
   rSin := 30 * SIN(rVar);

1. In the device tree, select the application and add a new visualization by clicking “Project ➔ Add Object ➔ Visualization”.
   ⇨ The respective visualization editor opens.
2. Add the “Visualization” object to the device tree below “Application”.
   ⇨ An empty visualization appears.
3. Open “Toolbox ➔ Special Controls”.
4. Drag the “Trace” element to the visualization editor.
   ⇨ The element properties are displayed on the right side.
5. Click the symbol in the “Trace” property.
   ⇒ The “Trace Configuration” dialog opens.

6. Click “Add Variable” to add an entry to the tree view of the trace configuration and select a project variable (for example, PLC_PRG.rSin).

7. Click the top node of the trace configuration.
   ⇒ The group “Record Settings” is shown on the right.

8. Select the MainTask option for the “Task” setting.
   Tip: The trace recording and the corresponding program should be executed in the same task.

9. Click “OK”.
   ⇒ The task configuration is applied.

10. Select the trace element and click “Visualization ➔ Add Elements for Trace Control”
    ⇒ The “Trace Wizard” dialog opens. By default, all control elements are activated there.

11. Click “OK” to close the dialog.
    ⇒ The control elements are added to the visualization and the control variables are declared. Then the control elements and the trace element are configured with the control variables.

12. Download the application to the controller and start it.

Example

The PLC_PRG program is running on the PLC. When you follow the "Getting Started" instructions, the following interface is displayed:

You can control the trace recording by clicking the buttons.
1.3.5.11 Displaying data curve with trend

A trend visualizes data that is used in the database of a trend recording. In contrast to the trace element, the trend element is particularly appropriate for long-term data recording.

The visualization of a trend encompasses the Trend element and the control elements. The three possible control elements can be seen in the illustration.

- Legend ①: Outputs the trend variables with values.
- Time range picker ②: Provides buttons for selecting predefined time ranges.
- Date range picker ③: encompasses control elements for navigation and zooming in the historical and current data on basis of the set date range.

A cursor is optionally available that enables the reading of a value at a certain time.

You can execute a trend visualization in the following clients:

- Target visualization
- Integrated visualization

See also

- Trend recording
- Chapter 1.3.5.11.1 “Getting started with trend visualization” on page 1165
- Chapter 1.3.5.18.1.35 “Visualization element 'Trend'” on page 1471
- Chapter 1.3.5.18.1.45 “Visualization element 'Date range picker'” on page 1521
- Chapter 1.3.5.18.1.46 “Visualization element 'Time range picker'” on page 1525

1.3.5.11.1 Getting started with trend visualization

When you execute a trend, it is best to proceed with user guidance and the help of the trend wizard.
Development of a visualization with trend

1. Create an empty standard project and program at least one variable into PLC_PRG.
   ⇒ PLC_PRG is declared and implemented
2. Add the “Visualization” object to the device tree below “Application”.
   ⇒ An empty visualization appears.
3. Open “Toolbox ➔ Special control elements”.
4. Drag the “Trend” element to the visualization
   ⇒ The “Trend Recording” dialog opens with the “Recording Settings”.
5. Select the task in which the trend recording will be executed.
   In general the trend recording runs in the same task as the main program, i.e. PLC_PRG.
   Therefore, select MainTask.
6. Add a trend variable with “Add Variable” and assign an IEC variable from PLC_PRG to the trend variable.
7. Click “OK” to close “Trend Configuration”.
   ⇒ There is a newly created object of the type Trend recording under “Trend Recording Manager”. The active visualization contains a trend element that is selected.
8. Click “Visualization ➔ Insert Elements for Controlling Trend Elements”.
   ⇒ The “Trend Wizard” dialog box opens.
9. By default, all three control elements are activated in the dialog. Click “OK” to close the dialog box.
   ⇒ The active visualization contains a trend with control elements.
10. Set the application containing the trend objects to active.
11. Compile the application with [F11].
12. Click “Online ➔ Login”.
13. Start the application with [F5].
   ⇒ The target visualization appears. The visualization contains the trend diagram with the value curve of the variable. The control elements enable user inputs.

See also
- Trend recording
- Chapter 1.3.5.11.2 “Programming a trend” on page 1168
- Chapter 1.3.5.19.2.18 “Command ‘Insert Elements for Controlling the Trend’” on page 1577

Example: Visualization of the sinusoidal trend of an IEC variable.
The following objects are implemented in the project:
- PLC_PRG
- Visualization_Trend1
- VisuWithTrend
PLC_PRG runs as part of the application on the controller.

```plaintext
PROGRAM PLC_PRG
VAR
  iVar : INT;
  rSin : REAL;
  rVar : REAL;
END_VAR

iVar := iVar + 1;
iVar := iVar MOD 33;

rVar := rVar + 0.1;
rSin := 30 * SIN(rVar);
```

Visualization_Trend1 is the object that contains the configuration of the trend recording.
**VisuWithTrend**  

VisuWithTrend is the object that visualizes the trend. The visualization contains four elements: one trend and three control elements. The properties of the trend are defined as follows.

<table>
<thead>
<tr>
<th>Properties</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Trend recording”</td>
<td>Visualization_Trend1</td>
</tr>
<tr>
<td>“Display cursor”</td>
<td>✅</td>
</tr>
<tr>
<td>“Display tool tip”</td>
<td>✅</td>
</tr>
<tr>
<td>“Show frame”</td>
<td>✅</td>
</tr>
<tr>
<td>“Date range picker”</td>
<td>Trend1DateRangeSelector</td>
</tr>
<tr>
<td>“Time picker”</td>
<td>Trend1TimeSelector</td>
</tr>
<tr>
<td>“Legend”</td>
<td>Trend1Legend</td>
</tr>
</tbody>
</table>

**VisuWithTrend at runtime**

**1.3.5.11.2 Programming a trend**

If you edit the properties of the trend you can modify the display of the trend recording. The trend wizard, which also edits the properties of the trend, additionally accesses the properties of the control elements.
The visualization task and the trend recording task generally run under the same application. If this is not the case, then the application containing the visualization task requires a data sources manager.

1. Select a trend in the active visualization editor.
   ⇒ The properties of the trend element are displayed on the right side.
2. Double-click the value field “Properties ➔ Application”.
3. Use the input assistant (… ) to select the application. You can also specify the name of the application directly.

See also

- Trend recording
- Data sources manager

Editing control elements

1. Select a trend in the active visualization editor.
2. Click “Visualization ➔ Insert Elements for Controlling Trend”.
   ⇒ The “Trend Wizard” dialog box opens.
3. Double-click in the configuration table on the “Element Type” of the element
   ⇒ A drop-down list appears.
4. Select an element type.
5. Click in the configuration table on the “Instance Name” of an element
6. Enter the new name.
   ⇒ The name is accepted by clicking “OK” to closed the dialog box.
7. Double-click in the configuration table on the “Position” of the element.
   ⇒ A drop-down list appears.
8. Select a position. You must select the position so that there is no overlap with other elements. Overlapping is indicated by an error message.

After closing the dialog box by clicking “OK”, the control element is moved to the new position.

See also

- Chapter 1.3.5.19.2.18 “Command ‘Insert Elements for Controlling the Trend’” on page 1577

Editing the trend recording

1. Select a trend in the active visualization editor.
   ⇒ The properties of the trend element are displayed on the right side.
2. Click the value field of “Properties ➔ Trend recording”
   ⇒ “Select trend recording” appears. The trend recordings available application-wide are listed under “Available trend recordings”.
3. Select a trend recording under “Available trend recordings”.
4. Click …
   ⇒ The trend recording is located under “Selected trend recording”.

2020/12/10
5. Click “OK” to confirm the entry.
   ⇨ The selected trend recording appears in “Values” in “Properties ➔ Trend recording”.

See also
- Trend recording

### Deleting a control element

A control element that was added with the help of the “Trend wizard” cannot be deleted via the Trend wizard dialog.

1. Select the control element of a trend in the active visualization editor.
2. Delete the element with [Del] or “Delete”.
3. Select the trend in the active visualization editor.
4. Delete the assigned value in “Properties ➔ Assigned control elements ➔ <control element>”.

**NOTICE!**
It is absolutely necessary to delete this reference manually. The property is not deleted automatically by deleting the control element.

### Add a further control element

1. Select a trend in the active visualization editor.
2. Click “Visualization ➔ Insert Elements for Controlling Trend”.
   ⇨ The “Trend Wizard” dialog box opens.
3. By default, all three control elements are activated in the dialog.
4. Determine the position of the control element.
5. Select an element type.
6. Edit the instance of the control element.
7. Click “OK” to close the dialog box.
   ⇨ The configured control element is added.

See also
- Chapter 1.3.5.19.2.18 “Command ‘Insert Elements for Controlling the Trend’” on page 1577

### Configuring the coordinate system of the trend diagram

1. Select a trend in the active visualization editor.
2. Use the command “Visualization ➔ Configure Appearance of Trend”.
   ⇨ The “Edit Appearance” dialog box opens.
3. Change the settings.
Reads a trend value at runtime

1. Open “View ➔ Element Properties”.
2. Select a trend element in your visualization.
   ➔ The properties of the trend element are displayed on the right side.
3. Activate the option “Properties ➔ Show cursor” and “Show tooltip”.
   ➔ A cursor is drawn in the coordinate system.
4. Activate the option “Properties ➔ Show tooltip”.
5. Load the application to the controller and start it.
6. If the diagram “runs”, then the date range has been placed in such a way that its end time is the current time.

Select the date range such that the diagram does **not** run. If necessary, drag the scrollbar to an earlier date range.

   ➔ A cursor is available. The tool tip of the cursor informs you of the trend values. For each trend variable, the legend displays the value at the point in time at which the cursor is positioned.

1.3.5.12 Displaying and editing text files

1.3.5.12.1 Configuring the display of a text file

In order to display a text file that is located on the controller, you need not only the element “Text Editor”, but also control elements for selecting, opening and closing the file. Optionally a text search function can be set up in the file with further control elements.

Example:
1. Drag an element “Text Editor” into the visualization editor.

2. Declare the control variables for the element, for example as global variables in the GVL object.
   
   Refer to the declaration of the control variables for this.

3. For the text editor, configure the property “Editing mode” with “Read only”.

4. Also configure the property “Control variables”.

Assign the following variables there:

- “Control variables ➔ File ➔ Variable” with \texttt{g\_sFileName}
- “Control variables ➔ File ➔ Open” with \texttt{g\_bFileOpen}
- “Control variables ➔ File ➔ Close” with \texttt{g\_bFileClose}
- “Control variables ➔ File ➔ New ➔ Variable” with \texttt{g\_bFileNew}
- “Control variables ➔ File ➔ Save ➔ Variable” with \texttt{g\_bFileSave}
- “Control variables ➔ Edit ➔ Variable” with \texttt{g\_sEditSearchFor}
- “Control variables ➔ Edit ➔ Find” with \texttt{g\_bEditFind}
- “Control variables ➔ Edit ➔ Find next occurrence” with \texttt{g\_bEditFindNext}
Declaring the control variables

VAR_GLOBAL

  g_sFileName: STRING := 'Readme.txt';
  g_bFileOpen : BOOL;
  g_bFileClose: BOOL;
  g_bFileNew: BOOL;
  g_bFileSave: BOOL;
  g_sEditSearchFor : STRING;
  g_bEditFind : BOOL;
  g_bEditFindNext : BOOL;
  g_usiErrorHandlingVarForErrorCode: USINT;
  g_bVarForContentChanged : BOOL;
  g_bVarForReadWriteMode: BOOL;
END_VAR

Configuring control elements for the file selection

1. Add an element “Label”.
2. Configure the property “Texts ➔ Text” with File:
3. Add an element “Rectangle” next to it, in which the user can then enter the file name:
4. Configure the property “Texts ➔ Text” with %s:
5. Configure the property “Texts ➔ Text variable” with g_sFileName.
6. Configure the property “Input configuration ➔ OnMouseclick” with “Write a variable”.
   In the dialog “Input Configuration”, select “Text input” as the “Input type”.
   Activate the option “Use text output variable”.
   ➔ The rectangle for the input of the file name is configured.
7. Add an element “Button” for opening the file.
8. Configure the property “Texts ➔ Text” with Open:
9. Configure the property “Input configuration ➔ OnMouseclick” with “Toggle a variable”.
    Assign g_bFileOpen as a variable.
   ➔ The button Open is configured.
10. Add a further element “Button” for closing the file.
11. Configure the property “Texts ➔ Text” with Close:
12. Configure the property “Input configuration ➔ OnMouseclick” with “Toggle a variable”.
    Assign g_bEditFile as a variable.
   ➔ The button Close is configured.

Control elements for searching for a text.

1. Add an element “Label”.
2. Configure the property “Texts ➔ Text” with Text:
3. Alongside it, add an element “Rectangle” for the input of the text to be found.
4. Configure the property “Texts ➔ Text” with %s:
5. Configure the property “Texts ➔ Text variable” with g_sEditSearchFor.
6. Configure the property “Input configuration ➔ OnMouseclick” with “Write a variable”.
   In the dialog “Input Configuration”, select “Text input” as the “Input type”.
   Activate the option “Use text output variable”.
   ➔ The rectangle is configured.
7. Add an element “Button” for starting the search.
8. Configure its property “Texts ➔ Text” with Find.
9. Configure the property “Input configuration ➔ OnMouseclick” with “Toggle a variable”.
   Assign g_bEditFind as a variable.
10. Also add the action “Execute ST-Code”.
    Program the action with: g_bEditFindNext := FALSE;
    ➔ The button is configured.
11. Add a further element “Button”.
12. Configure the property “Texts ➔ Text” with Find next.
13. Configure the property “Input Configuration ➔ OnMouseclick” with “Toggle a variable”.
    Assign g_bEditFind as a variable.
14. Also add the action “Execute ST code”.
    Program: g_bEditFindNext := TRUE;
    ➔ The button is configured.

See also
   ● § Chapter 1.3.5.18.1.41 “Visualization element 'Text editor'' on page 1494

1.3.5.12.2 Configuring the editing of a text file

In order to be able to create a new text file or edit an existing one on the controller with the text editor in the user interface, you need not only the element “Text Editor”, but also control elements for selecting, opening, closing, saving and creating a file.

Example:

Configuring the element “Text editor”, example:

1. Drag an element “Text Editor” into the visualization editor.
2. Declare the control variables for the element, for example as global variables in the GVL object.
   ➔ Refer below to the declaration of the control variables for this.
3. For the text editor, configure the property “Editing mode” with “Read/Write”.

![Image of text editor interface with file changed message]
4. Also configure the property “Control variables”.

Assign the following variables there:

- “Control variables ➔ File ➔ Variable” with g_sFileName
- “Control variables ➔ File ➔ Open” with g_bFileOpen
- “Control variables ➔ File ➔ Close” with g_bFileClose
- “Control variables ➔ File ➔ Save” with g_bFileSave
- “Control variables ➔ File ➔ New” with g_FileNew

```plaintext
VAR_GLOBAL

g_sFileName: STRING := 'Readme.txt';
g_bFileOpen : BOOL;
g_bFileClose: BOOL;
g_bFileSave: BOOL;
g_FileNew: BOOL;
g_usiErrorHandlingVarForErrorCode: USINT;
g_bVarForContentChanged : BOOL;
g_bVarForReadWriteMode: BOOL;
END_VAR
```

1. Add an element “Label”.

2. Configure it in the property “Texts ➔ Text” with File:

3. Add an element “Rectangle” next to it.

4. Configure its property “Texts ➔ Text” with %s.

5. Configure its property “Texts ➔ Text variable” with g_sFileName.

6. Configure the property “Input configuration ➔ OnMouseclick” with “Write a variable”.
   In the dialog “Input Configuration”, select “Text input” as the “Input type”.
   Activate the option “Use text output variable”.
   ⇧ The rectangle for the input of the file name is configured.

7. Add an element “Button”.


9. Configure the property “Input configuration ➔ OnMouseclick” with “Toggle a variable”.
   Assign g_bFileNew as a variable.
   ⇧ The button New is configured.

10. Add a further element “Button”.

11. Configure the property “Texts ➔ Text” with Open:

12. Configure the property “Input configuration ➔ OnMouseclick” with “Toggle a variable”.
    Assign g_bFileOpen as a variable.
    ⇧ The button Open is configured.

13. Add a further element “Button”.


15. Configure the property “Input configuration ➔ OnMouseclick” with “Toggle a variable”.
    Assign g_bFileSave as a variable.
    ⇧ The button Save is configured.

16. Add a further element “Button”.

17. Configure its property “Texts ➔ Text” with Close.
18. Configure the property “Input configuration ➔ OnMouseclick” with “Toggle a variable”.
Assign g_bEditFile as a variable.
⇒ The button Close is configured.

See also
● Chapter 1.3.5.18.1.41 “Visualization element ‘Text editor’” on page 1494

1.3.5.13 Configuring a variable assignment with unit conversion

A variable that was assigned in a visualization can be linked with a unit conversion. This causes the variable value to be converted according to a predefined rule and the result is edited in the visualization.

You have already configured the conversion rules in the editor of an object of the type “Unit Conversion”.

See also
● Using unit conversion

1.3.5.14 Using recipes in visualization elements

You can manage and use the recipes created in CODESYS by means of a visualization. For this purpose, the input configuration of a visualization element provides the following commands:

● “Read Recipe”
● “Write Recipe”
● “Load Recipe from File”
● “Save Recipe to File”
● “Create Recipe”
● “Delete Recipe”

See also
● Changing Values with Recipes
● Chapter 1.3.5.19.3.6 “Dialog ‘Input Configuration’” on page 1586
Requirement: The “Visualization” object is added to the project.

1. Create a recipe according to the instructions in the section "Changing Values with Recipes - Creating Recipes".
   Assign the following names:
   ● Recipe definition: “Recipes”
   ● Recipes: “Recipe1” and “Recipe2”
   ● Variables: iValue1 and iValue2
   Type in different variable values of both recipes.
2. Open the “Visualization” object in the editor.
3. Drag a “Button” element to the visualization. Label it "Load Recipe 1". You can specify the text by double-clicking the element or in the “Texts ➔ Text” property.
4. Click the value field of the “Input configuration”: “OnMouseDown” property.
   ➔ The “Input Configuration” dialog box opens.
5. Select “Execute command” in the left of the left side and click the button ➔ The configuration of the “Internal command” opens on the right side of the dialog.
6. Select the “Write Recipe” command from the drop-down list.
7. Click the button.
   ➔ The “WriteRecipe” command is added to the list.
8. Specify the first parameter as Recipes and the second parameter as Recipe1.
9. Click “OK” to close the dialog box.
10. Drag a second button to the visualization, name it "Load Recipe 2", and repeat steps 4 to 8. For step 7, specify Recipe2 as the second parameter.
11. Load the program to the controller and start it. Click the “Load Recipe 1” and “Load Recipe 2”, and monitor the variables iValue1 and iValue2.

The other recipe commands are assigned to visualization elements as described in this example. Refer to the help page of the input configuration for a description of the internal commands.

1.3.5.15 Creating a structured user interface

You can reference visualizations that are available or exist in the project in another visualization and thus reuse them. You obtain a structured user interface that consists of several visualizations. In principle you have the following possibilities to reference visualizations.

On the one hand you can display visualizations within a main visualization and toggle between them. The element “Frame” or “Tab control” serves here as a window area element that defines the display area for the referenced visualizations.

On the other hand you can configure a user input for a visualization that causes another visualization to open as a dialog. The requirements for this is that it has the visualization type “Dialog”. A dialog is used to collect inputs from the user.

In addition, you can declare an interface for a visualization that is to be referenced in order to vary the display of the visualization at runtime. A visualization is thereby instanced with different data and executed.
1.3.5.15.1 Displaying multiple visualizations in one visualization

You can reference other visualizations within a main visualization either in a frame or a tab control and then display them in the window pane of the element.

In the case of the “Frame” element, you can freely program which of the visualizations is displayed at which time. One option is to use the switch frame variable of the frame element, which automatically triggers a switch according to its value. You can also program additional controls which, after user input, trigger input actions that result in switching a visualization.

NOTICE!

Visualizations can be nested at any depth by means of frame elements. In order to use the “Switch to any visualization” frame selection type without any problems, a frame must not contain more than 21 referenced visualizations.

For more information, see also the description for the “Input configuration” of an element: Action “Switch frame visualization”.

Moreover, you can use the “Tab control” to reference visualizations. It is easy and advantageous that the tab element provides preconfigured control of the visualization switch.

In CODESYS Store, you will find the sample project “Visualization Switching”. There you will see a visualization that displays other visualizations in a frame area one after another at runtime. The visualization switch is controlled either by the user, programmatically, or via the FrameManager.

Switching frame visualizations by means of a variable

Connecting frame visualizations with a radio button

In the main visualization, the “Frame” element displays one of the referenced frame visualizations at runtime. The user can select the “Radio Button” element which is displayed in the frame.

1. Create a new standard project in CODESYS.
2. Select the application in the device tree and click “Add Object ➔ Visualization”.
3. In the “Add Visualization” dialog, specify the name VisuMain and click “Add” to close the dialog.
4. Select the application in the device tree and click “Add Object ➔ Visualization”.
5. In the “Add Visualization” dialog, specify the name Visu1 and click “Add” to close the dialog.
6. Select the application in the device tree and click “Add Object ➔ Visualization”.
7. In the “Add Visualization” dialog, specify the name Visu2 and click “Add” to close the dialog.
8. Select the application in the device tree and click “Add Object ➔ Visualization”.
9. In the “Add Visualization” dialog, specify the name Visu3 and click “Add” to close the dialog.
   In addition to the main visualization, there are three more visualization objects.
10. Open the Visu1 object.
11. In the “Visualization ToolBox”, in the “Basic” category, select and drag the “Radio Button”
element to the visualization editor.
   ➔ The “Properties” view of the element opens.

12. Configure the properties of the rectangle as follows:
   ● Property “Texts”, “Text” = Visu1
   ● Property “Text properties”, “Font” = “Title”
   ● Property “Colors”, “Normal state”, “Fill color” = “Light gray”

13. Program the object Visu2 accordingly.
    Properties of the rectangle:
    ● Property “Texts”, “Text” = Visu2
    ● Property “Text properties”, “Font” = “Title”
    ● Property “Colors”, “Normal state”, “Fill color” = “Gray”

14. Program the object Visu3 accordingly.
    Properties of the rectangle:
    ● Property “Texts”, “Text” = Visu3
    ● Property “Text properties”, “Font” = “Title”
    ● Property “Colors”, “Normal state”, “Fill color” = “Dark gray”

15. Open the VisuMain object.

16. In the “Visualization ToolBox”, in the “Basic” category, select and drag the “Frame” ele-
    ment to the visualization editor.
   ➔ The “Frame Configuration” dialog opens.

17. In the “Available Visualizations” window area, on the “By Visualization Name” tab, select
    the object Visu1. In “Selected Visualizations”, click “Add”.

18. Then select the object Visu2 and click “Add” in “Selected Visualizations”.

19. Then select the object Visu3 and click “Add” in “Selected Visualizations”.
20. Click “OK” to exit the dialog.

⇒ Now the frame element references the three selected visualizations. In the element properties of the frame element, the references (1) are listed in the “References” property. In addition to the visualization name, the corresponding index value (2) is also displayed.

Note: You can open the dialog when you click the “Configure” button in the value field of the “References” property. See (3). You can influence the index by means of the visualization order in the “Selected Visualizations” list.

21. In the “Visualization ToolBox”, in the “Common Controls” category, select and drag the “Radio Button” element to the visualization editor.

⇒ The “Properties” view of the element opens.

22. In the “Radio button settings”, “Radio Button”, click the “Create new” button.

⇒ This element has three switches to select from.
23. Configure the properties of the radio button as follows:
   - Property “Radio button settings”, “Areas”, “[0]”, “Text” = Visu1
   - Property “Radio button settings”, “Areas”, “[1]”, “Text” = Visu2
   - Property “Radio button settings”, “Areas”, “[2]”, “Text” = Visu3

24. In the PLC_PRG program, declare a local variable for the number of the visualization that is active.
   
   ```
   VAR
   iActiveVisu : INT; // Index of visu activated by the user END_VAR
   ```

25. Select the “Radio Button” element. In the value field of the “Variable” property, click [ ].

26. In the “Input Assistant” dialog, select the recently declared variable. Then exit the dialog.
   - Property of the radio button element:
     - Property “Variable” = PLC_PRG.iActiveVisu

27. Select the “Frame” element. Click in the value field of the “Switch frame variable”, “Variable” property. Specify the recently declared variable here as well.
   - Property of the frame element:
     - Property “Switch frame variable”, “Variable” = PLC_PRG.iActiveVisu

   The control variable of the radio button element is also the switch frame variable of the frame element. User input for the radio button switches the frame visualization.

28. Click “Build ➔ Build”.
29. Click “Online ➔ Login” and start the application.

- The visualization starts. One of the referenced visualizations is running in the frame. When you click an unselected option of the radio button, the visualization switches the contents in the frame to the desired visualization.

In the example, the switch frame variable is connected to an input variable. Instead, you can also set the switch frame variable programmatically in the IEC code.

In the main visualization, the “Frame” element displays one of the frame visualizations at runtime. The user can use buttons to control the display in the frame. The user input triggers the “Switch frame visualization” input action.

Switching frame visualizations by means of a follow-up action

Programming a visualization

1. Create a new standard project in CODESYS.
2. Select the application in the device tree and click “Add Object ➔ Visualization”.
3. In the “Add Visualization” dialog, specify the name VisuMain and click “Add” to close the dialog.
4. Select the application in the device tree and click “Add Object ➔ Visualization”.
5. In the “Add Visualization” dialog, specify the name Visu1 and click “Add” to close the dialog.
6. Select the application in the device tree and click “Add Object ➔ Visualization”.

PLC Automation with V3 CPUs
Programming with IEC 61131-3 editor > CODESYS Visualization
7. In the “Add Visualization” dialog, specify the name Visu2 and click “Add” to close the dialog.
8. Select the application in the device tree and click “Add Object ➔ Visualization”.
9. In the “Add Visualization” dialog, specify the name Visu3 and click “Add” to close the dialog.
   ⇒ In addition to the main visualization, there are three more visualization objects.
10. Open the Visu1 object.
11. In the “Visualization ToolBox”, in the “Basic” category, select and drag the “Rectangle” element to the visualization editor.
   ⇒ The “Properties” view of the element opens.
12. Configure the properties of the rectangle as follows:
   ● Property “Texts”, “Text” = Visu1
   ● Property “Text properties”, “Font” = “Title”
   ● Property “Colors”, “Normal state”, “Fill color” = “Light gray”

![Image of Visu1 rectangle configuration]

13. Program the object Visu2 accordingly.
   ⇒ Properties of the rectangle:
   ● Property “Texts”, “Text” = Visu2
   ● Property “Text properties”, “Font” = “Title”
   ● Property “Colors”, “Normal state”, “Fill color” = “Gray”
14. Program the object Visu3 accordingly.
   ⇒ Properties of the rectangle:
   ● Property “Texts”, “Text” = Visu3
   ● Property “Text properties”, “Font” = “Title”
   ● Property “Colors”, “Normal state”, “Fill color” = “Dark gray”
15. Open the VisuMain object.
16. In the “Visualization ToolBox”, in the “Basic” category, select and drag the “Frame” element to the visualization editor.
   ⇒ The “Frame Configuration” dialog opens.
17. In the “Available Visualizations” window area, on the “By Visualization Name” tab, select the object Visu1. In “Selected Visualizations”, click “Add”.
18. Then select the object Visu2 and click “Add” in “Selected Visualizations”.
19. Then select the object Visu3 and click “Add” in “Selected Visualizations”.

20. Click “OK” to exit the dialog.

⇒ Now the frame element references the three selected visualizations. In the element properties of the frame element, the references (1) are listed in the “References” property. In addition to the visualization name, the corresponding index value (2) is also displayed.

Note: You can open the dialog independently when you click the “Configure” button in the value field of the “References” property. See (3). You can influence the index by means of the visualization order in the “Selected Visualizations” list.

21. In the “Visualization ToolBox”, in the “Common Controls” category, select and drag the “Button” element to the visualization editor.

⇒ The element is selected and its properties are visible in the “Properties” view.


23. In the “Input configuration” “OnMouseDown” property, click “Configure”.

⇒ The “Input Configuration” dialog opens.

24. Select the “Switch frame visualization” action and click .

⇒ The action is displayed in the window on the right.

25. Configure the action:

- Select the “Switch local visualization” option.
- Set the “Visualization selection” to Visu1.
- Click “OK” to exit the dialog.

⇒ The follow-up action is configured in the “Input configuration” property.

Property “Input configuration”, “OnMouseDown”, “Switch frame visualization” = 0
26. Drag another “Button” element to the visualization editor. Configure the button accordingly.
   - Properties of the button:
     - Property “Texts”, “Text” = Visu2
     - Property “Input configuration”, “OnMouseDown”, “Switch frame visualization” = 1

27. Drag another “Button” element to the visualization editor. Configure the button accordingly.
   - Properties of the button:
     - Property “Texts”, “Text” = Visu3
     - Property “Input configuration”, “OnMouseDown”, “Switch frame visualization” = 2

28. Click “Build ➔ Build”.
29. Click “Online ➔ Login” for the device and start the application.
   - The visualization starts. One of the referenced visualizations is running in the frame. When you click one of the buttons, the visualization switches the contents in the frame to the respective visualization.

Displaying visualizations in a tab control
For the “Tab control”, the navigation of the referenced visualizations is provided automatically. The first of the referenced visualizations is in the foreground, while the others are hidden behind it. The user can navigate between them by means of the tabs which are provided automatically.

Configuring a tab control
1. Create a new standard project in CODESYS.
2. Select the application in the device tree and click “Add Object ➔ Visualization”.

2020/12/10
3. In the “Add Visualization” dialog, specify the name VisuMain and click “Add” to close the dialog.

4. Select the application in the device tree and click “Add Object ➔ Visualization”.

5. In the “Add Visualization” dialog, specify the name Visu1 and click “Add” to close the dialog.

6. Select the application in the device tree and click “Add Object ➔ Visualization”.

7. In the “Add Visualization” dialog, specify the name Visu2 and click “Add” to close the dialog.

8. Select the application in the device tree and click “Add Object ➔ Visualization”.

9. In the “Add Visualization” dialog, specify the name Visu3 and click “Add” to close the dialog.

⇒ In addition to the main visualization, there are three more visualization objects.

10. Open the Visu1 object.

11. Drag a “Rectangle” element to the visualization editor.

⇒ The “Properties” view of the element opens.

12. Configure the properties of the rectangle as follows:

- Property “Texts”, “Text” = Visu1
- Property “Text properties”, “Font” = “Title”
- Property “Colors”, “Normal state”, “Fill color” = “Light gray”

13. Program the object Visu2 accordingly.

⇒ Properties of the rectangle:

- Property “Texts”, “Text” = Visu2
- Property “Text properties”, “Font” = “Title”
- Property “Colors”, “Normal state”, “Fill color” = “Gray”

14. Program the object Visu3 accordingly.

⇒ Properties of the rectangle:

- Property “Texts”, “Text” = Visu3
- Property “Text properties”, “Font” = “Title”
- Property “Colors”, “Normal state”, “Fill color” = “Dark gray”

15. Open the VisuMain object.

16. In the “Visualization ToolBox”, in the “Basic” category, select and drag the “Frame” element to the visualization editor.

⇒ The “Frame Configuration” dialog opens.
17. In the “Available Visualizations” window area, on the “By Visualization Name” tab, select the object Visu1. In “Selected Visualizations”, click “Add”.

18. Then select the object Visu2 and click “Add” in “Selected Visualizations”.

19. Then select the object Visu3 and click “Add” in “Selected Visualizations”.

20. Click “OK” to exit the dialog.

➤ Now the frame element references the three selected visualizations. In the element properties of the frame element, the references (1) are listed in the “References” property. In addition to the visualization name, the corresponding index value (2) is also displayed.

Note: You can open the dialog “Frame Configuration” dialog independently when you click the “Configure” button in the value field of the “References” property. See (3). You can influence the index by means of the visualization order in the “Selected Visualizations” list.

21. In the “Visualization ToolBox”, in the “Common controls” category, select and drag the “Tab control” element to the visualization editor.

➤ The “Properties” view of the element opens.
22. Configure the properties of the tab as follows:
   - Property “Tab width”: 40
   - Property “References”, Visu1, “Header” = Visu1
   - Property “References”, Visu2, “Header” = Visu2
   - Property “References”, Visu3, “Header” = Visu3

23. Click “Build ➔ Build”.
24. Click “Online ➔ Login” for the device and start the application.
   ➔ The visualization starts. One of the referenced visualizations is running in the tab control. Click the tab to switch to the respective visualization.

See also
- Chapter 1.3.5.19.2.9 “Command 'Frame Selection’” on page 1564
- Chapter 1.3.5.18.1.6 “Visualization element 'Frame’” on page 1287
- Chapter 1.3.5.18.1.10 “Visualization element 'Tab control’” on page 1316

1.3.5.15.2 Calling a visualization with an interface

You can declare an interface for parameters for a visualization that is to be referenced. The actual parameters are passed to the interface (similar as in the case of a function block) when the visualization is called at runtime.
First of all, declare the interface variables in the visualization interface editor. Then configure the parameters that are transferred to the interface by assigning a data-type-compliant application variable to each interface variable. The assignment is configured in the “References” property in the case of a frame or a tab control.

Depending on the display variant, the parameter transfer of local variables (with the VAR scope) is limited. If you execute the visualization as an integrated visualization, you can only transfer local variables having a basic data type as parameters. If the visualization is called as CODESYS TargetVisu or CODESYS WebVisu, then you can also transfer parameters with a user-defined data type.

If you have configured visualization references and then save a change to the variable declaration for one of these visualizations in an interface editor, then the “Updating the Frame Parameters” dialog appears automatically. The dialog prompts you to edit the references. A list of all the visualizations affected is displayed there, so that the parameter transfers can be reassigned at the changed interface.

When the dialog is closed, the changes are accepted and the elements affected are displayed in the “References” property.

User-controlled update of the transfer parameters

If you have configured visualization references and then save a change to the variable declaration for one of these visualizations in an interface editor, then the “Updating the Frame Parameters” dialog appears automatically. The dialog prompts you to edit the references. A list of all the visualizations affected is displayed there, so that the parameter transfers can be reassigned at the changed interface.

When the dialog is closed, the changes are accepted and the elements affected are displayed in the “References” property.

Calling visualization with interface (VAR_IN_OUT)

- Requirement: The project contains a visualization and a main visualization. The main visualization contains an element that the visualization references.
  1. Open the visualization.
  2. Click “Visualization ➔ Interface Editor”.
  3. Declare a variable in the interface editor.
     - The visualization has an interface and the “Updating the Frame Parameters” dialog appears.
  4. Assign a type-compliant transfer parameter to the interface variables in all calls by entering an application variable in “Value”. Close the dialog.
     - A transfer parameter is assigned at the points where the visualization is to be referenced. These now appear in the main visualization in the “References” property.
The visPie visualization contains an animated, colored pie. The visMain main visualization calls the visPie visualization multiple times in a tab control. Color information, angle information, and label are transferred via the pieToDisplay interface variable. The pies vary at runtime.

Visualization visPie:

```
VAR_IN_OUT
  pieToDisplay : DATAPIE;
END_VAR
```

Main visualization visMain:

```
<table>
<thead>
<tr>
<th>&quot;References&quot;</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;visPie&quot;</td>
<td></td>
</tr>
<tr>
<td>&quot;Heading&quot;</td>
<td>A</td>
</tr>
<tr>
<td>pieToDisplay</td>
<td>PLC_PRG.pieA</td>
</tr>
<tr>
<td>&quot;visPie&quot;</td>
<td></td>
</tr>
<tr>
<td>&quot;Heading&quot;</td>
<td>B</td>
</tr>
<tr>
<td>pieToDisplay</td>
<td>PLC_PRG.pieB</td>
</tr>
<tr>
<td>&quot;visPie&quot;</td>
<td></td>
</tr>
<tr>
<td>&quot;Heading&quot;</td>
<td>C</td>
</tr>
<tr>
<td>pieToDisplay</td>
<td>PLC_PRG.pieC</td>
</tr>
</tbody>
</table>
```
DATAPIE (STRUCT)
TYPE DATAPIE : // Parameter type used in visPie
  STRUCT
    dwColor : DWORD; // Color data
    iStart : INT; // Angle data
    iEnd : INT;
    sLabel : STRING;
  END_STRUCT
END_TYPE

GVL
{attribute 'qualified_only'}
VAR_GLOBAL CONSTANT
  c_dwBLUE : DWORD := 16#FF0000FF; // Highly opaque
  c_dwGREEN : DWORD := 16#FF00FF00; // Highly opaque
  c_dwYELLOW : DWORD := 16#FFFF00FF; // Highly opaque
  c_dwGREY : DWORD :=16#88888888; // Semitransparent
  c_dwBLACK : DWORD := 16#88000000; // Semitransparent
  c_dwRED: DWORD := 16#FFFF0000; // Highly opaque
END_VAR

PLC_PRG
PROGRAM PLC_PRG
VAR
  iInit: BOOL := TRUE;
  pieA : DATAPIE; // Used as argument when visPie is called
  pieB : DATAPIE;
  pieC : DATAPIE;

  iDegree : INT; // Variable center angle for the pie element
  used for animation
END_VAR

IF iInit = TRUE THEN
  pieA.dwColor := GVL.c_dwBLUE;
  pieA.iStart := 0;
  pieA.sLabel := 'Blue';

  pieB.dwColor := GVL.c_dwGREEN;
  pieB.iStart := 22;
  pieB.sLabel := 'Green';

  pieC.dwColor := GVL.c_dwYELLOW;
  pieC.iStart := 45;
  pieC.sLabel := 'Yellow';

  iInit := FALSE;
END_IF

iDegree := (iDegree + 1) MOD 360;

pieA.iEnd := iDegree;
pieB.iEnd := iDegree;
pieC.iEnd := iDegree;

Main visualization visMain at runtime:
In order to obtain and output the instance name of a transfer parameter, you can implement an interface variable (data type STRING) with the pragma \{attribute 'parameterstringof'} in the VAR_INPUT scope.

The project contains a visualization and a main visualization. The main visualization contains elements that the visualization references.

1. Open the visualization.
2. Click “Visualization ➔ Interface Editor”.
3. Declare an interface variable (VAR_IN_OUT).
   \(\text{pieToDisplay} : \text{DATAPIE}\);
4. In the interface editor, declare a variable (VAR_INPUT) with attribute \{attribute 'parameterstringof'}.
   \(\{\text{attribute 'parameterstringof'} := \text{pieToDisplay}\}\)
   \(\text{sNameToDisplay} : \text{STRING}\);
5. Save the changes.
   \(\text{The “Updating the Frame Parameters” dialog does not open.}\)
6. Insert a “Text Field” element.
7. In the “Texts”, “Text” property, assign an output text to the text field.
   \(\text{Visualization of %s}\)
8. In the “Text variables” “Text variable” property, assign the interface variable to the text field.
   \(\text{sNameToDisplay visPie has a heading.}\)
The `visPie` visualization consists of one pie until now. The `visMain` main visualization calls `visPie` in a tab control three times with different transfer parameters.

The `visPie` is extended with a text field that outputs the name of the parameters actually passed to the visualization. For this, the interface of `visPie` is extended with a string variable that contains the instance name of the specified transfer parameter. At runtime, each pie is overwritten.

### Table 254: Properties of the “Text field” element:

<table>
<thead>
<tr>
<th>“Texts”, “Text”</th>
<th>Visualization of <code>%s</code></th>
</tr>
</thead>
<tbody>
<tr>
<td>“Text variables”, “Text variable”</td>
<td>sNameToDisplay</td>
</tr>
</tbody>
</table>

### Interface of the ‘visPie’ visualization:

```plaintext
VAR_INPUT
  {attribute 'parameterstringof' := 'pieToDisplay'}
  sNameToDisplay : STRING;
END_VAR
VAR_IN_OUT
  pieToDisplay : DATAPIE;
END_VAR
```

### Main visualization `visMain` at runtime:

The visualization is displayed in three tabs, each with its own color: blue, green, and yellow. The text field outputs the name of the parameter associated with each pie.
1.3.5.15.3 Calling a dialog in a visualization

You can configure a user input for a visualization that causes a referenced visualization to open as a dialog. For example, a user clicks on a button, whereupon a dialog opens requesting the input of values. A dialog is used to collect user inputs and, if it is modal, it can lead to inputs outside the dialog being blocked.

Only visualizations with the visualization type “Dialog” can be opened as dialog. The visualization type is configured in the dialog “Properties” of a visualization object.

Basic procedure:

☑ Requirement: The project contains a main visualization and a dialog.

1. Configure a user input for the main visualization with the action “OpenDialog” for the dialog.
   ➔ The opening of the dialog is configured.

2. Configure a user input for an element of the dialog with the action “CloseDialog”.
   Hint: in the case of non-modal dialogs you can also configure the user input for closing outside the dialog.
   ➔ The closing of the dialog is configured.

You can also use dialogs from the library instead of self-made dialogs. For example, if the library VisuDialogs is integrated in the project, you can use the dialogs VisuDialogs>Login or VisuDialogs.FileOpenSave contained in it.

See also

- Chapter 1.3.5.19.2.1 “Command ‘Interface Editor’” on page 1556
- Chapter 1.3.5.18.1.6 “Visualization element ‘Frame’” on page 1287
- Chapter 1.3.5.18.1.10 “Visualization element ‘Tab control’” on page 1316
- Chapter 1.3.5.19.3.3 “Dialog ‘Updating the Frame Parameters’” on page 1584

Configuring a visualization object as a dialog

1. Select the object in the view “Devices”, open the context menu and select the command “Properties”.
2. Select the tab “Visualization”.
3. Activate the option “Dialog” and close the dialog with “OK”.
   ➔ The visualization has the visualization type “Dialog” and can be called as such.

Configuring a dialog call

When calling a dialog, a user normally clicks on a button, whereupon a dialog opens requesting an input.

In the following example, a dialog representing a calendar enables a date to be entered.
Requirement: The project contains the visualizations `visMain` and `dlgCalender`.

1. Set the visualization type of `dlgCalender` to Dialog.
2. Drag a rectangle into the visualization `visMain`.
3. Configure the property “Texts ➔ Text” with the text `Due Date: %t[yyyy-MM-dd]`.
   Configure the property “Text variables ➔ Text variable” with `PLC_PRG.dateDue`.
4. Drag a button into the visualization.
5. Configure the property “Texts ➔ Text” with the text `Open dialog`.
   Configure the property “Input configuration ➔ OnMouseClick” for the action “Open Dialog” with `dlgCalender`.
   ➔ The user input for the opening of the dialog is configured.
6. Double-click on the dialog `dlgCalender`.
7. Drag the element “Date picker” into the visualization editor.
8. Configure the property “Texts ➔ Text” with `Due Date: %t[yyyy-MM-dd]`.
   Configure the property “Variable” with `PLC_PRG.dateCalender`.
   ➔ The element is configured.
9. Drag a button into the visualization editor.
10. Configure the property “Texts ➔ Text” with `OK`.
11. Configure the property “Input configuration ➔ OnMouseClick” for the action “Close Dialog” with `dlgCalender`, Result: OK.
12. Configure a further property “Input configuration ➔ OnMouseClick” for the action “Execute ST-Code” with `PLC_PRG.dateDue := PLC_PRG.dateCalendar;`.
   ➔ The user input for the closing of the dialog is configured.
13. Drag a further button into the visualization editor.
14. Configure the property “Texts ➔ Text” with `Cancel`.
15. Configure the property “Input configuration ➔ OnMouseClick” for the action “Close Dialog” with `dlgCalender`, Result: Cancel.
   ➔ The user input for the cancellation of the dialog is configured.
17. Compile, load and start the application.

**Variable declaration:**

```plaintext
PROGRAM PLC_PRG
VAR
  dateDue : DATE := DATE#2000-01-01;
  dateCalendar : DATE;
END_VAR
```

**Opening a dialog globally**

Normally a dialog appears only on the display variant on which the user has executed the triggering event.

However, you can configure the opening of the dialog in such a way that the dialog appears simultaneously on all active display variants configured under the visualization manager. This way, for example, an input request can appear simultaneously on all display variants although a user only entered something on the CODESYS TargetVisu.

If a user closes the dialog on a CODESYS TargetVisu display variant, it will be closed on all display variants.
You can open and close a global dialog with the functions `OpenDialog3` and `CloseDialog2` from the library `VisuElems`.

In the application code you can implement the access to a dialog that is managed in the dialog manager. The dialog manager automatically instances and manages all visualizations of the type “Dialog”. The access takes place via the internal visualization manager.

First of all, implement the access to the dialog manager by calling the `GetDialogManager()` method of the internal visualization manager. You can then use the methods of the dialog manager to program the program sequence of a dialog.

In the following example a button is configured so that it opens the preconfigured dialog `Login` when clicked on. The user can enter a name and a password in the dialog. The dialog `Login` is contained in the library `VisuDialogs`. You can also call a self-made dialog in the same way.

**Requirement:** The library `VisuDialogs` is integrated in the project.

1. Insert a new visualization `visMain` under the application.

2. Drag a button into the visualization editor. The visualization editor opens.

3. Enter in its property “Text” `Login`. The button is labelled.

4. Click on “Configure” in the property “Input configuration OnMouseDown”.

5. Select the input action “Execute ST-Code” and click on .

6. Enter the following function call in the ST editor: `OpenLoginDialog(pClientData)`; The main visualization contains a button. If a user clicks on the button, the dialog `Login` opens and the function `OpenLoginDialog()` is called.

7. Click on “Configure” in the property “Input configuration OnDialogClosed”.

8. Select the input action “Execute ST-Code” and click on .

9. Enter the following function call in the ST editor: `OnLoginDialogClosed(pClientData)`; If a user closes the dialog, the function `OnLoginDialogClosed()` is called.

```st
FUNCTION OpenLoginDialog : BOOL
VAR_INPUT
    pClientData : POINTER TO VisuStructClientData;
END_VAR

VAR
    dialogMan : IDialogManager;
```

2020/12/10
dialogMan := g_VisuManager.GetDialogManager(); // The DialogManager is provided via the implicitly available VisuManager IF dialogMan <> 0 AND pVisuClient <> 0 THEN loginDialog :=
dialogMan.GetDialog('VisuDialogs.Login'); // Dialog to be opened is specified
END_IF

FUNCTION OnLoginDialogClosed : BOOL
VAR_INPUT
    pClientData : POINTER TO VisuStructClientData;
END_VAR

VAR
    dialogMan : IDialogManager;
    loginDialog : IVisualisationDialog;
    pLoginInfo : POINTER TO Login_VISU_STRUCT;
    result : Visu_DialogResult;
    stPassword : STRING;
    stUsername : STRING;
END_VAR

dialogMan := g_VisuManager.GetDialogManager(); // The DialogManager is provided via the implicitly available VisuManager IF dialogMan <> 0 AND pVisuClient <> 0 THEN
    loginDialog :=
dialogMan.GetDialog('VisuDialogs/Login'); // Gets the login dialog
    IF loginDialog <> 0 THEN
        result := loginDialog.GetResult(); // Gets the result (OK, Cancel) of the dialog
        IF result = Visu_DialogResult.OK THEN
            loginDialog.SetResult(Visu_DialogResult.None); // Reset to default (none)
            pLoginInfo :=
dialogMan.GetClientInterface(loginDialog, pVisuClient); // Structure Login_VISU_STRUCT gets read;
            // In the following the structure parameters can be set
            IF pLoginInfo <> 0 THEN
                stPassword :=
            END_IF
        END_IF
    ELSE
        stPassword :=
    END_IF
END_FUNCTION

OnLoginDialogClosed() defines the reaction to the closing of a dialog.
pLoginInfo^.stPassword := ''; // Reset the password
stUsername := pLoginInfo^.stUsername;
END_IF
ELSIF result = Visu_DialogResult.Cancel THEN
loginDialog.SetResult(Visu_DialogResult.None); // React on 'Cancel'
ELSE
// nothing to do
END_IF
END_IF
END_IF
END_IF
See also
● Chapter 1.3.5.18.3 “Methods of the dialog manager” on page 1552

1.3.5.15.4 Call dialog with interfaces

You can define an interface for a visualization that is called as a dialog.

Create a visualization for this with visualization type “Dialog” and declare an interface for the dialog. The reference the visualization in a primary visualization by means of a user input and transfer the parameters to the interface.

If you call the visualization as an integrated visualization, then the parameter that are transferred must be variables of a basic data type. If the visualization is called as CODESYS Target-Visu or CODESYS WebVisu, then the parameters can have user-defined data types as well.

See also
● Chapter 1.3.5.19.2.1 “Command 'Interface Editor’” on page 1556
● Chapter 1.3.5.19.3.15 “Dialog 'Properties’ of Visualization Objects” on page 1603

Main procedure

1. Set the visualization types of the visualization to dialog.
2. Declare variables in the interface editor of the dialog.
   ⇒ The dialog has an interface. You can transfer parameters when calling the dialog.
3. Configure the elements of the dialog and use the interface variables.
4. Select an element in another visualization (usually the main visualization) for configuring how the dialog opens.
5. Click “Configure” in the property “Input configuration ⇒ OnMouseDown”.
   ⇒ The “Input Configuration” dialog box opens.
6. Select “Open dialog” in the list of selected input actions.
7. Select one from the “Dialog” drop-down list.
   ⇒ If the selected dialog has an interface, then the interface variables are listed below.
8. Assign a transfer parameter to the interface variables in the “Value” column.
9. Select the result for which the parameters were updated in the list “Update” ⇒ “and” ⇒ “Parameter in case of results”.
10. Activate the option “Open dialog modal”. Click “OK” to close the dialog box.
    ⇒ The dialog opening is configured.
Executing a dialog several times at the same time requires multiple instances of the dialog. These must have already been downloaded to the visualization device when downloading the application. For this purpose, set the number of instances to download in the visualization manager ("Visualizations" tab).

See also

- Chapter 1.3.5.19.3.15 “Dialog 'Properties' of Visualization Objects” on page 1603
- Chapter 1.3.5.19.3.6 “Dialog 'Input Configuration'” on page 1586
- Chapter 1.3.5.15.4 “Call dialog with interfaces” on page 1199

Example

the following application calls the “Change User Level” dialog and prompts the user to select a level and specify a password. If the password agrees, then the “OK” button is enabled. Then the user can close the dialog. The input of the level is also applied.
Dialog visChangeUserLevel:

Declaration of the interface of dialog visChangeUserLevel:

```plaintext
VAR_INPUT
    sTitle: STRING; // title of the dialog box
    sItfLevel0: STRING; // password level 0
    sItfLevel1: STRING; // password level 1
    sItfLevel2: STRING; // password level 2
    sItfLevel3: STRING; // password level 3
    sItfLevel4: STRING; // password level 4
    sItfLevel5: STRING; // password level 5
    sItfLevel6: STRING; // password level 6
    sItfLevel7: STRING; // password level 7
END_VAR
VAR_IN_OUT
    iItfLevel: INT; // user input: level
    sItfPwd: STRING; // user input: password
END_VAR
```

Table 255: Element list of the visChangeUserLevel dialog box:

<table>
<thead>
<tr>
<th>Type</th>
<th>Name</th>
<th>Element properties</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>#0 Image</td>
<td>Background</td>
<td>“Static ID”: VisuDialogs.ImagePoolDialogs.Login</td>
<td>The property assigns the image of a blank dialog with a gray background and a blank blue caption bar to the element. The image is included in the “VisuDialogs” library.</td>
</tr>
<tr>
<td>#1 Box</td>
<td>Title</td>
<td>“Texts ➔ Text”:<code>%s</code></td>
<td>Output with placeholder for text variable</td>
</tr>
<tr>
<td></td>
<td></td>
<td>“Text variables ➔ Text variable”: sItfTitle</td>
<td>Assignment of interface variable sItfTitle for which a parameter is transferred at call time.</td>
</tr>
<tr>
<td>#2 Radio button</td>
<td>Input level</td>
<td>“Variable”: iItfLevel</td>
<td>Assignment of interface variable iItfLevel for which a parameter is transferred at call time. Includes the user input at runtime.</td>
</tr>
</tbody>
</table>

“Number of columns”: 4
<table>
<thead>
<tr>
<th>Type</th>
<th>Name</th>
<th>Element properties</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>&quot;Radio button order&quot;: &quot;Left to right&quot;</td>
<td>Display</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&quot;Radio button settings&quot;</td>
<td>Label of eight radio buttons with numbers from 0 to 7</td>
</tr>
<tr>
<td></td>
<td></td>
<td>➔ Radio button ➔ Areas&quot;: [0] bis [7]&quot;</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>➔[&lt;n&gt;] ➔ Text&quot;: &lt;n&gt;</td>
<td></td>
</tr>
<tr>
<td>#3</td>
<td>Text field</td>
<td>&quot;Texts ➔ Text&quot;: %s</td>
<td>Output with placeholder for text variable</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&quot;Text variables ➔ Text variable&quot;:</td>
<td>Assignment of interface variable sItfPwd for which a parameter is transferred at call time.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>sItfPwd</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>&quot;Input configuration ➔ OnMouseDown ➔</td>
<td>In the “Input configuration” dialog, “Text input” is selected for the “Input type” drop-down</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Write variable&quot;:</td>
<td>list and the option “Use text output variable” is activated.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Variable:,InputType:Edit, Use text output variable : TRUE</td>
<td></td>
</tr>
<tr>
<td>#4</td>
<td>Text field</td>
<td>&quot;Texts ➔ Text&quot;: Level:</td>
<td>Label</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>#5</td>
<td>Text field</td>
<td>&quot;Texts ➔ Text&quot;: Password</td>
<td>Label</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>#6</td>
<td>Button</td>
<td>&quot;Texts ➔ Text&quot;: OK</td>
<td>Label</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&quot;Colors ➔ Color&quot;: Element base color</td>
<td>Configuration of the display in state-dependent colors. You can switch between colors.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&quot;Colors ➔ Alarm color&quot;: Alarm fill color</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>&quot;Color variables ➔ Toggle color&quot;:</td>
<td>If the password and the user input do not agree, then the expression is TRUE. Then the button is displayed in the alarm color.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>sItfPwd &lt;&gt; MUX(iItfLevel, sItfLevel0, sItfLevel1, sItfLevel2, sItfLevel3, sItfLevel4, sItfLevel5, sItfLevel6, sItfLevel7);</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>&quot;State variables ➔ Deactivate inputs&quot;:</td>
<td>If the password and the user input do not agree, then the expression is TRUE. The button is deactivated.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>sItfPwd &lt;&gt; MUX(iItfLevel, sItfLevel0, sItfLevel1, sItfLevel2, sItfLevel3, sItfLevel4, sItfLevel5, sItfLevel6, sItfLevel7);</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>&quot;Input configuration ➔ OnMouseDown ➔</td>
<td>If a user clicks the “OK” button, then the visChangeUserLevel dialog is closed and the para-</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Close dialog&quot;: Close Dialog:</td>
<td>meters are updated.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>visChangeUserLevel, Result : OK</td>
<td></td>
</tr>
<tr>
<td>#7</td>
<td>Button</td>
<td>&quot;Texts ➔ Text&quot;: Cancel</td>
<td>Label</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Type</td>
<td>Name</td>
<td>Element properties</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------</td>
<td>-------------------------------------------</td>
<td>-------------------------------------------------------------------------------------</td>
<td>-------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&quot;Colors ➔ Color&quot;: Element base color</td>
<td>Display</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&quot;Input configuration ➔ OnMouseDown ➔ Close dialog&quot;: Close Dialog: visChangeUserLevel,</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Result: Cancel</td>
<td>If a user clicks the &quot;Cancel&quot; button, then the visChangeUserLevel dialog is closed.</td>
</tr>
</tbody>
</table>
### Table 256: Element list of the visMain visualization:

<table>
<thead>
<tr>
<th>Type</th>
<th>Name</th>
<th>Element properties</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>#5 Text field</td>
<td>Button for change user level</td>
<td>“Texts ➔ Text”: %s</td>
<td>Output with placeholder.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>“Text variables ➔ Text variable”: PLC_PRG.iLevel</td>
<td>Assignment of the PLC_PRG.iLevel variables to the placeholder. Includes the level number.</td>
</tr>
<tr>
<td>#6 Button</td>
<td>Title</td>
<td>“Texts ➔ Text”: Change user level</td>
<td>If a user clicks the Change user level button, then the visChangeUserLevel dialog opens with the parameter list stored here.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>“Input configuration ➔ OnMouseDown ➔ Open dialog”:</td>
<td>Tip: Click “Configure” to view the stored configuration in the “Input Configuration” dialog (input action “Open dialog”).</td>
</tr>
</tbody>
</table>

### Table 257: Configuration of the call of dialog visChangeUserLevel:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sItfTitle</td>
<td>STRING</td>
<td>'ChangeUse user level'</td>
<td>Transfer of a string for the title.</td>
</tr>
<tr>
<td>sItfLevel0</td>
<td>STRING</td>
<td>'pwd0'</td>
<td>Transfer of a string as password for Level0.</td>
</tr>
<tr>
<td>sItfLevel1</td>
<td>STRING</td>
<td>'pwd1'</td>
<td>Transfer of a string as password for Level1.</td>
</tr>
<tr>
<td>sItfLevel2</td>
<td>STRING</td>
<td>'pwd2'</td>
<td>Transfer of a string as password for Level2.</td>
</tr>
<tr>
<td>Parameter</td>
<td>Type</td>
<td>Value</td>
<td>Description</td>
</tr>
<tr>
<td>----------------</td>
<td>----------</td>
<td>---------</td>
<td>---------------------------------------------------------</td>
</tr>
<tr>
<td>sItfLevel3</td>
<td>STRING</td>
<td>'pwd3'</td>
<td>Transfer of a string as password for Level3.</td>
</tr>
<tr>
<td>sItfLevel4</td>
<td>STRING</td>
<td>'pwd4'</td>
<td>Transfer of a string as password for Level4.</td>
</tr>
<tr>
<td>sItfLevel5</td>
<td>STRING</td>
<td>'pwd5'</td>
<td>Transfer of a string as password for Level5.</td>
</tr>
<tr>
<td>sItfLevel6</td>
<td>STRING</td>
<td>'pwd6'</td>
<td>Transfer of a string as password for Level6.</td>
</tr>
<tr>
<td>sItfLevel7</td>
<td>STRING</td>
<td>'pwd7'</td>
<td>Transfer of a string as password for Level7.</td>
</tr>
<tr>
<td>iItfLevel</td>
<td>INT</td>
<td>PLC_PRG.iLevel</td>
<td>Transfer of a variable for the level specified by the user.</td>
</tr>
<tr>
<td>sItfPwd</td>
<td>STRING</td>
<td>PLC_PRG.sPwd</td>
<td>Transfer of a variable for the password specified by the user.</td>
</tr>
</tbody>
</table>

Table 258: List “Update” and “Parameter in case of result”

<table>
<thead>
<tr>
<th>“Value”</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>“OK”</td>
<td>activated</td>
</tr>
</tbody>
</table>

“Open in dialog mode” activated input outside of the dialog is not possible.

**Application code PLC_PRG:**

```plaintext
PROGRAM PLC_PRG
VAR
iLevel: INT;
sPwd : STRING;
END_VAR
```
After clicking the button, the dialog opens and permits input. If the specified text agrees with the stored text, then “OK” is enabled:

After clicking “OK”, the selection is applied.

The example shows the procedure for multiple return values. However, the password can be returned more easily with a local variable in the dialog.
The variables declared in the interface of a visualization are available automatically as structure variables. They are identified by `<Name of visualization>_VISU_STRUCT`. Therefore, you can access the interface variables of visualizations that appear as a dialog. Normally you use the structure in the application code of a function that is called by a user input.

To pass a complex data structure, you can flag an interface variable of type `VAR_IN_OUT` with the pragma attribute `VAR_IN_OUT_AS_POINTER` and pass a pointer or reference to it as a parameter.

1. Declare the user data object (`DUT`).
2. In the interface editor of a dialog, declare an interface variable (`VAR_IN_OUT`) as a reference to the data object by assigning the attribute `VAR_IN_OUT_AS_POINTER` to the variable.
3. Program the user interface: use the dialog in a visualization or assign the dialog in the input configuration of a visualization element. Then access to the referenced data is possible.
Example:
Using an interface with the pragma "VAR_IN_OUT_AS_POINTER"

```
FUNCTION_BLOCK ControlFB
VAR_INPUT
END_VAR
VAR_OUTPUT
END_VAR
VAR
  bOk : BOOL := TRUE;
  nCounter : INT;
  nValue : INT;
END_VAR
nCounter := nCounter + 1;
```

Declaration of an interface variable with VAR_IN_OUT_AS_POINTER
User interface: dialog opens:

```
VAR_INPUT
  nValueIn : INT;
END_VAR
VAR_IN_OUT
  {attribute 'VAR_IN_OUT_AS_POINTER'}
    controller : ControlFB;
END_VAR
VAR_OUTPUT
  nValueOut : INT;
END_VAR
```
1.3.5.16 Configuring and executing display variants

You can execute the visualization created in CODESYS as a display variant. You can execute just one or also several display variants at the same time. The content of the visualization is thereby always the same in all variants, even if several variants are executed at the same time. If CODESYS is opened in addition, the same content of visualization is also displayed in the visualization editor at runtime (with the same content).

For the configuration of each display variant of your visualization, you use an corresponding object under the visualization manager. If no object is inserted under the visualization manager, the visualization is automatically displayed as an integrated visualization when the application is started.

The following object types are available to you:

- WebVisu

You can execute a display as CODESYS TargetVisu, but you can also execute your visualization any number of times as CODESYS WebVisu and thus also insert any number of objects with the object type WebVisu.

As soon as an object exists under the visualization manager, the visualization task VISU_TASK is automatically inserted under the task configuration. The task is automatically deleted if no more objects exist under the visualization manager or if the objects under it are excluded from compilation. You can configure an individual object accordingly in its dialog "Properties" on the tab "Compile".

See also

- "Chapter 1.3.5.18.4 “Attribute 'VAR_IN_OUT_AS_POINTER'” on page 1554"
If an error or an exception occurs in a visualization at runtime, the execution of the visualization is stopped without stopping the execution of the application. An error screen appears informing you of this. In addition, the error screen (usually) enables you to restart the visualization. This exception handling takes place from visualization profile CODESYS V3.5 SP7, compiler version 3.5.7.0 and a runtime system from version 3.5.7.0.

Select the command “Stop Execution at Handled Exceptions” in order to investigate the cause of the occurrence of exceptions and the error position.

See also
- ‘Stop execution on handled exceptions’

In order to programmatically identify a display variant, the VisuFbClientTagDataHelper library module from the VisuElemBase library is available to you. The library itself is referenced in VisuElems. The library module is typically called with VisuElems.VisuFbClientTagDataHelper.

Further information on this library module can be found in its documentation in the library manager.

See also
- Using library modules

### Executing as CODESYS WebVisu

**NOTICE!**

**Recommendations for data security**

In order to minimize the risk of breaches of data security, we recommend the following organizational and technical measures for the system on which your applications run:

As far as possible, avoid exposing the PLC and control networks to open networks and the Internet. For protection, use additional data link layers such as a VPN for remote access and install firewall mechanisms. Limit access to authorized persons, change any existing standard passwords during the initial commissioning and continue to change them regularly.

If you nevertheless wish to publish your web visualization, it is urgently recommended that you provide it at least with simple password protection in order to prevent anyone accessing your control functionality over the Internet. (See an example in the project SimpleWebvisuLogin.project, which is provided with the standard installation of the development system).

**Use the latest versions of the gateway server and the web server.**

You can execute a visualization as CODESYS WebVisu.

The requirement for this is that the runtime system contains a web server with WebVisu support. This enables communication between target system and web browser. The web server on the target system is started as soon as an application with WebVisu configuration is started and runs until all applications with WebVisu are ended. The device can then display visualizations in connected HTML5-capable web browsers.

The web-based display variant of the CODESYS Visualization enables remote access to a plant as well as its remote monitoring, service and diagnosis over the Internet. A web browser communicates by Java Script (optionally with SSL encryption) with the web server in the controller and displays the visualization by means of HTML5. This technology is supported by virtually all browsers and is thus also available on terminal devices with iOS or Android.
Configuring and starting display variants

- An executable visualization `visMain` exists in the project.
1. Select the object “Visualization manager” and select the command “Add object”.
2. Select the object “WebVisu” and enter the name `WebVisu_A`.
   - There is a new object in the device tree underneath the object “Visualization Manager”. The associated editor opens.
   - The visualization task `VISU_TASK` is automatically added under the task configuration.
3. Select the visualization `visMain` in the “Start Visualization”.
4. In “Name of .htm file”, enter the name `webvisuA`.
5. Click on “Show used visualizations” and check whether the selected visualization is activated for a download to the associated device.
   - The visualization is configured. The settings under “Scaling options” determine the window size and the scaling.
6. Start a suitable runtime system with web server and WebVisu support.
   - Configure the communication settings for your system.
   - The runtime system runs.
7. Compile, load and start the application.
   - The application and the web server run.
8. Start a web browser with the following address: `http://localhost:8080/webvisuA.htm`
   - The page is displayed and you can see the data of the application and operate the application.

See also
- Chapter 1.3.5.19.4.7 “Object ‘TargetVisu’” on page 1623

Calling a page in the web server

- Requirement: A visualization with WebVisu is started.
1. Start a current browser with JavaScript and support of HTML5-Canvas, e.g. Firefox, Chrome, IE>=9.
2. Enter the following address in the web browser:
   - `http://localhost:8080/webvisu.htm`
   - The page is displayed and you can see the data of the application and operate the application.
Identifying WebVisu

In order to be able to identify a WebVisu with the help of the library block `VisuFbClientTagDataHelper`, the WebVisu needs a name. In order to be able to specifically address it in the application, expand the URL call by the parameter `ClientName=<Name>`.


See also
- Chapter 1.3.5.19.4.8 “Object ‘WebVisu’” on page 1624

1.3.5.16.2 Executing as an Integrated Visualization

You can execute the visualization as an integrated visualization. In this case a display variant of the visualization runs on the development system **without** the visualization code being loaded to the controller.

Use the integrated visualization for the testing and diagnosis of your application, or for the service and commissioning of a plant.

The requirement for this is that there are no objects under the visualization manager. Alternatively, any objects located there can be excluded from compilation. You can configure an individual object accordingly in its dialog “Properties” on the tab “Compile”.

See also
- Chapter 1.3.5.19.3.15 “Dialog ‘Properties’ of Visualization Objects” on page 1603

Configuring and starting display variants

- A visualization project is open.
- Remove all objects from underneath the visualization manager or exclude the objects from compilation.
  - The VISU_TASK has been removed from under the task configuration.
- Load the application to the controller.
  - Now no visualization code will be transferred on loading the application.
- Start the application.
  - The visualization in the visualization editor is being executed. You can operate your application.

Use the command “Activate keyboard usage” in order to toggle between the keyboard usage of the integrated visualization and the keyboard usage of CODESYS.

See also
- Chapter 1.3.5.19.2.4 “Command ’Activate Keyboard Usage’” on page 1560
- Chapter 1.3.5.19.1 “Keyboard Shortcuts for Default Keyboard Action” on page 1555

Restrictions in the variable output

Numerical variable values, which are output within a text in an integrated visualization, are displayed according to the current display format. You can select the display format with the command “Debug ➔ Display”.
See also

- Command 'Display' – 'Binary', 'Decimal', 'Hexadecimal'

Data server restrictions
A variable value that is transferred via the data server is not output. The integrated visualization only outputs the initialization or the last transferred value.

The integrated visualization thus only enables a passive observation of the application.

Restrictions in variable types
VAR_INPUT variables behave like integrated visualizations such as VAR_IN_OUT variables during execution.

Restrictions in expressions and monitoring
Only the following expressions, which are also used in the monitoring mechanism of the development system, are supported in an integrated visualization.

Variable access:
- Example: PLC_PRG.myPou.nCounter

Array access:
- Access to an array of scalar data types, where a variable is used as an index
  Example: a[i]
- Access to an array of complex data types (structure, function block, array), where a variable is used as an index
  Example: a[i].x
- Access to a multidimensional array of all kinds of data types with one or more variable indices
  Example: a[i, 1, j].x
- Access to an array with constant index
  Example: a[3]
- Accesses like those described above in which simple operators are used for the calculations inside the index brackets.
  Example: a[i+3]
- Nested combinations of the complex expressions listed above
  Example: a[i + 4 * j].aInner[j * 3].x

Operators in index calculations:
- +, -, *, /, MOD

Pointer monitoring:
- Example: p^.x

Methods and function calls are not supported with the exception of the following:
- Standard string functions
- Type conversion functions
  Example: INT_TO_DWORD
- Operators such as SEL, MIN, etc.

Restrictions in the input action “Execute ST-Code”
When the input action “Execute ST-Code” is called, only a list of assignments is supported. If a list of assignments is used, the value of the left-hand side is not assigned until the next cycle. Processing in the next row immediately afterwards is not possible.
Example

```plaintext
PLC_PRG.n := 20 * PLC_PRG.m; // Don't use this!
IF PLC_PRG.n < MAX_COUNT THEN
  PLC_PRG.n := PLC_PRG.n + 1;
END_IF
//Use the following!
PLC_PRG.n := MIN(MAX_COUNT, PLC_PRG.n + 1);
```

Restrictions in the interface of a visualization

No interface (INTERFACE) may be declared in the interface editor of a visualization.

1.3.5.16.3 Configure File Transfer Mode

When downloading, usually files required by the visualization for displaying, are transmitted to the respective display unit. These are especially image files or text list files.

Alternatively, you can configure, that the visualization accesses local files. So no files are transferred with a download.

The following configuration is required to allow the visualization access to local files:

- The file paths for image files or text files lists are relative.
- The link type for image files is “Link to file”.

### Using local visualization files

- **Requirement:** You have opened a visualization project with an image pool.
  1. Open the image pool.
  2. Select for each image under “Link Type” the setting “Link to file”.
     - The image is linked.
  3. Select the command “Project ➔ Project Setting” and select the category “Visualization”.
  4. Insert in tab “General” in “Image files” the local paths of the image files with relative path names.
     - Example: `.\\images\`
     - Note: When no path is specified, the setting in dialog box “Options”, category “Visualization”, tab “File Options” setting “Image files” is used.
  5. Open the visualization manager.
  6. Activate under “Extended settings” the option “Visible”.
  7. Activate under “File Transfer Mode” the option “Use local visualization files”.
     - When downloading, no files are transferred. When displaying the visualization, the local files are used.

See also

- `Object 'Image pool'`
- `Object 'Image pool'`
  - Chapter 1.3.5.19.3.13 “Dialog ‘Project Settings’ - ‘Visualization’” on page 1602
1.3.5.17 Applying visualization styles

A visualization style is a collection of colors, fonts, images, and any values that are defined as style properties. When designing a visualization element, you can use these style properties only. The you have a uniform, style-dependent appearance.

An element that applies style colors and style fonts behaves according to the selected style design in each selected style. In this way, a style property, such as Element basic color, can be blue in one style and gray in another style. In contrast, if the color of an element has a fixed value, this color is fixed even when the style is switched.

All applicable styles are consistent because they define a fixed set of style properties. Therefore, you can switch smoothly between styles in order to customize your visualization. You can preview a style to get an impression of how it behaves.

CODESYS provides different styles, for example the styles Flat style and White Style. These provided styles are installed in the visualization style repository.

The selected style that applies to all visualizations in the application is set in the “Visualization Manager” object (“Settings” tab, “Style Settings” group, “Selected style”). In addition, the “Properties” view provides its style properties when designing an element. For each element, you can assign these styles instead of fixed values.

The style is applied to all visualizations that are below an application. The settings of the “Options - Visualization Styles” dialog are also considered for a library visualization or a visualization in the POUs view.

See also
- Chapter 1.3.5.19.3.11 “Dialog 'Project Environment' - 'Visualization Styles’” on page 1601
- Chapter 1.3.5.19.3.7 “Dialog 'Options' - 'Visualization Styles'” on page 1597

Designing visualization elements with style properties

The set style includes style properties. These are provided in the “Properties” view of an element in the drop-down list of the “Value” column. It is checked which style properties are appropriate for which property. For example, only style properties with color definitions are available for a color assignment.

A style can have directly defined visualization element properties. If this style is used in the project, then these properties are not configurable anymore.

Requirement: A project is open with a visualization.

1. Double-click the visualization.
2. Select an element.
3. Choose “View ➔ Element Properties”.
4. Click in the input field of a color in the window “Properties” (category “Colors”).
   - The list box opens with style properties. The style colors are based on the currently selected style.
5. Select a style property.
   - The visualization shows the element according to the style.

Example

A visualization uses the style CompanyStyle8, which defines the colors CompanyRed, CompanyBlue, and CompanyGreen. An element is selected in the visualization. You can configure the element in the “Properties” view. By clicking into the value field of the “Color” property, you receive a drop-down list with the entries CompanyRed, CompanyBlue, and CompanyGreen.
Switching visualization styles

When setting a style in the visualization manager, all complete styles in the repository are available for selection. It does not matter and it is not evident if a style have been derived from another style.

You can preview a style to get an impression of how it behaves.

How a visualization implements a style at runtime also depends on the display variant. For example, if a font that is defined in the style is not available, the display variant shows the visualization with a preset font.

☐ Requirement: A project is open with a visualization.

1. Double-click the “Visualization Manager” object in the device tree.
   ⇒ The editor opens.

2. Click in the input field of “Selected style” (“Settings” tab, “Style Settings” group).
   ⇒ All styles that are installed in the repository are listed.

3. Mouse over a style.
   ⇒ A preview of how the style is displayed appears in a new window.

4. Select a style.
   ⇒ The style is applied. The preview in “Style Settings” shows the new setting.

5. Double-click a visualization.
   ⇒ The visualization appears in the new style.

Updating versions

☐ Requirement: A project is open with a visualization.

1. Click “Project ➔ Project Environment”, “Visualization Styles” tab.
   ⇒ CODESYS lists all new versions of the currently used styles.

2. Click “Set All to Newest”.
   ⇒ The style is updated. Visualizations and their elements apply the new style.

1.3.5.17.1 Editing visualization styles in the visualization style editor

A style is an XML file with the file extension *.visustyle.xml. It contains a specific set of style properties. CODESYS checks the style properties in the consistency check.

You can create a new style or customize an existing style. The visualization style editor is available for this.

If you customize a style, then a new style is created as a hierarchy of styles. A hierarchy consists of at least two styles based on each other. The nesting depth is unrestricted. A hierarchy is identified simply with its top derived style. You can derive multiple different styles from one base style by extending the styles by differing style properties. This save memory and therefore should be your preferred method.

A base style does not have to be consistent for itself. Instead, you must identify it as an incomplete style. Only the top derived style must be consistent.
Example of a style hierarchy

In a partial style, you can combine any style properties to form efficient hierarchies without having to worry about consistency. For example, you can collect all image references into one partial style. Then you derive the style and define more style properties for colors. This style is also incomplete. You derive the style again and define more style properties for its fonts. The top style is now completely.

- **(1):** CompanyImg is a partial style defining image references.
- **(2):** CompanyColor is a partial, derived style based on CompanyImg and also defines colors.
- **(3):** PetrolStyle is a complete, derived style based on CompanyColor and also defines a special color.
- **(4):** The hierarchy of styles comprises PetrolStyle, CompanyColor, and CompanyImg.
In the visualization style editor, you can open a style, define its style properties, and localize its name. If the style is consistent, then you can install it in the visualization style repository. The editor is not integrated in CODESYS. However, you can start the editor in CODESYS.

**Names for style properties**

A style property is an entry for a specific color, a specific font, or a specific image reference. If this name contains a dash, then the Visualization Style Editor can sort the style properties by the prefixed terms before the dash and display them in a hierarchy. Otherwise the names can be sorted in alphabetical order or sequential order or in sequential order (as saved in the XML file). CODESYS displays the style properties in the order of names actually saved in the XML file for the style.

Example: `Element-Alarm-Fill-Color`

See also

- ☑ Chapter 1.3.5.20.3 “Editor ‘Visualization Style Editor’” on page 1947

**Starting the editor in CODESYS**

1. Double-click the “Visualization Manager” object.
   ⇒ The editor opens.
2. Click the symbol ☑ (“Settings” tab, “Style Settings” group).
3. Click “Open Style Editor” from the drop-down list.
   ⇒ The “Visualization Style Editor” opens.

**Starting the editor independent of CODESYS**

[Link]

Choose “Visualization Styles Editor” from the CODESYS install folder in the Start menu. If you have a standard installation, then this link is located in CODESYS (the program folder for CODESYS).

⇒ The visualization style editor opens.

**Deriving visualization styles**

![Info]

This is the recommended way to create a style that combines existing style properties with new ones.

**Starting the editor in CODESYS and deriving styles**

☑ Requirement: CODESYS is open with a project containing a visualization.

1. Double-click the “Visualization Manager” object in the device tree.
   ⇒ The editor opens.
2. Click the symbol ☑ (“Settings” tab, “Style Settings” group).
   ⇒ A list of commands opens.
3. Choose “Create and Edit Derived Style”.
   ⇒ The visualization style editor starts and the “Create a New Visualization Style” dialog box opens.
4. Type a name.
5. Select a directory.
6. Select a base style. The default style is set in CODESYS. You can also select a style from the repository.
   ⇒ The new style appears in the visualization style editor. The style properties from the base style are displayed yellow.
7. Add a new style property or modify an existing value.
8. Provide a version for the style and click “File ➔ Save and Install”.
   ⇒ The style is installed in the repository. The memory requirement is low because only the style property added in step 7 is saved.

Copying visualization styles

☑ Requirement: CODESYS is open with a project containing a visualization.
1. Double-click the “Visualization Manager” object in the device tree.
   ⇒ The editor opens.
2. Click the symbol ☰ (“Settings” tab, “Style Settings” group).
   ⇒ A list of commands opens.
3. Choose “Copy and Edit Style”.
   ⇒ The visualization style editor starts and the “Open Existing Style as a Copy” dialog box opens.
4. Select which style should be copied (“Style”).
5. Type a directory in “Destination” and click “OK”.
   ⇒ The new style appears in the visualization style editor. All style properties are identical to those in the copied style.
6. Type a name.
7. Add a new style property or modify the value of an existing style property.
8. Provide a version for the style and click “File ➔ Save and Install”.
   ⇒ The style is installed in the repository and the style properties are identical to the added style property, except for the style property added in step 8. The memory requirement is high because the common style properties are defined in both styles.

Creating new visualization styles

☑ Requirement: CODESYS is open with a project containing a visualization.
1. Double-click the “Visualization Manager” object in the device tree.
   ⇒ The editor opens.
2. Click the symbol ☰ (“Settings” tab, “Style Settings” group).
   ⇒ A list of commands opens.
3. Click “Open Style Editor”.
   ⇒ The visualization style editor opens.
4. Click “File ➔ New Style”.
   ⇒ The “Create a New Visualization Style” dialog box opens.
5. Type a name. Specify a base style.
6. Specify a directory and click “OK” to close the dialog box.
   ⇒ The new style appears in the visualization style editor.
7. Add a new style property.
8. Provide a version for the style and click “File ➔ Save and Install”.
   ⇒ The style is installed in the Visualization Styles Repository with the version number.

**Adding a style property**

Using the visualization style editor, you can edit a style to save and install as a new version.

☑  Requirement: The visualization style editor is open with a style.
1. Select a style property and click “Styles ➔ New Style (Afterwards)” in the “Style Properties” tab.
   ⇒ A new style property is added.
2. In the “General” tab, type a new version number in the “Version” setting.
3. Choose “File ➔ Save and Install”.
   ⇒ The changes are saved and the style is installed in the repository as the new version.

**Localizing style properties**

You can assign a language-dependent name to a style property. CODESYS displays a style property by its localized name, depending on the language settings in category “International Settings” (menu “Tools ➔ Options”).

☑  Requirement: The visualization style editor is open with a style.
1. Translate the name of the style property into the localized language in the “Localization” tab.
2. Provide a version for the file in the “General” tab.
3. Choose “Save and Install”.
   ⇒ The edited style is installed in the repository currently selected in CODESYS.
4. Update the style.
5. Set the language settings in CODESYS to the localized language.
6. Open a visualization and select an element. The style settings in its properties are displayed in the localized language.

**1.3.5.17.2 Managing visualization styles in repositories**

The styles that are listed in CODESYS in the drop-down lists of different dialogs and editors are all checked for consistency and installed in the visualization style repository. For derived styles, the hierarchy is checked completely and all styles of the hierarchy are installed. The repository is a version control system within the development system.

You can open a style as write-protected from the visualization style repository in the visualization style editor. The “Save” and “Save and Install” commands are not available there for read-only files. However, you can derive it as the basis for a new style or as a copy.

See also
- ☰ Chapter 1.3.5.19.2.20 “Command 'Visualization Style Repository’’ on page 1580
- ☰ Chapter 1.3.5.20.3 “Editor ‘Visualization Style Editor’” on page 1947
Installing styles to repositories

1. Click “Tools ➔ Visualization Style Repository”.
   ⇒ The “Visualization Styles” dialog box opens.

2. Select the “System” repository in the drop-down list of “Storage location”.
   ⇒ All versions of the installed styles are listed in “Installed Visualization Styles”.

3. Click on the “Install” button.
   ⇒ The “Select Visualization style(s)” dialog box opens.

4. Select a style file and click “Open” to close the dialog box.
   ⇒ The style is installed in the “System” repository. It appears now in the tree view below “Installed Visualization Styles”.

Uninstalling styles

1. Click “Tools ➔ Visualization Style Repository”.
   ⇒ The “Visualization Styles” dialog box opens.

2. Select a repository in the drop-down list of “Storage location”.
   ⇒ All versions of the installed styles are listed in “Installed Visualization Styles”.

3. Select a style there.

4. Click the “Uninstall” button.
   ⇒ The “Select Visualization Style(s)” dialog box opens.

Managing repositories

1. Click “Tools ➔ Visualization Style Repository”.
   ⇒ The “Visualization Styles” dialog box opens.

2. Click on the “Edit Locations” button.
   ⇒ The dialog makes it possible to manage other repositories.

1.3.5.18 Reference, programming

<table>
<thead>
<tr>
<th>Section</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.3.5.18.1</td>
<td>Visualization elements</td>
</tr>
<tr>
<td>1.3.5.18.2</td>
<td>Placeholders with format definition in the output text</td>
</tr>
<tr>
<td>1.3.5.18.3</td>
<td>Methods of the dialog manager</td>
</tr>
<tr>
<td>1.3.5.18.4</td>
<td>Attribute ‘VAR_IN_OUT_AS_POINTER’</td>
</tr>
<tr>
<td>1.3.5.18.5</td>
<td>Attribute ‘parameterstringof’</td>
</tr>
</tbody>
</table>
## Visualization elements

1. **1.3.5.18.1.1** Visualization element 'Rectangle', 'Round rectangle', 'Ellipse'...
2. **1.3.5.18.1.2** Visualization element 'Line'
3. **1.3.5.18.1.3** Visualization element 'Polygon', 'Polyline', 'Bézier curve'
4. **1.3.5.18.1.4** Visualization element 'Pie'
5. **1.3.5.18.1.5** Visualization element 'Image'
6. **1.3.5.18.1.6** Visualization element 'Frame'
7. **1.3.5.18.1.7** Visualization element 'Label'
8. **1.3.5.18.1.8** Visualization element 'Combo box integer'
9. **1.3.5.18.1.9** Visualization element 'Combo box array'
10. **1.3.5.18.1.10** Visualization element 'Tab control'
11. **1.3.5.18.1.11** Visualization element 'Button'
12. **1.3.5.18.1.12** Visualization element 'Group box'
13. **1.3.5.18.1.13** Visualization element 'Table'
14. **1.3.5.18.1.14** Visualization element 'Text field'
15. **1.3.5.18.1.15** Visualization element 'Scrollbar'
16. **1.3.5.18.1.16** Visualization element 'Slider'
17. **1.3.5.18.1.17** Visualization element 'SpinControl'
18. **1.3.5.18.1.18** Visualization element 'Invisible input'
19. **1.3.5.18.1.19** Visualization element 'Progress bar'
20. **1.3.5.18.1.20** Visualization element 'Checkbox'
21. **1.3.5.18.1.21** Visualization element 'Radio button'
22. **1.3.5.18.1.22** Visualization element 'Alarm table'
23. **1.3.5.18.1.23** Visualization element 'Alarm banner'
24. **1.3.5.18.1.24** Visualization element 'Bar display'
25. **1.3.5.18.1.25** Visualization element 'Meter 90°'
26. **1.3.5.18.1.26** Visualization element 'Meter 180°'
27. **1.3.5.18.1.27** Visualization element 'Meter'
28. **1.3.5.18.1.28** Visualization element 'Potentiometer'
29. **1.3.5.18.1.29** Visualization element 'Histogram'
30. **1.3.5.18.1.30** Visualization element 'Image switcher'
31. **1.3.5.18.1.31** Visualization element 'Lamp'
32. **1.3.5.18.1.32** Visualization element 'Dip switch', 'Power switch', 'Push switch', 'Push switch LED', 'Rocker switch'
33. **1.3.5.18.1.33** Visualization element 'Rotary switch'
34. **1.3.5.18.1.34** Visualization element 'Trace'
35. **1.3.5.18.1.35** Visualization element 'Trend'
36. **1.3.5.18.1.36** Visualization element 'Legend'
37. **1.3.5.18.1.37** Visualization element 'ActiveX'
38. **1.3.5.18.1.38** Visualization element 'Webbrowser'
39. **1.3.5.18.1.39** Visualization element 'Waiting symbol cube'
40. **1.3.5.18.1.40** Visualization element 'Waiting symbol flower'
41. **1.3.5.18.1.41** Visualization element 'Text editor'
42. **1.3.5.18.1.42** Visualization element 'Path3D'
43. **1.3.5.18.1.43** Visualization element 'Control Panel'
44. **1.3.5.18.1.44** Visualization element 'Cartesian XY Chart'
45. **1.3.5.18.1.45** Visualization element 'Date range picker'
46. **1.3.5.18.1.46** Visualization element 'Time range picker'
Visualization element 'Rectangle', 'Round rectangle', 'Ellipse'

Symbol:

Tag: "Basic"

The rectangle, round rectangle, and ellipse are the same element type. They can be converted into another element type by changing the "Element type" property.

Element properties

<table>
<thead>
<tr>
<th>&quot;Element name&quot;</th>
<th>Optional</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example: Werkstueck_3</td>
<td></td>
</tr>
<tr>
<td>Hint: Assign individual names for elements so that they are found faster in the element list.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>&quot;Type of element&quot;</th>
<th>&quot;Rectangle&quot;, &quot;Round rectangle&quot;, &quot;Ellipse&quot;</th>
</tr>
</thead>
</table>

Element property 'Position' The position defines the location and size of the element in the visualization window. This is based on the Cartesian coordinate system. The origin is located at the upper left corner of the window. The positive horizontal x-axis runs to the right. The positive vertical y-axis runs downwards.

<table>
<thead>
<tr>
<th>&quot;X&quot;</th>
<th>The x-coordinate of the upper left corner of the element</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Specified in pixels</td>
</tr>
<tr>
<td></td>
<td>Example: 10</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>&quot;Y&quot;</th>
<th>The y-coordinate of the upper left corner of the element</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Specified in pixels</td>
</tr>
<tr>
<td></td>
<td>Example: 10</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>&quot;Width&quot;</th>
<th>Specified in pixels</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Example: 150</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>&quot;Height&quot;</th>
<th>Specified in pixels</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Example: 30</td>
</tr>
</tbody>
</table>

Tip: You can change the values in "X", "Y", "Width", and "Height" by dragging the corresponding symbols to another position in the editor.
**“Angle”**

Static angle of rotation (in degrees)

Example: 35

The element is displayed rotated in the editor. The point of rotation is the center of the element. A positive value rotates clockwise.

Tip: You can change the value in the editor by focusing the element to the handle. When the cursor is displayed as a rotating arrow (🪂), you can rotate the element about its center as a handle.

(1): Handle

Note: If a dynamic angle of rotation is also configured in the property “Absolute movement ➔ Internal rotation”, then the static and dynamic angles of rotation are added in runtime mode. The static angle of rotation acts as an offset.

See also
- ¶ Chapter 1.3.5.3.2 “Positioning elements, adapting size and layer” on page 1113

**Radius setting**

Visible only when “Round rectangle” is selected in the “Type of element” property.

**“Radius”**

How the corners are rounded.

“From style”

“Relative to the element size”

“Explicit”: Allows for specifying a custom value in the “Value” setting.

**“Value”**

Radius of the rounded corners (in pixels)

Example: 5

Requirement: “Explicit” is selected in the “Radius” setting.

**Element property ‘Center’**

The properties contain fixed values for the coordinates of the point of rotation. This point of rotation is shown as the 🔄 symbol. The point is used as the center for rotating and scaling.

<table>
<thead>
<tr>
<th>“X”</th>
<th>X-coordinate of the point of rotation</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Y”</td>
<td>Y-coordinate of the point of rotation</td>
</tr>
</tbody>
</table>

You can also change the values by dragging the symbols ( ري) to other positions in the editor.

**Element property ‘Colors’**
<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Normal state</td>
<td>The normal state is in effect if the variable in “Color variables ➔ Toggle color” is not defined or it has the value <strong>FALSE</strong>.</td>
</tr>
<tr>
<td>Frame color</td>
<td>Frame and fill color for the corresponding state of the variable.</td>
</tr>
<tr>
<td>Fill color</td>
<td></td>
</tr>
<tr>
<td>Transparency</td>
<td>Transparency value (0 to 255) for defining the transparency of the selected color. Example: 255: The color is opaque. 0: The color is completely transparent.</td>
</tr>
<tr>
<td>Alarm state</td>
<td>The alarm state is in effect if the variable in “Color variables ➔ Toggle color” has the value <strong>TRUE</strong>.</td>
</tr>
<tr>
<td>Use gradient color</td>
<td>✅: The element is displayed with a gradient of two colors.</td>
</tr>
<tr>
<td>Gradient setting</td>
<td>The “Gradient editor” dialog box opens.</td>
</tr>
</tbody>
</table>

See also
- ❖ Chapter 1.3.5.19.3.5 “Dialog ‘Gradient Editor’” on page 1586

**Element property ‘Appearance’**

The properties contain fixed values for setting the look of the element.

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Line width</td>
<td>Value in pixels</td>
</tr>
<tr>
<td></td>
<td>Example: 2</td>
</tr>
<tr>
<td></td>
<td>Note: The values 0 and 1 both result in a line weight of 1 pixel. If no line should be displayed, then the “Line style” property must be set to the option “Invisible”.</td>
</tr>
<tr>
<td>Fill attributes</td>
<td>The way in which the element is filled.</td>
</tr>
<tr>
<td></td>
<td>✷ “Filled”: The element is filled with the color from property “Colors ➔ Fill color”.</td>
</tr>
<tr>
<td></td>
<td>✷ “Invisible”: The fill color is invisible.</td>
</tr>
<tr>
<td>Line style</td>
<td>Type of line representation</td>
</tr>
<tr>
<td></td>
<td>✷ “Solid”</td>
</tr>
<tr>
<td></td>
<td>✷ “Dashes”</td>
</tr>
<tr>
<td></td>
<td>✷ “Dots”</td>
</tr>
<tr>
<td></td>
<td>✷ “Dash Dot”</td>
</tr>
<tr>
<td></td>
<td>✷ “Dash Dot Dot”</td>
</tr>
<tr>
<td></td>
<td>✷ “not visible”</td>
</tr>
</tbody>
</table>

You can assign variables in the “Appearance variables” property for controlling the appearance dynamically. The fixed values here are overwritten.

See also
- ❖ “Element property ‘Appearance variables’” on page 1285

**Element property ‘Texts’**

The properties contain character strings for labeling the element. The character string can also contain a placeholder with a format definition. In runtime mode, the placeholder is replaced by the current value in the specified format.

CODESYS accepts the specified texts automatically into the “GlobalTextList” text list. Therefore, these texts can be localized.
### “Text”
Character string (without single straight quotation marks) for the labeling the element. Add a line break by pressing the keyboard shortcut [Ctrl] + [Enter].

**Example:** `Accesses: %i`  
The variable that contains the current value for the placeholder is specified in the property “Text variable ➔ Text”.

### “Tooltip”
Character string (without single straight quotation marks) that is displayed as the tooltip of an element.

**Example:** `Number of valid accesses`  
The variable that contains the current value for the placeholder is specified in the property “Text variable ➔ Tooltip”.

### See also
- “Element property ‘Text variables’” on page 1229
- Chapter 1.3.5.18.2 “Placeholders with format definition in the output text” on page 1546

### Element property ‘Text properties’
The properties contain fixed values for the text properties.

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Horizontal alignment”</td>
<td>Horizontal alignment of the text within the element.</td>
</tr>
<tr>
<td>“Vertical alignment”</td>
<td>Vertical alignment of the text within the element.</td>
</tr>
<tr>
<td>“Text format”</td>
<td>Definition for displaying texts that are too long</td>
</tr>
<tr>
<td></td>
<td>- “Default”: The long text is truncated.</td>
</tr>
<tr>
<td></td>
<td>- “Line break”: The text is split into parts.</td>
</tr>
<tr>
<td></td>
<td>- “Ellipsis”: The visible text ends with “...” indicating that it is not complete.</td>
</tr>
<tr>
<td>“Font”</td>
<td>Example: “Default”</td>
</tr>
<tr>
<td></td>
<td><img src="image" alt="Font dialog box" /> The “Font” dialog box opens.</td>
</tr>
<tr>
<td></td>
<td><img src="image" alt="Drop-down list" />: Drop-down list with style fonts.</td>
</tr>
<tr>
<td>“Font color”</td>
<td>Example: “Black”</td>
</tr>
<tr>
<td></td>
<td><img src="image" alt="Font color dialog box" />: The “Color” dialog box opens.</td>
</tr>
<tr>
<td></td>
<td><img src="image" alt="Drop-down list" />: Drop-down list with style colors.</td>
</tr>
<tr>
<td>“Transparency”</td>
<td>Whole number (value range from 0 to 255). This determines the transparency of the respective color.</td>
</tr>
<tr>
<td></td>
<td>Example: 255: The color is opaque.</td>
</tr>
<tr>
<td></td>
<td>0: The color is completely transparent.</td>
</tr>
<tr>
<td></td>
<td>Please note: If the color is a style color and already has a transparency value, then this property is write-protected.</td>
</tr>
</tbody>
</table>

### Element property ‘Absolute movement’
The properties contain IEC variables for controlling the position of the element dynamically. The reference point is the upper left corner of the element. In runtime mode, the entire element is moved.

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Movement”</td>
<td></td>
</tr>
<tr>
<td>“X”</td>
<td>Variable (numeric data type). Defines the X position (in pixels).</td>
</tr>
<tr>
<td></td>
<td>Example: PLC_PRG.iPos_X.</td>
</tr>
<tr>
<td></td>
<td>Increasing this value in runtime mode moves the element to the right.</td>
</tr>
<tr>
<td>Variable</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------</td>
<td>------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>&quot;Y&quot;</td>
<td>Variable (numeric data type). Defines the Y position (in pixels).</td>
</tr>
<tr>
<td>&quot;Rotation&quot;</td>
<td>Variable (numeric data type). Defines the angle of rotation (in degrees).</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>&quot;Scaling&quot;</td>
<td>Variable (integer data type). Causes centric stretching.</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>&quot;Interior rotation&quot;</td>
<td>Variable (numeric data type). Defines the angle of rotation (in degrees).</td>
</tr>
<tr>
<td>&quot;Use REAL values&quot;</td>
<td>Note: Only available if the device supports the use of REAL coordinates.</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

You can link the variables to a unit conversion.
The properties “X”, “Y”, “Rotation”, and “Interior rotation” are supported by the “Client Animation” functionality.

See also
- "Unit conversion"

Element property 'Relative movement'

The properties contains variables for moving the element. The reference point is the position of the element ("Position" property). The shape of the element can change.

<table>
<thead>
<tr>
<th>&quot;Movement top-left&quot;</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>“X”</td>
<td>Variable (integer data type). It contains the number (in pixels) that the left edge is moved horizontally. Incrementing the value moves the element to the right. Example: PLC_PRG.iDeltaX</td>
</tr>
<tr>
<td>“Y”</td>
<td>Variable (integer data type). It contains the number (in pixels) that the top edge is moved vertically. Incrementing the value moves the element to the down. Example: PLC_PRG.iDeltaY</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>&quot;Movement bottom-right&quot;</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>“X”</td>
<td>Variable (integer data type). It contains the number (in pixels) that the right edge is moved horizontally. Incrementing the value moves the element to the right. Example: PLC_PRG.iDeltaWidth</td>
</tr>
<tr>
<td>“Y”</td>
<td>Variable (integer data type). It contains the number (in pixels) that the bottom edge is moved vertically. Incrementing the value moves the element to the down. Example: PLC_PRG.iDeltaHeight</td>
</tr>
</tbody>
</table>

See also
- "Element property 'Absolute movement'" on page 1227

Element property 'Text variables'

These properties are variables with contents that replace a format definition.

<table>
<thead>
<tr>
<th>“Text variable&quot;</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Variable (data type compliant with the format definition). It contains what is printed instead of the format definition. Example: PLC_PRG.iAccesses</td>
<td></td>
</tr>
<tr>
<td>Note: The format definition is part of the text in the property “Texts ➔ Text”. Note: If you specify a variable of type enumeration with text list support, then the name of the enumeration data type is added automatically in angle brackets after the variable name. Example: PLC_PRG.enVar &lt;enumeration name&gt;. Then the symbolic value of the enumeration component is printed instead of the numeric value when text is printed. Refer to the help page for the enumerations.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>“Tooltip variable”</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Variable (data type compliant with the format definition). It contains what is printed instead of the format definition. Example: PLC_PRG.iAccessesInTooltip</td>
<td></td>
</tr>
<tr>
<td>Note: The format definition is part of the text in the property “Texts ➔ Tooltip&quot;</td>
<td></td>
</tr>
</tbody>
</table>
Element property 'Dynamic texts'

Dynamic texts are variably indexed texts of a text list. At runtime, the text is displayed that is currently indexed in the variable.

<table>
<thead>
<tr>
<th>“Text list”</th>
<th>Variable (string) or name of the text list as a fixed string in single straight quotation marks.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Example: 'Errorlist' * Drop-down list with the dialogs available in the text lists.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>“Text index”</th>
<th>Text list ID. This refers to the desired output text.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>• As fixed string with the ID in single straight quotation marks.</td>
</tr>
<tr>
<td></td>
<td>Example: '1'</td>
</tr>
<tr>
<td></td>
<td>• As a variable (STRING) for dynamically controlling the text output.</td>
</tr>
<tr>
<td></td>
<td>Example: strTextID * Sample assignment: PLC_PRG.strTextID := '1';</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>“Tooltip index”</th>
<th>Text list ID. This refers to the desired output text.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>• As fixed string with the ID in single straight quotation marks.</td>
</tr>
<tr>
<td></td>
<td>Example: '2'</td>
</tr>
<tr>
<td></td>
<td>• As a variable (STRING) for dynamically controlling the text output.</td>
</tr>
<tr>
<td></td>
<td>Example: strToolTipID * Sample assignment: PLC_PRG.strToolTipID := '2';</td>
</tr>
</tbody>
</table>

See also

- Text list

Element property 'Font variables'

The variables allow for dynamic control of the text display.

See also

- Chapter 1.3.5.18.2 “Placeholders with format definition in the output text” on page 1546
- “Element property ‘Texts’” on page 1226
- Enumerations
### “Font name”
Variable (STRING). Includes the font of the text.
Example: PLC_PRG.stFontVar := 'Arial';
The selection of fonts corresponds to the default “Font” dialog.

### “Size”
Variable (numeric data type). Contains the font size (in pixels or points). The applied unit is specified in brackets after the variable name.
- <pt>: Points (default)
  Example: PLC_PRG.iFontHeight <pt>
  Code: iFontHeight : INT := 12;
- <px>: Pixels
  Example: PLC_PRG.iFontHeight <px>
  Code: iFontHeight : INT := 19;

If you click in the value field, a drop-down list opens on the right for setting the unit.

Hint: The font size is specified in points (example: Arial 12). Use points when the variable font size should match a font, for example if a font is set in the property “Text property ⇒ Font”.

### “Flags”
Variable (DWORD). Contains the flags for displaying fonts.
Flags:
- 1: Italics
- 2: Bold
- 4: Underline
- 8: Strikethrough

Note: You can combine the font displays by adding the coding of the flags. For example, a bold and underlined text: PLC_PRG.dwFontType := 6;

### “Character set”
Variable (DWORD). Contains a character set number for the font.
The selection of character set numbers corresponds to the “Script” setting of the standard “Font” dialog.

### “Color”
Variable (DWORD). Includes the color of the text.
Example: PLC_PRG.dwColorFont := 16#FF000000;

### “Flags for text alignment”
Variable (integer data type). Contains the coding for text alignment.
Example: PLC_PRG.dwTextAlignment.
Coding:
- 0: Top left
- 1: Horizontal center
- 2: Right
- 4: Vertical center
- 8: Bottom

Note: You can combine the text alignments by adding the coding of the flags. For example, a vertical and horizontal centered text: PLC_PRG.dwFontType := 5;
Fixed values for displaying texts are set in “Text properties”.

See also
- "Element property 'Text properties'" on page 1227

Element property 'Colorvariables'

The Element property is used as an interface for project variables to dynamically control colors at runtime.

<table>
<thead>
<tr>
<th>“Toggle color”</th>
<th>The property controls the toggled color at runtime.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value assignment:</td>
<td></td>
</tr>
<tr>
<td>● FALSE:</td>
<td>The element is displayed with the color specified in the “Color” property.</td>
</tr>
<tr>
<td>● TRUE:</td>
<td>The element is displayed with the color specified in the “Alarm color” property.</td>
</tr>
<tr>
<td>Assignment options:</td>
<td></td>
</tr>
<tr>
<td>● Placeholder for the user input variable</td>
<td>&lt;toggle/tap variable&gt;”</td>
</tr>
<tr>
<td></td>
<td>&lt;NOT toggle/tap variable&gt;”</td>
</tr>
<tr>
<td>Note: Specify a variable for the mouse events “Tap” or “Toggle” in the input configuration of the element. Only then is the placeholder set. If you configure a variable in both “Toggle” and “Tap”, then the variable specified in “Tap” is used.</td>
<td></td>
</tr>
<tr>
<td>Hint: Click the symbol to insert the placeholder “&lt;toggle/tap variable&gt;”. When you activate the “Inputconfiguration”, “Tap FALSE” property, then the “&lt;NOT toggle/tap variable&gt;” placeholder is displayed.</td>
<td></td>
</tr>
<tr>
<td>● Instance path of a project variable (BOOL)</td>
<td>Example: PLC_PRG.xColorIsToggled</td>
</tr>
<tr>
<td>Note: In the code, declare and implement the variable specified here. Its value assignment determines when the color changes.</td>
<td></td>
</tr>
</tbody>
</table>

| “Normal state”       | The properties listed below control the color depending on the state. The normal state is in effect if the variable in “Color variables”, “Toggle color” is not defined or it has the value FALSE. The alarm state is in effect if the variable in “Colorvariables”, “Toggle color” has the value TRUE. |
| “Alarm state”        |                                                      |

<table>
<thead>
<tr>
<th>“Frame color”</th>
<th>Assignment options:</th>
</tr>
</thead>
<tbody>
<tr>
<td>● Variable (DWORD)</td>
<td>for the frame color</td>
</tr>
<tr>
<td>Example: PLC_PRG.dwBorderColor</td>
<td></td>
</tr>
<tr>
<td>● Color literal</td>
<td>Example of green and opaque: 16#FF00FF00</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>“Filling color”</th>
<th>Assignment options:</th>
</tr>
</thead>
<tbody>
<tr>
<td>● Variable (DWORD)</td>
<td>for the fill color</td>
</tr>
<tr>
<td>Example: PLC_PRG.dwFillColor</td>
<td></td>
</tr>
<tr>
<td>● Color literal</td>
<td>Example of gray and opaque: 16#FF888888</td>
</tr>
</tbody>
</table>
The transparency part of the color value is evaluated only if the “Activate semi-transparent drawing” option of the visualization manager is selected.

Select the “Advanced” option in the toolbar of the properties view. Then all element properties are visible.

See also
- Chapter 1.3.5.8.3 “Animating a color display” on page 1152

Element property 'Appearance variables'

| “Line width” | Variable (integer data type). Contains the line weight (in pixels). |
| “Fill attributes” | Variable (DWORD). Controls whether the fill color of the element is visible. |
| - Variable value = 0: Filled |
| - Variable value > 0: Invisible; no fill color |
| “Line style” | Variable (DWORD). Controls the line style. |
| - Coding: |
| - 0: Solid line |
| - 1: Dashed line |
| - 2: Dotted line |
| - 3: Line type "Dash Dot" |
| - 3: Line type "Dash Dot Dot" |
| - 8: Invisible; no line |

Fixed values can be set in the “Appearance” property. These values can be overwritten by dynamic variables at runtime.

See also
- “Element property ‘Appearance’” on page 1238

Element property 'State variables'

| “Invisible” | Variable (BOOL). Toggles the visibility of the element. |
| - TRUE: The element is not visible at runtime. |
| “Deactivate inputs” | Variable (BOOL). Toggles the operability of the element. |
| - TRUE: User inputs do not have any effect in runtime more. The element is shown as deactivated. |
The “Invisible” property is supported by the “Client Animation” functionality.

See also

These properties are available only when you select the “Preview: Support client animations and overlay of native elements” option in the Visualization Manager.

You can animate a movement, a rotation, and the visibility of a visualization element.

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Animation duration”</td>
<td>Time that the element executes an animation (in milliseconds)</td>
</tr>
<tr>
<td>Example: 500</td>
<td></td>
</tr>
<tr>
<td>“Move to the foreground”</td>
<td>Property value (BOOL)</td>
</tr>
<tr>
<td>TRUE: At runtime, the element is displayed in the foreground.</td>
<td></td>
</tr>
<tr>
<td>FALSE: At runtime, the element is displayed in the layer where it was inserted in the visualization editor.</td>
<td></td>
</tr>
</tbody>
</table>

See also

- Chapter 1.3.5.21.5 “Using client animation” on page 1962

Element property 'Input configuration'

The properties contain the configurations for the user input when using the mouse or keyboard. A user input defines an event and one or more actions that are executed when an event occurs.

The “Configure” button opens the “Input configuration” dialog box for creating or modifying a user input.

Configured user inputs are briefly listed below the events. They each include the action that is triggered and the setting in short form.

Example: “Execute ST code”: #PLC_PRG.i_x := 0;

<table>
<thead>
<tr>
<th>Event</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>“OnDialogClosed”</td>
<td>Input event: The user closes the dialog box.</td>
</tr>
<tr>
<td>“OnMouseClick”</td>
<td>Input event: The user clicks the mouse button completely. The mouse button is clicked and released.</td>
</tr>
<tr>
<td>“OnMouseDown”</td>
<td>Input event: The user clicks down on the mouse button.</td>
</tr>
<tr>
<td>“OnMouseEnter”</td>
<td>Input event: The user drags the mouse pointer to the element.</td>
</tr>
<tr>
<td>“OnMouseLeave”</td>
<td>Input event: The user drags the mouse pointer away from the element.</td>
</tr>
<tr>
<td>“OnMouseMove”</td>
<td>Input event: The user moves the mouse pointer over the element area.</td>
</tr>
<tr>
<td>“OnMouseUp”</td>
<td>Input event: The user releases the mouse button over the element area.</td>
</tr>
</tbody>
</table>
When a mouse click event occurs, the variable defined in "Variable" is described in the application. The coding depends on the options "Tap FALSE" and "Tap on enter if captured".

<table>
<thead>
<tr>
<th>&quot;Variable&quot;</th>
</tr>
</thead>
<tbody>
<tr>
<td>Variable (BOOL) that is set on mouse click.</td>
</tr>
<tr>
<td>Example: PLC_PRG.bIsTapped</td>
</tr>
<tr>
<td>TRUE: A mouse click event exists. It lasts while the user presses the mouse button over the element. It ends when the button is released.</td>
</tr>
<tr>
<td>FALSE: A mouse click event does not exist.</td>
</tr>
<tr>
<td>Requirement: The &quot;Tap FALSE&quot; option is not activated.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>&quot;Tap FALSE&quot;</th>
</tr>
</thead>
<tbody>
<tr>
<td>☑: The mouse click event leads to a complementary value in &quot;Variable&quot;.</td>
</tr>
<tr>
<td>TRUE: A mouse click event does not exist.</td>
</tr>
<tr>
<td>FALSE: While the mouse click event exists.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>&quot;Tap on enter if captured&quot;</th>
</tr>
</thead>
<tbody>
<tr>
<td>☑: During user input, it is also taken into consideration whether the mouse pointer is dragged within the element area or not while the mouse button is pressed.</td>
</tr>
<tr>
<td>TRUE: While the mouse click event exists and the mouse pointer is moved over the element area.</td>
</tr>
<tr>
<td>FALSE: A mouse click event does not exist. Or the user moves the mouse pointer outside of the element area while the mouse button is pressed.</td>
</tr>
<tr>
<td>The value is TRUE again as soon as the user moves the pointer back to the element area. The mouse is then captured.</td>
</tr>
</tbody>
</table>

With the onset of a mouse click event, the variable is set; when the mouse click event is completed, the variable is reset.

<table>
<thead>
<tr>
<th>&quot;Variable&quot;</th>
</tr>
</thead>
<tbody>
<tr>
<td>Variable (BOOL). Its value toggled when the mouse click event is ended. This is when the user releases the mouse button while the mouse pointer is over the element area.</td>
</tr>
<tr>
<td>If the user releases the mouse button while the mouse pointer is outside of the element area, then the mouse click event is not ended and the value is not toggled.</td>
</tr>
<tr>
<td>Tip: The user can cancel a started toggle input by dragging the mouse pointer out of the element area.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>&quot;Toggle on up if captured&quot;</th>
</tr>
</thead>
<tbody>
<tr>
<td>☑: The value toggles regardless of where the mouse pointer is when the mouse button is released. The mouse is then captured.</td>
</tr>
</tbody>
</table>

Keyboard shortcut on the element for triggering specific input actions.

When the keyboard shortcut event occurs, the input actions in the "Event(s)" property are triggered. In this way, it is not the input action itself that leads to this input action, but the mouse input action.

<table>
<thead>
<tr>
<th>&quot;Key&quot;</th>
</tr>
</thead>
<tbody>
<tr>
<td>Key pressed for input action.</td>
</tr>
<tr>
<td>Example: [T]</td>
</tr>
<tr>
<td>Note: The following properties appear when a key is selected.</td>
</tr>
</tbody>
</table>
**“Event(s)”**

- “None”
- “Mouse down”: Pressing the key triggers the input actions that are configured in the “OnMouseDown” property.
- “Mouse up”: Releasing the key triggers the input actions that are configured in the “OnMouseUp” property.
- “Mouse down/up”: Pressing and releasing the key triggers the input actions that are configured in the “OnMouseDown” property and the “OnMouseUp” property.

**“Shift”**  
✔: Combination with the Shift key  
Example: [Shift]+[T].

**“Control”**  
✔: Combination with the Ctrl key  
Example: [Ctrl]+[T].

**“Alt”**  
✔: Combination with the Alt key  
Example: [Alt]+[T].

---

All keyboard shortcuts and their actions that are configured in the visualization are listed in the “Keyboard configuration” tab.

---

See also

- § Chapter 1.3.5.19.2.2 “Command ‘Keyboard Configuration’” on page 1558
- § Chapter 1.3.5.19.3.6 “Dialog ‘Input Configuration’” on page 1586

---

**Element property 'Access rights'**  

Requirement: User management is set up for the visualization.

**“Access rights”**  
Opens the “Access rights” dialog. There you can edit the access privileges for the element.

Status messages:

- “Not set. Full rights.”: Access rights for all user groups : “operable”
- “Rights are set: Limited rights”: Access is restricted for at least one group.

See also

- § Chapter 1.3.5.19.3.1 “Dialog ‘Access Rights’” on page 1583

---

Visualization element 'Line'

Symbol:

Tag: “Basic”

The element draws a single line.
### Element properties

<table>
<thead>
<tr>
<th>&quot;Element name&quot;</th>
<th>Optional.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example: Separator_Header</td>
<td></td>
</tr>
<tr>
<td>Hint: Assign individual names for elements so that they are found faster in the element list.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>&quot;Type of element&quot;</th>
<th>Line</th>
</tr>
</thead>
</table>

#### Element property 'Position'

The following properties define the position and length of the element in the visualization window. These are based on the Cartesian coordinate system. The origin is located at the upper left corner of the window. The positive horizontal x-axis runs to the right. The positive vertical y-axis runs downwards.

<table>
<thead>
<tr>
<th>&quot;Dots&quot;</th>
<th>&quot;[0]&quot;: Coordinates of the starting point</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>&quot;[1]&quot;: Coordinate of the end point</td>
</tr>
<tr>
<td></td>
<td>You can also change the values by dragging the box symbols (□) to other positions in the editor.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>&quot;Angle&quot;</th>
<th>Static angle of rotation (in degrees).</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Example: 35</td>
</tr>
<tr>
<td></td>
<td>The element is displayed rotated in the editor. The point of rotation is the center of the element. A positive value rotates clockwise.</td>
</tr>
<tr>
<td></td>
<td>Tip: You can change the value in the editor by focusing the element to the handle. When the cursor is displayed as a rotating arrow (↻), you can rotate the element about its center as a handle.</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
</tr>
<tr>
<td><img src="image.png" alt="Diag_1" /></td>
<td></td>
</tr>
<tr>
<td>(1): Handle</td>
<td></td>
</tr>
<tr>
<td>Note: If a dynamic angle of rotation is also configured in the property &quot;Absolute movement ➔ Internal rotation&quot;, then the static and dynamic angles of rotation are added in runtime mode. The static angle of rotation acts as an offset.</td>
<td></td>
</tr>
</tbody>
</table>

#### See also

- 

#### Element property 'Center'

The properties contain fixed values for the coordinates of the point of rotation. This point of rotation is shown as the symbol. The point is used as the center for rotating and scaling.

<table>
<thead>
<tr>
<th>&quot;X&quot;</th>
<th>X-coordinate of the point of rotation</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Y&quot;</td>
<td>Y-coordinate of the point of rotation</td>
</tr>
</tbody>
</table>
You can also change the values by dragging the symbols (intéractor) to other positions in the editor.

### Element property 'Colors'

The properties contain fixed values for setting colors.

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Color”</td>
<td>Color of the line in normal state. Please note that the normal state is in effect if the expression in the “Color variables ➔ Toggle color” property is not defined or it has the value FALSE.</td>
</tr>
<tr>
<td>“Alarm color”</td>
<td>Color of the line in alarm state. Please note that the alarm state is in effect if the expression in the “Color variables ➔ Toggle color” property has the value TRUE.</td>
</tr>
<tr>
<td>“Transparency”</td>
<td>Value (0 to 255) for defining the transparency of the selected color. Example 255: The color is opaque. 0: The color is completely transparent.</td>
</tr>
</tbody>
</table>

See also
- 

### Element property 'Appearance'

The properties contain fixed values for setting the look of the element.

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Line width”</td>
<td>Value in pixels</td>
</tr>
<tr>
<td></td>
<td>Example: 2</td>
</tr>
<tr>
<td></td>
<td>Note: The values 0 and 1 both result in a line weight of one pixel. If no line should be displayed, then the “Line style” property must be set to the option “Invisible”.</td>
</tr>
<tr>
<td>“Line style”</td>
<td>Type of line representation</td>
</tr>
<tr>
<td></td>
<td>“Solid”</td>
</tr>
<tr>
<td></td>
<td>“Dashes”</td>
</tr>
<tr>
<td></td>
<td>“Dots”</td>
</tr>
<tr>
<td></td>
<td>“Dash Dot”</td>
</tr>
<tr>
<td></td>
<td>“Dash Dot Dot”</td>
</tr>
<tr>
<td></td>
<td>“not visible”</td>
</tr>
</tbody>
</table>

You can assign variables in the “Appearance variables” property for controlling the appearance dynamically. The fixed values are defined here.

See also
- Element property 'Appearance variables’ on page 1285

### Element property 'Texts'

The properties contains character strings for labeling the element. The character string can also contain a placeholder with a format definition. In runtime mode, the placeholder is replaced by the current value in the specified format.

CODESYS accepts the specified texts automatically into the “GlobalTextList” text list. Therefore, these texts can be localized.
| **“Text”** | Character string (without single straight quotation marks) for the labeling the element. Add a line break by pressing the keyboard shortcut [Ctrl] + [Enter].  
Example: `Accesses: %i`  
The variable that contains the current value for the placeholder is specified in the property **“Text variable”**. |
| **“Tooltip”** | Character string (without single straight quotation marks) that is displayed as the tooltip of an element.  
Example: `Number of valid accesses`  
The variable that contains the current value for the placeholder is specified in the property **“Text variable”**. |

See also
- “Element property ‘Text variables’” on page 1229
- “Chapter 1.3.5.18.2 “Placeholders with format definition in the output text” on page 1546

### Element property ‘Text properties’

The properties contain fixed values for the text properties.

| **“Horizontal alignment”** | Horizontal alignment of the text within the element. |
| **“Vertical alignment”** | Vertical alignment of the text within the element. |
| **“Text format”** | Definition for displaying texts that are too long  
- **“Default”**: The long text is truncated.  
- **“Line break”**: The text is split into parts.  
- **“Ellipsis”**: The visible text ends with "..." indicating that it is not complete. |
| **“Font”** | Example: **“Default”**  
[▼]: The “Font” dialog box opens.  
[▼]: Drop-down list with style fonts. |
| **“Font color”** | Example: **“Black”**  
[▼]: The “Color” dialog box opens.  
[▼]: Drop-down list with style colors. |
| **“Transparency”** | Whole number (value range from 0 to 255). This determines the transparency of the respective color.  
Example: **255**: The color is opaque.  
0: The color is completely transparent.  
Please note: If the color is a style color and already has a transparency value, then this property is write-protected. |

### Element property ‘Absolute movement’

The properties contain IEC variables for controlling the position of the element dynamically. The reference point is the upper left corner of the element. In runtime mode, the entire element is moved.

| **“Movement”** |  |
| **“X”** | Variable (numeric data type). Defines the X position (in pixels).  
Example: `PLC_PRG.iPos_X`.  
Increasing this value in runtime mode moves the element to the right. |
### “Y”
Variable (numeric data type). Defines the Y position (in pixels).

**Example:** PLC_PRG.iPos_Y.

Increasing this value in runtime mode moves the element downwards.

### “Rotation”
Variable (numeric data type). Defines the angle of rotation (in degrees).

**Example:** PLC_PRG.iAngle1.

The midpoint of the element rotates at the “Center” point. This rotation point is shown as the symbol.

In runtime mode, the alignment of the element remains the same with respect to the coordinate system of the visualization. Increasing the value rotates the element to the right.

### “Scaling”
Variable (integer data type). Causes centric stretching.

**Example:** PLC_PRG.iScaling.

The reference point is the “Center” property.

The value 1 shrinks the element by a factor of 0.001. The value 1000 returns the element to its original size.

### “Interior rotation”
Variable (numeric data type). Defines the angle of rotation (in degrees).

**Example:** PLC_PRG.iAngle2.

In runtime mode, the element rotates about the point of rotation specified in “Center” according to the value of the variable. In addition, the alignment of the element rotates according to the coordinate system of the visualization. Increasing the value in the code rotates clockwise.

The rotation point is shown as the symbol.

Note: If a static angle of rotation is specified in the property “Position ➔ Angle”, then the static angle of rotation is added to the variable angle of rotation (offset) when the visualization is executed.

### “Use REAL values”
Note: Only available if the device supports the use of REAL coordinates.

- The properties of the absolute movement are interpreted as REAL values. The values are not rounded.

- The option allows for the individual fine-tuning of drawing the element, for example for the visualization of a smoother rotation.

- Hint: If a horizontal or vertical line is drawn blurry on a specific visualization platform, then this can be corrected by an offset of 0.5px in the direction of the line thickness.

---

You can link the variables to a unit conversion.
The properties “X”, “Y”, “Rotation”, and “Interior rotation” are supported by the “Client Animation” functionality.

See also
- Unit conversion

Element property 'Relative movement'
The properties contains variables for moving the element. The reference point is the position of the element (“Position” property). The shape of the element can change.

<table>
<thead>
<tr>
<th>“Movement point[0]”</th>
<th>Variable (numeric data type). It contains the number (in pixels) that the starting point of the line is moved. Incrementing the X value moves the element to the right. Incrementing the Y value moves the element to the down.</th>
</tr>
</thead>
<tbody>
<tr>
<td>“X”</td>
<td></td>
</tr>
<tr>
<td>“Y”</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>“Movement point[1]”</th>
<th>Variable (numeric data type). It contains the number (in pixels) that the end point of the line is moved. Incrementing the X value moves the element to the right. Incrementing the Y value moves the element to the down.</th>
</tr>
</thead>
<tbody>
<tr>
<td>“X”</td>
<td></td>
</tr>
<tr>
<td>“Y”</td>
<td></td>
</tr>
</tbody>
</table>

See also
- “Element property ‘Absolute movement’” on page 1227

Element property 'Text variables'
These properties are variables with contents that replace a format definition.

| “Text variable” | Variable (data type compliant with the format definition). It contains what is printed instead of the format definition. Example: PLC_PRG.iAccesses
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Note: The format definition is part of the text in the property “Texts ➔ Text”. Note: If you specify a variable of type enumeration with text list support, then the name of the enumeration data type is added automatically in angle brackets after the variable name. Example: PLC_PRG.enVar &lt;enumeration name&gt;. Then the symbolic value of the enumeration component is printed instead of the numeric value when text is printed. Refer to the help page for the enumerations.</td>
</tr>
</tbody>
</table>

| “Tooltip variable” | Variable (data type compliant with the format definition). It contains what is printed instead of the format definition. Example: PLC_PRG.iAccessesInTooltip
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Note: The format definition is part of the text in the property “Texts ➔ Tooltip”.</td>
</tr>
</tbody>
</table>

See also
- “Chapter 1.3.5.18.2 “Placeholders with format definition in the output text” on page 1546
- “Element property ‘Texts’” on page 1226
- Enumerations

Element property 'Dynamic texts'
Dynamic texts are variably indexed texts of a text list. At runtime, the text is displayed that is currently indexed in the variable.
| **Text list** | Variable (string) or name of the text list as a fixed string in single straight quotation marks.  
Example: 'Errorlist'  
▼: Drop-down list with the dialogs available in the text lists. |
| **Text index** | Text list ID. This refers to the desired output text.  
● As fixed string with the ID in single straight quotation marks.  
Example: '1'  
● As a variable (STRING) for dynamically controlling the text output.  
Example: strTextID  
Sample assignment: PLC_PRG.strTextID := '1'; |
| **Tooltip index** | Text list ID. This refers to the desired output text.  
● As fixed string with the ID in single straight quotation marks.  
Example: '2'  
● As a variable (STRING) for dynamically controlling the text output.  
Example: strToolTipID  
Sample assignment: PLC_PRG.strToolTipID := '2'; |

See also  
● Text list

### Element property 'Font variables'

| **Font name** | Variable (STRING). Includes the font of the text.  
Example: PLC_PRG.stFontVar := 'Arial';  
The selection of fonts corresponds to the default “Font” dialog. |
| **Size** | Variable (numeric data type). Contains the font size (in pixels or points). The applied unit is specified in brackets after the variable name.  
● <pt>: Points (default)  
Example: PLC_PRG.iFontHeight <pt>  
Code: iFontHeight : INT := 12;  
● <px>: Pixels  
Example: PLC_PRG.iFontHeight <px>  
Code: iFontHeight : INT := 19;  
If you click in the value field, a drop-down list opens on the right for setting the unit.  
Hint: The font size is specified in points (example: Arial 12). Use points when the variable font size should match a font, for example if a font is set in the property “Text property ➔ Font”.

### Flags
Variable (DWORD). Contains the flags for displaying fonts.

#### Flags:
- 1: Italics
- 2: Bold
- 4: Underline
- 8: Strikethrough

Note: You can combine the font displays by adding the coding of the flags. For example, a bold and underlined text: `PLC_PRG.dwFontType := 6;`

### Character set
Variable (DWORD). Contains a character set number for the font.

The selection of character set numbers corresponds to the “Script” setting of the standard “Font” dialog.

### Color
Variable (DWORD). Includes the color of the text.

Example: `PLC_PRG.dwColorFont := 16#FF000000;`

### Flags for text alignment
Variable (integer data type). Contains the coding for text alignment.

#### Coding:
- 0: Top left
- 1: Horizontal center
- 2: Right
- 4: Vertical center
- 8: Bottom

Note: You can combine the text alignments by adding the coding of the flags. For example, a vertical and horizontal centered text: `PLC_PRG.dwFontType := 5;`

---

Fixed values for displaying texts are set in "Text properties".

See also
- "Element property 'Text properties'" on page 1227

**Element property 'Color variables'**
The Element property is used as an interface for project variables to dynamically control colors at runtime.
The property controls the toggled color at runtime.

Value assignment:

- **FALSE**: The element is displayed with the color specified in the “Color” property.
- **TRUE**: The element is displayed with the color specified in the “Alarm color” property.

Assigning the property:

- Placeholder for the user input variable
  - “<toggle/tap variable>”
  - “<NOT toggle/tap variable>”

  The color change is not controlled by its own variable, but by a user input variable.
  
  Note: Specify a variable for the mouse events “Tap” or “Toggle” in the input configuration of the element. Only then is the placeholder set. If you configure a variable in both “Toggle” and “Tap”, then the variable specified in “Tap” is used.

  Hint: Click the symbol to insert the placeholder “<toggle/tap variable>”. When you activate the “Input configuration”, “Tap FALSE” property, then the “<NOT toggle/tap variable>” placeholder is displayed.

- Instance path of a project variable (BOOL)
  Example: PLC_PRG.xColorIsToggled

  Note: In the code, declare and implement the variable specified here. Its value assignment determines when the color changes.

---

**“Color”**

- Variable (DWORD) for the color
  Example: PLC_PRG.dwColor

- Color literal
  Example of gray and opaque: `16#FF888888`

  Please note that the normal state is in effect if the expression in the “Colorvariables ➔ Toggle color” property is not defined or it has the value FALSE.

---

**“Alarm color”**

Color variable in the alarm state

- Variable (DWORD) for the alarm color
  Example: PLC_PRG.dwAlarmColor

- Color literal
  Example of red and opaque: `16#FFFF0000`

  Please note that the alarm state is in effect if the expression in the “Colorvariables ➔ Toggle color” property has the value TRUE.

---

The transparency part of the color value is evaluated only if the “Activate semi-transparent drawing” option of the visualization manager is selected.

Select the “Advanced” option in the toolbar of the properties view. Then all element properties are visible.

See also

- Chapter 1.3.5.8.3 “Animating a color display” on page 1152
### Element property 'State variables'

The variables control the element behavior dynamically.

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>“Invisible”</strong></td>
<td>Variable (BOOL). Toggles the visibility of the element. TRUE: The element is not visible at runtime.</td>
</tr>
<tr>
<td><strong>“Deactivate inputs”</strong></td>
<td>Variable (BOOL). Toggles the operability of the element. TRUE: User inputs do not have any effect in runtime more. The element is shown as deactivated.</td>
</tr>
</tbody>
</table>

> The “Invisible” property is supported by the “Client Animation” functionality.

See also
- 

### Element property 'Line width variable'

Dynamic definition of the weight of a line element using a variable.

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>“Integer value”</strong></td>
<td>Variable (integer data type). Defines the line weight of the element (in pixels). This overwrites the fixed value that is defined in “Appearance ➔ Line weight”. Note: The value 0 codes the same as 1 and sets the line weight to one pixel.</td>
</tr>
</tbody>
</table>

### Element property 'Line style variable'

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
</table>
| **“Integer value”** | Variable (integer data type). Defines the appearance of the line at runtime.  
  - 1: Solid  
  - 2: Dashes  
  - 3: Dots  
  - 4: Dash Dot  
  - 5: Dash Dot Dot  
  - 6: Invisible: The line is not drawn. |

These properties are available only when you select the “Preview: Support client animations and overlay of native elements” option in the Visualization Manager.

You can animate a movement, a rotation, and the visibility of a visualization element.

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
</table>
| **“Animation duration”** | Time that the element executes an animation (in milliseconds)  
  Example: 500                                                                                         |
| **“Move to the foreground”** | Property value (BOOL)  
  TRUE: At runtime, the element is displayed in the foreground.  
  FALSE: At runtime, the element is displayed in the layer where it was inserted in the visualization editor. |

See also
- [Chapter 1.3.5.21.5 “Using client animation” on page 1962](#)
The properties contain the configurations for the user input when using the mouse or keyboard. A user input defines an event and one or more actions that are executed when an event occurs.

The "Configure" button opens the "Input configuration" dialog box for creating or modifying a user input. Configured user inputs are briefly listed below the events. They each include the action that is triggered and the setting in short form.

**Example:** "Execute ST code": `PLC_PRG.i_x := 0;`

<table>
<thead>
<tr>
<th>Event</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;OnDialogClosed&quot;</td>
<td>Input event: The user closes the dialog box.</td>
</tr>
<tr>
<td>&quot;OnMouseClick&quot;</td>
<td>Input event: The user clicks the mouse button completely. The mouse button is clicked and released.</td>
</tr>
<tr>
<td>&quot;OnMouseDown&quot;</td>
<td>Input event: The user clicks down on the mouse button.</td>
</tr>
<tr>
<td>&quot;OnMouseEnter&quot;</td>
<td>Input event: The user drags the mouse pointer to the element.</td>
</tr>
<tr>
<td>&quot;OnMouseLeave&quot;</td>
<td>Input event: The user drags the mouse pointer away from the element.</td>
</tr>
<tr>
<td>&quot;OnMouseMove&quot;</td>
<td>Input event: The user moves the mouse pointer over the element area.</td>
</tr>
<tr>
<td>&quot;OnMouseUp&quot;</td>
<td>Input event: The user releases the mouse button over the element area.</td>
</tr>
</tbody>
</table>

| "Tap"                  | When a mouse click event occurs, the variable defined in "Variable" is described in the application. The coding depends on the options "Tap FALSE" and "Tap on enter if captured". |
| "Variable"             | Variable (BOOL) that is set on mouse click.                                 |
| Example: PLC_PRG.bIsTapped | TRUE: A mouse click event exists. It lasts while the user presses the mouse button over the element. It ends when the button is released. |
|                        | FALSE: A mouse click event does not exist.                                 |
| Requirement:           | The "Tap FALSE" option is not activated.                                   |

| "Tap FALSE"            | ☑: The mouse click event leads to a complementary value in "Variable".      |
|                        | TRUE: A mouse click event does not exist.                                  |
|                        | FALSE: While the mouse click event exists.                                 |

| "Tap on enter if captured" | ☑: During user input, it is also taken into consideration whether the mouse pointer is dragged within the element area or not while the mouse button is pressed. |
|                           | TRUE: While the mouse click event exists and the mouse pointer is moved over the element area. |
|                           | FALSE: A mouse click event does not exist. Or the user moves the mouse pointer outside of the element area while the mouse button is pressed. |
|                           | The value is TRUE again as soon as the user moves the pointer back to the element area. The mouse is then captured. |
### "Shift"
With the onset of a mouse click event, the variable is set; when the mouse click event is completed, the variable is reset.

### "Variable"
Variable (BOOL). Its value toggled when the mouse click event is ended. This is when the user releases the mouse button while the mouse pointer is over the element area.

If the user releases the mouse button while the mouse pointer is outside of the element area, then the mouse click event is not ended and the value is not toggled.

Tip: The user can cancel a started toggle input by dragging the mouse pointer out of the element area.

### "Toggle on up if captured"
- The value toggles regardless of where the mouse pointer is when the mouse button is released. The mouse is then captured.

### "Keyboard shortcuts"
Keyboard shortcut on the element for triggering specific input actions.

When the keyboard shortcut event occurs, the input actions in the "Event(s)" property are triggered. In this way, it is not the input action itself that leads to this input action, but the mouse input action.

#### "Key"
Key pressed for input action.

Example: [T]

Note: The following properties appear when a key is selected.

#### "Event(s)"
- "None"
- "Mouse down": Pressing the key triggers the input actions that are configured in the "OnMouseDown" property.
- "Mouse up": Releasing the key triggers the input actions that are configured in the "OnMouseUp" property.
- "Mouse down/up": Pressing and releasing the key triggers the input actions that are configured in the "OnMouseDown" property and the "OnMouseUp" property.

#### "Shift"
- Combination with the Shift key
  - Example: [Shift]+[T].

#### "Control"
- Combination with the Ctrl key
  - Example: [Ctrl]+[T].

#### "Alt"
- Combination with the Alt key
  - Example: [Alt]+[T].

All keyboard shortcuts and their actions that are configured in the visualization are listed in the "Keyboard configuration" tab.

See also
- Chapter 1.3.5.19.2.2 “Command ‘Keyboard Configuration’” on page 1558
- Chapter 1.3.5.19.3.6 “Dialog ‘Input Configuration’” on page 1586

Requirement: User management is set up for the visualization.
“Access rights” Opens the “Access rights” dialog. There you can edit the access privileges for the element.

Status messages:
- “Not set. Full rights.”: Access rights for all user groups: “operable”
- “Rights are set: Limited rights”: Access is restricted for at least one group.

See also:
- § Chapter 1.3.5.19.3.1 “Dialog ‘Access Rights’” on page 1583

Visualisation element ‘Polygon’, ‘Polyline’, ‘Bézier curve’

Symbol:

Tag: “Basic”

The polygon, polyline, and Bézier curve are the same element type. They can be converted into another type by changing the “Element type” property.

Elements can be dragged and dropped to the editor. The element is then drawn with five points: [0] to [4].

Other positions are added as follows: Move the mouse pointer over a corner point; the mouse pointer changes shape. Now if you press and hold [Ctrl] and click the left mouse button, another point is created. You can delete a point by pressing and holding [Shift]+[Ctrl] and click the selected point.

As an alternative, you can select the element in the toolbox area and in the editor click multiple times. At the same time, a connecting line is drawn from one point to the other. End by double-clicking the element or right-clicking it one time.

Element properties

<table>
<thead>
<tr>
<th>“Element name”</th>
<th>Optional.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hint: Assign individual names for elements so that they are found faster in the element list.</td>
<td></td>
</tr>
<tr>
<td>Example: Werkstueck_1</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>“Type of element”</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>“Polygon”</td>
<td></td>
</tr>
<tr>
<td>“Polyline”</td>
<td></td>
</tr>
<tr>
<td>“Bézier curve”</td>
<td></td>
</tr>
</tbody>
</table>

Element property ‘Position’

The following properties define the position of the corner points in the visualization window. These are based on the Cartesian coordinate system. The origin is located at the upper left corner of the window. The positive horizontal x-axis runs to the right. The positive vertical y-axis runs downwards.
### “Dots”

[0..n]: Coordinates of the corner points

Specified in pixels

You can also change the values by dragging the box symbols (✓) to other positions in the editor.

### “Angle”

Static angle of rotation (in degrees).

Example: 35

The element is displayed rotated in the editor. The point of rotation is the center of the element. A positive value rotates clockwise.

Tip: You can change the value in the editor by focusing the element to the handle. When the cursor is displayed as a rotating arrow (✓), you can rotate the element about its center as a handle.

(1): Handle

Note: If a dynamic angle of rotation is also configured in the property “Absolute movement ➔ Internal rotation”, then the static and dynamic angles of rotation are added in runtime mode. The static angle of rotation acts as an offset.

### Element property ‘Center’

The properties contain fixed values for the coordinates of the point of rotation. This point of rotation is shown as the symbol. The point is used as the center for rotating and scaling.

<table>
<thead>
<tr>
<th>“X”</th>
<th>X-coordinate of the point of rotation</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Y”</td>
<td>Y-coordinate of the point of rotation</td>
</tr>
</tbody>
</table>

You can also change the values by dragging the symbols (✓) to other positions in the editor.

### Element property ‘Colors’

<table>
<thead>
<tr>
<th>“Normal state”</th>
<th>The normal state is in effect if the variable in “Color variables ➔ Toggle color” is not defined or it has the value FALSE.</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Frame color”</td>
<td>Frame and fill color for the corresponding state of the variable.</td>
</tr>
<tr>
<td>“Fill color”</td>
<td></td>
</tr>
<tr>
<td>“Transparency”</td>
<td>Transparency value (0 to 255) for defining the transparency of the selected color. Example: 255: The color is opaque. 0: The color is completely transparent.</td>
</tr>
<tr>
<td>“Alarm state”</td>
<td>The alarm state is in effect if the variable in “Color variables ➔ Toggle color” has the value TRUE.</td>
</tr>
<tr>
<td>“Use gradient color”</td>
<td>[✓]: The element is displayed with a gradient of two colors.</td>
</tr>
<tr>
<td>“Gradient setting”</td>
<td>The “Gradient editor” dialog box opens.</td>
</tr>
</tbody>
</table>
Element property 'Appearance'
The properties contain fixed values for setting the look of the element.

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Line width</td>
<td>Value in pixels&lt;br&gt;Note: The values 0 and 1 both result in a line weight of 1 pixel. If no line should be displayed, then the “Line style” property must be set to the option “Invisible”.</td>
</tr>
<tr>
<td>Fill attributes</td>
<td>The way in which the element is filled.&lt;br&gt;● “Filled”: The element is filled with the color from property “Colors ➔ Fill color”.&lt;br&gt;● “Invisible”: The fill color is invisible.</td>
</tr>
<tr>
<td>Line style</td>
<td>Type of line representation&lt;br&gt;● “Solid”&lt;br&gt;● “Dashes”&lt;br&gt;● “Dots”&lt;br&gt;● “Dash Dot”&lt;br&gt;● “Dash Dot Dot”&lt;br&gt;● “not visible”</td>
</tr>
</tbody>
</table>

You can assign variables in the “Appearance variables” property for controlling the appearance dynamically. The fixed values here are overwritten.

See also
● § Chapter 1.3.5.19.3.5 “Dialog ‘Gradient Editor’” on page 1586

Element property 'Texts'
The properties contains character strings for labeling the element. The character string can also contain a placeholder with a format definition. In runtime mode, the placeholder is replaced by the current value in the specified format.

CODESYS accepts the specified texts automatically into the “GlobalTextList” text list. Therefore, these texts can be localized.

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Text</td>
<td>Character string (without single straight quotation marks) for the labeling the element. Add a line break by pressing the keyboard shortcut [Ctrl] + [Enter].&lt;br&gt;Example: Accesses: %i&lt;br&gt;The variable that contains the current value for the placeholder is specified in the property “Text variable ➔ Text”.</td>
</tr>
<tr>
<td>Tooltip</td>
<td>Character string (without single straight quotation marks) that is displayed as the tooltip of an element.&lt;br&gt;Example: Number of valid accesses.&lt;br&gt;The variable that contains the current value for the placeholder is specified in the property “Text variable ➔ Tooltip”.</td>
</tr>
</tbody>
</table>
See also
- “Element property ‘Text variables’” on page 1229
- Chapter 1.3.5.18.2 “Placeholders with format definition in the output text” on page 1546

**Element property ‘Text properties’**
The properties contain fixed values for the text properties.

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Horizontal alignment”</td>
<td>Horizontal alignment of the text within the element.</td>
</tr>
<tr>
<td>“Vertical alignment”</td>
<td>Vertical alignment of the text within the element.</td>
</tr>
<tr>
<td>“Text format”</td>
<td>Definition for displaying texts that are too long</td>
</tr>
<tr>
<td></td>
<td>- “Default”: The long text is truncated.</td>
</tr>
<tr>
<td></td>
<td>- “Line break”: The text is split into parts.</td>
</tr>
<tr>
<td></td>
<td>- “Ellipsis”: The visible text ends with ”...” indicating that it is not complete.</td>
</tr>
<tr>
<td>“Font”</td>
<td>Example: “Default”</td>
</tr>
<tr>
<td></td>
<td>- The “Font” dialog box opens.</td>
</tr>
<tr>
<td></td>
<td>- Drop-down list with style fonts.</td>
</tr>
<tr>
<td>“Font color”</td>
<td>Example: “Black”</td>
</tr>
<tr>
<td></td>
<td>- The “Color” dialog box opens.</td>
</tr>
<tr>
<td></td>
<td>- Drop-down list with style colors.</td>
</tr>
<tr>
<td>“Transparency”</td>
<td>Whole number (value range from 0 to 255). This determines the transparency of the respective color.</td>
</tr>
<tr>
<td></td>
<td>Example: 255: The color is opaque.</td>
</tr>
<tr>
<td></td>
<td>0: The color is completely transparent.</td>
</tr>
<tr>
<td></td>
<td>Please note: If the color is a style color and already has a transparency value, then this property is write-protected.</td>
</tr>
</tbody>
</table>

**Element property ‘Absolute movement’**
The properties contain IEC variables for controlling the position of the element dynamically. The reference point is the upper left corner of the element. In runtime mode, the entire element is moved.

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Movement”</td>
<td></td>
</tr>
<tr>
<td>“X”</td>
<td>Variable (numeric data type). Defines the X position (in pixels).</td>
</tr>
<tr>
<td></td>
<td>Example: PLC_PRG.iPos_X.</td>
</tr>
<tr>
<td></td>
<td>Increasing this value in runtime mode moves the element to the right.</td>
</tr>
<tr>
<td>“Y”</td>
<td>Variable (numeric data type). Defines the Y position (in pixels).</td>
</tr>
<tr>
<td></td>
<td>Example: PLC_PRG.iPos_Y.</td>
</tr>
<tr>
<td></td>
<td>Increasing this value in runtime mode moves the element downwards.</td>
</tr>
</tbody>
</table>
**“Rotation”**  
Variable (numeric data type). Defines the angle of rotation (in degrees).

Example: PLC_PRG.iAngle1.

The midpoint of the element rotates at the “Center” point. This rotation point is shown as the symbol.

In runtime mode, the alignment of the element remains the same with respect to the coordinate system of the visualization. Increasing the value rotates the element to the right.

---

**“Scaling”**  
Variable (integer data type). Causes centric stretching.

Example: PLC_PRG.iScaling.

The reference point is the “Center” property.

The value 1 shrinks the element by a factor of 0.001. The value 1000 returns the element to its original size.

---

**“Interior rotation”**  
Variable (numeric data type). Defines the angle of rotation (in degrees).

Example: PLC_PRG.iAngle2.

In runtime mode, the element rotates about the point of rotation specified in “Center” according to the value of the variable. In addition, the alignment of the element rotates according to the coordinate system of the visualization. Increasing the value in the code rotates clockwise.

The rotation point is shown as the symbol.

Note: If a static angle of rotation is specified in the property “Position → Angle”, then the static angle of rotation is added to the variable angle of rotation (offset) when the visualization is executed.

---

**“Use REAL values”**  
Note: Only available if the device supports the use of REAL coordinates.

- The properties of the absolute movement are interpreted as REAL values. The values are not rounded.

The option allows for the individual fine-tuning of drawing the element, for example for the visualization of a smoother rotation.

Hint: If a horizontal or vertical line is drawn blurry on a specific visualization platform, then this can be corrected by an offset of 0.5px in the direction of the line thickness.
The properties “X”, “Y”, “Rotation”, and “Interior rotation” are supported by the “Client Animation” functionality.

See also

- Unit conversion

### Element property ‘Dynamic points’

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
</table>
| **“Array of points”** variable (POINTER TO) | Points to an array of the structure VisuElems.VisuStructPoint. The elements iX and iY of VisuStructPoint contain the xy-coordinates of a point. The current number of array elements implicitly contains the variable in the property “Number of points”. The variable that is assigned to the property “Number of points” contains the number of array elements and therefore the number of corner points.

Example:
```plaintext
pPoints : POINTER TO ARRAY[0..100] OF VisuElems.VisuStructPoint;
```

| **“Number of points”** variable (integer data type) | Contains the number of array elements and therefore the number of corner points for displaying the element.

Example:
```plaintext
PLC_PRG.iNumberOfPoints := 24;
```

In the example, the element has 24 points. This definition is necessary because the individual points are defined by a pointer and this does not allow control over the number of points.

Note: In this way, it is possible to adapt the display of the element dynamically by updating the number of corner points.

### Element property ‘Text variables’

These properties are variables with contents that replace a format definition.

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
</table>
| **“Text variable”** variable (data type compliant with the format definition) | It contains what is printed instead of the format definition.

Example:
```plaintext
PLC_PRG.iAccesses
```

Note: The format definition is part of the text in the property “Texts ➔ Text”.

Note: If you specify a variable of type enumeration with text list support, then the name of the enumeration data type is added automatically in angle brackets after the variable name. Example: PLC_PRG.enVar <enumeration name>. Then the symbolic value of the enumeration component is printed instead of the numeric value when text is printed. Refer to the help page for the enumerations.

| **“Tooltip variable”** variable (data type compliant with the format definition) | It contains what is printed instead of the format definition.

Example:
```plaintext
PLC_PRG.iAccessesInTooltip
```

Note: The format definition is part of the text in the property “Texts ➔ Tooltip”.

See also

- “Chapter 1.3.5.18.2 “Placeholders with format definition in the output text” on page 1546
- “Element property ‘Texts’” on page 1226
- Enumerations
**Element property 'Dynamic texts'**

Dynamic texts are variably indexed texts of a text list. At runtime, the text is displayed that is currently indexed in the variable.

| “Text list” | Variable (string) or name of the text list as a fixed string in single straight quotation marks.  
| | Example: 'Errorlist'  
| | ◀: Drop-down list with the dialogs available in the text lists. |

| “Text index” | Text list ID. This refers to the desired output text.  
| | ● As fixed string with the ID in single straight quotation marks.  
| | Example: '1'  
| | ● As a variable (STRING) for dynamically controlling the text output.  
| | Example: strTextID  
| | Sample assignment: PLC_PRG.strTextID := '1'; |

| “Tooltip index” | Text list ID. This refers to the desired output text.  
| | ● As fixed string with the ID in single straight quotation marks.  
| | Example: '2'  
| | ● As a variable (STRING) for dynamically controlling the text output.  
| | Example: strToolTipID  
| | Sample assignment: PLC_PRG.strToolTipID := '2'; |

**See also**

- "Text list"

**Element property 'Font variables'**

The variables allow for dynamic control of the text display.

| “Font name” | Variable (STRING). Includes the font of the text.  
| | Example: PLC_PRG.stFontVar := 'Arial';  
| | The selection of fonts corresponds to the default “Font” dialog. |

| “Size” | Variable (numeric data type). Contains the font size (in pixels or points). The applied unit is specified in brackets after the variable name.  
| | ● <pt>: Points (default)  
| | Example: PLC_PRG.iFontHeight <pt>  
| | Code: iFontHeight : INT := 12;  
| | ● <px>: Pixels  
| | Example: PLC_PRG.iFontHeight <px>  
| | Code: iFontHeight : INT := 19; |

If you click in the value field, a drop-down list opens on the right for setting the unit.

**Hint:** The font size is specified in points (example: Arial 12). Use points when the variable font size should match a font, for example if a font is set in the property “Text property ➔ Font”.

---

[Image of a drop-down list with options 'pt' and 'px']
### "Flags"
Variable (DWORD). Contains the flags for displaying fonts.

Flags:
- 1: Italics
- 2: Bold
- 4: Underline
- 8: Strikethrough

Note: You can combine the font displays by adding the coding of the flags. For example, a bold and underlined text: `PLC_PRG.dwFontType := 6;`

### "Character set"
Variable (DWORD). Contains a character set number for the font.
The selection of character set numbers corresponds to the “Script” setting of the standard “Font” dialog.

### "Color"
Variable (DWORD). Includes the color of the text.
Example: `PLC_PRG.dwColorFont := #FF000000;`

### "Flags for text alignment"
Variable (integer data type). Contains the coding for text alignment.
Example: `PLC_PRG.dwTextAlignment;
Coding:
- 0: Top left
- 1: Horizontal center
- 2: Right
- 4: Vertical center
- 8: Bottom

Note: You can combine the text alignments by adding the coding of the flags. For example, a vertical and horizontal centered text: `PLC_PRG.dwFontType := 5;`

---

**Fixed values for displaying texts are set in "Text properties".**

See also
- § “Element property 'Text properties’” on page 1227

**Element property 'Colorvariables’**
The Element property is used as an interface for project variables to dynamically control colors at runtime.
### "Toggle color"

The property controls the toggled color at runtime.

**Value assignment:**
- **FALSE:** The element is displayed with the color specified in the "Color" property.
- **TRUE:** The element is displayed with the color specified in the "Alarm color" property.

**Assignment options:**
- Placeholder for the user input variable
  - "<toggle/tap variable>"
  - "<NOT toggle/tap variable>"

The color change is not controlled by its own variable, but by a user input variable.

Note: Specify a variable for the mouse events "Tap" or "Toggle" in the input configuration of the element. Only then is the placeholder set. If you configure a variable in both "Toggle" and "Tap", then the variable specified in "Tap" is used.

Hint: Click the symbol to insert the placeholder "<toggle/tap variable>". When you activate the "Inputconfiguration", "Tap FALSE" property, then the "<NOT toggle/tap variable>" placeholder is displayed.

- **Instance path of a project variable** (BOOL)
  
  Example: PLC_PRG.xColorIsToggled

  Note: In the code, declare and implement the variable specified here. Its value assignment determines when the color changes.

---

### "Normal state"

### "Alarm state"

The properties listed below control the color depending on the state. The normal state is in effect if the variable in "Color variables", "Toggle color" is not defined or it has the value FALSE. The alarm state is in effect if the variable in "Colorvariables", "Toggle color" has the value TRUE.

---

### "Frame color"

**Assignment options:**
- **Variable (DWORD) for the frame color**
  
  Example: PLC_PRG.dwBorderColor

- **Color literal**
  
  Example of green and opaque: 16#FF00FF00

---

### "Filling color"

**Assignment options:**
- **Variable (DWORD) for the fill color**
  
  Example: PLC_PRG.dwFillColor

- **Color literal**
  
  Example of gray and opaque: 16#FF888888

---

The transparency part of the color value is evaluated only if the "Activate semi-transparent drawing" option of the visualization manager is selected.

Select the "Advanced" option in the toolbar of the properties view. Then all element properties are visible.

See also
- Chapter 1.3.5.8.3 “Animating a color display” on page 1152
The properties contain IEC variables for controlling the appearance of the element dynamically.

**Element property 'Appearance variables'**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Line width&quot;</td>
<td>Variable (integer data type). Contains the line weight (in pixels).</td>
</tr>
<tr>
<td>&quot;Fill attributes&quot;</td>
<td>Variable (DWORD). Controls whether the fill color of the element is visible.</td>
</tr>
<tr>
<td>Variable value = 0: Filled</td>
<td></td>
</tr>
<tr>
<td>Variable value &gt; 0: Invisible; no fill color</td>
<td></td>
</tr>
<tr>
<td>&quot;Line style&quot;</td>
<td>Variable (DWORD). Controls the line style.</td>
</tr>
<tr>
<td>Coding:</td>
<td></td>
</tr>
<tr>
<td>0: Solid line</td>
<td></td>
</tr>
<tr>
<td>1: Dashed line</td>
<td></td>
</tr>
<tr>
<td>2: Dotted line</td>
<td></td>
</tr>
<tr>
<td>3: Line type &quot;Dash Dot&quot;</td>
<td></td>
</tr>
<tr>
<td>3: Line type &quot;Dash Dot Dot&quot;</td>
<td></td>
</tr>
<tr>
<td>8: Invisible; no line</td>
<td></td>
</tr>
</tbody>
</table>

Fixed values can be set in the "Appearance" property. These values can be overwritten by dynamic variables at runtime.

See also

- § “Element property ‘Appearance’” on page 1238

**Element property 'State variables'**

The variables control the element behavior dynamically.

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Invisible&quot;</td>
<td>Variable (BOOL). Toggles the visibility of the element.</td>
</tr>
<tr>
<td>TRUE: The element is not visible at runtime.</td>
<td></td>
</tr>
<tr>
<td>&quot;Deactivate inputs&quot;</td>
<td>Variable (BOOL). Toggles the operability of the element.</td>
</tr>
<tr>
<td>TRUE: User inputs do not have any effect in runtime more. The element is shown as deactivated.</td>
<td></td>
</tr>
</tbody>
</table>

The “Invisible” property is supported by the "Client Animation" functionality.

See also

- 

These properties are available only when you select the “Preview: Support client animations and overlay of native elements” option in the Visualization Manager.

You can animate a movement, a rotation, and the visibility of a visualization element.
<table>
<thead>
<tr>
<th><strong>Animation duration</strong></th>
<th>Time that the element executes an animation (in milliseconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Example:</strong></td>
<td>500</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Move to the foreground</strong></th>
<th>Property value (BOOL)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>TRUE:</strong></td>
<td>At runtime, the element is displayed in the foreground.</td>
</tr>
<tr>
<td><strong>FALSE:</strong></td>
<td>At runtime, the element is displayed in the layer where it was inserted in the visualization editor.</td>
</tr>
</tbody>
</table>

**See also**
- Chapter 1.3.5.21.5 “Using client animation” on page 1962

**Element property 'Input configuration'**
The properties contain the configurations for the user input when using the mouse or keyboard. A user input defines an event and one or more actions that are executed when an event occurs.

The “Configure” button opens the “Input configuration” dialog box for creating or modifying a user input. Configured user inputs are briefly listed below the events. They each include the action that is triggered and the setting in short form.

**Example:** "Execute ST code": # PLC_PRG.i_x := 0;

<table>
<thead>
<tr>
<th><strong>OnDialogClosed</strong></th>
<th>Input event: The user closes the dialog box.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>OnMouseClick</strong></td>
<td>Input event: The user clicks the mouse button completely. The mouse button is clicked and released.</td>
</tr>
<tr>
<td><strong>OnMouseDown</strong></td>
<td>Input event: The user clicks down on the mouse button.</td>
</tr>
<tr>
<td><strong>OnMouseEnter</strong></td>
<td>Input event: The user drags the mouse pointer to the element.</td>
</tr>
<tr>
<td><strong>OnMouseLeave</strong></td>
<td>Input event: The user drags the mouse pointer away from the element.</td>
</tr>
<tr>
<td><strong>OnMouseMove</strong></td>
<td>Input event: The user moves the mouse pointer over the element area.</td>
</tr>
<tr>
<td><strong>OnMouseUp</strong></td>
<td>Input event: The user releases the mouse button over the element area.</td>
</tr>
</tbody>
</table>

**Tap**
When a mouse click event occurs, the variable defined in “Variable” is described in the application. The coding depends on the options “Tap FALSE” and “Tap on enter if captured”.

<table>
<thead>
<tr>
<th><strong>Variable</strong></th>
<th>Variable (BOOL) that is set on mouse click.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Example:</strong></td>
<td>PLC_PRG.bIsTapped</td>
</tr>
<tr>
<td><strong>TRUE:</strong></td>
<td>A mouse click event exists. It lasts while the user presses the mouse button over the element. It ends when the button is released.</td>
</tr>
<tr>
<td><strong>FALSE:</strong></td>
<td>A mouse click event does not exist.</td>
</tr>
</tbody>
</table>

Requirement: The “Tap FALSE” option is not activated.
### "Tap FALSE"

- **True**: The mouse click event leads to a complementary value in "Variable".
- **False**: A mouse click event does not exist.
- **False**: While the mouse click event exists.

### "Tap on enter if captured"

- **True**: During user input, it is also taken into consideration whether the mouse pointer is dragged within the element area or not while the mouse button is pressed.
- **False**: While the mouse click event exists and the mouse pointer is moved over the element area.
- **False**: A mouse click event does not exist. Or the user moves the mouse pointer outside of the element area while the mouse button is pressed.

The value is **True** again as soon as the user moves the pointer back to the element area. The mouse is then captured.

### "Shift"

With the onset of a mouse click event, the variable is set; when the mouse click event is completed, the variable is reset.

### "Variable"

Variable (BOOL). Its value toggled when the mouse click event is ended. This is when the user releases the mouse button while the mouse pointer is over the element area.

If the user releases the mouse button while the mouse pointer is outside of the element area, then the mouse click event is not ended and the value is not toggled.

Tip: The user can cancel a started toggle input by dragging the mouse pointer out of the element area.

### "Toggle on up if captured"

- **True**: The value toggles regardless of where the mouse pointer is when the mouse button is released. The mouse is then captured.

### "Keyboard shortcuts"

Keyboard shortcut on the element for triggering specific input actions.

When the keyboard shortcut event occurs, the input actions in the "Event(s)" property are triggered. In this way, it is not the input action itself that leads to this input action, but the mouse input action.

### "Key"

Key pressed for input action.

Example: [T]

Note: The following properties appear when a key is selected.

### "Event(s)"

- "None"
- "Mouse down": Pressing the key triggers the input actions that are configured in the "OnMouseDown" property.
- "Mouse up": Releasing the key triggers the input actions that are configured in the "OnMouseUp" property.
- "Mouse down/up": Pressing and releasing the key triggers the input actions that are configured in the "OnMouseDown" property and the "OnMouseUp" property.

### "Shift"

- **True**: Combination with the Shift key

Example: [Shift]+[T].

### "Control"

- **True**: Combination with the Ctrl key

Example: [Ctrl]+[T].

### "Alt"

- **True**: Combination with the Alt key

Example: [Alt]+[T].
All keyboard shortcuts and their actions that are configured in the visualization are listed in the “Keyboard configuration” tab.

See also
- % Chapter 1.3.5.19.2.2 “Command 'Keyboard Configuration’” on page 1558
- % Chapter 1.3.5.19.3.6 “Dialog 'Input Configuration’” on page 1586

Element property 'Access rights'

| "Access rights" | Opens the “Access rights” dialog. There you can edit the access privileges for the element. Status messages:  
| "Not set. Full rights.": Access rights for all user groups: "operable"  
| "Rights are set: Limited rights": Access is restricted for at least one group. |

See also
- % Chapter 1.3.5.19.3.1 “Dialog 'Access Rights’” on page 1583

See also
- 

Visualization element 'Pie'

Symbol:

Tag: “Basic”
The element draws a pie of any angle.

Element properties

| "Element name" | Example: Error_rate_part_1 |
| "Type of element" | Pie |

The position defines the location and size of the element in the visualization window. This is based on the Cartesian coordinate system. The origin is located at the upper left corner of the window. The positive horizontal x-axis runs to the right. The positive vertical y-axis runs downwards.
<table>
<thead>
<tr>
<th><strong>“X”</strong></th>
<th>The x-coordinate of the upper left corner of the element</th>
<th>Specified in pixels</th>
<th>Example: 10</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>“Y”</strong></td>
<td>The y-coordinate of the upper left corner of the element</td>
<td>Specified in pixels</td>
<td>Example: 10</td>
</tr>
<tr>
<td><strong>“Width”</strong></td>
<td>Specified in pixels</td>
<td>Example: 150</td>
<td></td>
</tr>
<tr>
<td><strong>“Height”</strong></td>
<td>Specified in pixels</td>
<td>Example: 30</td>
<td></td>
</tr>
</tbody>
</table>

Tip: You can change the values in “X”, “Y”, “Width”, and “Height” by dragging the corresponding symbols to another position in the editor.

<table>
<thead>
<tr>
<th><strong>“Angle”</strong></th>
<th>Static angle of rotation (in degrees).</th>
<th>Example: 35</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>The element is displayed rotated in the editor. The point of rotation is the center of the element. A positive value rotates clockwise.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Tip: You can change the value in the editor by focusing the element to the handle. When the cursor is displayed as a rotating arrow, you can rotate the element about its center as a handle.</td>
<td></td>
</tr>
</tbody>
</table>

(1): Handle

Note: If a dynamic angle of rotation is also configured in the property “Absolute movement ➔ Internal rotation”, then the static and dynamic angles of rotation are added in runtime mode. The static angle of rotation acts as an offset.

See also

- Chapter 1.3.5.3.2 “Positioning elements, adapting size and layer” on page 1113
### Property "Begin"

Start angle of the pie. If you also define a variable for the start, then the start angle is calculated from the sum of the values for "Begin" and "Variable for begin".

**Example:**
- "Begin": 330
- "End": 90

### Property "End"

End angle of the pie. If you also define a variable for the end, then the end angle is calculated from the sum of the values for "End" and "Variable for end".

The pie is drawn clockwise from the start angle to the end angle.

### Property "Variable for begin"

The start of the sector is defined dynamically by a variable.

### Property "Variable for end"

The end of the sector is defined dynamically by a variable.

### Property "Only show circle line"

The pie is drawn without the radius line or filling color.

### Element property 'Center'

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;X&quot;</td>
<td>Display of the center coordinates. You cannot modify these values here in the properties.</td>
</tr>
<tr>
<td>&quot;Y&quot;</td>
<td>If the pie is selected in the editor, then the center of the pie (as well as the center of the enveloping box) is visualized with the symbol 📑. Moreover, the element is decorated with a position, begin, and end boxes that you can move.</td>
</tr>
</tbody>
</table>

The center coordinates change when you move the center symbol 📑 in the editor. This also changes the size of the pie so that the position box 🖍 retains its position and the center remains in the middle of the element.

### Element property 'Colors'

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Normal state&quot;</td>
<td>The normal state is in effect if the variable in &quot;Color variables ➔ Toggle color&quot; is not defined or it has the value FALSE.</td>
</tr>
<tr>
<td>&quot;Frame color&quot;</td>
<td>Frame and fill color for the corresponding state of the variable.</td>
</tr>
<tr>
<td>&quot;Fill color&quot;</td>
<td></td>
</tr>
<tr>
<td>&quot;Transparency&quot;</td>
<td>Transparency value (0 to 255) for defining the transparency of the selected color. Example: 255: The color is opaque. 0: The color is completely transparent.</td>
</tr>
</tbody>
</table>
### “Alarm state”
The alarm state is in effect if the variable in “Color variables → Toggle color” has the value TRUE.

### “Use gradient color”
- The element is displayed with a gradient of two colors.

### “Gradient setting”
The “Gradient editor” dialog box opens.

See also
- ☀ Chapter 1.3.5.19.3.5 “Dialog ‘Gradient Editor’” on page 1586

### Element property ‘Appearance’
The properties contain fixed values for setting the look of the element.

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
</table>
| **“Line width”** | Value in pixels
| Example: 2       | Note: The values 0 and 1 both result in a line weight of 1 pixel. If no line should be displayed, then the “Line style” property must be set to the option "Invisible". |
| **“Fill attributes”** | The way in which the element is filled.
| “Filled”         | The element is filled with the color from property “Colors → Fill color”.
| “Invisible”      | The fill color is invisible. |
| **“Line style”** | Type of line representation
| “Solid”          | |
| “Dashes”         | |
| “Dots”           | |
| “Dash Dot”       | |
| “Dash Dot Dot”   | |
| “not visible”    | |

You can assign variables in the “Appearance variables” property for controlling the appearance dynamically. The fixed values here are overwritten.

See also
- ☀ “Element property ‘Appearance variables’” on page 1285

### Element property ‘Texts’
The properties contains character strings for labeling the element. The character string can also contain a placeholder with a format definition. In runtime mode, the placeholder is replaced by the current value in the specified format.

CODESYS accepts the specified texts automatically into the “GlobalTextList” text list. Therefore, these texts can be localized.
**“Text”**
Character string (without single straight quotation marks) for the labeling the element. Add a line break by pressing the keyboard shortcut [Ctrl] + [Enter].

Example: Accesses: %i

The variable that contains the current value for the placeholder is specified in the property “Text variable ➔ Text”.

**“Tooltip”**
Character string (without single straight quotation marks) that is displayed as the tooltip of an element.

Example: Number of valid accesses.

The variable that contains the current value for the placeholder is specified in the property “Text variable ➔ Tooltip”.

See also
- ☰ “Element property 'Text variables“ on page 1229
- ☰ Chapter 1.3.5.18.2 “Placeholders with format definition in the output text” on page 1546

The properties contain fixed values for the text properties.

<table>
<thead>
<tr>
<th>“Horizontal alignment”</th>
<th>Horizontal alignment of the text within the element.</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Vertical alignment”</td>
<td>Vertical alignment of the text within the element.</td>
</tr>
<tr>
<td>“Text format”</td>
<td>Definition for displaying texts that are too long</td>
</tr>
<tr>
<td></td>
<td>• “Default”: The long text is truncated.</td>
</tr>
<tr>
<td></td>
<td>• “Line break”: The text is split into parts.</td>
</tr>
<tr>
<td></td>
<td>• “Ellipsis”: The visible text ends with “…” indicating that it is not complete.</td>
</tr>
<tr>
<td>“Font”</td>
<td>Example: “Default”</td>
</tr>
<tr>
<td></td>
<td>▶: The “Font” dialog box opens.</td>
</tr>
<tr>
<td></td>
<td>▼: Drop-down list with style fonts.</td>
</tr>
<tr>
<td>“Font color”</td>
<td>Example: “Black”</td>
</tr>
<tr>
<td></td>
<td>▶: The “Color” dialog box opens.</td>
</tr>
<tr>
<td></td>
<td>▼: Drop-down list with style colors.</td>
</tr>
<tr>
<td>“Transparency”</td>
<td>Whole number (value range from 0 to 255). This determines the transparency of the respective color.</td>
</tr>
<tr>
<td></td>
<td>Example: 255: The color is opaque.</td>
</tr>
<tr>
<td></td>
<td>0: The color is completely transparent.</td>
</tr>
<tr>
<td></td>
<td>Please note: If the color is a style color and already has a transparency value, then this property is write-protected.</td>
</tr>
</tbody>
</table>

The properties contain IEC variables for controlling the position of the element dynamically. The reference point is the upper left corner of the element. In runtime mode, the entire element is moved.

<table>
<thead>
<tr>
<th>“Movement”</th>
</tr>
</thead>
<tbody>
<tr>
<td>“X”</td>
</tr>
<tr>
<td>Variable (integer data type). Defines the X position (in pixels).</td>
</tr>
<tr>
<td>Example: PLC_PRG.iPos_X.</td>
</tr>
<tr>
<td>Increasing this value in runtime mode moves the element to the right.</td>
</tr>
<tr>
<td>Property</td>
</tr>
<tr>
<td>--------------</td>
</tr>
<tr>
<td>“Y”</td>
</tr>
<tr>
<td>“Scaling”</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>“Interior rotation”</td>
</tr>
</tbody>
</table>

---

You can link the variables to a unit conversion.

The “X”, “Y”, and “Interior rotation” properties are supported by the "Client Animation" functionality.

See also
- **Unit conversion**

Element property 'Text variables'

These properties are variables with contents that replace a format definition.
### "Text variable"

Variable (data type compliant with the format definition). It contains what is printed instead of the format definition.

**Example:** PLC_PRG.iAccesses

Note: The format definition is part of the text in the property "Texts ➤ Text".

Note: If you specify a variable of type enumeration with text list support, then the name of the enumeration data type is added automatically in angle brackets after the variable name. Example: PLC_PRG.enVar <enumeration name>. Then the symbolic value of the enumeration component is printed instead of the numeric value when text is printed. Refer to the help page for the enumerations.

### "Tooltip variable"

Variable (data type compliant with the format definition). It contains what is printed instead of the format definition.

**Example:** PLC_PRG.iAccessesInTooltip

Note: The format definition is part of the text in the property "Texts ➤ Tooltip".

See also

- ¶ Chapter 1.3.5.18.2 “Placeholders with format definition in the output text” on page 1546
- ¶ “Element property 'Texts’” on page 1226
- ¶ Enumerations

#### Element property 'Dynamic texts'

Dynamic texts are variably indexed texts of a text list. At runtime, the text is displayed that is currently indexed in the variable.

<table>
<thead>
<tr>
<th>&quot;Text list&quot;</th>
<th>Variable (string) or name of the text list as a fixed string in single straight quotation marks.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Example:</strong></td>
<td>'Errorlist'</td>
</tr>
<tr>
<td>‣: Drop-down list with the dialogs available in the text lists.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>&quot;Text index&quot;</th>
<th>Text list ID. This refers to the desired output text.</th>
</tr>
</thead>
</table>
| ‣ As fixed string with the ID in single straight quotation marks. | Example: '1'
| ‣ As a variable (STRING) for dynamically controlling the text output. | Example: strTextID
| Sample assignment: PLC_PRG.strTextID := '1'; |

<table>
<thead>
<tr>
<th>&quot;Tooltip index&quot;</th>
<th>Text list ID. This refers to the desired output text.</th>
</tr>
</thead>
</table>
| ‣ As fixed string with the ID in single straight quotation marks. | Example: '2'
| ‣ As a variable (STRING) for dynamically controlling the text output. | Example: strToolTipID
| Sample assignment: PLC_PRG.strToolTipID := '2'; |

See also

- ¶ Text list

#### Element property 'Font variables'

The variables allow for dynamic control of the text display.
<table>
<thead>
<tr>
<th><strong>“Font name”</strong></th>
<th>Variable (STRING). Includes the font of the text.</th>
<th>Example: PLC_PRG.stFontVar := 'Arial';</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>The selection of fonts corresponds to the default “Font” dialog.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>“Size”</strong></th>
<th>Variable (numeric data type). Contains the font size (in pixels or points). The applied unit is specified in brackets after the variable name.</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>• &lt;pt&gt;: Points (default)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Example: PLC_PRG.iFontHeight &lt;pt&gt;</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Code: iFontHeight : INT := 12;</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• &lt;px&gt;: Pixels</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Example: PLC_PRG.iFontHeight &lt;px&gt;</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Code: iFontHeight : INT := 19;</td>
<td></td>
</tr>
</tbody>
</table>

If you click in the value field, a drop-down list opens on the right for setting the unit.

Hint: The font size is specified in points (example: Arial 12). Use points when the variable font size should match a font, for example if a font is set in the property “Text property ➔ Font”.

<table>
<thead>
<tr>
<th><strong>“Flags”</strong></th>
<th>Variable (DWORD). Contains the flags for displaying fonts.</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Flags:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• 1: Italics</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• 2: Bold</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• 4: Underline</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• 8: Strikethrough</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Note: You can combine the font displays by adding the coding of the flags. For example, a bold and underlined text: PLC_PRG.dwFontType := 6;</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>“Character set”</strong></th>
<th>Variable (DWORD). Contains a character set number for the font.</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>The selection of character set numbers corresponds to the “Script” setting of the standard “Font” dialog.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>“Color”</strong></th>
<th>Variable (DWORD). Includes the color of the text.</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Example: PLC_PRG.dwColorFont := 16#FF000000;</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>“Flags for text alignment”</strong></th>
<th>Variable (integer data type). Contains the coding for text alignment.</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Example: PLC_PRG.dwTextAlignment.</td>
<td></td>
</tr>
<tr>
<td>Coding:</td>
<td>• 0: Top left</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• 1: Horizontal center</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• 2: Right</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• 4: Vertical center</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• 8: Bottom</td>
<td></td>
</tr>
<tr>
<td>Note:</td>
<td>You can combine the text alignments by adding the coding of the flags. For example, a vertical and horizontal centered text: PLC_PRG.dwFontType := 5;</td>
<td></td>
</tr>
</tbody>
</table>
Fixed values for displaying texts are set in “Text properties”.

See also
- "Element property 'Text properties'“ on page 1227

**Element property 'Color variables'**

The Element property is used as an interface for project variables to dynamically control colors at runtime.

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Toggle color&quot;</td>
<td>The property controls the toggled color at runtime.</td>
</tr>
<tr>
<td>Value assignment:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• FALSE: The element is displayed with the color specified in the “Color” property.</td>
</tr>
<tr>
<td></td>
<td>• TRUE: The element is displayed with the color specified in the “Alarm color” property.</td>
</tr>
<tr>
<td>Assignment options:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Placeholder for the user input variable</td>
</tr>
<tr>
<td></td>
<td>– &quot;&lt;toggle/tap variable&gt;&quot;</td>
</tr>
<tr>
<td></td>
<td>– &quot;&lt;NOT toggle/tap variable&gt;&quot;</td>
</tr>
<tr>
<td></td>
<td>The color change is not controlled by its own variable, but by a user input variable.</td>
</tr>
<tr>
<td></td>
<td>Note: Specify a variable for the mouse events “Tap” or “Toggle” in the input configuration of the element. Only then is the placeholder set. If you configure a variable in both “Toggle” and “Tap”, then the variable specified in “Tap” is used.</td>
</tr>
<tr>
<td></td>
<td>Hint: Click the symbol to insert the placeholder “&lt;toggle/tap variable&gt;”. When you activate the “Inputconfiguration”, “Tap FALSE” property, then the “&lt;NOT toggle/tap variable&gt;” placeholder is displayed.</td>
</tr>
<tr>
<td></td>
<td>• Instance path of a project variable (BOOL)</td>
</tr>
<tr>
<td></td>
<td>Example: PLC_PRG.xColorIsToggeled</td>
</tr>
<tr>
<td></td>
<td>Note: In the code, declare and implement the variable specified here. Its value assignment determines when the color changes.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Normal state&quot;</td>
<td>The properties listed below control the color depending on the state. The normal state is in effect if the variable in “Color variables”, “Toggle color” is not defined or it has the value FALSE.</td>
</tr>
<tr>
<td>&quot;Alarm state&quot;</td>
<td>The alarm state is in effect if the variable in “Color variables”, “Toggle color” has the value TRUE.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Property</th>
<th>Assignment options:</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Frame color&quot;</td>
<td>• Variable (DWORD) for the frame color</td>
</tr>
<tr>
<td></td>
<td>Example: PLC_PRG.dwBorderColor</td>
</tr>
<tr>
<td></td>
<td>• Color literal</td>
</tr>
<tr>
<td></td>
<td>Example of green and opaque: 16#FF00FF00</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Property</th>
<th>Assignment options:</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Filling color&quot;</td>
<td>• Variable (DWORD) for the fill color</td>
</tr>
<tr>
<td></td>
<td>Example: PLC_PRG.dwFillColor</td>
</tr>
<tr>
<td></td>
<td>• Color literal</td>
</tr>
<tr>
<td></td>
<td>Example of gray and opaque: 16#FF888888</td>
</tr>
</tbody>
</table>
The transparency part of the color value is evaluated only if the "Activate semi-transparent drawing" option of the visualization manager is selected.

Select the "Advanced" option in the toolbar of the properties view. Then all element properties are visible.

See also
- ¶ Chapter 1.3.5.8.3 “Animating a color display” on page 1152
- ¶

Element property 'Appearance variables'

The properties contain IEC variables for controlling the appearance of the element dynamically.

| "Line width" | Variable (integer data type). Contains the line weight (in pixels). |
| "Fill attributes" | Variable (DWORD). Controls whether the fill color of the element is visible. |
| - Variable value = 0: Filled |
| - Variable value > 0: Invisible; no fill color |
| "Line style" | Variable (DWORD). Controls the line style. |
| - Coding: |
| - 0: Solid line |
| - 1: Dashed line |
| - 2: Dotted line |
| - 3: Line type "Dash Dot" |
| - 3: Line type "Dash Dot Dot" |
| - 8: Invisible; no line |

Fixed values can be set in the “Appearance” property. These values can be overwritten by dynamic variables at runtime.

See also
- ¶ “Element property ‘Appearance’” on page 1238

Element property 'State variables'

The variables control the element behavior dynamically.

| "Invisible" | Variable (BOOL). Toggles the visibility of the element. |
| TRUE: The element is not visible at runtime. |
| "Deactivate inputs" | Variable (BOOL). Toggles the operability of the element. |
| TRUE: User inputs do not have any effect in runtime more. The element is shown as deactivated. |
The “Invisible” property is supported by the “Client Animation” functionality.

See also

These properties are available only when you select the “Preview: Support client animations and overlay of native elements” option in the Visualization Manager.

You can animate a movement, a rotation, and the visibility of a visualization element.

<table>
<thead>
<tr>
<th>“Animation duration”</th>
<th>Time that the element executes an animation (in milliseconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Example: 500</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>“Move to the foreground”</th>
<th>Property value (BOOL)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>TRUE: At runtime, the element is displayed in the foreground.</td>
</tr>
<tr>
<td></td>
<td>FALSE: At runtime, the element is displayed in the layer where it was inserted in the visualization editor.</td>
</tr>
</tbody>
</table>

See also

- Chapter 1.3.5.21.5 “Using client animation” on page 1962

**Element property 'Input configuration'**

The properties contain the configurations for the user input when using the mouse or keyboard. A user input defines an event and one or more actions that are executed when an event occurs.

The “Configure” button opens the “Input configuration” dialog box for creating or modifying a user input.

Configured user inputs are briefly listed below the events. They each include the action that is triggered and the setting in short form.

Example: “Execute ST code”: #PLC_PRG.i_x := 0;

<table>
<thead>
<tr>
<th>“OnDialogClosed”</th>
<th>Input event: The user closes the dialog box.</th>
</tr>
</thead>
<tbody>
<tr>
<td>“OnMouseClick”</td>
<td>Input event: The user clicks the mouse button completely. The mouse button is clicked and released.</td>
</tr>
<tr>
<td>“OnMouseDown”</td>
<td>Input event: The user clicks down on the mouse button.</td>
</tr>
<tr>
<td>“OnMouseEnter”</td>
<td>Input event: The user drags the mouse pointer to the element.</td>
</tr>
<tr>
<td>“OnMouseLeave”</td>
<td>Input event: The user drags the mouse pointer away from the element.</td>
</tr>
<tr>
<td>“OnMouseMove”</td>
<td>Input event: The user moves the mouse pointer over the element area.</td>
</tr>
<tr>
<td>“OnMouseUp”</td>
<td>Input event: The user releases the mouse button over the element area.</td>
</tr>
</tbody>
</table>
### “Tap”
When a mouse click event occurs, the variable defined in “Variable” is described in the application. The coding depends on the options “Tap FALSE” and “Tap on enter if captured”.

### “Variable”
Variable (BOOL) that is set on mouse click.
- **Example**: PLC_PRG.bIsTapped
  - TRUE: A mouse click event exists. It lasts while the user presses the mouse button over the element. It ends when the button is released.
  - FALSE: A mouse click event does not exist.
- **Requirement**: The “Tap FALSE” option is not activated.

### “Tap FALSE”
- **☑️**: The mouse click event leads to a complementary value in “Variable”.
- **TRUE**: A mouse click event does not exist.
- **FALSE**: While the mouse click event exists.

### “Tap on enter if captured”
- **☑️**: During user input, it is also taken into consideration whether the mouse pointer is dragged within the element area or not while the mouse button is pressed.
- **TRUE**: While the mouse click event exists and the mouse pointer is moved over the element area.
- **FALSE**: A mouse click event does not exist. Or the user moves the mouse pointer outside of the element area while the mouse button is pressed.
  - The value is **TRUE** again as soon as the user moves the pointer back to the element area. The mouse is then captured.

### “Shift”
With the onset of a mouse click event, the variable is set; when the mouse click event is completed, the variable is reset.

### “Variable”
Variable (BOOL). Its value toggled when the mouse click event is ended. This is when the user releases the mouse button while the mouse pointer is over the element area.
- If the user releases the mouse button while the mouse pointer is outside of the element area, then the mouse click event is not ended and the value is not toggled.
- **Tip**: The user can cancel a started toggle input by dragging the mouse pointer out of the element area.

### “Toggle on up if captured”
- **☑️**: The value toggles regardless of where the mouse pointer is when the mouse button is released. The mouse is then captured.

### “Keyboard shortcuts”
Keyboard shortcut on the element for triggering specific input actions.
When the keyboard shortcut event occurs, the input actions in the “Event(s)” property are triggered. In this way, it is not the input action itself that leads to this input action, but the mouse input action.

### “Key”
Key pressed for input action.
- **Example**: [T]
- **Note**: The following properties appear when a key is selected.
**"Event(s)"**

- **"None"**
- **"Mouse down"**: Pressing the key triggers the input actions that are configured in the "OnMouseDown" property.
- **"Mouse up"**: Releasing the key triggers the input actions that are configured in the "OnMouseUp" property.
- **"Mouse down/up"**: Pressing and releasing the key triggers the input actions that are configured in the "OnMouseDown" property and the "OnMouseUp" property.

**"Shift"**
- Combination with the Shift key
- Example: [Shift]+[T].

**"Control"**
- Combination with the Ctrl key
- Example: [Ctrl]+[T].

**"Alt"**
- Combination with the Alt key
- Example: [Alt]+[T].

---

> All keyboard shortcuts and their actions that are configured in the visualization are listed in the "Keyboard configuration" tab.

See also

- [Chapter 1.3.5.19.2.2 “Command 'Keyboard Configuration’” on page 1558](#)
- [Chapter 1.3.5.19.3.6 “Dialog 'Input Configuration’” on page 1586](#)

**Element property 'Access rights'**

Requirement: User management is set up for the visualization.

**"Access rights"**

Opens the "Access rights" dialog. There you can edit the access privileges for the element.

Status messages:

- **"Not set. Full rights."**: Access rights for all user groups : “operable”
- **"Rights are set: Limited rights."**: Access is restricted for at least one group.

See also

- [Chapter 1.3.5.19.3.1 “Dialog 'Access Rights’” on page 1583](#)

---

**Visualization element 'Image'**

Symbol:

Tag: "Basic"

The element adds an image to the visualization. The displayed image is managed in the image pool and referenced in the visualization element by means of a static ID. In addition, you can change the displayed image dynamically by using a variable instead of the static ID.
With the “Background” command, you can define a background for the entire visualization.

Directories that contain the images for use in visualizations can be defined in the project settings (category “Visualization”).

Element properties

<table>
<thead>
<tr>
<th>“Element name”</th>
<th>Example: Status bar</th>
</tr>
</thead>
<tbody>
<tr>
<td>Optional</td>
<td>Hint: Assign individual names for elements so that they are found faster in the element list.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>“Type of element”</th>
<th>“Image”</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Identifier for static assignment</td>
</tr>
<tr>
<td></td>
<td>ID of the image file on, as it is defined in the corresponding image pool. If the image is not included in the global image pool in the POU view, then the instance path must be specified. Then the name of the image pool is preceded to make the entry unique. Example: imagepool2.button_image.</td>
</tr>
<tr>
<td></td>
<td>When entering a new ID, a file selection opens. The selected file is saved to the “GlobalImagePool”.</td>
</tr>
<tr>
<td></td>
<td>See also: Help for the “Image pool” object.</td>
</tr>
</tbody>
</table>

| “Show frame” | [✓] The image file is displayed with a frame. |
| “Clipping” | Requirement: “Scaling type” is “Fixed”. |
| [✓] Only part of the visualization is displayed that fits in the element frame. |

| “Transparent” | [✓] The image pixels that have the “Transparent color” are displayed as transparent. |

| “Transparent color” | Effective only if the “Transparent” option is activated. |
|                     | The button opens the color selection dialog. This is where you select the transparent color. |

<table>
<thead>
<tr>
<th>“Scaling type”</th>
<th>Definition how an image fits in the element frame.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>• “Isotropic”: The entire image is shown in the element frame, either larger or smaller. As a result, the proportion of height and width are retained.</td>
</tr>
<tr>
<td></td>
<td>If the alignment of the elements to each other should also be retained within a scaled frame element, then note the following. Unwanted horizontal or vertical offsets can be prevented by setting the properties “Horizontal alignment” and “Vertical alignment” to “Centered”. The alignment of the elements is retained and there are no resulting horizontal or vertical offsets. Example: A lamp is centered above a switch. The lamp should remain in the horizontally centered position, even if the frame is resized.</td>
</tr>
<tr>
<td></td>
<td>• “Anisotropic”: The image resizes automatically to the dimensions of the element frame, filling the entire element frame. As a result, the proportions are not retained.</td>
</tr>
<tr>
<td></td>
<td>• “Fixed”: The image retains its original size, even if the element frame is resized. Please note also that the “Clipping” option is selected. For each reassignment of an image ID, the element size is adapted automatically to the image size.</td>
</tr>
<tr>
<td>“Horizontal alignment”</td>
<td>Horizontal alignment of the element within the element frame:</td>
</tr>
<tr>
<td>------------------------</td>
<td>-------------------------------------------------------------</td>
</tr>
<tr>
<td></td>
<td>● “Left”</td>
</tr>
<tr>
<td></td>
<td>● “Centered”</td>
</tr>
<tr>
<td></td>
<td>● “Right”</td>
</tr>
<tr>
<td>Requirement</td>
<td>The scaling type of the image is “Isotropic” or “Fixed”.</td>
</tr>
<tr>
<td>Note</td>
<td>If the visualization is referenced, then the horizontal alignment takes effect within the frame position.</td>
</tr>
<tr>
<td>Note</td>
<td>The property “Variable” is shown below.</td>
</tr>
</tbody>
</table>

| “Variable”             | Enumeration variable (ENUM VisuElemBase.VisuEnumHorizontalAlignment). Contains the horizontal alignment. |   |
|                       | Enumeration:                                                 |   |
|                       | TYPE VisuElemBase.VisuEnumHorizontalAlignment                  |   |
|                       | LEFT                                                        |   |
|                       | HCENTER                                                     |   |
|                       | RIGHT                                                       |   |
|                       | END_TYPE                                                    |   |
| Example                | PLC_PRG.eHorizontalAlignment                                |   |
| Declaration            | PROGRAM PLC_PRG                                              |   |
|                       | VAR                                                         |   |
|                       | eHorizontalAlignment :                                       |   |
|                       | VisuElemBase.VisuEnumHorizontalAlignment :=                  |   |
|                       | VisuElemBase.VisuEnumHorizontalAlignment.HCENTER;            |   |
|                       | END_VAR                                                     |   |
**Vertical alignment**

Vertical alignment of the element within the element frame:
- "Top"
- "Centered"
- "Bottom"

Requirement: The scaling type of the image is "Isotropic" or "Fixed".

Note: If the visualization is referenced, then the horizontal alignment takes effect within the frame position.

* The "Variable" property is shown below.

**Variable**

Enumeration variable (ENUM VisuElemBase.VisuEnumVerticalAlignment). Contains the vertical alignment.

**Enumeration:**

```
TYPE VisuElemBase.VisuEnumVerticalAlignment
  DOWN
  VCENTER
  BOTTOM
END_TYPE
```

Example: PLC_PRG.eHorizontalAlignment

**Declaration:**

```
PROGRAM PLC_PRG
VAR
  eVerticalAlignment :
    VisuElemBase.VisuEnumVerticalAlignment :=
    VisuElemBase.VisuEnumVerticalAlignment.VCENTER;
END_VAR
```

See also

- *Object 'Image Pool'*

**Element property 'Position'**

The position defines the location and size of the element in the visualization window. This is based on the Cartesian coordinate system. The origin is located at the upper left corner of the window. The positive horizontal x-axis runs to the right. The positive vertical y-axis runs downwards.

**"X"**

The x-coordinate of the upper left corner of the element

Specified in pixels

Example: 10

**"Y"**

The y-coordinate of the upper left corner of the element

Specified in pixels

Example: 10

**"Width"**

Specified in pixels

Example: 150

**"Height"**

Specified in pixels

Example: 30

Tip: You can change the values in "X", "Y", "Width", and "Height" by dragging the corresponding symbols to another position in the editor.
**“Angle”**

Static angle of rotation (in degrees).

Example: 35

The element is displayed rotated in the editor. The point of rotation is the center of the element. A positive value rotates clockwise.

Tip: You can change the value in the editor by focusing the element to the handle. When the cursor is displayed as a rotating arrow, you can rotate the element about its center as a handle.

![Image of handle](image)

(1): Handle

Note: If a dynamic angle of rotation is also configured in the property “Absolute movement ➔ Internal rotation”, then the static and dynamic angles of rotation are added in runtime mode. The static angle of rotation acts as an offset.

See also

- § Chapter 1.3.5.3.2 “Positioning elements, adapting size and layer” on page 1113

**Element property 'Center’**

The properties contain fixed values for the coordinates of the point of rotation. This point of rotation is shown as the symbol. The point is used as the center for rotating and scaling.

| **“X”** | X-coordinate of the point of rotation |
| **“Y”** | Y-coordinate of the point of rotation |

You can also change the values by dragging the symbols (aleza) to other positions in the editor.

**Element property 'Colors’**

The properties contain fixed values for setting colors.

| **“Color”** | Color for the frame |
| **Requirement**: “Show frame” property is activated. |
| Please note that the normal state is in effect if the expression in the “Color variables ➔ Toggle color” property is not defined or it has the value FALSE. |

| **“Alarm color”** | Color for the frame in alarm state |
| **Requirement**: “Show frame” property is activated. |
| Please note that the alarm state is in effect if the expression in the “Color variables ➔ Toggle color” property has the value TRUE. |

| **“Transparency”** | Value (0 to 255) for defining the transparency of the selected color. |
| Example 255: The color is opaque. 0: The color is completely transparent. |
The properties contain fixed values for setting the look of the element.

**Element property 'Appearance'**

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Line width&quot;</td>
<td>Value in pixels&lt;br&gt;Note: The values 0 and 1 both result in a line weight of one pixel. If no line should be displayed, then the &quot;Line style&quot; property must be set to the option &quot;Invisible&quot;.</td>
</tr>
</tbody>
</table>
| "Line style"  | Type of line representation<br>- "Solid"
  - "Dashes"
  - "Dots"
  - "Dash Dot"
  - "Dash Dot Dot"
  - "not visible" |

You can assign variables in the "Appearance variables" property for controlling the appearance dynamically. The fixed values are defined here.

See also

- “Element property ‘Appearance variables’” on page 1285

**Element property 'Texts'**

The properties contain character strings for labeling the element. The character string can also contain a placeholder with a format definition. In runtime mode, the placeholder is replaced by the current value in the specified format.

CODESYS accepts the specified texts automatically into the “GlobalTextList” text list. Therefore, these texts can be localized.

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Text&quot;</td>
<td>Character string (without single straight quotation marks) for the labeling of the element. Add a line break by pressing the keyboard shortcut [Ctrl] + [Enter].&lt;br&gt;Example: Accesses: %i&lt;br&gt;The variable that contains the current value for the placeholder is specified in the property &quot;Text variable ➔ Text&quot;.</td>
</tr>
<tr>
<td>&quot;Tooltip&quot;</td>
<td>Character string (without single straight quotation marks) that is displayed as the tooltip of an element.&lt;br&gt;Example: Number of valid accesses.&lt;br&gt;The variable that contains the current value for the placeholder is specified in the property &quot;Text variable ➔ Tooltip&quot;.</td>
</tr>
</tbody>
</table>

See also

- “Element property ‘Text variables’” on page 1229
- Chapter 1.3.5.18.2 “Placeholders with format definition in the output text” on page 1546
The properties contain fixed values for the text properties.

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Horizontal alignment&quot;</td>
<td>Horizontal alignment of the text within the element.</td>
</tr>
<tr>
<td>&quot;Vertical alignment&quot;</td>
<td>Vertical alignment of the text within the element.</td>
</tr>
<tr>
<td>&quot;Text format&quot;</td>
<td>Definition for displaying texts that are too long</td>
</tr>
<tr>
<td></td>
<td>● &quot;Default&quot;: The long text is truncated.</td>
</tr>
<tr>
<td></td>
<td>● &quot;Line break&quot;: The text is split into parts.</td>
</tr>
<tr>
<td></td>
<td>● &quot;Ellipsis&quot;: The visible text ends with &quot;...&quot; indicating that it is not complete.</td>
</tr>
<tr>
<td>&quot;Font&quot;</td>
<td>Example: &quot;Default&quot;</td>
</tr>
<tr>
<td></td>
<td>The &quot;Font&quot; dialog box opens.</td>
</tr>
<tr>
<td></td>
<td>▼: Drop-down list with style fonts.</td>
</tr>
<tr>
<td>&quot;Font color&quot;</td>
<td>Example: &quot;Black&quot;</td>
</tr>
<tr>
<td></td>
<td>The &quot;Color&quot; dialog box opens.</td>
</tr>
<tr>
<td></td>
<td>▼: Drop-down list with style colors.</td>
</tr>
<tr>
<td>&quot;Transparency&quot;</td>
<td>Whole number (value range from 0 to 255). This determines the transparency of the respective color.</td>
</tr>
<tr>
<td></td>
<td>Example: 255: The color is opaque.</td>
</tr>
<tr>
<td></td>
<td>0: The color is completely transparent.</td>
</tr>
<tr>
<td></td>
<td>Please note: If the color is a style color and already has a transparency value, then this property is write-protected.</td>
</tr>
</tbody>
</table>

Variable (STRING). Contains the image ID. The contents of the string correspond to the description of the "Static ID" property.

Example: PLC_PRG.stImageID := 'ImagePool_A.Image3';

See also
- Chapter 1.3.5.18.1.5 "Visualization element 'Image'" on page 1272
- Object 'Image Pool'

You can use this element property for animating a series of image files.

Variable (integer data type). Contains the version of the image.

If the variable changes, then the visualization re-reads the image referenced in the "Image ID" property and displays it.

The visualization displays animations when the image file on the controller is updated continuously, thus incrementing the version variable. The application must be programmed for this.

Possible applications
- Displaying graphics that are generated by the application
- Displaying images that are refreshed by a camera
The properties contain IEC variables for controlling the position of the element dynamically. The reference point is the upper left corner of the element. In runtime mode, the entire element is moved.

### "Movement"

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>Variable (numeric data type). Defines the X position (in pixels).</td>
<td>PLC_PRG.iPos_X. Increasing this value in runtime mode moves the element to the right.</td>
</tr>
<tr>
<td>Y</td>
<td>Variable (numeric data type). Defines the Y position (in pixels).</td>
<td>PLC_PRG.iPos_Y. Increasing this value in runtime mode moves the element downwards.</td>
</tr>
</tbody>
</table>

### "Rotation"

Variable (numeric data type). Defines the angle of rotation (in degrees).

Example: PLC_PRG.iAngle1.

The midpoint of the element rotates at the “Center” point. This rotation point is shown as the symbol.

In runtime mode, the alignment of the element remains the same with respect to the coordinate system of the visualization. Increasing the value rotates the element to the right.

### "Scaling"

Variable (integer data type). Causes centric stretching.

Example: PLC_PRG.iScaling.

The reference point is the “Center” property.

The value 1 shrinks the element by a factor of 0.001. The value 1000 returns the element to its original size.

### "Interior rotation"

Variable (numeric data type). Defines the angle of rotation (in degrees).

Example: PLC_PRG.iAngle2.

In runtime mode, the element rotates about the point of rotation specified in “Center” according to the value of the variable. In addition, the alignment of the element rotates according to the coordinate system of the visualization. Increasing the value in the code rotates clockwise.

The rotation point is shown as the symbol.

Note: If a static angle of rotation is specified in the property “Position → Angle”, then the static angle of rotation is added to the variable angle of rotation (offset) when the visualization is executed.

You can link the variables to a unit conversion.
The properties “X”, “Y”, “Rotation”, and “Interior rotation” are supported by the "Client Animation" functionality.

See also
- Unit conversion

Element property ‘Relative movement’

The properties contains variables for moving the element. The reference point is the position of the element (‘Position’ property). The shape of the element can change.

<table>
<thead>
<tr>
<th>“Movement top-left”</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>“X”</td>
<td>Variable (integer data type). It contains the number (in pixels) that the left edge is moved horizontally. Incrementing the value moves the element to the right. Example: PLC_PRG.iDeltaX</td>
</tr>
<tr>
<td>“Y”</td>
<td>Variable (integer data type). It contains the number (in pixels) that the top edge is moved vertically. Incrementing the value moves the element to the down. Example: PLC_PRG.iDeltaY</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>“Movement bottom-right”</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>“X”</td>
<td>Variable (integer data type). It contains the number (in pixels) that the right edge is moved horizontally. Incrementing the value moves the element to the right. Example: PLC_PRG.iDeltaWidth</td>
</tr>
<tr>
<td>“Y”</td>
<td>Variable (integer data type). It contains the number (in pixels) that the bottom edge is moved vertically. Incrementing the value moves the element to the down. Example: PLC_PRG.iDeltaHeight</td>
</tr>
</tbody>
</table>

See also
- “Element property ‘Absolute movement”’ on page 1227

Element property 'Text variables'

These properties are variables with contents that replace a format definition.

<table>
<thead>
<tr>
<th>“Text variable”</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Variable (data type compliant with the format definition). It contains what is printed instead of the format definition. Example: PLC_PRG.iAccesses</td>
<td></td>
</tr>
<tr>
<td>Note: The format definition is part of the text in the property “Texts ➔ Text”. Note: If you specify a variable of type enumeration with text list support, then the name of the enumeration data type is added automatically in angle brackets after the variable name. Example: PLC_PRG.enVar &lt;enumeration name&gt;. Then the symbolic value of the enumeration component is printed instead of the numeric value when text is printed. Refer to the help page for the enumerations.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>“Tooltip variable”</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Variable (data type compliant with the format definition). It contains what is printed instead of the format definition. Example: PLC_PRG.iAccessesInTooltip</td>
<td></td>
</tr>
<tr>
<td>Note: The format definition is part of the text in the property “Texts ➔ Tooltip”.</td>
<td></td>
</tr>
</tbody>
</table>
Dynamic texts are variably indexed texts of a text list. At runtime, the text is displayed that is currently indexed in the variable.

<table>
<thead>
<tr>
<th><strong>“Text list”</strong></th>
<th>Variable (string) or name of the text list as a fixed string in single straight quotation marks.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example:</td>
<td>'Errorlist'</td>
</tr>
<tr>
<td></td>
<td>Drop-down list with the dialogs available in the text lists.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>“Text index”</strong></th>
<th>Text list ID. This refers to the desired output text.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>• As fixed string with the ID in single straight quotation marks.</td>
</tr>
<tr>
<td></td>
<td>Example: '1'</td>
</tr>
<tr>
<td></td>
<td>• As a variable (STRING) for dynamically controlling the text output.</td>
</tr>
<tr>
<td></td>
<td>Example: strTextID</td>
</tr>
<tr>
<td></td>
<td>Sample assignment: PLC_PRG.strTextID := '1';</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>“Tooltip index”</strong></th>
<th>Text list ID. This refers to the desired output text.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>• As fixed string with the ID in single straight quotation marks.</td>
</tr>
<tr>
<td></td>
<td>Example: '2'</td>
</tr>
<tr>
<td></td>
<td>• As a variable (STRING) for dynamically controlling the text output.</td>
</tr>
<tr>
<td></td>
<td>Example: strToolTipID</td>
</tr>
<tr>
<td></td>
<td>Sample assignment: PLC_PRG.strToolTipID := '2';</td>
</tr>
</tbody>
</table>

See also

- **Text list**

The variables allow for dynamic control of the text display.

Element property 'Font variables'

- **See also**
  - Chapter 1.3.5.18.2 “Placeholders with format definition in the output text” on page 1546
  - “Element property ‘Texts’” on page 1226
  - Enumerations
| **Font name** | Variable (STRING). Includes the font of the text.  
Example: PLC_PRG.stFontVar := 'Arial';  
The selection of fonts corresponds to the default “Font” dialog. |
| **Size** | Variable (numeric data type). Contains the font size (in pixels or points). The applied unit is specified in brackets after the variable name.  
- `<pt>`: Points (default)  
  Example: PLC_PRG.iFontHeight <pt>  
  Code: iFontHeight : INT := 12;  
- `<px>`: Pixels  
  Example: PLC_PRG.iFontHeight <px>  
  Code: iFontHeight : INT := 19;  
  If you click in the value field, a drop-down list opens on the right for setting the unit.  
  Hint: The font size is specified in points (example: Arial 12). Use points when the variable font size should match a font, for example if a font is set in the property “Text property → Font”. |
| **Flags** | Variable (DWORD). Contains the flags for displaying fonts.  
Flags:  
- 1: Italics  
- 2: Bold  
- 4: Underline  
- 8: Strikethrough  
Note: You can combine the font displays by adding the coding of the flags. For example, a bold and underlined text: PLC_PRG.dwFontType := 6; |
| **Character set** | Variable (DWORD). Contains a character set number for the font.  
The selection of character set numbers corresponds to the “Script” setting of the standard “Font” dialog. |
| **Color** | Variable (DWORD). Includes the color of the text.  
Example: PLC_PRG.dwColorFont := 16#FF000000; |
| **Flags for text alignment** | Variable (integer data type). Contains the coding for text alignment.  
Example: PLC_PRG.dwTextAlignment.  
Coding:  
- 0: Top left  
- 1: Horizontal center  
- 2: Right  
- 4: Vertical center  
- 8: Bottom  
Note: You can combine the text alignments by adding the coding of the flags. For example, a vertical and horizontal centered text: PLC_PRG.dwFontType := 5; |
Fixed values for displaying texts are set in “Text properties”.

See also

- “Element property 'Text properties'” on page 1227

**Element property 'Colorvariables'**

The Element property is used as an interface for project variables to dynamically control colors at runtime.
"Toggle color"  
The property controls the toggled color at runtime. 

Value assignment: 
- **FALSE**: The element is displayed with the color specified in the "Color" property. 
- **TRUE**: The element is displayed with the color specified in the "Alarm color" property. 

Assigning the property: 
- Placeholder for the user input variable 
  - "<toggle/tap variable>" 
  - "<NOT toggle/tap variable>" 

The color change is not controlled by its own variable, but by a user input variable.  

Note: Specify a variable for the mouse events "Tap" or "Toggle" in the input configuration of the element. Only then is the placeholder set. If you configure a variable in both "Toggle" and "Tap", then the variable specified in "Tap" is used. 

Hint: Click the symbol ✈ to insert the placeholder "<toggle/tap variable>". When you activate the "Input configuration", "Tap FALSE" property, then the "<NOT toggle/tap variable>" placeholder is displayed. 

- Instance path of a project variable (BOOL) 
  Example: PLC_PRG.xColorIsToggeled 

Note: In the code, declare and implement the variable specified here. Its value assignment determines when the color changes. 

"Color"  
Color variable for the frame 

- Variable (DWORD) for the color 
  Example: PLC_PRG.dwColor 

- Color literal 
  Example of gray and opaque: 16#FF888888 

Requirement: "Show frame" property is activated. 

Please note that the normal state is in effect if the expression in the "Colorvariables ➜ Toggle color" property is not defined or it has the value FALSE. 

"Alarm color"  
Color variable for the frame in alarm state 

- Variable (DWORD) for the alarm color 
  Example: PLC_PRG.dwAlarmColor 

- Color literal 
  Example of red and opaque: 16#FFFF0000 

Please note that the alarm state is in effect if the expression in the "Colorvariables ➜ Toggle color" property has the value TRUE. 

The transparency part of the color value is evaluated only if the "Activate semi-transparent drawing" option of the visualization manager is selected. 

Select the "Advanced" option in the toolbar of the properties view. Then all element properties are visible.
Element property 'Appearance variables'
The properties contain variables for controlling the appearance of the element dynamically.

| “Line width” | Variable (integer data type). Contains the line weight (in pixels).
|              | Note: The values 0 and 1 both result in a line weight of one pixel. If no line should be displayed, then the “Line style” property must be set to the option “Invisible”.

| “Line style” | Variable (DWORD). Controls the line style.
|             | Coding:
|             | ● 0: Solid line
|             | ● 1: Dashed line
|             | ● 2: Dotted line
|             | ● 3: Line type "Dash Dot"
|             | ● 3: Line type "Dash Dot Dot"
|             | ● 8: Invisible: The line is not drawn.

Fixed values can be set in the “Appearance” property. These values can be overwritten by dynamic variables at runtime.

Element property 'State variables'
The variables control the element behavior dynamically.

| “Invisible” | Variable (BOOL). Toggles the visibility of the element.
|            | TRUE: The element is not visible at runtime.

| “Deactivate inputs” | Variable (BOOL). Toggles the operability of the element.
|                     | TRUE: User inputs do not have any effect in runtime more. The element is shown as deactivated.

The “Invisible” property is supported by the “Client Animation” functionality.

These properties are available only when you select the “Preview: Support client animations and overlay of native elements” option in the Visualization Manager.

You can animate a movement, a rotation, and the visibility of a visualization element.
**“Animation duration”**
Time that the element executes an animation (in milliseconds)
Example: 500

**“Move to the foreground”**
Property value (BOOL)
TRUE: At runtime, the element is displayed in the foreground.
FALSE: At runtime, the element is displayed in the layer where it was inserted in the visualization editor.

See also
- § Chapter 1.3.5.21.5 “Using client animation” on page 1962

**Element property 'Input configuration’**
The properties contain the configurations for the user input when using the mouse or keyboard.
User input is a user event from the perspective of the element.

The “Configure” button opens the “Input configuration” dialog box for creating or modifying a user input configuration.
A configuration contains one or more input actions for the respective input event. Existing input actions are displayed below it.
Example: “Execute ST code”: #PLC_PRG.i_x := 0;

<table>
<thead>
<tr>
<th>Input event</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;OnDialogClosed&quot;</td>
<td>Input event: The user closes the dialog box.</td>
</tr>
<tr>
<td>&quot;OnMouseClick&quot;</td>
<td>Input event: A user clicks the element completely. The mouse button is clicked and released.</td>
</tr>
<tr>
<td>&quot;OnMouseDown&quot;</td>
<td>Input event: A user clicks down on the element only.</td>
</tr>
<tr>
<td>&quot;OnMouseEnter&quot;</td>
<td>Input event: A user drags the mouse pointer to the element.</td>
</tr>
<tr>
<td>&quot;OnMouseLeave&quot;</td>
<td>Input event: A user drags the mouse pointer away from the element.</td>
</tr>
<tr>
<td>&quot;OnMouseMove&quot;</td>
<td>Input event: A user moves the mouse pointer over the element area.</td>
</tr>
<tr>
<td>&quot;OnMouseUp&quot;</td>
<td>Input event: The user releases the mouse button over the element area.</td>
</tr>
</tbody>
</table>

See also
- § Chapter 1.3.5.19.3.6 “Dialog ‘Input Configuration’” on page 1586

**“Hotkeys”**
Keyboard shortcut on the element for triggering specific input actions.
When the keyboard shortcut event occurs, the input actions in the “Event(s)” property are triggered.

<table>
<thead>
<tr>
<th>Key pressed for input action</th>
<th>Example: [T]</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
</table>
| “Event(s)”                | • “None”  
  • “Mouse down”: Pressing the key triggers the input actions that are configured in the “OnMouseDown” property.  
  • “Mouse up”: Releasing the key triggers the input actions that are configured in the “OnMouseUp” property.  
  • “Mouse down/up”: Pressing and releasing the key triggers the input actions that are configured in the “OnMouseDown” property and the “OnMouseUp” property.  |

**“Shift”**
☑: Combination with the Shift key
Example: [Shift]+[T].
“Control”  
☑ Combination with the Ctrl key  
Example: [Ctrl]+[T].

“Alt”  
☑ Combination with the Alt key  
Example: [Alt]+[T].

All keyboard shortcuts and their actions that are configured in the visualization are listed in the “Keyboard configuration” tab.

See also  
● ☞ Chapter 1.3.5.19.2.2 “Command 'Keyboard Configuration'' on page 1558

Element property 'Access rights'

Requirement: User management is set up for the visualization.

“Access rights”  
Opens the “Access rights” dialog. There you can edit the access privileges for the element.  
Status messages:  
● “Not set. Full rights.”: Access rights for all user groups: “operable”  
● “Rights are set: Limited rights”: Access is restricted for at least one group.

See also  
● ☞ Chapter 1.3.5.19.3.1 “Dialog 'Access Rights’” on page 1583

See also  
● Project Settings - Visualization  
● ☞ Chapter 1.3.5.19.2.10 “Command 'Background'” on page 1565

Visualization element 'Frame'

Symbol:

Tag: “Basic”

The element serves as a frame in which to display one or more already existing visualizations. You get a structured user interface. The size of the frame can be fixed or scaled. The display area of the referenced visualization then adapts itself to the frame size.

Element properties

<table>
<thead>
<tr>
<th>“Element name”</th>
<th>Example: refVisUserInfo</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Type of element”</td>
<td>“Frame”</td>
</tr>
</tbody>
</table>

2020/12/10 3ADR010583, 1, en_US 1287
| “Clipping” | Fixed size. Only that part of the referenced visualization that fits inside the frame is displayed. Requirement: “Scaling type” property is “Fixed”. |
| “Show frame” | Displays the frame |
| ● “No frame”: The displayed area of the frame does not have borders. |
| ● “Frame”: The displayed area of the frame has borders. |
| ● “No frame with offset”: The displayed area of the frame does not have a border and the displayed area of the referenced visualization is reduced inwards by one pixel as compared to the frame area. The gap prevents the referenced visualization from touching any adjacent elements. |

| “Scaling type” | The method with which the height and width of the referenced visualization are scaled. |
| ● “Isotropic”: The visualization is scaled to the size of the element. The visualization retains its proportions with a fixed height/width ratio. |
| ● “Anisotropic”: The visualization is scaled to the size of the element. The height and width are adapted to the element independently of each other. |
| ● “Fixed”: the visualization is displayed in its original size without taking into account the size of the element. |
| ● “Fixed and scrollable”: The visualization is displayed fixed in the element. If it is larger than the element, the element will be provided with scrollbars. Please note: assign variables to the properties “Scroll position variable horizontal” or “Scroll position variable vertical”. You can then edit the data of the scrollbar position in the application. |

Element properties ‘Scrollbar settings’ The properties contain variables for the position of the scrollboxes in the scrollbars. You can then edit the data of the scrollbox position in the application.

| Requirement: the property “Scaling type” is “fixed and scrollable”. |
| “Scroll position variable horizontal” | Variable (integer data type, also as array). Contains the position of the horizontal or vertical scrollbox. The array contains the position for every display variant. If the visualization runs on several display variants, then the position changes are decoupled from each other. |
| “Scroll position variable vertical” | Example: |
| PLC_FRG.iScrollHor[CURRENTCLIENTID] PLC_FRG.iScrollVer[CURRENTCLIENTID] The variable is declared as an array in the example. |
| iScrollHor: ARRAY[0..20] OF INT; iScrollVer: ARRAY[0..20] OF INT; CURRENTCLIENTID indexes the current display variant. You can combine the variables with a unit conversion. |

See also
- Unit conversion
**Deactivation of the background character**

The background is drawn. The non-animated element of the referenced visualization is drawn as a background bitmap in order to optimize the performance of the visualization.

Consequence: Elements can be displayed in an unexpected order at runtime. For example, an animated element can push itself behind the frame at runtime.

☑️ Background character is deactivated in order to avoid the behaviour described above.

---

### Element property 'References'

Contains the currently configured visualization references as a subnode

#### "References"

Clicking “Configure” opens the “Frame Configuration” dialog. This is used to manage the referenced visualizations.

Caution: Visualizations can be nested at any depth by means of frame elements. In order to use the “Switch to any visualization” frame selection type without any problems, a frame must not contain more than 21 referenced visualizations. For more information, see also the description for the “Input configuration” of an element: Action “Switch frame visualization”.

List of the currently referenced visualizations

Visualizations that have a button also have this displayed as a subnode. Each interface variable is listed with the currently assigned transfer parameters.

Example:

```plaintext
vis_FormA
```

- `iDataToDisplay_1:PLC_PRG.iVar1`
- `iDataToDisplay_2:PLC_PRG.iVar2`

Hint: You can change the assignment of the variables to an interface variable here and edit the value field. Or click the “Configure” button instead.

---

See also
- ☑️ Chapter 1.3.5.19.2.1 “Command ‘Interface Editor’” on page 1556
- ☑️ Chapter 1.3.5.19.3.6 “Dialog ‘Input Configuration’” on page 1586

---

### Element property 'Position'

The position defines the location and size of the element in the visualization window. These are based on the Cartesian coordinate system. The origin is located at the upper left corner of the window. The positive horizontal x-axis runs to the right. The positive vertical y-axis runs downwards.

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>&quot;X&quot;</strong></td>
<td>X coordinate of the upper left corner of the element Specified in pixels. Example: 10.</td>
</tr>
<tr>
<td><strong>&quot;Y&quot;</strong></td>
<td>Y coordinate of the upper left corner of the element Specified in pixels. Example: 10.</td>
</tr>
<tr>
<td><strong>&quot;Width&quot;</strong></td>
<td>Specified in pixels. Example: 150</td>
</tr>
<tr>
<td><strong>&quot;Height&quot;</strong></td>
<td>Specified in pixels. Example: 30</td>
</tr>
</tbody>
</table>
The properties contain fixed values for the coordinates of the point of rotation. This point of rotation is shown as the symbol. The point is used as the center for rotating and scaling.

### Element property 'Center'

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>X-coordinate of the point of rotation</td>
</tr>
<tr>
<td>Y</td>
<td>Y-coordinate of the point of rotation</td>
</tr>
</tbody>
</table>

The properties contain fixed values for the colors.

### Element property 'Colors'

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Color</td>
<td>Color of the frame&lt;br&gt;▼: Selection list with style colors appears&lt;br&gt;: Standard dialog “Color” opens for selecting a color.</td>
</tr>
<tr>
<td>Alarm color</td>
<td>Color with which the element is filled during the alarm state. Please note: Alarm state is when the value of the boolean variable in the property “Color variables  Toggle color” is not defined or its value is FALSE.</td>
</tr>
<tr>
<td>Transparency</td>
<td>Integer number (value range from 255 to 0). Specifies the transparency of the associated color.&lt;br&gt;255: The color is opaque.&lt;br&gt;0: The color is fully transparent. Please note: If the color is a style color and already contains a transparency value, then this property is write-protected.</td>
</tr>
</tbody>
</table>

The properties contain fixed values for setting the look of the element.

### Element property 'Appearance'

See also
- Chapter 1.3.5.3.2 “Positioning elements, adapting size and layer” on page 1113
- Chapter 1.3.5.3.3 “Assigning a color” on page 1115
<table>
<thead>
<tr>
<th><strong>“Line width”</strong></th>
<th>Value in pixels</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example: 2</td>
<td></td>
</tr>
</tbody>
</table>

Note: The values 0 and 1 both result in a line weight of one pixel. If no line should be displayed, then the “Line style” property must be set to the option “Invisible”.

<table>
<thead>
<tr>
<th><strong>“Line style”</strong></th>
<th>Type of line representation</th>
</tr>
</thead>
<tbody>
<tr>
<td>● “Solid”</td>
<td></td>
</tr>
<tr>
<td>● “Dashes”</td>
<td></td>
</tr>
<tr>
<td>● “Dots”</td>
<td></td>
</tr>
<tr>
<td>● “Dash Dot”</td>
<td></td>
</tr>
<tr>
<td>● “Dash Dot Dot”</td>
<td></td>
</tr>
<tr>
<td>● “not visible”</td>
<td></td>
</tr>
</tbody>
</table>

You can assign variables in the “Appearance variables” property for controlling the appearance dynamically. The fixed values are defined here.

See also
- § “Element property ‘Appearance variables’” on page 1285

**Element property 'Texts’**

The properties contain character strings for labeling the element. The character string can also contain a placeholder with a format definition. In runtime mode, the placeholder is replaced by the current value in the specified format.

CODESYS accepts the specified texts automatically into the “GlobalTextList” text list. Therefore, these texts can be localized.

<table>
<thead>
<tr>
<th><strong>“Text”</strong></th>
<th>Character string (without single straight quotation marks) for the labeling the element. Add a line break by pressing the keyboard shortcut [Ctrl] + [Enter].</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example:</td>
<td>Accesses: %i</td>
</tr>
<tr>
<td></td>
<td>The variable that contains the current value for the placeholder is specified in the property “Text variable” ➔ “Text”.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>“Tooltip”</strong></th>
<th>Character string (without single straight quotation marks) that is displayed as the tooltip of an element.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example:</td>
<td>Number of valid accesses.</td>
</tr>
<tr>
<td></td>
<td>The variable that contains the current value for the placeholder is specified in the property “Text variable” ➔ “Tooltip”.</td>
</tr>
</tbody>
</table>

See also
- § “Element property ‘Text variables’” on page 1229
- § Chapter 1.3.5.18.2 “Placeholders with format definition in the output text” on page 1546

**Element property 'Text properties’**

The properties contain fixed values for the text properties.

<table>
<thead>
<tr>
<th><strong>“Horizontal alignment”</strong></th>
<th>Horizontal alignment of the text within the element.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>“Vertical alignment”</strong></td>
<td>Vertical alignment of the text within the element.</td>
</tr>
<tr>
<td><strong>Text format</strong></td>
<td>Definition for displaying texts that are too long</td>
</tr>
<tr>
<td>----------------</td>
<td>---------------------------------------------------</td>
</tr>
<tr>
<td>● &quot;Default&quot;</td>
<td>The long text is truncated.</td>
</tr>
<tr>
<td>● &quot;Line break&quot;</td>
<td>The text is split into parts.</td>
</tr>
<tr>
<td>● &quot;Ellipsis&quot;</td>
<td>The visible text ends with &quot;...&quot; indicating that it is not complete.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Font</strong></th>
<th>Example: &quot;Default&quot;</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>▲: The &quot;Font&quot; dialog box opens.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Font color</strong></th>
<th>Example: &quot;Black&quot;</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>▲: The &quot;Color&quot; dialog box opens.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Transparency</strong></th>
<th>Whole number (value range from 0 to 255). This determines the transparency of the respective color.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example: 255</td>
<td>The color is opaque.</td>
</tr>
<tr>
<td>0</td>
<td>The color is completely transparent.</td>
</tr>
</tbody>
</table>

Please note: If the color is a style color and already has a transparency value, then this property is write-protected.

### Element property 'Absolute movement'

The properties contain IEC variables for controlling the position of the element dynamically. The reference point is the upper left corner of the element. In runtime mode, the entire element is moved.

<table>
<thead>
<tr>
<th><strong>Movement</strong></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>X</strong></td>
<td>Variable (numeric data type). Defines the X position (in pixels).</td>
</tr>
<tr>
<td>Example:</td>
<td>PLC_PRG.iPos_X.</td>
</tr>
<tr>
<td>Increasing this value in runtime mode moves the element to the right.</td>
<td></td>
</tr>
</tbody>
</table>

| **Y**        | Variable (numeric data type). Defines the Y position (in pixels). |
| Example:     | PLC_PRG.iPos_Y. |
| Increasing this value in runtime mode moves the element downwards. |

| **Rotation** | Variable (numeric data type). Defines the angle of rotation (in degrees). |
| Example:     | PLC_PRG.iAngle1. |
| The midpoint of the element rotates at the “Center” point. This rotation point is shown as the symbol. |
| In runtime mode, the alignment of the element remains the same with respect to the coordinate system of the visualization. Increasing the value rotates the element to the right. |

| **Scaling** | Variable (integer data type). Causes centric stretching. |
| Example:    | PLC_PRG.iScaling. |
| The reference point is the “Center” property. |
| The value 1 shrinks the element by a factor of 0.001. The value 1000 returns the element to its original size. |
"Interior rotation"  
Variable (numeric data type). Defines the angle of rotation (in degrees).

Example: PLC_PRG.iAngle2.

In runtime mode, the element rotates about the point of rotation specified in "Center" according to the value of the variable. In addition, the alignment of the element rotates according to the coordinate system of the visualization. Increasing the value in the code rotates clockwise.

The rotation point is shown as the ‼ symbol.

Note: If a static angle of rotation is specified in the property “Position → Angle”, then the static angle of rotation is added to the variable angle of rotation (offset) when the visualization is executed.

You can link the variables to a unit conversion.

The properties “X”, “Y”, “Rotation”, and “Interior rotation” are supported by the "Client Animation" functionality.

See also
- Unit conversion
- "Element property ‘Absolute movement’" on page 1227

---

Element property ‘Relative movement’

The properties contains variables for moving the element. The reference point is the position of the element (“Position” property). The shape of the element can change.

| “Movement top-left” |  
| "X"      | Variable (integer data type). It contains the number (in pixels) that the left edge is moved horizontally. Incrementing the value moves the element to the right. Example: PLC_PRG.iDeltaX |
| "Y"      | Variable (integer data type). It contains the number (in pixels) that the top edge is moved vertically. Incrementing the value moves the element to the down. Example: PLC_PRG.iDeltaY |

| “Movement bottom-right” |  
| "X"      | Variable (integer data type). It contains the number (in pixels) that the right edge is moved horizontally. Incrementing the value moves the element to the right. Example: PLC_PRG.iDeltaWidth |
| "Y"      | Variable (integer data type). It contains the number (in pixels) that the bottom edge is moved vertically. Incrementing the value moves the element to the down. Example: PLC_PRG.iDeltaHeight |

See also
- ☞ “Element property ‘Absolute movement’” on page 1227
### Element property 'Text variables'

These properties are variables with contents that replace a format definition.

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Text variable&quot;</td>
<td>Variable (data type compliant with the format definition). It contains what is printed instead of the format definition.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Example: PLC_PRG.iAccesses</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Note: The format definition is part of the text in the property “Texts ➔ Text”.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Note: If you specify a variable of type enumeration with text list support, then the name of the enumeration data type is added automatically in angle brackets after the variable name. Example: PLC_PRG.enVar &lt;enumeration name&gt;. Then the symbolic value of the enumeration component is printed instead of the numeric value when text is printed. Refer to the help page for the enumerations.</td>
<td></td>
</tr>
<tr>
<td>&quot;Tooltip variable&quot;</td>
<td>Variable (data type compliant with the format definition). It contains what is printed instead of the format definition.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Example: PLC_PRG.iAccessesInTooltip</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Note: The format definition is part of the text in the property “Texts ➔ Tooltip”.</td>
<td></td>
</tr>
</tbody>
</table>

See also
- ☛ Chapter 1.3.5.18.2 “Placeholders with format definition in the output text” on page 1546
- ☛ “Element property 'Texts'” on page 1226
- Enumerations

### Element property 'Dynamic texts'

Dynamic texts are variably indexed texts of a text list. At runtime, the text is displayed that is currently indexed in the variable.

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Text list&quot;</td>
<td>Variable (string) or name of the text list as a fixed string in single straight quotation marks.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Example: 'Errorlist'</td>
<td></td>
</tr>
<tr>
<td></td>
<td>◄: Drop-down list with the dialogs available in the text lists.</td>
<td></td>
</tr>
<tr>
<td>&quot;Text index&quot;</td>
<td>Text list ID. This refers to the desired output text.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- As fixed string with the ID in single straight quotation marks.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Example: '1'</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- As a variable (STRING) for dynamically controlling the text output.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Example: strTextID</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Sample assignment: PLC_PRG.strTextID := '1';</td>
<td></td>
</tr>
<tr>
<td>&quot;Tooltip index&quot;</td>
<td>Text list ID. This refers to the desired output text.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- As fixed string with the ID in single straight quotation marks.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Example: '2'</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- As a variable (STRING) for dynamically controlling the text output.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Example: strToolTipID</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Sample assignment: PLC_PRG.strToolTipID := '2';</td>
<td></td>
</tr>
</tbody>
</table>

See also
- Text list

### Element property 'Font variables'

The variables allow for dynamic control of the text display.
### “Font name”
Variable (STRING). Includes the font of the text.
Example: `PLC_PRG.stFontVar := 'Arial';`

The selection of fonts corresponds to the default “Font” dialog.

### “Size”
Variable (numeric data type). Contains the font size (in pixels or points). The applied unit is specified in brackets after the variable name.
- **<pt>:** Points (default)
  Example: `PLC_PRG.iFontHeight <pt>`
  Code: `iFontHeight : INT := 12;`
- **<px>:** Pixels
  Example: `PLC_PRG.iFontHeight <px>`
  Code: `iFontHeight : INT := 19;`

If you click in the value field, a drop-down list opens on the right for setting the unit.

Hint: The font size is specified in points (example: Arial 12). Use points when the variable font size should match a font, for example if a font is set in the property “Text property → Font”.

### “Flags”
Variable (DWORD). Contains the flags for displaying fonts.
Flags:
- **1**: Italics
- **2**: Bold
- **4**: Underline
- **8**: Strikethrough

Note: You can combine the font displays by adding the coding of the flags. For example, a bold and underlined text: `PLC_PRG.dwFontType := 6;`

### “Character set”
Variable (DWORD). Contains a character set number for the font.
The selection of character set numbers corresponds to the “Script” setting of the standard “Font” dialog.

### “Color”
Variable (DWORD). Includes the color of the text.
Example: `PLC_PRG.dwColorFont:= 16#FF000000;`

### “Flags for text alignment”
Variable (integer data type). Contains the coding for text alignment.
Example: `PLC_PRG.dwTextAlignment.`
Coding:
- **0**: Top left
- **1**: Horizontal center
- **2**: Right
- **4**: Vertical center
- **8**: Bottom

Note: You can combine the text alignments by adding the coding of the flags. For example, a vertical and horizontal centered text: `PLC_PRG.dwFontType := 5;`
Fixed values for displaying texts are set in “Text properties”.

See also

- “Element property 'Text properties'” on page 1227

**Element property 'Colorvariables'**

The Element property is used as an interface for project variables to dynamically control colors at runtime.
"Toggle color"  

The property controls the toggled color at runtime.

Value assignment:
- **FALSE**: The element is displayed with the color specified in the "Color" property.
- **TRUE**: The element is displayed with the color specified in the "Alarm color" property.

Assigning the property:
- Placeholder for the user input variable
  - "<toggle/tap variable>"
  - "<NOT toggle/tap variable>"

The color change is not controlled by its own variable, but by a user input variable.

Note: Specify a variable for the mouse events "Tap" or "Toggle" in the input configuration of the element. Only then is the placeholder set. If you configure a variable in both "Toggle" and "Tap", then the variable specified in "Tap" is used.

Hint: Click the symbol to insert the placeholder "<toggle/tap variable>". When you activate the "Input configuration", "Tap FALSE" property, then the "<NOT toggle/tap variable>" placeholder is displayed.

- Instance path of a project variable (BOOL)
  Example: PLC_PRG.xColorIsToggled

Note: In the code, declare and implement the variable specified here. Its value assignment determines when the color changes.

"Color"  

Color variable for the frame

- Variable (DWORD) for the color
  Example: PLC_PRG.dwColor
- Color literal
  Example of gray and opaque: 16#FF888888

Requirement: "Show frame" property is activated.

Please note that the normal state is in effect if the expression in the "Colorvariables Toggle color" property is not defined or it has the value FALSE.

"Alarm color"  

Color variable for the frame in alarm state

- Variable (DWORD) for the alarm color
  Example: PLC_PRG.dwAlarmColor
- Color literal
  Example of red and opaque: 16#FFFF0000

Please note that the alarm state is in effect if the expression in the "Colorvariables Toggle color" property has the value TRUE.

The transparency part of the color value is evaluated only if the "Activate semi-transparent drawing" option of the visualization manager is selected.

Select the "Advanced" option in the toolbar of the properties view. Then all element properties are visible.
The properties contain variables for controlling the appearance of the element dynamically.

### Element property 'Appearance variables'

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Line width&quot;</td>
<td>Variable (integer data type). Contains the line weight (in pixels). Note: The values 0 and 1 both result in a line weight of one pixel. If no line should be displayed, then the &quot;Line style&quot; property must be set to the option &quot;Invisible&quot;.</td>
</tr>
</tbody>
</table>

**Fixed values can be set in the “Appearance” property. These values can be overwritten by dynamic variables at runtime.**

### Element property 'Switch frame variable'

The variable controls the switching of the referenced visualizations. This variable indexes one of the referenced frame visualizations and this is displayed in the frame. When the value of the variable changes, it switches to the recently indexed visualization.

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Variable&quot;</td>
<td>Variable (integer data type) that contains the index of the active visualization Example: PLC_PRG.uiIndexVisu Hint: The &quot;Frame Configuration&quot; dialog includes a list of referenced visualizations. The visualizations are automatically numerically indexed via the order in the list. Note: This variant of switching usually affects all connected display variants. Array element (integer data type) for index access via CURRENTCLIENTID Example: PLC_PRG.aIndexVisu[CURRENTCLIENTID] Note: This variant of switching applies to the current client only, and therefore only on one display variant. That is the display variant where the value change was triggered (for example, by means of user input).</td>
</tr>
</tbody>
</table>

### Element property 'State variables'

The variables control the element behavior dynamically.
“Invisible” Variable (BOOL). Toggles the visibility of the element.
TRUE: The element is not visible at runtime.

“Deactivate inputs” Variable (BOOL). Toggles the operability of the element.
TRUE: User inputs do not have any effect in runtime more. The element is shown as deactivated.

The “Invisible” property is supported by the “Client Animation” functionality.

See also

●

These properties are available only when you select the “Preview: Support client animations and overlay of native elements” option in the Visualization Manager.

You can animate a movement, a rotation, and the visibility of a visualization element.

“Animation duration” Time that the element executes an animation (in milliseconds)
Example: 500

“Move to the foreground” Property value (BOOL)
TRUE: At runtime, the element is displayed in the foreground.
FALSE: At runtime, the element is displayed in the layer where it was inserted in the visualization editor.

See also

●  Chapter 1.3.5.21.5 “Using client animation” on page 1962

Element property 'Input configuration'
The properties contain the configurations for the user input when using the mouse or keyboard. A user input defines an event and one or more actions that are executed when an event occurs.

The “Configure” button opens the “Input configuration” dialog box for creating or modifying a user input. Configured user inputs are briefly listed below the events. They each include the action that is triggered and the setting in short form.

Example: “Execute ST code”:

```
PLC_PRG.i_x := 0;
```

“OnDialogClosed” Input event: The user closes the dialog box.

“OnMouseClick” Input event: The user clicks the mouse button completely. The mouse button is clicked and released.

“OnMouseDown” Input event: The user clicks down on the mouse button.

“OnMouseEnter” Input event: The user drags the mouse pointer to the element.

“OnMouseLeave” Input event: The user drags the mouse pointer away from the element.

“OnMouseMove” Input event: The user moves the mouse pointer over the element area.

“OnMouseUp” Input event: The user releases the mouse button over the element area.
### "Tap"

When a mouse click event occurs, the variable defined in "Variable" is described in the application. The coding depends on the options "Tap FALSE" and "Tap on enter if captured".

### "Variable"

Variable (BOOL) that is set on mouse click.

**Example:** PLC_PRG.bIsTapped

**TRUE:** A mouse click event exists. It lasts while the user presses the mouse button over the element. It ends when the button is released.

**FALSE:** A mouse click event does not exist.

**Requirement:** The "Tap FALSE" option is not activated.

### "Tap FALSE"

<select>
- **TRUE:** The mouse click event leads to a complementary value in "Variable".
- **FALSE:** A mouse click event does not exist.

### "Tap on enter if captured"

<select>
- **TRUE:** While the mouse click event exists and the mouse pointer is moved over the element area.
- **FALSE:** A mouse click event does not exist. Or the user moves the mouse pointer outside of the element area while the mouse button is pressed.

The value is **TRUE** again as soon as the user moves the pointer back to the element area. The mouse is then captured.

### "Shift"

With the onset of a mouse click event, the variable is set; when the mouse click event is completed, the variable is reset.

### "Variable"

Variable (BOOL). Its value toggled when the mouse click event is ended. This is when the user releases the mouse button while the mouse pointer is over the element area.

If the user releases the mouse button while the mouse pointer is outside of the element area, then the mouse click event is not ended and the value is not toggled.

**Tip:** The user can cancel a started toggle input by dragging the mouse pointer out of the element area.

### "Toggle on up if captured"

<select>
- **TRUE:** The value toggles regardless of where the mouse pointer is when the mouse button is released. The mouse is then captured.

### "Keyboard shortcuts"

Keyboard shortcut on the element for triggering specific input actions.

When the keyboard shortcut event occurs, the input actions in the "Event(s)" property are triggered. In this way, it is not the input action itself that leads to this input action, but the mouse input action.

### "Key"

Key pressed for input action.

**Example:** [T]

**Note:** The following properties appear when a key is selected.
“Event(s)”
- “None”
- “Mouse down”: Pressing the key triggers the input actions that are configured in the “OnMouseDown” property.
- “Mouse up”: Releasing the key triggers the input actions that are configured in the “OnMouseUp” property.
- “Mouse down/up”: Pressing and releasing the key triggers the input actions that are configured in the “OnMouseDown” property and the “OnMouseUp” property.

“Shift”
- ✔ Combination with the Shift key
  Example: [Shift]+[T].

“Control”
- ✔ Combination with the Ctrl key
  Example: [Ctrl]+[T].

“Alt”
- ✔ Combination with the Alt key
  Example: [Alt]+[T].

---

All keyboard shortcuts and their actions that are configured in the visualization are listed in the “Keyboard configuration” tab.

See also
- ☰ Chapter 1.3.5.19.2.2 “Command 'Keyboard Configuration’” on page 1558
- ☰ Chapter 1.3.5.19.3.6 “Dialog 'Input Configuration’” on page 1586

Element property 'Access rights'
Requirement: User management is set up for the visualization.

“Access rights”
- Opens the “Access rights” dialog. There you can edit the access privileges for the element.
  Status messages:
  - “Not set, Full rights.”: Access rights for all user groups: “operable”
  - “Rights are set: Limited rights”: Access is restricted for at least one group.

See also
- ☰ Chapter 1.3.5.19.3.1 “Dialog 'Access Rights’” on page 1583

See also
- ☰
  - ☰ Chapter 1.3.5.19.3.6 “Dialog 'Input Configuration’” on page 1586

Visualization element 'Label'
Symbol:

Tag: “Common controls”
The element is used to label visualizations.
**Element properties**

<table>
<thead>
<tr>
<th>“Element name”</th>
<th>Optional</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hint: Assign individual names for elements so that they are found faster in the element list.</td>
<td></td>
</tr>
<tr>
<td>Example: Header_Parameter</td>
<td></td>
</tr>
</tbody>
</table>

| “Type of element” | “Label” |

**Element property 'Texts'**

The property requires a string that can contain a placeholder with a format definition. At runtime, the placeholder is replaced by the current value. Specify the actual value in the “Text variables” property.

This text is entered automatically into the GlobalTextList text list and can be localized there.

<table>
<thead>
<tr>
<th>“Text”</th>
<th>Character string (without single straight quotation marks)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example: Main page %s</td>
<td></td>
</tr>
</tbody>
</table>

See also

- Chapter 1.3.5.18.2 “Placeholders with format definition in the output text” on page 1546

**Element property 'Position'**

The position defines the location and size of the element in the visualization window. These are based on the Cartesian coordinate system. The origin is located at the upper left corner of the window. The positive horizontal x-axis runs to the right. The positive vertical y-axis runs downwards.

<table>
<thead>
<tr>
<th>“X”</th>
<th>X coordinate of the upper left corner of the element</th>
</tr>
</thead>
<tbody>
<tr>
<td>Specified in pixels.</td>
<td></td>
</tr>
<tr>
<td>Example: 10.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>“Y”</th>
<th>Y coordinate of the upper left corner of the element</th>
</tr>
</thead>
<tbody>
<tr>
<td>Specified in pixels.</td>
<td></td>
</tr>
<tr>
<td>Example: 10.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>“Width”</th>
<th>Specified in pixels.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example: 150</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>“Height”</th>
<th>Specified in pixels.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example: 30</td>
<td></td>
</tr>
</tbody>
</table>

You can also change the values by dragging the box symbols (_drag_) to other positions in the editor.

See also

- Chapter 1.3.5.3.2 “Positioning elements, adapting size and layer” on page 1113

**Element property 'Center'**

The properties contain fixed values for the coordinates of the point of rotation. This point of rotation is shown as the _rotate_ symbol. The point is used as the center for rotating and scaling.
"X"  X-coordinate of the point of rotation

"Y"  Y-coordinate of the point of rotation

You can also change the values by dragging the symbols (🔗) to other positions in the editor.

Element property 'Text properties'

The properties contain fixed values for the text properties.

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Horizontal alignment&quot;</td>
<td>Horizontal alignment of the text within the element.</td>
</tr>
<tr>
<td>&quot;Vertical alignment&quot;</td>
<td>Vertical alignment of the text within the element.</td>
</tr>
<tr>
<td>&quot;Text format&quot;</td>
<td>Definition for displaying texts that are too long</td>
</tr>
<tr>
<td></td>
<td>- &quot;Default&quot;: The long text is truncated.</td>
</tr>
<tr>
<td></td>
<td>- &quot;Line break&quot;: The text is split into parts.</td>
</tr>
<tr>
<td></td>
<td>- &quot;Ellipsis&quot;: The visible text ends with &quot;...&quot; indicating that it is not complete.</td>
</tr>
<tr>
<td>&quot;Font&quot;</td>
<td>Example: &quot;Default&quot;</td>
</tr>
<tr>
<td></td>
<td>▶: The &quot;Font&quot; dialog box opens.</td>
</tr>
<tr>
<td></td>
<td>▼: Drop-down list with style fonts.</td>
</tr>
<tr>
<td>&quot;Font color&quot;</td>
<td>Example: &quot;Black&quot;</td>
</tr>
<tr>
<td></td>
<td>▶: The &quot;Color&quot; dialog box opens.</td>
</tr>
<tr>
<td></td>
<td>▼: Drop-down list with style colors.</td>
</tr>
<tr>
<td>&quot;Transparency&quot;</td>
<td>Whole number (value range from 0 to 255). This determines the transparency of</td>
</tr>
<tr>
<td></td>
<td>the respective color.</td>
</tr>
<tr>
<td></td>
<td>Example: 255: The color is opaque.</td>
</tr>
<tr>
<td></td>
<td>0: The color is completely transparent.</td>
</tr>
<tr>
<td></td>
<td>Please note: If the color is a style color and already has a transparency value, then this property is write-protected.</td>
</tr>
</tbody>
</table>

Element property 'Absolute movement'

The properties contain IEC variables for controlling the position of the element dynamically. The reference point is the upper left corner of the element. In runtime mode, the entire element is moved.

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Movement&quot;</td>
<td></td>
</tr>
<tr>
<td>&quot;X&quot;</td>
<td>Variable (numeric data type). Defines the X position (in pixels).</td>
</tr>
<tr>
<td></td>
<td>Example: PLC_PRG.iPos_X.</td>
</tr>
<tr>
<td></td>
<td>Increasing this value in runtime mode moves the element to the right.</td>
</tr>
<tr>
<td>&quot;Y&quot;</td>
<td>Variable (numeric data type). Defines the Y position (in pixels).</td>
</tr>
<tr>
<td></td>
<td>Example: PLC_PRG.iPos_Y.</td>
</tr>
<tr>
<td></td>
<td>Increasing this value in runtime mode moves the element downwards.</td>
</tr>
</tbody>
</table>
### Rotation

Variable (numeric data type). Defines the angle of rotation (in degrees).

**Example:** PLC_PRG.iAngle1.

The midpoint of the element rotates at the “Center” point. This rotation point is shown as the symbol. In runtime mode, the alignment of the element remains the same with respect to the coordinate system of the visualization. Increasing the value rotates the element to the right.

### Interior rotation

Variable (numeric data type). Defines the angle of rotation (in degrees).

**Example:** PLC_PRG.iAngle2.

In runtime mode, the element rotates about the point of rotation specified in “Center” according to the value of the variable. In addition, the alignment of the element rotates according to the coordinate system of the visualization. Increasing the value in the code rotates clockwise.

The rotation point is shown as the symbol.

Note: If a static angle of rotation is specified in the “Position ➔ Angle” property, then the static angle of rotation is added to the variable angle of rotation (offset) when the visualization is executed.

---

You can link the variables to a unit conversion.

The “X”, “Y”, “Rotation”, and “Interior rotation” properties are supported by the “Client Animation” functionality.

See also

- **Unit conversion**

---

**Element property 'State variables'**

The variables control the element behavior dynamically.

<table>
<thead>
<tr>
<th><strong>Invisible</strong></th>
<th>Variable (BOOL). Toggles the visibility of the element.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>TRUE</strong></td>
<td>The element is not visible at runtime.</td>
</tr>
</tbody>
</table>

The “Invisible” property is supported by the “Client Animation” functionality.
These properties are available only when you select the “Preview: Support client animations and overlay of native elements” option in the Visualization Manager.

You can animate a movement, a rotation, and the visibility of a visualization element.

<table>
<thead>
<tr>
<th><strong>Property</strong></th>
<th>Description</th>
</tr>
</thead>
</table>
| **Animation duration** | Time that the element executes an animation (in milliseconds)  
Example: 500 |
| **Move to the foreground** | Property value (BOOL)  
TRUE: At runtime, the element is displayed in the foreground.  
FALSE: At runtime, the element is displayed in the layer where it was inserted in the visualization editor. |

See also

● Chapter 1.3.5.21.5 “Using client animation” on page 1962

Requirement: User management is set up for the visualization.

<table>
<thead>
<tr>
<th><strong>Property</strong></th>
<th>Description</th>
</tr>
</thead>
</table>
| **Access rights** | Opens the “Access rights” dialog. There you can edit the access privileges for the element.  
Status messages:  
● “Not set. Full rights.”: Access rights for all user groups: “operable”  
● “Rights are set: Limited rights”: Access is restricted for at least one group. |

See also

● Chapter 1.3.5.19.3.1 “Dialog ‘Access Rights’” on page 1583

Visualization element ‘Combo box integer’

Symbol:

Tag: “Common controls”

The element shows values as a drop-down list. When the user clicks an entry, the ID of the entry is written to an integer variable. The entries in the drop-down list can be from a list and contain images from an image pool.

Element properties
<table>
<thead>
<tr>
<th>&quot;Element name&quot;</th>
<th>Example: List of product numbers</th>
</tr>
</thead>
<tbody>
<tr>
<td>Optional</td>
<td>Hint: Assign individual names for elements so that they are found faster in the element list.</td>
</tr>
<tr>
<td>&quot;Type of element&quot;</td>
<td>&quot;Combo box integer&quot;</td>
</tr>
</tbody>
</table>

Element property 'Position' The position defines the location and size of the element in the visualization window. These are based on the Cartesian coordinate system. The origin is located at the upper left corner of the window. The positive horizontal x-axis runs to the right. The positive vertical y-axis runs downwards.

<table>
<thead>
<tr>
<th>“X”</th>
<th>X coordinate of the upper left corner of the element</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Specified in pixels.</td>
</tr>
<tr>
<td></td>
<td>Example: 10.</td>
</tr>
<tr>
<td>“Y”</td>
<td>Y coordinate of the upper left corner of the element</td>
</tr>
<tr>
<td></td>
<td>Specified in pixels.</td>
</tr>
<tr>
<td></td>
<td>Example: 10.</td>
</tr>
<tr>
<td>“Width”</td>
<td>Specified in pixels.</td>
</tr>
<tr>
<td></td>
<td>Example: 150</td>
</tr>
<tr>
<td>“Height”</td>
<td>Specified in pixels.</td>
</tr>
<tr>
<td></td>
<td>Example: 30</td>
</tr>
</tbody>
</table>

You can also change the values by dragging the box symbols (Dragging) to other positions in the editor.

See also
- Chapter 1.3.5.3.2 “Positioning elements, adapting size and layer” on page 1113

Element property ‘Center’ The properties contain fixed values for the coordinates of the point of rotation. This point of rotation is shown as the symbol. The point is used as the center for rotating and scaling.

<table>
<thead>
<tr>
<th>“X”</th>
<th>X-coordinate of the point of rotation</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Y”</td>
<td>Y-coordinate of the point of rotation</td>
</tr>
</tbody>
</table>

You can also change the values by dragging the symbols (Drag) to other positions in the editor.
### “Variable”

At runtime, saves the text list ID of the list entry that the user clicks. If only one image pool is displayed, then the image ID is saved.

**Property value**

- **Variable (integer data type)**
  
  *Example:* `PLC_PRG.iIDComboboxEntry`

- **Enumeration variable with text list support**
  
  *Example:* `PLC_PRG.eMyCombobox<COMBO>`

### “Text list”

Displayed as combo box. Every text list entry becomes a combo box entry.

**Transfer value**

- **Text list identifier as string**
  
  *Example:* `'TextList_A'`

**Note:** The IDs of the text list must be lie within the value range of DWORD or DINT.

- **Empty**
  
  - When an enumeration variable with text list support is specified in the “Variable” property
  
  - When only one image pool is visualized

### “Image pool”

Displayed as combo box. Every image in the image pool becomes a combo box entry.

*Example:* `'ImagePool_A'`

---

**See also**

- **Enumerations**
- © Chapter 1.3.5.6 “Setting up multiple languages” on page 1143

---

**Element property ‘Settings of the list’**

Displayed list that expands when a visualization user clicks into the element.

<table>
<thead>
<tr>
<th><strong>Number of rows setting</strong></th>
<th><strong>From style</strong>: “Explicit”. Then the “Number of visible rows” property appears below it.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Number of visible rows</strong></td>
<td>Number of visible lines of the combo box drop-down list defined here</td>
</tr>
<tr>
<td></td>
<td>- Integer literal</td>
</tr>
<tr>
<td></td>
<td><em>Example:</em> 5</td>
</tr>
<tr>
<td></td>
<td>- Variable (integer data type)</td>
</tr>
<tr>
<td></td>
<td><em>Example:</em> <code>PLC_PRG.iNumberOfVisibleRows</code></td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> The property is available when the “Number of rows setting” property is set to “Explicit”.</td>
</tr>
<tr>
<td><strong>Row height</strong></td>
<td><strong>From style</strong>:</td>
</tr>
<tr>
<td></td>
<td>- Literal</td>
</tr>
<tr>
<td></td>
<td><em>Example:</em> 20</td>
</tr>
<tr>
<td><strong>Height of image</strong></td>
<td>Image height (in pixels) of the image displayed in the drop-down list entry</td>
</tr>
<tr>
<td></td>
<td>- “From style”:</td>
</tr>
<tr>
<td></td>
<td>- Integer literal</td>
</tr>
<tr>
<td></td>
<td><em>Example:</em> 30</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> Images are displayed only when a value is specified in the “Image pool” property.</td>
</tr>
</tbody>
</table>
### "Width of image"

Image width (in pixels) of the image displayed in the drop-down list entry

- **"From style"**:
  - Literal
    - Example: 30

Note: Images are displayed only when a value is specified in the “Image pool” property.

### "Offset of image"

Makes the images in the selection list appear offset (in pixels) from the left margin. An offset of 0 means that the images are displayed directly on the margin.

- **"From style"**:
  - Literal
    - Example: 4

Note: Images are displayed only when a value is specified in the “Image pool” property.

### "Scrollbar size"

Size of the scrollbar (in pixels). The scrollbar is displayed when more entries are specified in the drop-down list than in “Number of visible rows”.

Default: 20

---

**Element property 'Texts'**

### "Tooltip"

Character string (without single straight quotation marks) that is displayed as the tooltip of an element in runtime mode

Example: Products of customer A

Hint: The text is accepted automatically into the “GlobalTextList” text list and can be localized there.

---

**Element property 'Value range'**

### "Limit valuerange"

Limits the text list to one subrange. This subrange is displayed by the combobox.

Requirement: A value is specified in the “Text list” property.

- : Only the subrange that is defined by the “Minimum value” “Maximum value” properties is displayed as a drop-down list.

### "Minimum value"

ID of the text list entry from which a combobox entry is displayed

- Literal ($ANY_NUM$)
  - Example: 5
- Variable (integer data type)
  - Example: PLC_PRG.iFirstEntry
### Maximum value
ID of the text list entry up to which combo box entries are displayed

- **Literal (ANY_NUM)**
  - Example: 10
- **Variable (integer data type)**
  - Example: PLC_PRG.iLastEntry

### Filter missing text entries
☑: Text list is refreshed and any unused texts (IDs) are removed.

**Requirement:** A value is specified in the “Text list” property.

---

**Element property 'Text properties'**

The properties contain fixed values for the text properties.

| “Usage of” | “Default style values”: The values of the visualization style are used.  
|            | “Individual settings”: The "Individual text properties" property group is shown  
|            | The values can be customized here. |
| "Individual text properties" | Requirement: The “Individual settings” text property is defined. |
| “Horizontal alignment” | Horizontal alignment of the text within the element. |
| “Vertical alignment” | Vertical alignment of the text within the element. |
| “Font” | Example: “Default”  
|        | ☐: The “Font” dialog box opens.  
|        | ▼: Drop-down list with style fonts. |
| “Font color” | Example: “Black”  
|        | ☐: The “Color” dialog box opens.  
|        | ▼: Drop-down list with style colors. |
| “Transparency” | Whole number (value range from 0 to 255). This determines the transparency of the respective color.  
|      | Example: 255: The color is opaque.  
|      | 0: The color is completely transparent.  
|      | Please note: If the color is a style color and already has a transparency value, then this property is write-protected. |

---

**Element property 'Absolute movement'**

The properties contain IEC variables for controlling the position of the element dynamically. The reference point is the upper left corner of the element. In runtime mode, the entire element is moved.

| “Movement” |  |
| "X" | Variable (numeric data type). Defines the X position (in pixels).  
|      | Example: PLC_PRG.iPos_X.  
|      | Increasing this value in runtime mode moves the element to the right. |
| "Y" | Variable (numeric data type). Defines the Y position (in pixels).  
|      | Example: PLC_PRG.iPos_Y.  
|      | Increasing this value in runtime mode moves the element downwards. |
### "Rotation"
Variable (numeric data type). Defines the angle of rotation (in degrees).

**Example:** PLC_PRG.iAngle1.

The midpoint of the element rotates at the "Center" point. This rotation point is shown as the symbol.

In runtime mode, the alignment of the element remains the same with respect to the coordinate system of the visualization. Increasing the value rotates the element to the right.

### "Interior rotation"
Variable (numeric data type). Defines the angle of rotation (in degrees).

**Example:** PLC_PRG.iAngle2.

In runtime mode, the element rotates about the point of rotation specified in "Center" according to the value of the variable. In addition, the alignment of the element rotates according to the coordinate system of the visualization. Increasing the value in the code rotates clockwise.

The rotation point is shown as the symbol.

Note: If a static angle of rotation is specified in the "Position Angle" property, then the static angle of rotation is added to the variable angle of rotation (offset) when the visualization is executed.

---

You can link the variables to a unit conversion.

The "X", "Y", "Rotation", and "Interior rotation" properties are supported by the "Client Animation" functionality.

See also
- Unit conversion

---

**Element property 'State variables'**

The variables control the element behavior dynamically.

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Invisible&quot;</td>
<td>Variable (BOOL). Toggles the visibility of the element. TRUE: The element is not visible at runtime.</td>
</tr>
<tr>
<td>&quot;Deactivate inputs&quot;</td>
<td>Variable (BOOL). Toggles the operability of the element. TRUE: User inputs do not have any effect in runtime more. The element is shown as deactivated.</td>
</tr>
</tbody>
</table>
The “Invisible” property is supported by the “Client Animation” functionality.

See also

- These properties are available only when you select the “Preview: Support client animations and overlay of native elements” option in the Visualization Manager. You can animate a movement, a rotation, and the visibility of a visualization element.

<table>
<thead>
<tr>
<th>“Animation duration”</th>
<th>Time that the element executes an animation (in milliseconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Example: 500</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>“Move to the foreground”</th>
<th>Property value (BOOL)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>TRUE: At runtime, the element is displayed in the foreground.</td>
</tr>
<tr>
<td></td>
<td>FALSE: At runtime, the element is displayed in the layer where it was inserted in the visualization editor.</td>
</tr>
</tbody>
</table>

See also

- Chapter 1.3.5.21.5 “Using client animation” on page 1962

Requirement: User management is set up for the visualization.

<table>
<thead>
<tr>
<th>“Access rights”</th>
<th>Opens the “Access rights” dialog. There you can edit the access privileges for the element.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Status messages:</td>
<td>• “Not set. Full rights.”: Access rights for all user groups: “operable”</td>
</tr>
<tr>
<td></td>
<td>• “Rights are set: Limited rights”: Access is restricted for at least one group.</td>
</tr>
</tbody>
</table>

See also

- Chapter 1.3.5.19.3.1 “Dialog 'Access Rights’” on page 1583

See also

- 

Visualization element ‘Combo box array’

Symbol:

Tag: “Common controls”

The element shows values of an array as a drop-down list. When the visualization user clicks an entry, the array index of the entry is written to an integer variable.
Element properties

<table>
<thead>
<tr>
<th>“Element name”</th>
<th>Optional</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hint: Assign individual names for elements so that they are found faster in the element list.</td>
<td></td>
</tr>
<tr>
<td>Example: List_Product_Number</td>
<td></td>
</tr>
</tbody>
</table>

| “Type of element” | “Combo box array” |

The position defines the location and size of the element in the visualization window. These are based on the Cartesian coordinate system. The origin is located at the upper left corner of the window. The positive horizontal x-axis runs to the right. The positive vertical y-axis runs downwards.

| “X” | X coordinate of the upper left corner of the element |
| Specified in pixels. |
| Example: 10 |

| “Y” | Y coordinate of the upper left corner of the element |
| Specified in pixels. |
| Example: 10 |

| “Width” | Specified in pixels. |
| Example: 150 |

| “Height” | Specified in pixels. |
| Example: 30 |

You can also change the values by dragging the box symbols (строенное) to other positions in the editor.

See also
● § Chapter 1.3.5.3.2 “Positioning elements, adapting size and layer” on page 1113

Element property 'Center'

The properties contain fixed values for the coordinates of the point of rotation. This point of rotation is shown as the symbol. The point is used as the center for rotating and scaling.

| “X” | X-coordinate of the point of rotation |
| “Y” | Y-coordinate of the point of rotation |

You can also change the values by dragging the symbols (строено) to other positions in the editor.
**Variable**

At runtime, saves the array index of the list entry that the user clicks.

**Property value**

- Variable (integer data type)
  
  Example: PLC_PRG.iIndexComboBoxEntry

- Enumeration variable with text list support
  
  Example: PLC_PRG.eMyComboBox<COMBO>

  Note: Value range of the enumeration value that lies within the DWORD or DINT value range

**Data array**

Displayed as combo box. Every array component becomes a combo box entry.

**Property value**

- Array variable (ARRAY[...] OF)
  
  Example: PLC_PRG.astrComboBox


See also

- Enumerations
- Chapter 1.3.5.6 “Setting up multiple languages” on page 1143

**Element property 'Columns'**

The "ComboBox – Array" element visualizes an array variable or structure variable in a tabular view. The index of array elements or structure members is shown in a column or row. Two-dimensional arrays or structure arrays are shown in several columns. You specify the visualized variable in the "Data array" property. If a variable is assigned there, then you can specify the display of the table columns where the array elements are shown. You can customize each column that is assigned to an index [<n>].

**Columns**

- [<n>]

  Due to the structure of the variable that is defined in "Data array", CODESYS determines the number of columns and defines them with the index <n>.

  Example: StringTable : ARRAY [0..2, 0..4] OF STRING := ['BMW', 'Audi', 'Mercedes', 'VW', 'Fiat', '150', '150', '150', '150', '100', 'blue', 'gray', 'silver', 'blue', 'red'];: three columns are formed [0], [1] and [2].

**Max. array index**

Optional. Variable (integer data type) or value. Defines up to which array index the data is displayed.

**Row height**

Height of the rows (in pixels).

**Number visible rows**

Optional. If the array is larger than the number of visible rows, then a scrollbar is included.

**Scrollbar size**

Width of the vertical scrollbar (in pixels).

Table 259: “Element property 'Columns: Column [<n>]’”

<table>
<thead>
<tr>
<th>“Width”</th>
<th>Column width (in pixels).</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Image column”</td>
<td>Images can be displayed in the column. Images are used from the global image pool or user-defined image pools. The image IDs are shown in the cells of the table as defined in the image pool.</td>
</tr>
<tr>
<td>“Image configuration”</td>
<td></td>
</tr>
</tbody>
</table>
### “Fill mode”
- **“Fill cell”**
  The image resizes to the dimensions of the cell without fixing the height/width ratio.
- **“Centered”**
  The image is centered in the cell and retains its proportions (height-width ratio).

### “Transparency”
[✔]: The color that is specified in “Transparent color” is displayed as transparent.

### “Transparent color”
When the “Transparent” property is enabled, the color specified here is not displayed. Pixels with this color are transparent.

### “Text alignment in column”
- **“Left”**
- **“Centered”**
- **“Right”**

## Element property ‘Texts’

### “Tooltip”
Character string (without single straight quotation marks) that is displayed as the tooltip of an element in runtime mode

**Example:** Products of customer A

**Hint:** The text is accepted automatically into the “GlobalTextList” text list and can be localized there.

### See also

## Element property 'Text properties'

The properties contain fixed values for the text properties.

### “Usage of”
- **“Default style values”**: The values of the visualization style are used.
- **“Individual settings”**: The “Individual text properties” property group is shown
  The values can be customized here.

### “Individual text properties”
Requirement: The “Individual settings” text property is defined.

### “Font”
**Example:** “Default”

![Font dialog box opens.](image)

- Drop-down list with style fonts.

### “Font color”
**Example:** “Black”

![Color dialog box opens.](image)

- Drop-down list with style colors.

### “Transparency”
Whole number (value range from 0 to 255). This determines the transparency of the respective color.

**Example:** 255: The color is opaque.

0: The color is completely transparent.

Please note: If the color is a style color and already has a transparency value, then this property is write-protected.
Element property 'Absolute movement'  
The properties contain IEC variables for controlling the position of the element dynamically. The reference point is the upper left corner of the element. In runtime mode, the entire element is moved.

## "Movement"

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>X</strong></td>
<td>Variable (numeric data type). Defines the X position (in pixels).</td>
<td>PLC_PRG.iPos_X.</td>
</tr>
<tr>
<td></td>
<td>Increasing this value in runtime mode moves the element to the right.</td>
<td></td>
</tr>
<tr>
<td><strong>Y</strong></td>
<td>Variable (numeric data type). Defines the Y position (in pixels).</td>
<td>PLC_PRG.iPos_Y.</td>
</tr>
<tr>
<td></td>
<td>Increasing this value in runtime mode moves the element downwards.</td>
<td></td>
</tr>
</tbody>
</table>

## "Rotation"

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Rotation</strong></td>
<td>Variable (numeric data type). Defines the angle of rotation (in degrees).</td>
<td>PLC_PRG.iAngle1.</td>
</tr>
<tr>
<td></td>
<td>The midpoint of the element rotates at the &quot;Center&quot; point. This rotation point is shown as the symbol.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>In runtime mode, the alignment of the element remains the same with respect to the coordinate system of the visualization. Increasing the value rotates the element to the right.</td>
<td></td>
</tr>
</tbody>
</table>

## "Interior rotation"

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Interior rotation</strong></td>
<td>Variable (numeric data type). Defines the angle of rotation (in degrees).</td>
<td>PLC_PRG.iAngle2.</td>
</tr>
<tr>
<td></td>
<td>In runtime mode, the element rotates about the point of rotation specified in &quot;Center&quot; according to the value of the variable. In addition, the alignment of the element rotates according to the coordinate system of the visualization. Increasing the value in the code rotates clockwise.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>The rotation point is shown as the symbol.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Note: If a static angle of rotation is specified in the &quot;Position ➔ Angle&quot; property, then the static angle of rotation is added to the variable angle of rotation (offset) when the visualization is executed.</td>
<td></td>
</tr>
</tbody>
</table>

You can link the variables to a unit conversion.

The "X", "Y", "Rotation", and "Interior rotation" properties are supported by the "Client Animation" functionality.
See also

- Unit conversion

Element property 'State variables'
The variables control the element behavior dynamically.

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Invisible&quot;</td>
<td>Variable (BOOL). Toggles the visibility of the element. TRUE: The element is not visible at runtime.</td>
</tr>
<tr>
<td>&quot;Deactivate inputs&quot;</td>
<td>Variable (BOOL). Toggles the operability of the element. TRUE: User inputs do not have any effect in runtime more. The element is shown as deactivated.</td>
</tr>
</tbody>
</table>

The "Invisible" property is supported by the "Client Animation" functionality.

See also

These properties are available only when you select the "Preview: Support client animations and overlay of native elements" option in the Visualization Manager.

You can animate a movement, a rotation, and the visibility of a visualization element.

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Animation duration&quot;</td>
<td>Time that the element executes an animation (in milliseconds) Example: 500</td>
</tr>
<tr>
<td>&quot;Move to the foreground&quot;</td>
<td>Property value (BOOL) TRUE: At runtime, the element is displayed in the foreground. FALSE: At runtime, the element is displayed in the layer where it was inserted in the visualization editor.</td>
</tr>
</tbody>
</table>

See also

- Chapter 1.3.5.21.5 “Using client animation” on page 1962

Visualization element 'Tab control'
Symbol:

Tag: “Common controls”
The element displays selected visualizations in tabs. The tabs can be used by means of the tab header without having to configure an input configuration. A visualization user switches between visualizations by clicking the tab header.

### Element properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Example</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Element name&quot;</td>
<td>Assembly A</td>
<td>Required, Assign individual names for elements so that they are found faster in the element list.</td>
</tr>
<tr>
<td>&quot;Type of element&quot;</td>
<td>&quot;Tab control&quot;</td>
<td></td>
</tr>
<tr>
<td>&quot;Tab width&quot;</td>
<td>Width of the tab (in pixels). If there is not space for all tab headers, then a scrollbar is added. Example: 30</td>
<td></td>
</tr>
<tr>
<td>&quot;Tab height&quot;</td>
<td>Height of the tab (in pixels)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Integer literal</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• &quot;From style&quot;</td>
</tr>
<tr>
<td>&quot;Scaling type&quot;</td>
<td>The method with which the height and width of the referenced visualization are scaled.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• &quot;Isotropic&quot;: The visualization is scaled to the size of the element. The visualization retains its proportions with a fixed height/width ratio.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• &quot;Anisotropic&quot;: The visualization is scaled to the size of the element. The height and width are adapted to the element independently of each other.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• &quot;Fixed&quot;: The visualization is displayed in its original size without taking into account the size of the element.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• &quot;Fixed and scrollable&quot;: The visualization is displayed fixed in the element. If it is larger than the element, the element will be provided with scrollbars.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Please note: assign variables to the properties &quot;Scroll position variable horizontal&quot; or &quot;Scroll position variable vertical&quot;. You can then edit the data of the scrollbar position in the application.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>&quot;Deactivate background drawing&quot;</th>
<th>The non-animated elements of the referenced visualization are displayed as background images in order to optimize the performance of the visualization. Result: At runtime, the elements can be displayed in any order, for example when an element moves behind the frame at runtime.</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Deactivate background drawing&quot;</td>
<td>□: Deactivates the background display in order to prevent the behavior described above The property is not available for the following settings:</td>
</tr>
<tr>
<td></td>
<td>• The &quot;Scaling type&quot; property is set to &quot;Fixed and scrollable&quot;</td>
</tr>
<tr>
<td></td>
<td>• The client animation functionality is enabled.</td>
</tr>
</tbody>
</table>

See also

- The properties include variables for the position of the scroll boxes in the scrollbars. You can process the data for the scroll box position in the application.
**Requirement:** The “Scaling type” property is “Fixed and scrollable”.

<table>
<thead>
<tr>
<th>Scaling type property</th>
<th>Description</th>
</tr>
</thead>
</table>
| “Scroll position variable horizontal”                      | Variable (integer data type, also array). Includes the position of the horizontal or vertical scroll box. The array contains the position for each display variant. If the visualization is running on multiple display variants, then the position changes are disconnected from each other. Example:   
  PLC_PRG.iScrollHor[CURRENTCLIENTID]  
  PLC_PRG.iScrollVer[CURRENTCLIENTID]  
  In this example, the variable is declared as an array:  
  iScrollHor: ARRAY[0..20] OF INT;  
  iScrollVer: ARRAY[0..20] OF INT;  
  CURRENTCLIENTID indicates the current display variant. |
| “Scroll position variable, vertical”                       |                                                                                                  |

See also

- Unit conversion

**Element property 'References'**

<table>
<thead>
<tr>
<th>References</th>
<th>Description</th>
</tr>
</thead>
</table>
| Clicking “Configure” opens the “Frame Configuration” dialog. You can select an existing visualization there.  
  Selected visualization references are shown in the properties.  
  Selected visualization references are listed here as subordinate properties. |
| Name pf the visualization reference (example: PLC_PRG.S1)                                      |                                                                                               |

<table>
<thead>
<tr>
<th>Heading</th>
<th>Tab caption (example: Panel)</th>
</tr>
</thead>
</table>
| “Image ID”                                                                                      | Image ID in the theme <image pool name>.<ID>  
  Example: Imagepool_A.1 for the image with ID 1 in Imagepool_A  
  Interface parameter of the visualization reference  
  Example: iX  
  If the visualization has an interface, then their parameters are displayed here as subordinate properties.  
  Variable (data type conforms to data type of the interface parameter). Includes the initialization value for the instantiation of the visualization. |

See also

- Chapter 1.3.5.19.2.1 “Command 'Interface Editor” on page 1556  
- Chapter 1.3.5.19.2.9 “Command 'Frame Selection” on page 1564

**Element property 'Position'**

The position defines the location and size of the element in the visualization window. These are based on the Cartesian coordinate system. The origin is located at the upper left corner of the window. The positive horizontal x-axis runs to the right. The positive vertical y-axis runs downwards.
<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>“X”</td>
<td>X coordinate of the upper left corner of the element</td>
</tr>
<tr>
<td></td>
<td>Specified in pixels.</td>
</tr>
<tr>
<td></td>
<td>Example: 10.</td>
</tr>
<tr>
<td>“Y”</td>
<td>Y coordinate of the upper left corner of the element</td>
</tr>
<tr>
<td></td>
<td>Specified in pixels.</td>
</tr>
<tr>
<td></td>
<td>Example: 10.</td>
</tr>
<tr>
<td>“Width”</td>
<td>Specified in pixels.</td>
</tr>
<tr>
<td></td>
<td>Example: 150</td>
</tr>
<tr>
<td>“Height”</td>
<td>Specified in pixels.</td>
</tr>
<tr>
<td></td>
<td>Example: 30</td>
</tr>
</tbody>
</table>

You can also change the values by dragging the box symbols (⬇) to other positions in the editor.

See also
- Chapter 1.3.5.3.2 “Positioning elements, adapting size and layer” on page 1113

### Element property ‘Center’

The properties contain fixed values for the coordinates of the point of rotation. This point of rotation is shown as the ⚙ symbol. The point is used as the center for rotating and scaling.

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>“X”</td>
<td>X-coordinate of the point of rotation</td>
</tr>
<tr>
<td>“Y”</td>
<td>Y-coordinate of the point of rotation</td>
</tr>
</tbody>
</table>

You can also change the values by dragging the symbols (⬇) to other positions in the editor.

### Element property ‘Switch frame variable’

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Variable”</td>
<td>Variable (integer data type). Specifies the index of the active visualization.</td>
</tr>
<tr>
<td></td>
<td>Example: PLC_PRG.uiActiveVisuID.</td>
</tr>
<tr>
<td></td>
<td>Tip: The “Frame Configuration” dialog box includes a list of selected visualizations. The visualizations are ordered automatically in numeric order in the list.</td>
</tr>
</tbody>
</table>

See also
- Chapter 1.3.5.19.2.9 “Command ‘Frame Selection’” on page 1564

### Element property ‘Absolute movement’

The properties contain IEC variables for controlling the position of the element dynamically. The reference point is the upper left corner of the element. In runtime mode, the entire element is moved.
**“Movement”**

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
<th>Example</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>“X”</strong></td>
<td>Variable (numeric data type). Defines the X position (in pixels).</td>
<td>PLC_PRG.iPos_X</td>
<td>Increasing this value in runtime mode moves the element to the right.</td>
</tr>
<tr>
<td><strong>“Y”</strong></td>
<td>Variable (numeric data type). Defines the Y position (in pixels).</td>
<td>PLC_PRG.iPos_Y</td>
<td>Increasing this value in runtime mode moves the element downwards.</td>
</tr>
<tr>
<td><strong>“Rotation”</strong></td>
<td>Variable (numeric data type). Defines the angle of rotation (in degrees).</td>
<td>PLC_PRG.iAngle1</td>
<td>The midpoint of the element rotates at the “Center” point. This rotation point is shown as the symbol. In runtime mode, the alignment of the element remains the same with respect to the coordinate system of the visualization. Increasing the value rotates the element to the right.</td>
</tr>
<tr>
<td><strong>“Interior rotation”</strong></td>
<td>Variable (numeric data type). Defines the angle of rotation (in degrees).</td>
<td>PLC_PRG.iAngle2</td>
<td>In runtime mode, the element rotates about the point of rotation specified in “Center” according to the value of the variable. In addition, the alignment of the element rotates according to the coordinate system of the visualization. Increasing the value in the code rotates clockwise. The rotation point is shown as the symbol. Note: If a static angle of rotation is specified in the “Position ➔ Angle” property, then the static angle of rotation is added to the variable angle of rotation (offset) when the visualization is executed.</td>
</tr>
</tbody>
</table>

You can link the variables to a unit conversion.

The “X”, “Y”, “Rotation”, and “Interior rotation” properties are supported by the “Client Animation” functionality.

See also
- Unit conversion
**Element properties 'State variables'**

The variables control the element behavior dynamically.

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Invisible&quot;</td>
<td>Variable (BOOL). Toggles the visibility of the element. TRUE: The element is not visible at runtime.</td>
</tr>
<tr>
<td>&quot;Deactivate inputs&quot;</td>
<td>Variable (BOOL). Toggles the operability of the element. TRUE: User inputs do not have any effect in runtime more. The element is shown as deactivated.</td>
</tr>
</tbody>
</table>

The "Invisible" property is supported by the "Client Animation" functionality.

See also

- 

These properties are available only when you select the “Preview: Support client animations and overlay of native elements” option in the Visualization Manager.

You can animate a movement, a rotation, and the visibility of a visualization element.

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Animation duration&quot;</td>
<td>Time that the element executes an animation (in milliseconds)</td>
</tr>
<tr>
<td>Example: 500</td>
<td></td>
</tr>
<tr>
<td>&quot;Move to the foreground&quot;</td>
<td>Property value (BOOL)</td>
</tr>
<tr>
<td>TRUE: At runtime, the element is displayed in the foreground.</td>
<td></td>
</tr>
<tr>
<td>FALSE: At runtime, the element is displayed in the layer where it was inserted in the visualization editor.</td>
<td></td>
</tr>
</tbody>
</table>

See also

- ☞ Chapter 1.3.5.21.5 “Using client animation” on page 1962

See also

- 

**Visualization element 'Button'**

Symbol:

![Symbol](image)

Tag: “Common controls”

This element triggers an action, such as setting a variable.

**Element properties**
**Element property 'Position'**

The position defines the location and size of the element in the visualization window. These are based on the Cartesian coordinate system. The origin is located at the upper left corner of the window. The positive horizontal x-axis runs to the right. The positive vertical y-axis runs downwards.

| “X” | X coordinate of the upper left corner of the element  
Specified in pixels.  
Example: 10. |
| “Y” | Y coordinate of the upper left corner of the element  
Specified in pixels.  
Example: 10. |
| “Width” | Specified in pixels.  
Example: 150 |
| “Height” | Specified in pixels.  
Example: 30 |

You can also change the values by dragging the box symbols (_drag_box_) to other positions in the editor.

See also

- Chapter 1.3.5.3.2 “Positioning elements, adapting size and layer” on page 1113

**Element property 'Center'**

The properties contain fixed values for the coordinates of the point of rotation. This point of rotation is shown as the _circle_ symbol. The point is used as the center for rotating and scaling.

| “X” | X-coordinate of the point of rotation |
| “Y” | Y-coordinate of the point of rotation |

You can also change the values by dragging the symbols ( _circle_ ) to other positions in the editor.

**Element property 'Colors'**

The properties contain fixed values for setting colors.
**“Color”**

Color for the element in its normal state.

Please note that the normal state is in effect if the expression in the “Color variables ➔ Toggle color” property is not defined or it has the value `FALSE`.

**“Alarm color”**

Color for the element in alarm state.

Please note that the alarm state is in effect if the expression in the “Color variables ➔ Toggle color” property has the value `TRUE`.

**“Transparency”**

Value (0 to 255) for defining the transparency of the selected color.

Example: 255: The color is opaque. 0: The color is completely transparent.

**“Use gradient color”**

The element is displayed with a color gradient.

**“Gradient setting”**

The “Color gradient editor” dialog box opens.

See also

- ☝️ Chapter 1.3.5.19.3.5 “Dialog ‘Gradient Editor’” on page 1586

---

**Element property ’Image’**

**“Static ID”**

Reference to an image in an image pool of the format `<name of image pool>.<image ID>` (example: `image_pool.GreenButton`).

If the image is from the “GlobalImagePool”, then you can omit the name of the image pool because CODESYS always searches this pool first.

[...]: The “Input Assistant” dialog box opens and lists all available image pools and images in the entire project.

**“Scale type”**

Behavior of the image when resizing the button.

- **“Isotropic”**: The image retains its proportions. The ratio of height to width is retained, even if you change the height or width of the button separately.
- **“Anisotropic”**: The image resizes to the dimensions of the button.
- **“Fixed”**: The image retains its original size, even if you change the size of the button.

**“Transparency”**

The visualization displays the image with the transparency color that is selected in “Transparency color”.

**“Transparency color”**

Color that is transparent in the image (example: “White”). If the image background that is referenced by “Static ID” is white, then this background is displayed transparent.

Clicking [...] opens a color selection dialog.

Requirement: The “Transparency” option is activated.

**“Horizontal alignment”**

Horizontal alignment of the image

- **“Left”**
- **“Centered”**
- **“Right”**

**“Vertical alignment”**

Vertical alignment of the image

- **“Top”**
- **“Centered”**
- **“Bottom”**

---

**Element property ’Texts’**

The properties contain character strings for labeling the element. The character string can also contain a placeholder with a format definition. In runtime mode, the placeholder is replaced by the current value in the specified format.
CODESYS accepts the specified texts automatically into the “GlobalTextList” text list. Therefore, these texts can be localized.

|“Text”| Character string (without single straight quotation marks) for the labeling the element. Add a line break by pressing the keyboard shortcut [Ctrl] + [Enter]. | Example: Accesses: %i
The variable that contains the current value for the placeholder is specified in the property “Text variable ⇒ Text”. |
|“Tooltip”| Character string (without single straight quotation marks) that is displayed as the tooltip of an element. | Example: Number of valid accesses.
The variable that contains the current value for the placeholder is specified in the property “Text variable ⇒ Tooltip”.

See also
- “Element property ‘Text variables’” on page 1229
- Chapter 1.3.5.18.2 “Placeholders with format definition in the output text” on page 1546

The properties contain fixed values for the text properties.

|“Horizontal alignment”| Horizontal alignment of the text within the element. |
|“Vertical alignment”| Vertical alignment of the text within the element. |
|“Text format”| Definition for displaying texts that are too long | “Default”: The long text is truncated. 
“Line break”: The text is split into parts. 
“Ellipsis”: The visible text ends with “...” indicating that it is not complete. |
▼: Drop-down list with style fonts. |
|“Font color”| Example: “Black” | The “Color” dialog box opens. 
▼: Drop-down list with style colors. |
|“Transparency”| Whole number (value range from 0 to 255). This determines the transparency of the respective color. | Example: 255: The color is opaque. 
0: The color is completely transparent. 
Please note: If the color is a style color and already has a transparency value, then this property is write-protected. |

The properties contain IEC variables for controlling the position of the element dynamically. The reference point is the upper left corner of the element. In runtime mode, the entire element is moved.
### "Movement"

| **X** | Variable (numeric data type). Defines the X position (in pixels).  
**Example:** PLC_PRG.iPos_X.  
Increasing this value in runtime mode moves the element to the right. |
| **Y** | Variable (numeric data type). Defines the Y position (in pixels).  
**Example:** PLC_PRG.iPos_Y.  
Increasing this value in runtime mode moves the element downwards. |
| **Rotation** | Variable (numeric data type). Defines the angle of rotation (in degrees).  
**Example:** PLC_PRG.iAngle1.  
The midpoint of the element rotates at the "Center" point. This rotation point is shown as the + symbol.  
In runtime mode, the alignment of the element remains the same with respect to the coordinate system of the visualization. Increasing the value rotates the element to the right. |
| **Scaling** | Variable (integer data type). Causes centric stretching.  
**Example:** PLC_PRG.iScaling.  
The reference point is the "Center" property.  
The value 1 shrinks the element by a factor of 0.001. The value 1000 returns the element to its original size. |

---

**You can link the variables to a unit conversion.**

**The properties "X", "Y", "Rotation", and "Interior rotation" are supported by the "Client Animation" functionality.**

---

**Element property 'Relative movement'**

The properties contains variables for moving the element. The reference point is the position of the element ("Position" property). The shape of the element can change.

### "Movement top-left"

| **X** | Variable (integer data type). It contains the number (in pixels) that the left edge is moved horizontally. Incrementing the value moves the element to the right.  
**Example:** PLC_PRG.iDeltaX |
<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
</table>
| \(Y\) Variable (integer data type). It contains the number (in pixels) that the top edge is moved vertically. Incrementing the value moves the element to the down.  
Example: PLC_PRG.iDeltaY |
| \(Movement \text{ bottom-right}\) Variable (integer data type). It contains the number (in pixels) that the \(right\) edge is moved horizontally. Incrementing the value moves the element to the right.  
Example: PLC_PRG.iDeltaWidth|
| \(Y\) Variable (integer data type). It contains the number (in pixels) that the bottom edge is moved vertically. Incrementing the value moves the element to the down.  
Example: PLC_PRG.iDeltaHeight |

See also
- “Element property ‘Absolute movement’” on page 1227

**Element property ‘Text variables’**
These properties are variables with contents that replace a format definition.

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
</table>
| \(\text{Text variable}\) Variable (data type compliant with the format definition). It contains what is printed instead of the format definition.  
Example: PLC_PRG.iAccesses |

Note: The format definition is part of the text in the property “Texts \(\rightarrow\) Text”.
Note: If you specify a variable of type enumeration with text list support, then the name of the enumeration data type is added automatically in angle brackets after the variable name. Example: PLC_PRG.enVar <enumeration name>. Then the symbolic value of the enumeration component is printed instead of the numeric value when text is printed. Refer to the help page for the enumerations. |

| \(\text{Tooltip variable}\) Variable (data type compliant with the format definition). It contains what is printed instead of the format definition.  
Example: PLC_PRG.iAccessesInTooltip |

Note: The format definition is part of the text in the property “Texts \(\rightarrow\) Tooltip”. |

See also
- Chapter 1.3.5.18.2 “Placeholders with format definition in the output text” on page 1546
- “Element property ‘Texts’” on page 1226
- Enumerations

**Element property ‘Dynamic texts’**
Dynamic texts are variably indexed texts of a text list. At runtime, the text is displayed that is currently indexed in the variable.
| **Text list** | Variable (string) or name of the text list as a fixed string in single straight quotation marks.  
Example: 'Errorlist'  
▼: Drop-down list with the dialogs available in the text lists. |
| **Text index** | Text list ID. This refers to the desired output text.  
- As fixed string with the ID in single straight quotation marks.  
  Example: '1'  
- As a variable (STRING) for dynamically controlling the text output.  
  Example: strTextID  
  Sample assignment: PLC_PRG.strTextID := '1'; |
| **Tooltip index** | Text list ID. This refers to the desired output text.  
- As fixed string with the ID in single straight quotation marks.  
  Example: '2'  
- As a variable (STRING) for dynamically controlling the text output.  
  Example: strToolTipID  
  Sample assignment: PLC_PRG.strToolTipID := '2'; |

**See also**  
- Text list

**Element property 'Font variables'**  
The variables allow for dynamic control of the text display.

| **Font name** | Variable (STRING). Includes the font of the text.  
Example: PLC_PRG.stFontVar := 'Arial';  
The selection of fonts corresponds to the default “Font” dialog. |
| **Size** | Variable (numeric data type). Contains the font size (in pixels or points). The applied unit is specified in brackets after the variable name.  
- <pt>: Points (default)  
  Example: PLC_PRG.iFontHeight <pt>  
  Code: iFontHeight : INT := 12;  
- <px>: Pixels  
  Example: PLC_PRG.iFontHeight <px>  
  Code: iFontHeight : INT := 19;  

If you click in the value field, a drop-down list opens on the right for setting the unit.  

Hint: The font size is specified in points (example: Arial 12). Use points when the variable font size should match a font, for example if a font is set in the property “Text property ➔ Font”.

---

2020/12/10  
3ADR010583, 1, en_US  
1327
**"Flags"**

Variable (DWORD). Contains the flags for displaying fonts.

Flags:
- 1: Italics
- 2: Bold
- 4: Underline
- 8: Strikethrough

Note: You can combine the font displays by adding the coding of the flags. For example, a bold and underlined text: `PLC_PRG.dwFontType := 6;`

---

**"Character set"**

Variable (DWORD). Contains a character set number for the font.

The selection of character set numbers corresponds to the “Script” setting of the standard “Font” dialog.

---

**"Color"**

Variable (DWORD). Includes the color of the text.

Example: `PLC_PRG.dwColorFont := 16#FF000000;`

---

**"Flags for text alignment"**

Variable (integer data type). Contains the coding for text alignment.

Example: `PLC_PRG.dwTextAlignment.`

Coding:
- 0: Top left
- 1: Horizontal center
- 2: Right
- 4: Vertical center
- 8: Bottom

Note: You can combine the text alignments by adding the coding of the flags. For example, a vertical and horizontal centered text: `PLC_PRG.dwFontType := 5;`

---

*Fixed values for displaying texts are set in "Text properties".*

---

**See also**
- § "Element property 'Text properties'" on page 1227

---

**Element property 'Colorvariables'**

The Element property is used as an interface for project variables to dynamically control colors at runtime.
**“Toggle color”**

The property controls the toggled color at runtime.

**Value assignment:**
- **FALSE:** The element is displayed with the color specified in the “Color” property.
- **TRUE:** The element is displayed with the color specified in the “Alarm color” property.

**Assigning the property:**
- Placeholder for the user input variable
  - `<toggle/tap variable>`
  - `<NOT toggle/tap variable>`

  The color change is not controlled by its own variable, but by a user input variable.

  Note: Specify a variable for the mouse events “Tap” or “Toggle” in the input configuration of the element. Only then is the placeholder set. If you configure a variable in both “Toggle” and “Tap”, then the variable specified in “Tap” is used.

  Hint: Click the symbol to insert the placeholder “<toggle/tap variable>”. When you activate the “Inputconfiguration”, “Tap FALSE” property, then the “<NOT toggle/tap variable>” placeholder is displayed.

  - Instance path of a project variable (BOOL)
    Example: PLC_PRG.xColorIsToggeled

    Note: In the code, declare and implement the variable specified here. Its value assignment determines when the color changes.

**“Color”**

Color variable for the frame

- **Variable (DWORD) for the color**
  Example: PLC_PRG.dwColor

- **Color literal**
  Example of gray and opaque: 16#FF888888

**Requirement:** “Show frame” property is activated.

Please note that the normal state is in effect if the expression in the “Colorvariables ➔ Toggle color” property is not defined or it has the value FALSE.

**“Alarm color”**

Color variable for the frame in alarm state

- **Variable (DWORD) for the alarm color**
  Example: PLC_PRG.dwAlarmColor

- **Color literal**
  Example of red and opaque: 16#FFFF0000

Please note that the alarm state is in effect if the expression in the “Colorvariables ➔ Toggle color” property has the value TRUE.

---

The transparency part of the color value is evaluated only if the “Activate semi-transparent drawing” option of the visualization manager is selected.

Select the “Advanced” option in the toolbar of the properties view. Then all element properties are visible.
See also
- Chapter 1.3.5.8.3 “Animating a color display” on page 1152

**Element property 'State variables'**

The variables control the element behavior dynamically.

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>“Invisible”</strong></td>
<td>Variable (BOOL). Toggles the visibility of the element. TRUE: The element is not visible at runtime.</td>
</tr>
<tr>
<td><strong>“Deactivate inputs”</strong></td>
<td>Variable (BOOL). Toggles the operability of the element. TRUE: User inputs do not have any effect in runtime more. The element is shown as deactivated.</td>
</tr>
</tbody>
</table>

The “Invisible” property is supported by the "Client Animation" functionality.

See also
- 

**Element property 'Button state variable'**

At runtime, the property controls whether the button is displayed as pressed or not.

Values:
- FALSE: The button is displayed as not pressed.
- TRUE: The button is displayed as pressed.

Argument passed to the property:
- Placeholder for the user input variable to couple the representation of the button with the input variable.
  - “<toggle/tap variable>”
  - “<NOT toggle/tap variable>”

Note: Specify a variable for the mouse events “Tap” or “Toggle” in the input configuration of the button. Only then is the placeholder set. If you configure a variable in both “Toggle” and “Tap”, then the variable specified in “Tap” is used.

Hint: Click the symbol 🖼 to insert the placeholder “<toggle/tap variable>”.

When you activate the “Inputconfiguration”, “Tap FALSE” property, then the “<NOT toggle/tap variable>” placeholder is displayed.
- Instance path of a project variable (BOOL)
  Example: prgA.xButtonState

Note: Implement a value assignment in the code for the variable specified here.

See also
- 

**Element property 'Image ID variable'**

<table>
<thead>
<tr>
<th>“Image ID”</th>
<th>Variable (STRING). Contains the image ID. The contents of the string corresponds to the description of the “Static ID” property.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example:</td>
<td>PLC_PRG.stImageID := 'ImagePool_A.Image3';</td>
</tr>
</tbody>
</table>

See also

- "Chapter 1.3.5.18.1.5 “Visualization element 'Image'” on page 1272"
- "Object 'Image Pool'"

These properties are available only when you select the “Preview: Support client animations and overlay of native elements” option in the Visualization Manager.

You can animate a movement, a rotation, and the visibility of a visualization element.

<table>
<thead>
<tr>
<th>“Animation duration”</th>
<th>Time that the element executes an animation (in milliseconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example:</td>
<td>500</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>“Move to the foreground”</th>
<th>Property value (BOOL)</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRUE</td>
<td>At runtime, the element is displayed in the foreground.</td>
</tr>
<tr>
<td>FALSE</td>
<td>At runtime, the element is displayed in the layer where it was inserted in the visualization editor.</td>
</tr>
</tbody>
</table>

See also

- "Chapter 1.3.5.21.5 “Using client animation” on page 1962"

**Element property 'Input configuration'**

The properties contain the configurations for the user input when using the mouse or keyboard. A user input defines an event and one or more actions that are executed when an event occurs.

The “Configure” button opens the “Input configuration” dialog box for creating or modifying a user input.

Configured user inputs are briefly listed below the events. They each include the action that is triggered and the setting in short form.

**Example:** "Execute ST code":  # PLC_PRG.i_x := 0;

<table>
<thead>
<tr>
<th>“OnDialogClosed”</th>
<th>Input event: The user closes the dialog box.</th>
</tr>
</thead>
<tbody>
<tr>
<td>“OnMouseClicked”</td>
<td>Input event: The user clicks the mouse button completely. The mouse button is clicked and released.</td>
</tr>
<tr>
<td>“OnMouseDown”</td>
<td>Input event: The user clicks down on the mouse button.</td>
</tr>
<tr>
<td>“OnMouseEnter”</td>
<td>Input event: The user drags the mouse pointer to the element.</td>
</tr>
<tr>
<td>“OnMouseLeave”</td>
<td>Input event: The user drags the mouse pointer away from the element.</td>
</tr>
<tr>
<td>“OnMouseMove”</td>
<td>Input event: The user moves the mouse pointer over the element area.</td>
</tr>
<tr>
<td>“OnMouseUp”</td>
<td>Input event: The user releases the mouse button over the element area.</td>
</tr>
</tbody>
</table>
**“Tap”**

When a mouse click event occurs, the variable defined in “Variable” is described in the application. The coding depends on the options “Tap FALSE” and “Tap on enter if captured”.

**“Variable”**

Variable (BOOL) that is set on mouse click.

Example: PLC_PRG.bIsTapped

TRUE: A mouse click event exists. It lasts while the user presses the mouse button over the element. It ends when the button is released.

FALSE: A mouse click event does not exist.

Requirement: The “Tap FALSE” option is not activated.

**“Tap FALSE”**

☑: The mouse click event leads to a complementary value in “Variable”.

TRUE: A mouse click event does not exist.

FALSE: While the mouse click event exists.

**“Tap on enter if captured”**

☑: During user input, it is also taken into consideration whether the mouse pointer is dragged within the element area or not while the mouse button is pressed.

TRUE: While the mouse click event exists and the mouse pointer is moved over the element area.

FALSE: A mouse click event does not exist. Or the user moves the mouse pointer outside of the element area while the mouse button is pressed.

The value is TRUE again as soon as the user moves the pointer back to the element area. The mouse is then captured.

**“Shift”**

With the onset of a mouse click event, the variable is set; when the mouse click event is completed, the variable is reset.

**“Variable”**

Variable (BOOL). Its value toggled when the mouse click event is ended. This is when the user releases the mouse button while the mouse pointer is over the element area.

If the user releases the mouse button while the mouse pointer is outside of the element area, then the mouse click event is not ended and the value is not toggled.

Tip: The user can cancel a started toggle input by dragging the mouse pointer out of the element area.

**“Toggle on up if captured”**

☑: The value toggles regardless of where the mouse pointer is when the mouse button is released. The mouse is then captured.

**“Keyboard shortcuts”**

Keyboard shortcut on the element for triggering specific input actions.

When the keyboard shortcut event occurs, the input actions in the “Event(s)” property are triggered. In this way, it is not the input action itself that leads to this input action, but the mouse input action.

**“Key”**

Key pressed for input action.

Example: [T]

Note: The following properties appear when a key is selected.
<table>
<thead>
<tr>
<th><strong>Event(s)</strong></th>
<th><strong>None</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>&quot;Mouse down&quot;: Pressing the key triggers the input actions that are configured in the &quot;OnMouseDown&quot; property.</td>
</tr>
<tr>
<td></td>
<td>&quot;Mouse up&quot;: Releasing the key triggers the input actions that are configured in the &quot;OnMouseUp&quot; property.</td>
</tr>
<tr>
<td></td>
<td>&quot;Mouse down/up&quot;: Pressing and releasing the key triggers the input actions that are configured in the &quot;OnMouseDown&quot; property and the &quot;OnMouseUp&quot; property.</td>
</tr>
<tr>
<td><strong>Shift</strong></td>
<td>![Checkmark] Combination with the Shift key</td>
</tr>
<tr>
<td></td>
<td>Example: [Shift]+[T].</td>
</tr>
<tr>
<td><strong>Control</strong></td>
<td>![Checkmark] Combination with the Ctrl key</td>
</tr>
<tr>
<td></td>
<td>Example: [Ctrl]+[T].</td>
</tr>
<tr>
<td><strong>Alt</strong></td>
<td>![Checkmark] Combination with the Alt key</td>
</tr>
<tr>
<td></td>
<td>Example: [Alt]+[T].</td>
</tr>
</tbody>
</table>

All keyboard shortcuts and their actions that are configured in the visualization are listed in the "Keyboard configuration" tab.

See also
- Chapter 1.3.5.19.2.2 “Command ‘Keyboard Configuration’” on page 1558
- Chapter 1.3.5.19.3.6 “Dialog ‘Input Configuration’” on page 1586

**Element property 'Access rights'**

Requirement: User management is set up for the visualization.

<table>
<thead>
<tr>
<th><strong>Access rights</strong></th>
<th>Opens the “Access rights” dialog. There you can edit the access privileges for the element.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Status messages:</td>
</tr>
<tr>
<td></td>
<td>- “Not set. Full rights.”: Access rights for all user groups: “operable”</td>
</tr>
<tr>
<td></td>
<td>- “Rights are set: Limited rights.”: Access is restricted for at least one group.</td>
</tr>
</tbody>
</table>

See also
- Chapter 1.3.5.19.3.1 “Dialog ‘Access Rights’” on page 1583

**Visualization element 'Group box’**

Symbol:

Tag: "Common controls"

The element provides visual grouping of visualization elements. The group box can be nested many times.
You can also use drag&drop to add elements to a group box. To do this, simply drag the element to the group box. The appearance of the cursor changes (a small plus sign is shown). When you click the [Shift] key at the same time, the element is not added.

You can remove elements from the group box by dragging them out of the group box.

---

**Element properties**

<table>
<thead>
<tr>
<th>“Element name”</th>
<th>Example: Parameter axis 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Optional</td>
<td></td>
</tr>
<tr>
<td>Hint: Assign individual names for elements so that they are found faster in the element list.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>“Type of element”</th>
<th>“Group box”</th>
</tr>
</thead>
</table>

| “Clipping” | [ ]: Elements are clipped that protrude beyond the size of the group box. |

---

**Element property 'Position'**

The position defines the location and size of the element in the visualization window. These are based on the Cartesian coordinate system. The origin is located at the upper left corner of the window. The positive horizontal x-axis runs to the right. The positive vertical y-axis runs downwards.

<table>
<thead>
<tr>
<th>“X”</th>
<th>X coordinate of the upper left corner of the element</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Specified in pixels.</td>
</tr>
<tr>
<td></td>
<td>Example: 10.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>“Y”</th>
<th>Y coordinate of the upper left corner of the element</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Specified in pixels.</td>
</tr>
<tr>
<td></td>
<td>Example: 10.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>“Width”</th>
<th>Specified in pixels.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Example: 150</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>“Height”</th>
<th>Specified in pixels.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Example: 30</td>
</tr>
</tbody>
</table>

You can also change the values by dragging the box symbols (¶) to other positions in the editor.

See also

- § Chapter 1.3.5.3.2 “Positioning elements, adapting size and layer” on page 1113

**Element property 'Center'**

The properties contain fixed values for the coordinates of the point of rotation. This point of rotation is shown as the ± symbol. The point is used as the center for rotating and scaling.
"X"  X-coordinate of the point of rotation

"Y"  Y-coordinate of the point of rotation

You can also change the values by dragging the symbols (_drag) to other positions in the editor.

Element property 'Texts'
The properties contain character strings for labeling the element. CODESYS accepts the specified texts automatically into the "GlobalTextList" text list. Therefore, these texts can be localized.

| "Text" | Character string (without single straight quotation marks) for the labeling the element.  
Example: Axis 1. |
| "Tooltip" | Character string (without single straight quotation marks) that is displayed as the tooltip of an element.  
Example: Parameters of Axis 1. |

See also

Element property 'Text properties'
The properties contain fixed values for the text properties.

| "Font" | Example: "Default"  
_text: The "Font" dialog box opens.  
▼: Drop-down list with style fonts. |
| "Font color" | Example: "Black"  
_text: The "Color" dialog box opens.  
▼: Drop-down list with style colors. |
| "Transparency" | Whole number (value range from 0 to 255). This determines the transparency of the respective color.  
Example: 255: The color is opaque.  
0: The color is completely transparent.  
Please note: If the color is a style color and already has a transparency value, then this property is write-protected. |

Element property 'Absolute movement'
The properties contain IEC variables for controlling the position of the element dynamically. The reference point is the upper left corner of the element. In runtime mode, the entire element is moved.
### Movement

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>X</strong></td>
<td>Variable (numeric data type). Defines the X position (in pixels).</td>
<td><code>PLC_PRG.iPos_X</code></td>
</tr>
<tr>
<td></td>
<td>Increasing this value in runtime mode moves the element to the right.</td>
<td></td>
</tr>
<tr>
<td><strong>Y</strong></td>
<td>Variable (numeric data type). Defines the Y position (in pixels).</td>
<td><code>PLC_PRG.iPos_Y</code></td>
</tr>
<tr>
<td></td>
<td>Increasing this value in runtime mode moves the element downwards.</td>
<td></td>
</tr>
</tbody>
</table>

### Rotation

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Rotation</strong></td>
<td>Variable (numeric data type). Defines the angle of rotation (in degrees).</td>
<td><code>PLC_PRG.iAngle1</code></td>
</tr>
<tr>
<td></td>
<td>The midpoint of the element rotates at the “Center” point. This rotation point is shown as the <code>+</code> symbol.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>In runtime mode, the alignment of the element remains the same with respect to the coordinate system of the visualization. Increasing the value rotates the element to the right.</td>
<td></td>
</tr>
</tbody>
</table>

### Scaling

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Scaling</strong></td>
<td>Variable (integer data type). Causes centric stretching.</td>
<td><code>PLC_PRG.iScaling</code></td>
</tr>
<tr>
<td></td>
<td>The reference point is the “Center” property.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>The value 1 shrinks the element by a factor of 0.001. The value 1000 returns the element to its original size.</td>
<td></td>
</tr>
</tbody>
</table>

### Interior rotation

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Interior rotation</strong></td>
<td>Variable (numeric data type). Defines the angle of rotation (in degrees).</td>
<td><code>PLC_PRG.iAngle2</code></td>
</tr>
<tr>
<td></td>
<td>In runtime mode, the element rotates about the point of rotation specified in “Center” according to the value of the variable. In addition, the alignment of the element rotates according to the coordinate system of the visualization. Increasing the value in the code rotates clockwise.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>The rotation point is shown as the <code>+</code> symbol.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Note: If a static angle of rotation is specified in the property “Position ➔ Angle”, then the static angle of rotation is added to the variable angle of rotation (offset) when the visualization is executed.</td>
<td></td>
</tr>
</tbody>
</table>

You can link the variables to a unit conversion.

The properties “X”, “Y”, “Rotation”, and “Interior rotation” are supported by the “Client Animation” functionality.
Element property 'State variables'
The variables control the element behavior dynamically.

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Invisible&quot;</td>
<td>Variable (BOOL). Toggles the visibility of the element. TRUE: The element is not visible at runtime.</td>
</tr>
<tr>
<td>&quot;Deactivate inputs&quot;</td>
<td>Variable (BOOL). Toggles the operability of the element. TRUE: User inputs do not have any effect in runtime more. The element is shown as deactivated.</td>
</tr>
</tbody>
</table>

The "Invisible" property is supported by the "Client Animation" functionality.

These properties are available only when you select the "Preview: Support client animations and overlay of native elements" option in the Visualization Manager.

You can animate a movement, a rotation, and the visibility of a visualization element.

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Animation duration&quot;</td>
<td>Time that the element executes an animation (in milliseconds) Example: 500</td>
</tr>
</tbody>
</table>
| "Move to the foreground" | Property value (BOOL)  
TRUE: At runtime, the element is displayed in the foreground.  
FALSE: At runtime, the element is displayed in the layer where it was inserted in the visualization editor. |

Visualization element 'Table'
Symbol:

Tag: “Common controls”
The element displays data that can be represented as an array in a table. Therefore, the data type of the visualizing variable can be 1) a one-dimensional array, 2) a maximum two-dimensional array, 3) an array of an array, 4) an array of structures, or 5) an array of a function block.

### Element properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>“Element name”</strong></td>
<td>Example: Data set component 1</td>
</tr>
<tr>
<td>Optional</td>
<td>Hint: Assign individual names for elements so that they are found faster in the element list.</td>
</tr>
<tr>
<td><strong>“Type of element”</strong></td>
<td>Table</td>
</tr>
<tr>
<td><strong>“Data array”</strong></td>
<td>Array whose data is visualized as a table. Variable (ARRAY) whose data type determines the number of columns and rows in the table. Array types</td>
</tr>
<tr>
<td></td>
<td>● One-dimensional array: The table has one column.</td>
</tr>
<tr>
<td></td>
<td>● Two-dimensional array: The second dimension determines the number of columns.</td>
</tr>
<tr>
<td></td>
<td>● Array of an array: The number of array elements of the back array determines the number of columns.</td>
</tr>
<tr>
<td></td>
<td>● Array of a structure: The number of structure members determines the number of columns.</td>
</tr>
<tr>
<td></td>
<td>● Array of a function block: The number of local variables determines the number of columns.</td>
</tr>
<tr>
<td>Example:</td>
<td>PLC_PRG.aiTable</td>
</tr>
<tr>
<td>Declaration:</td>
<td>aiTable : ARRAY[0..3, 0..4] OF INT := [4(1, 2, 3, 4, 5)];</td>
</tr>
<tr>
<td></td>
<td>Hint: If the declaration of the array changes, then the table can be refreshed by placing the cursor in the data array value field and pressing the [Enter] key.</td>
</tr>
<tr>
<td><strong>“Max. array index”</strong></td>
<td>Top index limit for the displayed table. Limits the number of displayed rows. The index begins at 0.</td>
</tr>
<tr>
<td></td>
<td>● Variable (integer data type)</td>
</tr>
<tr>
<td></td>
<td>Example: PLC_PRG.iUpperIndexBoundToDisplay</td>
</tr>
<tr>
<td></td>
<td>● Integer literal</td>
</tr>
<tr>
<td></td>
<td>Example: 4 is displayed as 5 in the row of the table.</td>
</tr>
</tbody>
</table>

See also

- Data Type ‘ARRAY’

### Element property ‘Columns’

The table element shows the values of a variable in a tabular view. The array elements of structure members are shown in a column or in a row. Two-dimensional arrays or arrays of a structure are shown in multiple columns. The visualized variable is defined in the “Data array” property. When a variable is assigned there, you can specify the display of the table columns where the array elements are shown. An individual configuration is possible for each column that is assigned to an index [<n>].

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>“Show row header”</strong></td>
<td>☑: The row header is visible.</td>
</tr>
<tr>
<td>Example:</td>
<td>For an array, the index of the array element is displayed in the header.</td>
</tr>
<tr>
<td><strong>“Show column header”</strong></td>
<td>☑: The column label is visible.</td>
</tr>
</tbody>
</table>
Table 260: “Element property ‘Columns: Column [<n>]’”

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Row height”</td>
<td>Height of the rows (in pixels)</td>
</tr>
<tr>
<td>“Row header width”</td>
<td>Width of the row label</td>
</tr>
<tr>
<td>“Scrollbar size”</td>
<td>Size of the scrollbar (in pixels)</td>
</tr>
<tr>
<td>“Column header”</td>
<td>By default, the name of the array or structure is applied as the heading with the index or structure member for the column. If an array of a function block has been selected for “Data array”, then the name of the array is applied to the column header with the local variables of the function block that belong to the column. The column label can be changed here by specifying a new title.</td>
</tr>
<tr>
<td>“Width”</td>
<td>Column width (in pixels)</td>
</tr>
<tr>
<td>“Image column”</td>
<td>Images can be displayed in the column. Images are used from the global image pool or custom image pools. The image IDs are shown in the cells of the table as they are defined in the image pool.</td>
</tr>
<tr>
<td>“Image configuration”</td>
<td></td>
</tr>
<tr>
<td>“Fill mode”</td>
<td>● Fill cell: The image resizes to the dimensions of the cell without fixing the height/width ratio. ● Centered: The image is centered in the cell and retains its proportions (height/width ratio).</td>
</tr>
<tr>
<td>“Transparency”</td>
<td>[✓]: The color which is specified in “Transparent color” is displayed as transparent.</td>
</tr>
<tr>
<td>“Transparent color”</td>
<td>This color is displayed as transparent. Requirement: The “Transparency” property is activated.</td>
</tr>
<tr>
<td>“Text alignment of header”</td>
<td>Alignment of the column header:</td>
</tr>
<tr>
<td></td>
<td>● Left</td>
</tr>
<tr>
<td></td>
<td>● Centered</td>
</tr>
<tr>
<td></td>
<td>● Right</td>
</tr>
<tr>
<td>“Use template”</td>
<td>[✓]: Another visualization element (type “Rectangle”, “Round rectangle”, or “Ellipse”) is inserted into each line of this table column. The properties list is extended automatically with the properties of this element in “Template”.</td>
</tr>
<tr>
<td>“Text alignment of the headline from the template”</td>
<td>Requirement: The “Use template” property is activated. [✓]: When activated, the settings for font (size) and alignment in the inserted template are also applied to the column header.</td>
</tr>
<tr>
<td>“Template”</td>
<td>Requirement: The “Use template” property is activated. The properties of all elements assigned to the column are listed in “Template”. They can be modified there as described in &quot;Rectangle, Round rectangle and Ellipse&quot;</td>
</tr>
</tbody>
</table>

See also
- Chapter 1.3.5.18.1.1 “Visualization element 'Rectangle', 'Round rectangle', 'Ellipse'” on page 1224

Element property ‘Position’
The position defines the location and size of the element in the visualization window. These are based on the Cartesian coordinate system. The origin is located at the upper left corner of the window. The positive horizontal x-axis runs to the right. The positive vertical y-axis runs downwards.
<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>“X”</td>
<td>X coordinate of the upper left corner of the element. Specified in pixels. Example: 10.</td>
</tr>
<tr>
<td>“Width”</td>
<td>Specified in pixels. Example: 150</td>
</tr>
<tr>
<td>“Height”</td>
<td>Specified in pixels. Example: 30</td>
</tr>
</tbody>
</table>

You can also change the values by dragging the box symbols (_) to other positions in the editor.

See also

● Chapter 1.3.5.3.2 “Positioning elements, adapting size and layer” on page 1113

**Element property ‘Center’**

The properties contain fixed values for the coordinates of the point of rotation. This point of rotation is shown as the ◆ symbol. The point is used as the center for rotating and scaling.

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>“X”</td>
<td>X-coordinate of the point of rotation</td>
</tr>
<tr>
<td>“Y”</td>
<td>Y-coordinate of the point of rotation</td>
</tr>
</tbody>
</table>

You can also change the values by dragging the symbols (◆) to other positions in the editor.

**Element property ‘Text properties’**

The properties contain fixed values for the text properties.

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Horizontal alignment”</td>
<td>Horizontal alignment of the text within the element.</td>
</tr>
<tr>
<td>“Vertical alignment”</td>
<td>Vertical alignment of the text within the element.</td>
</tr>
<tr>
<td>“Font”</td>
<td>Example: “Default”</td>
</tr>
<tr>
<td></td>
<td><img src="icon" alt="The “Font” dialog box opens." /></td>
</tr>
<tr>
<td></td>
<td><img src="icon" alt="Drop-down list with style fonts." /></td>
</tr>
</tbody>
</table>
**Font color**

Example: "Black"

- The "Color" dialog box opens.
- Drop-down list with style colors.

**Transparency**

Whole number (value range from 0 to 255). This determines the transparency of the respective color.

Example: 255: The color is opaque.
0: The color is completely transparent.

Please note: If the color is a style color and already has a transparency value, then this property is write-protected.

---

**Element property 'Dynamic texts'**

Dynamic texts are variably indexed texts of a text list. At runtime, the text is displayed that is currently indexed in the variable.

**Text list**

Variable (string) or name of the text list as a fixed string in single straight quotation marks.

Example: 'Errorlist'

- Drop-down list with the dialogs available in the text lists.

**Text index**

Text list ID. This refers to the desired output text.

- As fixed string with the ID in single straight quotation marks.
  Example: '1'
- As a variable (STRING) for dynamically controlling the text output.
  Example: strTextID
  Sample assignment: PLC_PRG.strTextID := '1';

**Tooltip index**

Text list ID. This refers to the desired output text.

- As fixed string with the ID in single straight quotation marks.
  Example: '2'
- As a variable (STRING) for dynamically controlling the text output.
  Example: strToolTipID
  Sample assignment: PLC_PRG.strToolTipID := '2';

---

See also

- Text list

---

**Element property 'Font variables'**

The variables enable dynamic control of the text display.

**Font name**

Variable (STRING). Includes the font of the text.

Example: PLC_PRG.stFontVar := 'Arial';

The selection of fonts corresponds to the default "Font" dialog box.

**Size**

Variable (integer data type). Contains the font size (in pixels).

Example: PLC_PRG.iFontHeight := 16;

The selection of font sizes corresponds to the default "Font" dialog box.
**Flags**

Variable (DWORD). Contains the flags for displaying fonts.

Flags:
- 1: Italics
- 2: Bold
- 4: Underline
- 8: Strikethrough

Note: You can combine the font displays by adding the coding of the flags. For example, a bold and underlined text:

```plaintext
PLC_PRG.dwFontType := 6;
```

**Charset**

Variable (DWORD). Contains a character set number for the font.

The selection of character set numbers corresponds to the “Script” setting of the standard “Font” dialog box.

**Color**

Variable (DWORD). Includes the color of the text.

Example:

```plaintext
PLC_PRG.dwColorFont := 16#FF000000;
```

**Flags for text alignment**

Variable (integer data type). Contains the coding for text alignment.

Example:

```plaintext
PLC_PRG.dwTextAlignment := 5;
```

Coding:
- 0: Top left
- 1: Horizontal center
- 2: Right
- 4: Vertical center
- 8: Bottom

Note: You can combine the text alignments by adding the coding of the flags. For example, a vertical and horizontal centered text:

```plaintext
PLC_PRG.dwFontType := 5;
```

### Fixed values for displaying texts are set in “Text properties”.

See also

- “Element property ‘Text properties’” on page 1227

**Element property 'Absolute movement'**

The properties contain IEC variables for controlling the position of the element dynamically. The reference point is the upper left corner of the element. In runtime mode, the entire element is moved.

<table>
<thead>
<tr>
<th><strong>“Movement”</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>“X”</strong></td>
</tr>
<tr>
<td>Variable (numeric data type). Defines the X position (in pixels).</td>
</tr>
<tr>
<td>Example: PLC_PRG.iPos_X.</td>
</tr>
<tr>
<td>Increasing this value in runtime mode moves the element to the right.</td>
</tr>
</tbody>
</table>

| **“Y”** |
| Variable (numeric data type). Defines the Y position (in pixels). |
| Example: PLC_PRG.iPos_Y. |
| Increasing this value in runtime mode moves the element downwards. |
### “Rotation”

Variable (numeric data type). Defines the angle of rotation (in degrees).

**Example:** PLC_PRG.iAngle1.

The midpoint of the element rotates at the “Center” point. This rotation point is shown as the symbol.

In runtime mode, the alignment of the element remains the same with respect to the coordinate system of the visualization. Increasing the value rotates the element to the right.

### “Interior rotation”

Variable (numeric data type). Defines the angle of rotation (in degrees).

**Example:** PLC_PRG.iAngle2.

In runtime mode, the element rotates about the point of rotation specified in “Center” according to the value of the variable. In addition, the alignment of the element rotates according to the coordinate system of the visualization. Increasing the value in the code rotates clockwise.

The rotation point is shown as the symbol.

Note: If a static angle of rotation is specified in the “Position » Angle” property, then the static angle of rotation is added to the variable angle of rotation (offset) when the visualization is executed.

---

**You can link the variables to a unit conversion.**

---

The “X”, “Y”, “Rotation”, and “Interior rotation” properties are supported by the “Client Animation” functionality.

---

**Element property ‘State variables’**

The variables control the element behavior dynamically.

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Invisible”</td>
<td>Variable (BOOL). Toggles the visibility of the element. TRUE: The element is not visible at runtime.</td>
</tr>
<tr>
<td>“Deactivate inputs”</td>
<td>Variable (BOOL). Toggles the operability of the element. TRUE: User inputs do not have any effect in runtime more. The element is shown as deactivated.</td>
</tr>
</tbody>
</table>
The “Invisible” property is supported by the “Client Animation” functionality.

See also

Element property ‘Selection’

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Background color on selection”</td>
<td>Fill color of the selected row.</td>
</tr>
<tr>
<td>“Selection font color”</td>
<td>Font color of the selected row.</td>
</tr>
<tr>
<td>“Selection type”</td>
<td>Selection when clicking the table row.</td>
</tr>
<tr>
<td></td>
<td>• No selection: No selection</td>
</tr>
<tr>
<td></td>
<td>• Cell selection: The clicked cell only.</td>
</tr>
<tr>
<td></td>
<td>• Row selection: Row of the clicked cell.</td>
</tr>
<tr>
<td></td>
<td>• Column selection: Column of the clicked cell.</td>
</tr>
<tr>
<td></td>
<td>• Row and column selection: Row and column of the clicked cell.</td>
</tr>
<tr>
<td>“Frame around selected cells”</td>
<td>☑️: A frame is drawn around the selected cells.</td>
</tr>
<tr>
<td>“Variable for selected column”</td>
<td>Variable (INT). Contains the array index of the “Column” of the selected cell.</td>
</tr>
<tr>
<td></td>
<td>Warning: This index represents the correct position in the array only if no columns have been removed from the table in the display.</td>
</tr>
<tr>
<td>“Variable for selected row”</td>
<td>Variable (INT). Contains the array index of the “Row” of the selected cell.</td>
</tr>
<tr>
<td>“Variable for valid column selection”</td>
<td>Variable (BOOL). TRUE: The “Variable for selected column” variable contains a valid value.</td>
</tr>
</tbody>
</table>

These properties are available only when you select the “Preview: Support client animations and overlay of native elements” option in the Visualization Manager.

You can animate a movement, a rotation, and the visibility of a visualization element.

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Animation duration”</td>
<td>Time that the element executes an animation (in milliseconds)</td>
</tr>
<tr>
<td></td>
<td>Example: 500</td>
</tr>
<tr>
<td>“Move to the foreground”</td>
<td>Property value (BOOL)</td>
</tr>
<tr>
<td></td>
<td>TRUE: At runtime, the element is displayed in the foreground.</td>
</tr>
<tr>
<td></td>
<td>FALSE: At runtime, the element is displayed in the layer where it was inserted in the visualization editor.</td>
</tr>
</tbody>
</table>

See also

- ☑️ Chapter 1.3.5.21.5 “Using client animation” on page 1962
Requirement: User management is set up for the visualization.

| "Access rights" | Opens the "Access rights" dialog. There you can edit the access privileges for the element. Status messages: ● "Not set. Full rights.": Access rights for all user groups: “operable” ● "Rights are set: Limited rights" : Access is restricted for at least one group. |

See also

● § Chapter 1.3.5.19.3.1 “Dialog ‘Access Rights’” on page 1583

See also

●

● § Chapter 1.3.5.9.1 “Displaying structured variable values in tables” on page 1155

● Data Type ‘ARRAY’

Visualization element 'Text field'  

Symbol: 

Tag: “Common controls”

The element is used for the following:

● Static output of text. The contents of a variable can be part of the text.
● Showing a tooltip. The text is managed as static text and can also be defined so that the contents of a variable are also displayed.
● Dynamic output of text. Texts of a text list are displayed dynamically.
● Input of text. For example, a user can input a number or a text literal.

See also

●

Element properties

| "Element name" | Optional
| Example: FileName_A |
| Hint: Assign individual names for elements so that they are found faster in the element list. |

| "Type of element" | “Text field” |

Element property ‘Position’  
The position defines the location and size of the element in the visualization window. These are based on the Cartesian coordinate system. The origin is located at the upper left corner of the window. The positive horizontal x-axis runs to the right. The positive vertical y-axis runs downwards.
### "X"
X coordinate of the upper left corner of the element
Specified in pixels.
Example: 10.

### "Y"
Y coordinate of the upper left corner of the element
Specified in pixels.
Example: 10.

### "Width"
Specified in pixels.
Example: 150

### "Height"
Specified in pixels.
Example: 30

---

You can also change the values by dragging the box symbols (.getDrawable) to other positions in the editor.

---

See also
- Chapter 1.3.5.3.2 “Positioning elements, adapting size and layer” on page 1113

---

**Element property 'Colors'**

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Normal state&quot;</td>
<td>The normal state is in effect if the variable in “Color variables ➔ Toggle color” is not defined or it has the value FALSE.</td>
</tr>
<tr>
<td>&quot;Frame color&quot;</td>
<td>Frame and fill color for the corresponding state of the variable.</td>
</tr>
<tr>
<td>&quot;Fill color&quot;</td>
<td></td>
</tr>
<tr>
<td>&quot;Transparency&quot;</td>
<td>Transparency value (0 to 255) for defining the transparency of the selected color. Example: 255: The color is opaque. 0: The color is completely transparent.</td>
</tr>
<tr>
<td>&quot;Alarm state&quot;</td>
<td>The alarm state is in effect if the variable in “Color variables ➔ Toggle color” has the value TRUE.</td>
</tr>
</tbody>
</table>

See also
- Chapter 1.3.5.19.3.5 “Dialog ‘Gradient Editor’” on page 1586

---

**Element property 'Appearance'**
The properties contain fixed values for setting the look of the element.
"Line width" Value in pixels
Example: 2
Note: The values 0 and 1 both result in a line weight of 1 pixel. If no line should be displayed, then the “Line style” property must be set to the option “Invisible”.

"Fill attributes" The way in which the element is filled.
- “Filled”: The element is filled with the color from property “Colors ➔ Fill color”.
- “Invisible”: The fill color is invisible.

"Line style" Type of line representation
- “Solid”
- “Dashes”
- “Dots”
- “Dash Dot”
- “Dash Dot Dot”
- “not visible”

You can assign variables in the “Appearance variables” property for controlling the appearance dynamically. The fixed values here are overwritten.

See also
- § “Element property 'Appearance variables” on page 1285

Element property 'Texts'
The properties contains character strings for labeling the element. The character string can also contain a placeholder with a format definition. In runtime mode, the placeholder is replaced by the current value in the specified format.
CODESYS accepts the specified texts automatically into the “GlobalTextList” text list. Therefore, these texts can be localized.

"Text" Character string (without single straight quotation marks) for the labeling the element. Add a line break by pressing the keyboard shortcut [Ctrl] + [Enter].
Example: Accesses: %i
The variable that contains the current value for the placeholder is specified in the property “Text variable ➔ Text”.

"Tooltip" Character string (without single straight quotation marks) that is displayed as the tooltip of an element.
Example: Number of valid accesses.
The variable that contains the current value for the placeholder is specified in the property “Text variable ➔ Tooltip”.

See also
- § “Element property 'Text variables” on page 1229
- § Chapter 1.3.5.18.2 “Placeholders with format definition in the output text” on page 1546

Element property 'Text properties'
The properties contain fixed values for the text properties.
**“Horizontal alignment”**
Horizontal alignment of the text within the element.

**“Vertical alignment”**
Vertical alignment of the text within the element.

**“Text format”**
Definition for displaying texts that are too long
- **“Default”**: The long text is truncated.
- **“Line break”**: The text is split into parts.
- **“Ellipsis”**: The visible text ends with "..." indicating that it is not complete.

**“Font”**
Example: **“Default”**
- The “Font” dialog box opens.
- Drop-down list with style fonts.

**“Font color”**
Example: **“Black”**
- The “Color” dialog box opens.
- Drop-down list with style colors.

**“Transparency”**
Whole number (value range from 0 to 255). This determines the transparency of the respective color.
Example: **255**: The color is opaque.
Example: **0**: The color is completely transparent.

Please note: If the color is a style color and already has a transparency value, then this property is write-protected.

---

**Element property ’Text variables’**
These properties are variables with contents that replace a format definition.

<table>
<thead>
<tr>
<th>“Text variable”</th>
<th>Variable (data type compliant with the format definition). It contains what is printed instead of the format definition.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example: <strong>PLC_PRG.iAccesses</strong></td>
<td>Note: The format definition is part of the text in the property “Texts ➔ Text”. Note: If you specify a variable of type enumeration with text list support, then the name of the enumeration data type is added automatically in angle brackets after the variable name. Example: <strong>PLC_PRG.enVar &lt;enumeration name&gt;</strong>. Then the symbolic value of the enumeration component is printed instead of the numeric value when text is printed. Refer to the help page for the enumerations.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>“Tooltip variable”</th>
<th>Variable (data type compliant with the format definition). It contains what is printed instead of the format definition.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example: <strong>PLC_PRG.iAccessesInTooltip</strong></td>
<td>Note: The format definition is part of the text in the property “Texts ➔ Tooltip”.</td>
</tr>
</tbody>
</table>

See also
- ☀ Chapter 1.3.5.18.2 “Placeholders with format definition in the output text” on page 1546
- ☀ “Element property ’Texts’” on page 1226
- Enumerations

**Element property ’Dynamic texts’**
Dynamic texts are variably indexed texts of a text list. At runtime, the text is displayed that is currently indexed in the variable.
| **Text list** | Variable (string) or name of the text list as a fixed string in single straight quotation marks.  
Example: 'Errorlist'  
▼: Drop-down list with the dialogs available in the text lists. |
|------------------|--------------------------------------------------|
| **Text index** | Text list ID. This refers to the desired output text.  
- As fixed string with the ID in single straight quotation marks.  
  Example: '1'  
- As a variable (STRING) for dynamically controlling the text output.  
  Example: strTextID  
  Sample assignment: PLC_PRG.strTextID := '1'; |
| **Tooltip index** | Text list ID. This refers to the desired output text.  
- As fixed string with the ID in single straight quotation marks.  
  Example: '2'  
- As a variable (STRING) for dynamically controlling the text output.  
  Example: strToolTipID  
  Sample assignment: PLC_PRG.strToolTipID := '2'; |

See also  
- **Text list**

**Element property 'Font variables'**  
The variables allow for dynamic control of the text display.

| **Font name** | Variable (STRING). Includes the font of the text.  
Example: PLC_PRG.stFontVar := 'Arial';  
The selection of fonts corresponds to the default "Font" dialog. |
|------------------|--------------------------------------------------|
| **Size** | Variable (numeric data type). Contains the font size (in pixels or points). The applied unit is specified in brackets after the variable name.  
- <pt>: Points (default)  
  Example: PLC_PRG.iFontHeight <pt>  
  Code: iFontHeight : INT := 12;  
- <px>: Pixels  
  Example: PLC_PRG.iFontHeight <px>  
  Code: iFontHeight : INT := 19;  
If you click in the value field, a drop-down list opens on the right for setting the unit.  
Hint: The font size is specified in points (example: Arial 12). Use points when the variable font size should match a font, for example if a font is set in the property "Text property ➤ Font". |
### Flags

Variable (DWORD). Contains the flags for displaying fonts.

Flags:
- 1: Italics
- 2: Bold
- 4: Underline
- 8: Strikethrough

Note: You can combine the font displays by adding the coding of the flags. For example, a bold and underlined text: `PLC_PRG.dwFontType := 6;` 

### Character set

Variable (DWORD). Contains a character set number for the font.

The selection of character set numbers corresponds to the "Script" setting of the standard "Font" dialog.

### Color

Variable (DWORD). Includes the color of the text.

Example: `PLC_PRG.dwColorFont := 16#FF000000;` 

### Flags for text alignment

Variable (integer data type). Contains the coding for text alignment.

Example: `PLC_PRG.dwTextAlignment := 5;` 

Coding:
- 0: Top left
- 1: Horizontal center
- 2: Right
- 4: Vertical center
- 8: Bottom

Note: You can combine the text alignments by adding the coding of the flags. For example, a vertical and horizontal centered text: `PLC_PRG.dwFontType := 5;` 

---

**Fixed values for displaying texts are set in "Text properties".**

See also

- "Element property 'Text properties'" on page 1227

**Element property 'Colorvariables'**

The Element property is used as an interface for project variables to dynamically control colors at runtime.
### “Toggle color”

The property controls the toggled color at runtime.

**Value assignment:**
- **FALSE:** The element is displayed with the color specified in the “Color” property.
- **TRUE:** The element is displayed with the color specified in the “Alarm color” property.

**Assignment options:**
- Placeholder for the user input variable
  - `<toggle/tap variable>`
  - `<NOT toggle/tap variable>`

The color change is not controlled by its own variable, but by a user input variable.

Note: Specify a variable for the mouse events “Tap” or “Toggle” in the input configuration of the element. Only then is the placeholder set. If you configure a variable in both “Toggle” and “Tap”, then the variable specified in “Tap” is used.

Hint: Click the symbol to insert the placeholder “<toggle/tap variable>”. When you activate the “Input configuration”, “Tap FALSE” property, then the “<NOT toggle/tap variable>” placeholder is displayed.

- Instance path of a project variable (BOOL)
  - Example: PLC_PRG.xColorIsToggeled

Note: In the code, declare and implement the variable specified here. Its value assignment determines when the color changes.

### “Normal state”

### “Alarm state”

The properties listed below control the color depending on the state. The normal state is in effect if the variable in “Color variables”, “Toggle color” is not defined or it has the value FALSE. The alarm state is in effect if the variable in “Color variables”, “Toggle color” has the value TRUE.

### “Frame color”

**Assignment options:**
- Variable (DWORD) for the frame color
  - Example: PLC_PRG.dwBorderColor
- Color literal
  - Example of green and opaque: 16#FF00FF00

### “Filling color”

**Assignment options:**
- Variable (DWORD) for the fill color
  - Example: PLC_PRG.dwFillColor
- Color literal
  - Example of gray and opaque: 16#FF888888

---

The transparency part of the color value is evaluated only if the “Activate semi-transparent drawing” option of the visualization manager is selected.

Select the “Advanced” option in the toolbar of the properties view. Then all element properties are visible.

See also
- Chapter 1.3.5.8.3 “Animating a color display” on page 1152
**Element property 'Absolute movement'**
The properties contain IEC variables for controlling the position of the element dynamically. The reference point is the upper left corner of the element. In runtime mode, the entire element is moved.

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
<th>Example</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>“Movement”</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>“X”</strong></td>
<td>Variable (numeric data type). Defines the X position (in pixels).</td>
<td>PLC_PRG.iPos_X</td>
<td>Increasing this value in runtime mode moves the element to the right.</td>
</tr>
<tr>
<td><strong>“Y”</strong></td>
<td>Variable (numeric data type). Defines the Y position (in pixels).</td>
<td>PLC_PRG.iPos_Y</td>
<td>Increasing this value in runtime mode moves the element downwards.</td>
</tr>
<tr>
<td><strong>“Rotation”</strong></td>
<td>Variable (numeric data type). Defines the angle of rotation (in degrees).</td>
<td>PLC_PRG.iAngle1</td>
<td>The midpoint of the element rotates at the “Center” point. This rotation point is shown as the symbol.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>In runtime mode, the alignment of the element remains the same with respect to the coordinate system of the visualization. Increasing the value rotates the element to the right.</td>
</tr>
<tr>
<td><strong>“Interior rotation”</strong></td>
<td>Variable (numeric data type). Defines the angle of rotation (in degrees).</td>
<td>PLC_PRG.iAngle2</td>
<td>In runtime mode, the element rotates about the point of rotation specified in “Center” according to the value of the variable. In addition, the alignment of the element rotates according to the coordinate system of the visualization. Increasing the value in the code rotates clockwise.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The rotation point is shown as the symbol. Note: If a static angle of rotation is specified in the &quot;Position Angle&quot; property, then the static angle of rotation is added to the variable angle of rotation (offset) when the visualization is executed.</td>
</tr>
</tbody>
</table>

You can link the variables to a unit conversion.

The “X”, “Y”, “Rotation”, and “Interior rotation” properties are supported by the "Client Animation" functionality.
See also
- [Unit conversion](#)

### Element property 'State variables'

The variables control the element behavior dynamically.

<table>
<thead>
<tr>
<th>Variable (BOOL)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Invisible&quot;</td>
<td>Toggles the visibility of the element. TRUE: The element is not visible at runtime.</td>
</tr>
<tr>
<td>&quot;Deactivate inputs&quot;</td>
<td>Toggles the operability of the element. TRUE: User inputs do not have any effect in runtime more. The element is shown as deactivated.</td>
</tr>
</tbody>
</table>

The "Invisible" property is supported by the "Client Animation" functionality.

See also
- [Unit conversion](#)

### Element property 'Selection and caret configuration'

The variables allow for controlling the caret position and the selection of the text.

<table>
<thead>
<tr>
<th>Variable (integer data type)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Caret position&quot;</td>
<td>Contains the position of the caret.</td>
</tr>
<tr>
<td>&quot;Selection start&quot;</td>
<td>Contains the position of the first selected character. Example: PLC_PRG.iSelStart</td>
</tr>
<tr>
<td>&quot;Selection end&quot;</td>
<td>Contains the position of the last selected character. Example: PLC_PRG.iSelEnd</td>
</tr>
<tr>
<td>&quot;All selected&quot;</td>
<td>Toggles the selection of the entered text. TRUE: The text in the text field is selected. FALSE: The selection starts with the value in “Selection start” and ends with “Selection end”.</td>
</tr>
</tbody>
</table>

These properties are available only when you select the “Preview: Support client animations and overlay of native elements” option in the Visualization Manager.

You can animate a movement, a rotation, and the visibility of a visualization element.

<table>
<thead>
<tr>
<th>Variable (BOOL)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Animation duration&quot;</td>
<td>Time that the element executes an animation (in milliseconds). Example: 500</td>
</tr>
<tr>
<td>&quot;Move to the foreground&quot;</td>
<td>Property value (BOOL). TRUE: At runtime, the element is displayed in the foreground. FALSE: At runtime, the element is displayed in the layer where it was inserted in the visualization editor.</td>
</tr>
</tbody>
</table>
The properties contain the configurations for the user input when using the mouse or keyboard. A user input defines an event and one or more actions that are executed when an event occurs.

The “Configure” button opens the “Input configuration” dialog box for creating or modifying a user input. Configured user inputs are briefly listed below the events. They each include the action that is triggered and the setting in short form.

### Example: “Execute ST code”

```plc
PLC_PRG.i_x := 0;
```

<table>
<thead>
<tr>
<th>Event</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>“OnDialogClosed”</td>
<td>Input event: The user closes the dialog box.</td>
</tr>
<tr>
<td>“OnMouseClick”</td>
<td>Input event: The user clicks the mouse button completely. The mouse button is clicked and released.</td>
</tr>
<tr>
<td>“OnMouseDown”</td>
<td>Input event: The user clicks down on the mouse button.</td>
</tr>
<tr>
<td>“OnMouseEnter”</td>
<td>Input event: The user drags the mouse pointer to the element.</td>
</tr>
<tr>
<td>“OnMouseMove”</td>
<td>Input event: The user moves the mouse pointer over the element area.</td>
</tr>
<tr>
<td>“OnMouseLeave”</td>
<td>Input event: The user drags the mouse pointer away from the element.</td>
</tr>
<tr>
<td>“OnMouseUp”</td>
<td>Input event: The user releases the mouse button over the element area.</td>
</tr>
</tbody>
</table>

### “Tap”

When a mouse click event occurs, the variable defined in “Variable” is described in the application. The coding depends on the options “Tap FALSE” and “Tap on enter if captured”.

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Tap FALSE”</td>
<td>✔️: The mouse click event leads to a complementary value in “Variable”.</td>
</tr>
<tr>
<td></td>
<td>TRUE: A mouse click event does not exist.</td>
</tr>
<tr>
<td></td>
<td>FALSE: While the mouse click event exists.</td>
</tr>
</tbody>
</table>

### “Tap on enter if captured”

✔️: During user input, it is also taken into consideration whether the mouse pointer is dragged within the element area or not while the mouse button is pressed.

TRUE: While the mouse click event exists and the mouse pointer is moved over the element area.

FALSE: A mouse click event does not exist. Or the user moves the mouse pointer outside of the element area while the mouse button is pressed. The value is TRUE again as soon as the user moves the pointer back to the element area. The mouse is then captured.
**“Shift”**  
With the onset of a mouse click event, the variable is set; when the mouse click event is completed, the variable is reset.

**“Variable”**  
Variable (BOOL). Its value toggled when the mouse click event is ended. This is when the user releases the mouse button while the mouse pointer is over the element area.

If the user releases the mouse button while the mouse pointer is outside of the element area, then the mouse click event is not ended and the value is not toggled.

Tip: The user can cancel a started toggle input by dragging the mouse pointer out of the element area.

**“Toggle on up if captured”**  
☑: The value toggles regardless of where the mouse pointer is when the mouse button is released. The mouse is then captured.

**“Keyboard shortcuts”**  
Keyboard shortcut on the element for triggering specific input actions.

When the keyboard shortcut event occurs, the input actions in the “Event(s)” property are triggered. In this way, it is not the input action itself that leads to this input action, but the mouse input action.

**“Key”**  
Key pressed for input action.

Example: [T]

Note: The following properties appear when a key is selected.

**“Event(s)”**
- “None”
- “Mouse down”: Pressing the key triggers the input actions that are configured in the “OnMouseDown” property.
- “Mouse up”: Releasing the key triggers the input actions that are configured in the “OnMouseUp” property.
- “Mouse down/up”: Pressing and releasing the key triggers the input actions that are configured in the “OnMouseDown” property and the “OnMouseUp” property.

**“Shift”**
☑: Combination with the Shift key

Example: [Shift]+[T].

**“Control”**
☑: Combination with the Ctrl key

Example: [Ctrl]+[T].

**“Alt”**
☑: Combination with the Alt key

Example: [Alt]+[T].

---

All keyboard shortcuts and their actions that are configured in the visualization are listed in the “Keyboard configuration” tab.

See also
- Chapter 1.3.5.19.2.2 “Command ‘Keyboard Configuration’” on page 1558
- Chapter 1.3.5.19.3.6 “Dialog ‘Input Configuration’” on page 1586

**Element property ‘Access rights’**

Requirement: User management is set up for the visualization.
“Access rights” Opens the “Access rights” dialog. There you can edit the access privileges for the element.
Status messages:
- “Not set. Full rights.”: Access rights for all user groups: “operable”
- “Rights are set: Limited rights”: Access is restricted for at least one group.

See also
- ☞ Chapter 1.3.5.19.3.1 “Dialog ‘Access Rights’” on page 1583

Visualization element ‘Scrollbar’
Symbol:

Tag: “Common controls”
The element sets the value of a variable, depending on the position of the scrollbar.

Element properties

<table>
<thead>
<tr>
<th>“Element name”</th>
<th>Example: Speed Conveyor Belt 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Optional</td>
<td>Hint: Assign individual names for elements so that they are found faster in the element list.</td>
</tr>
<tr>
<td>“Type of element”</td>
<td>“Scrollbar”</td>
</tr>
</tbody>
</table>

Element property ‘Center’
The properties contain fixed values for the coordinates of the point of rotation. This point of rotation is shown as the symbol. The point is used as the center for rotating and scaling.

<table>
<thead>
<tr>
<th>“X”</th>
<th>X-coordinate of the point of rotation</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Y”</td>
<td>Y-coordinate of the point of rotation</td>
</tr>
</tbody>
</table>

You can also change the values by dragging the symbols (④) to other positions in the editor.

<table>
<thead>
<tr>
<th>“Value”</th>
<th>Variable as type integer that includes the position of the scrollbar.</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Minimum value”</td>
<td>Smallest value of the scrollbar (fixed value or variable).</td>
</tr>
<tr>
<td>“Maximum value”</td>
<td>Largest value of the scrollbar (fixed value or variable).</td>
</tr>
</tbody>
</table>
"Page size"  Page size
  ● As a fixed value, for example 10
  ● As a variable of data type integer
  Requirement: Visible when the "Move to click" property is not selected.

"Move to click"  Behavior of the scrollbar at visualization runtime when it is clicked:
  ☑: The scrollbar moves to the clicked position.
  ☐: The scrollbar moves to one "Page size" in the direction of the click.

Element property 'Position'  The position defines the location and size of the element in the visualization window. These are based on the Cartesian coordinate system. The origin is located at the upper left corner of the window. The positive horizontal x-axis runs to the right. The positive vertical y-axis runs downwards.

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;X&quot;</td>
<td>X coordinate of the upper left corner of the element Specified in pixels. Example: 10.</td>
</tr>
<tr>
<td>&quot;Y&quot;</td>
<td>Y coordinate of the upper left corner of the element Specified in pixels. Example: 10.</td>
</tr>
<tr>
<td>&quot;Width&quot;</td>
<td>Specified in pixels. Example: 150</td>
</tr>
<tr>
<td>&quot;Height&quot;</td>
<td>Specified in pixels. Example: 30</td>
</tr>
</tbody>
</table>

You can also change the values by dragging the box symbols (tracked) to other positions in the editor.

See also
  ● Chapter 1.3.5.3.2 “Positioning elements, adapting size and layer” on page 1113

Element property 'Bar'  The property defines the representation of scaling and direction of travel.
**Orientation**  
Alignment of the slider; defined by the ratio of width to height.  
- **“Horizontal”**  
- **“Vertical”**  
You can modify the alignment in the visualization editor by using the pointing device to adjust the width and height of the scrollbar.

**Running direction**  
The drop-down list varies depending on the alignment of the slider.  
**Horizontal**  
- **“Left to right”: Scale starts at the left.**  
- **“Right to left”: Scale starts at the right.**  
**Vertical**  
- **“Bottom to top”: Scale starts at the bottom.**  
- **“Top to bottom”: Scale starts at the top.**

**Element property ‘Colors’**  
The properties contain fixed values for setting colors.

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Color</strong></td>
<td>Color for the element in its normal state. Please note that the normal state is in effect if the expression in the “Color variables ➔ Toggle color” property is not defined or it has the value FALSE.</td>
</tr>
<tr>
<td><strong>Alarm color</strong></td>
<td>Color for the element in alarm state. Please note that the alarm state is in effect if the expression in the “Color variables ➔ Toggle color” property has the value TRUE.</td>
</tr>
<tr>
<td><strong>Transparency</strong></td>
<td>Value (0 to 255) for defining the transparency of the selected color. Example 255: The color is opaque. 0: The color is completely transparent.</td>
</tr>
</tbody>
</table>

**Element property ‘Texts’**  
The properties contains character strings for labeling the element. The character string can also contain a placeholder with a format definition. In runtime mode, the placeholder is replaced by the current value in the specified format.  
CODESYS accepts the specified texts automatically into the “GlobalTextList” text list. Therefore, these texts can be localized.

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
</table>
| **Text**       | Character string (without single straight quotation marks) for the labeling the element. Add a line break by pressing the keyboard shortcut [Ctrl] + [Enter]. Example: Accesses: %i  
The variable that contains the current value for the placeholder is specified in the property “Text variable ➔ Text”. |
| **Tooltip**    | Character string (without single straight quotation marks) that is displayed as the tooltip of an element. Example: Number of valid accesses.  
The variable that contains the current value for the placeholder is specified in the property “Text variable ➔ Tooltip” |
Element property 'Text properties'
The properties contain fixed values for the text properties.

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Horizontal alignment&quot;</td>
<td>Horizontal alignment of the text within the element.</td>
</tr>
<tr>
<td>&quot;Vertical alignment&quot;</td>
<td>Vertical alignment of the text within the element.</td>
</tr>
<tr>
<td>&quot;Font&quot;</td>
<td>Example: &quot;Default&quot;&lt;br&gt;The &quot;Font&quot; dialog box opens.&lt;br&gt;▼: Drop-down list with style fonts.</td>
</tr>
<tr>
<td>&quot;Font color&quot;</td>
<td>Example: &quot;Black&quot;&lt;br&gt;The &quot;Color&quot; dialog box opens.&lt;br&gt;▼: Drop-down list with style colors.</td>
</tr>
<tr>
<td>&quot;Transparency&quot;</td>
<td>Whole number (value range from 0 to 255). This determines the transparency of the respective color.&lt;br&gt;Example: 255: The color is opaque.&lt;br&gt;0: The color is completely transparent.&lt;br&gt;Please note: If the color is a style color and already has a transparency value, then this property is write-protected.</td>
</tr>
</tbody>
</table>

Element property 'Text variables'
These properties are variables with contents that replace a format definition.

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Text variable&quot;</td>
<td>Variable (data type compliant with the format definition). It contains what is printed instead of the format definition.&lt;br&gt;Example: PLC_PRG.iAccesses&lt;br&gt;Note: The format definition is part of the text in the property “Texts ➔ Text”.&lt;br&gt;Note: If you specify a variable of type enumeration with text list support, then the name of the enumeration data type is added automatically in angle brackets after the variable name. Example: PLC_PRG.enVar &lt;enumeration name&gt;. Then the symbolic value of the enumeration component is printed instead of the numeric value when text is printed. Refer to the help page for the enumerations.</td>
</tr>
<tr>
<td>&quot;ToolTip variable&quot;</td>
<td>Variable (data type compliant with the format definition). It contains what is printed instead of the format definition.&lt;br&gt;Example: PLC_PRG.iAccessesInTooltip&lt;br&gt;Note: The format definition is part of the text in the property “Texts ➔ Tooltip”.</td>
</tr>
</tbody>
</table>

See also
- ❆ "Element property 'Text variables'" on page 1229
- ❆ Chapter 1.3.5.18.2 “Placeholders with format definition in the output text” on page 1546
- ❆ "Element property 'Texts'" on page 1226
- Enumerations
### Element property 'Dynamic texts'

Dynamic texts are variably indexed texts of a text list. At runtime, the text is displayed that is currently indexed in the variable.

| **“Text list”** | Variable (string) or name of the text list as a fixed string in single straight quotation marks.  
Example: 'Errorlist'  
▼: Drop-down list with the dialogs available in the text lists. |
| --- | --- |
| **“Text index”** | Text list ID. This refers to the desired output text.  
- As fixed string with the ID in single straight quotation marks.  
  Example: '1'  
- As a variable (STRING) for dynamically controlling the text output.  
  Example: strTextID  
  Sample assignment: PLC_PRG.strTextID := '1'; |
| **“Tooltip index”** | Text list ID. This refers to the desired output text.  
- As fixed string with the ID in single straight quotation marks.  
  Example: '2'  
- As a variable (STRING) for dynamically controlling the text output.  
  Example: strToolTipID  
  Sample assignment: PLC_PRG.strToolTipID := '2'; |

See also
- **Text list**

### Element property 'Font variables'

The variables allow for dynamic control of the text display.

| **“Font name”** | Variable (STRING). Includes the font of the text.  
Example: PLC_PRG.stFontVar := 'Arial';  
The selection of fonts corresponds to the default “Font” dialog. |
| --- | --- |
| **“Size”** | Variable (numeric data type). Contains the font size (in pixels or points). The applied unit is specified in brackets after the variable name.  
- <pt>: Points (default)  
  Example: PLC_PRG.iFontHeight <pt>  
  Code: iFontHeight : INT := 12;  
- <px>: Pixels  
  Example: PLC_PRG.iFontHeight <px>  
  Code: iFontHeight : INT := 19; |

If you click in the value field, a drop-down list opens on the right for setting the unit.

Hint: The font size is specified in points (example: Arial 12). Use points when the variable font size should match a font, for example if a font is set in the property “Text property ➔ Font”.

---

---
“Flags” Variable (DWORD). Contains the flags for displaying fonts.
Flags:
● 1: Italics
● 2: Bold
● 4: Underline
● 8: Strikethrough
Note: You can combine the font displays by adding the coding of the flags. For example, a bold and underlined text: PLC_PRG.dwFontType := 6;

“Character set” Variable (DWORD). Contains a character set number for the font.
The selection of character set numbers corresponds to the “Script” setting of the standard “Font” dialog.

“Color” Variable (DWORD). Includes the color of the text.
Example: PLC_PRG.dwColorFont := 16#FF000000;

“Flags for text alignment” Variable (integer data type). Contains the coding for text alignment.
Example: PLC_PRG.dwTextAlignment.
Coding:
● 0: Top left
● 1: Horizontal center
● 2: Right
● 4: Vertical center
● 8: Bottom
Note: You can combine the text alignments by adding the coding of the flags. For example, a vertical and horizontal centered text: PLC_PRG.dwFontType := 5;

Fixed values for displaying texts are set in “Text properties”.

See also
● § “Element property ‘Text properties’” on page 1227

Element property ‘Colorvariables’ The Element property is used as an interface for project variables to dynamically control colors at runtime.
<table>
<thead>
<tr>
<th><strong>“Toggle color”</strong></th>
<th>The property controls the toggled color at runtime.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Value assignment:</strong></td>
<td></td>
</tr>
<tr>
<td>● <strong>FALSE</strong>: The element is displayed with the color specified in the “Color” property.</td>
<td></td>
</tr>
<tr>
<td>● <strong>TRUE</strong>: The element is displayed with the color specified in the “Alarm color” property.</td>
<td></td>
</tr>
<tr>
<td><strong>Assignment options:</strong></td>
<td></td>
</tr>
<tr>
<td>● Placeholder for the user input variable</td>
<td></td>
</tr>
<tr>
<td>– “&lt;toggle/tap variable&gt;”</td>
<td></td>
</tr>
<tr>
<td>– “&lt;NOT toggle/tap variable&gt;”</td>
<td></td>
</tr>
</tbody>
</table>

The color change is not controlled by its own variable, but by a user input variable.

Note: Specify a variable for the mouse events “Tap” or “Toggle” in the input configuration of the element. Only then is the placeholder set. If you configure a variable in both “Toggle” and “Tap”, then the variable specified in “Tap” is used.

Hint: Click the symbol to insert the placeholder “<toggle/tap variable>”. When you activate the “Inputconfiguration”, “Tap FALSE” property, then the “<NOT toggle/tap variable>” placeholder is displayed.

● Instance path of a project variable (BOOL)
  Example: PLC_PRG.xColorIsToggled

Note: In the code, declare and implement the variable specified here. Its value assignment determines when the color changes.

| **“Normal state”** | The properties listed below control the color depending on the state. The normal state is in effect if the variable in “Color variables”, “Toggle color” is not defined or it has the value FALSE. |
| **“Alarm state”** | The alarm state is in effect if the variable in “Color variables”, “Toggle color” has the value TRUE. |

<table>
<thead>
<tr>
<th><strong>“Frame color”</strong></th>
<th>Assignment options:</th>
</tr>
</thead>
<tbody>
<tr>
<td>● Variable (DWORD) for the frame color</td>
<td></td>
</tr>
<tr>
<td>Example: PLC_PRG.dwBorderColor</td>
<td></td>
</tr>
<tr>
<td>● Color literal</td>
<td></td>
</tr>
<tr>
<td>Example of green and opaque: 16#FF00FF00</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>“Filling color”</strong></th>
<th>Assignment options:</th>
</tr>
</thead>
<tbody>
<tr>
<td>● Variable (DWORD) for the fill color</td>
<td></td>
</tr>
<tr>
<td>Example: PLC_PRG.dwFillColor</td>
<td></td>
</tr>
<tr>
<td>● Color literal</td>
<td></td>
</tr>
<tr>
<td>Example of gray and opaque: 16#FF888888</td>
<td></td>
</tr>
</tbody>
</table>

The transparency part of the color value is evaluated only if the “Activate semi-transparent drawing” option of the visualization manager is selected.

Select the “Advanced” option in the toolbar of the properties view. Then all element properties are visible.

See also

● Chapter 1.3.5.8.3 “Animating a color display” on page 1152
The properties contain IEC variables for controlling the position of the element dynamically. The reference point is the upper left corner of the element. In runtime mode, the entire element is moved.

<table>
<thead>
<tr>
<th>“Movement”</th>
</tr>
</thead>
</table>
| **“X”** | Variable (numeric data type). Defines the X position (in pixels).  
**Example:** PLC_PRG.iPos_X.  
Increasing this value in runtime mode moves the element to the right. |
| **“Y”** | Variable (numeric data type). Defines the Y position (in pixels).  
**Example:** PLC_PRG.iPos_Y.  
Increasing this value in runtime mode moves the element downwards. |
| **“Rotation”** | Variable (numeric data type). Defines the angle of rotation (in degrees).  
**Example:** PLC_PRG.iAngle1.  
The midpoint of the element rotates at the “Center” point. This rotation point is shown as the symbol.  
In runtime mode, the alignment of the element remains the same with respect to the coordinate system of the visualization. Increasing the value rotates the element to the right. |
| **“Interior rotation”** | Variable (numeric data type). Defines the angle of rotation (in degrees).  
**Example:** PLC_PRG.iAngle2.  
In runtime mode, the element rotates about the point of rotation specified in “Center” according to the value of the variable. In addition, the alignment of the element rotates according to the coordinate system of the visualization. Increasing the value in the code rotates clockwise.  
The rotation point is shown as the symbol.  
Note: If a static angle of rotation is specified in the “Position ➔ Angle” property, then the static angle of rotation is added to the variable angle of rotation (offset) when the visualization is executed. |

You can link the variables to a unit conversion.

The “X”, “Y”, “Rotation”, and “Interior rotation” properties are supported by the “Client Animation” functionality.
The variables control the element behavior dynamically.

**Element property 'State variables'**

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Invisible&quot;</td>
<td>Variable (BOOL). Toggles the visibility of the element. TRUE: The element is not visible at runtime.</td>
</tr>
<tr>
<td>&quot;Deactivate inputs&quot;</td>
<td>Variable (BOOL). Toggles the operability of the element. TRUE: User inputs do not have any effect in runtime more. The element is shown as deactivated.</td>
</tr>
</tbody>
</table>

The "Invisible" property is supported by the "Client Animation" functionality.

These properties are available only when you select the "Preview: Support client animations and overlay of native elements" option in the Visualization Manager.

You can animate a movement, a rotation, and the visibility of a visualization element.

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Animation duration&quot;</td>
<td>Time that the element executes an animation (in milliseconds) Example: 500</td>
</tr>
<tr>
<td>&quot;Move to the foreground&quot;</td>
<td>Property value (BOOL) TRUE: At runtime, the element is displayed in the foreground. FALSE: At runtime, the element is displayed in the layer where it was inserted in the visualization editor.</td>
</tr>
</tbody>
</table>

Requirement: User management is set up for the visualization.

**Element property 'Access rights'**

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Access rights&quot;</td>
<td>Opens the &quot;Access rights&quot; dialog. There you can edit the access privileges for the element. Status messages: &quot;Not set. Full rights.&quot;: Access rights for all user groups: &quot;operable&quot; &quot;Rights are set: Limited rights&quot;: Access is restricted for at least one group.</td>
</tr>
</tbody>
</table>

See also

- Chapter 1.3.5.21.5 “Using client animation” on page 1962

- Chapter 1.3.5.19.3.1 “Dialog ‘Access Rights’” on page 1583
Visualization element ‘Slider’

Symbol:

Tag: “Common controls”

The element adjusts the value of a variable, depending on the position of the slider within the slider bar. You define the value range of the slider bar by means of the scale start and scale end.

**Element properties**

<table>
<thead>
<tr>
<th>“Element name”</th>
<th>Example: Speed controller conveyor belt 1</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Optional</td>
</tr>
<tr>
<td></td>
<td>Hint: Assign individual names for elements so that they are found faster in the element list.</td>
</tr>
</tbody>
</table>

| “Type of element”       | “Slider”                                   |

**Element property ‘Position’**

The position defines the location and size of the element in the visualization window. These are based on the Cartesian coordinate system. The origin is located at the upper left corner of the window. The positive horizontal x-axis runs to the right. The positive vertical y-axis runs downwards.

<table>
<thead>
<tr>
<th>“X”</th>
<th>X coordinate of the upper left corner of the element</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Specified in pixels.</td>
</tr>
<tr>
<td></td>
<td>Example: 10.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>“Y”</th>
<th>Y coordinate of the upper left corner of the element</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Specified in pixels.</td>
</tr>
<tr>
<td></td>
<td>Example: 10.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>“Width”</th>
<th>Specified in pixels.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Example: 150</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>“Height”</th>
<th>Specified in pixels.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Example: 30</td>
</tr>
</tbody>
</table>

You can also change the values by dragging the box symbols (¶) to other positions in the editor.

See also

- Chapter 1.3.5.3.2 “Positioning elements, adapting size and layer” on page 1113
Element property 'Center'

The properties contain fixed values for the coordinates of the point of rotation. This point of rotation is shown as the symbol. The point is used as the center for rotating and scaling.

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>X-coordinate of the point of rotation</td>
</tr>
<tr>
<td>Y</td>
<td>Y-coordinate of the point of rotation</td>
</tr>
</tbody>
</table>

You can also change the values by dragging the symbols (⊕) to other positions in the editor.

---

Element property 'Variable'

Variable (numeric data type)

Example: PLC_PRG.rSlider

When executed, the variable assigns a value that corresponds to the position of the slider in the bar.

---

Element property 'Page size'

Page size

- As a fixed value, for example 10
- As an IEC variable of data type integer

Requirement: The "Move to click" element property is not activated.

---

Element property 'Move to click'

Function of the slider at visualization runtime when it is clicked:

- The slider moves to the clicked position.
- The slider moves to the value (defined in the "Page size" element property) in the direction of the click.

---

Element property 'Scale'

- "Show scale": The element has a visible scale.
  
  Note: This option is available for the slider only.

- "Scale start": Least value of the scale and the lower limit of the value range for the element.
  
  Example: 0

  The property "Variable" is shown below.

- "Variable": Variable (integer data type). Contains the scale start.
  
  Example: PLC_PRG.iScaleStart

  Declaration:

  ```
  PROGRAM PLC_PRG
  VAR
    iScaleStart : INT := 0;
  END_VAR
  ```

- "Scale end": Greatest value of the scale and the upper limit of the value range for the element.
  
  Example: 100

  The property "Variable" is shown below.
<table>
<thead>
<tr>
<th><strong>Variable</strong></th>
<th>Variable (integer data type). Contains the scale end.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Example:</strong></td>
<td>PLC_PRG.iScaleEnd</td>
</tr>
<tr>
<td><strong>Declaration:</strong></td>
<td></td>
</tr>
<tr>
<td>PROGRAM PLC_PRG</td>
<td></td>
</tr>
<tr>
<td>VAR</td>
<td></td>
</tr>
<tr>
<td>iScaleEnd : INT := 120;</td>
<td></td>
</tr>
<tr>
<td>END_VAR</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Main scale</strong></th>
<th>Distance between two tick marks on the rough scale.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Example:</strong></td>
<td>10</td>
</tr>
<tr>
<td>✿: The property &quot;Variable&quot; is shown below.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Variable</strong></th>
<th>Variable (integer data type). Contains the distance.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Example:</strong></td>
<td>PLC_PRG.iMainScale</td>
</tr>
<tr>
<td><strong>Declaration:</strong></td>
<td></td>
</tr>
<tr>
<td>PROGRAM PLC_PRG</td>
<td></td>
</tr>
<tr>
<td>VAR</td>
<td></td>
</tr>
<tr>
<td>iMainScale : INT := 20;</td>
<td></td>
</tr>
<tr>
<td>END_VAR</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Subscale</strong></th>
<th>Distance between two dashes on the fine scale. You can hide the fine scale by setting the value to 0.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Example:</strong></td>
<td>2</td>
</tr>
<tr>
<td>✿: The property &quot;Variable&quot; is shown below.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Variable</strong></th>
<th>Variable (integer data type). Contains the distance.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Example:</strong></td>
<td>PLC_PRG.iSubScale</td>
</tr>
<tr>
<td><strong>Declaration:</strong></td>
<td></td>
</tr>
<tr>
<td>PROGRAM PLC_PRG</td>
<td></td>
</tr>
<tr>
<td>VAR</td>
<td></td>
</tr>
<tr>
<td>iMainScale : INT := 5;</td>
<td></td>
</tr>
<tr>
<td>END_VAR</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Scale format (C Syntax)</strong></th>
<th>Formatting of the scale label (example: %d %s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Note: This property is available for the slider only.</td>
<td></td>
</tr>
</tbody>
</table>

| **Scale proportion** | Size of the scale (in %) of the total size |

**Element property 'Bar'**
The property defines the representation of scaling and direction of travel.
"Diagram type" | The drop-down list varies depending on the alignment of the diagram.  
Horizontal  
- "Top": Scale is above the slider.  
- "Bottom": Scale is below the slider.  
- "Top and bottom": Two scales frame the slider above and below.  
Vertical  
- Left: Scale is left of the slider.  
- Right: Scale is right of the slider.  
- Left and right: Two scales frame the slider on the left and the right.

"Orientation" | Alignment of the slider; defined by the ratio of width to height.  
- "Horizontal"  
- "Vertical"  
You can modify the alignment in the visualization editor by using the pointing device to adjust the width and height of the scrollbar.

"Running direction" | The drop-down list varies depending on the alignment of the slider.  
Horizontal  
- "Left to right": Scale starts at the left.  
- "Right to left": Scale starts at the right.  
Vertical  
- "Bottom to top": Scale starts at the bottom.  
- "Top to bottom": Scale starts at the top.

Element property 'Absolute movement' | The properties contain IEC variables for controlling the position of the element dynamically. The reference point is the upper left corner of the element. In runtime mode, the entire element is moved.

"Movement" |  
"X" | Variable (numeric data type). Defines the X position (in pixels).  
Example: PLC_PRG.iPos_X.  
Increasing this value in runtime mode moves the element to the right.

"Y" | Variable (numeric data type). Defines the Y position (in pixels).  
Example: PLC_PRG.iPos_Y.  
Increasing this value in runtime mode moves the element downwards.

"Rotation" | Variable (numeric data type). Defines the angle of rotation (in degrees).  
Example: PLC_PRG.iAngle1.  
The midpoint of the element rotates at the "Center" point. This rotation point is shown as the symbol.  
In runtime mode, the alignment of the element remains the same with respect to the coordinate system of the visualization. Increasing the value rotates the element to the right.
"Interior rotation"

Variable (numeric data type). Defines the angle of rotation (in degrees).

**Example:** PLC_PRG.iAngle2.

In runtime mode, the element rotates about the point of rotation specified in "Center" according to the value of the variable. In addition, the alignment of the element rotates according to the coordinate system of the visualization. Increasing the value in the code rotates clockwise.

The rotation point is shown as the symbol.

Note: If a static angle of rotation is specified in the "Position ➔ Angle" property, then the static angle of rotation is added to the variable angle of rotation (offset) when the visualization is executed.

You can link the variables to a unit conversion.

The “X”, “Y”, “Rotation”, and “Interior rotation” properties are supported by the "Client Animation" functionality.

See also

- Unit conversion
- 

**Element property 'State variables'**

The variables control the element behavior dynamically.

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Invisible&quot;</td>
<td>Variable (BOOL). Toggles the visibility of the element. TRUE: The element is not visible at runtime.</td>
</tr>
<tr>
<td>&quot;Deactivate inputs&quot;</td>
<td>Variable (BOOL). Toggles the operability of the element. TRUE: User inputs do not have any effect in runtime more. The element is shown as deactivated.</td>
</tr>
</tbody>
</table>

The "Invisible" property is supported by the "Client Animation" functionality.

See also

- 

These properties are available only when you select the “Preview: Support client animations and overlay of native elements” option in the Visualization Manager.

You can animate a movement, a rotation, and the visibility of a visualization element.
**"Animation duration"**

Time that the element executes an animation (in milliseconds)

**Example:** 500

---

**"Move to the foreground"**

Property value (BOOL)

**TRUE:** At runtime, the element is displayed in the foreground.

**FALSE:** At runtime, the element is displayed in the layer where it was inserted in the visualization editor.

See also

- § Chapter 1.3.5.21.5 “Using client animation” on page 1962

---

**Element property 'Access rights'**

Requirement: User management is set up for the visualization.

**"Access rights"**

Opens the “Access rights” dialog. There you can edit the access privileges for the element.

Status messages:

- “Not set. Full rights.”: Access rights for all user groups: “operable”
- “Rights are set: Limited rights”: Access is restricted for at least one group.

See also

- § Chapter 1.3.5.19.3.1 “Dialog ‘Access Rights’” on page 1583

See also

-  

---

**Visualization element 'SpinControl'**

Symbol:

![Symbol](Image)

Tag: “Common controls”

The element increments or decrements the value of a variable in defined intervals.

---

**Element properties**

**"Element name"**

**Example:** Speed controller conveyor belt

Optional

Hint: Assign individual names for elements so that they are found faster in the element list.

**"Type of element"**

“SpinControl”

---

**Element property 'Position'**

The position defines the location and size of the element in the visualization window. These are based on the Cartesian coordinate system. The origin is located at the upper left corner of the window. The positive horizontal x-axis runs to the right. The positive vertical y-axis runs downwards.
"X"  
X coordinate of the upper left corner of the element  
Specified in pixels.  
Example: 10.

"Y"  
Y coordinate of the upper left corner of the element  
Specified in pixels.  
Example: 10.

"Width"  
Specified in pixels.  
Example: 150

"Height"  
Specified in pixels.  
Example: 30

---

You can also change the values by dragging the box symbols (ドラッグ) to other positions in the editor.

See also
- Chapter 1.3.5.3.2 “Positioning elements, adapting size and layer” on page 1113

Element property 'Center'  
The properties contain fixed values for the coordinates of the point of rotation. This point of rotation is shown as the symbol. The point is used as the center for rotating and scaling.

"X"  
X-coordinate of the point of rotation

"Y"  
Y-coordinate of the point of rotation

---

You can also change the values by dragging the symbols (ドラッグ) to other positions in the editor.

"Variable"  
Variable (numeric data type)  
Beispiel PLC_PRG.iTemp

"Number format"  
Format of the value in printf syntax  
Example: %d, %.2f

"Interval"  
Interval used for modification of the value
**Element property 'Text properties'**

- **Minimum value**
  - Lower limit of the output value
  - fixed value
  - Variable (INT)

- **Maximum value**
  - Upper limit of the output value
  - fixed value
  - Variable (INT)

The properties contain fixed values for the text properties.

**Usage of**

- “Default style values”: The values of the visualization style are used.
- “Individual settings”: The "Individual text properties" property group is shown. The values can be customized here.

**“Individual text properties”**
Requirement: The “Individual settings” text property is defined.

- **Horizontal alignment**
  - Horizontal alignment of the text within the element.

- **Vertical alignment**
  - Vertical alignment of the text within the element.

- **Font**
  - Example: “Default”
  - The “Font” dialog box opens.
  - †: Drop-down list with style fonts.

- **Font color**
  - Example: “Black”
  - †: The “Color” dialog box opens.
  - †: Drop-down list with style colors.

- **Transparency**
  - Whole number (value range from 0 to 255). This determines the transparency of the respective color.
  - Example: 255: The color is opaque.
  - 0: The color is completely transparent.
  - Please note: If the color is a style color and already has a transparency value, then this property is write-protected.

**Element property 'Colorvariables'**

The Element property is used as an interface for project variables to dynamically control colors at runtime.
**“Toggle color”**

The property controls the toggled color at runtime.

**Value assignment:**
- **FALSE:** The element is displayed with the color specified in the “Color” property.
- **TRUE:** The element is displayed with the color specified in the “Alarm color” property.

**Assigning the property:**
- Placeholder for the user input variable
  - `<toggle/tap variable>`
  - `<NOT toggle/tap variable>`

The color change is not controlled by its own variable, but by a user input variable.

*Note:* Specify a variable for the mouse events “Tap” or “Toggle” in the input configuration of the element. Only then is the placeholder set. If you configure a variable in both “Toggle” and “Tap”, then the variable specified in “Tap” is used.

*Hint:* Click the symbol to insert the placeholder “<toggle/tap variable>”. When you activate the “Input configuration”, “Tap FALSE” property, then the “<NOT toggle/tap variable>” placeholder is displayed.

- **Instance path of a project variable (BOOL)**
  - Example: PLC_PRG.xColorIsToggled

*Note:* In the code, declare and implement the variable specified here. Its value assignment determines when the color changes.

---

**The transparency part of the color value is evaluated only if the “Activate semi-transparent drawing” option of the visualization manager is selected.**

**Select the “Advanced” option in the toolbar of the properties view. Then all element properties are visible.**

See also
- § Chapter 1.3.5.8.3 “Animating a color display” on page 1152
- §

**Element property 'Absolute movement'**

The properties contain IEC variables for controlling the position of the element dynamically. The reference point is the upper left corner of the element. In runtime mode, the entire element is moved.

**“Movement”**

<table>
<thead>
<tr>
<th><strong>“X”</strong></th>
<th>Variable (numeric data type). Defines the X position (in pixels).</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Example: PLC_PRG.iPos_X.</td>
</tr>
<tr>
<td></td>
<td>Increasing this value in runtime mode moves the element to the right.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>“Y”</strong></th>
<th>Variable (numeric data type). Defines the Y position (in pixels).</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Example: PLC_PRG.iPos_Y.</td>
</tr>
<tr>
<td></td>
<td>Increasing this value in runtime mode moves the element downwards.</td>
</tr>
</tbody>
</table>
"Rotation" Variable (numeric data type). Defines the angle of rotation (in degrees).

Example: PLC_PRG.iAngle1.
The midpoint of the element rotates at the “Center” point. This rotation point is shown as the symbol.
In runtime mode, the alignment of the element remains the same with respect to the coordinate system of the visualization. Increasing the value rotates the element to the right.

"Interior rotation" Variable (numeric data type). Defines the angle of rotation (in degrees).

Example: PLC_PRG.iAngle2.
In runtime mode, the element rotates about the point of rotation specified in “Center” according to the value of the variable. In addition, the alignment of the element rotates according to the coordinate system of the visualization. Increasing the value in the code rotates clockwise.
The rotation point is shown as the symbol.
Note: If a static angle of rotation is specified in the “Position Angle” property, then the static angle of rotation is added to the variable angle of rotation (offset) when the visualization is executed.

You can link the variables to a unit conversion.

The “X”, “Y”, “Rotation”, and “Interior rotation” properties are supported by the "Client Animation" functionality.

Element property ‘State variables’ The variables control the element behavior dynamically.

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Invisible&quot;</td>
<td>Variable (BOOL). Toggles the visibility of the element. TRUE: The element is not visible at runtime.</td>
</tr>
<tr>
<td>&quot;Deactivate inputs&quot;</td>
<td>Variable (BOOL). Toggles the operability of the element. TRUE: User inputs do not have any effect in runtime more. The element is shown as deactivated.</td>
</tr>
</tbody>
</table>
The “Invisible” property is supported by the “Client Animation” functionality.

See also

These properties are available only when you select the “Preview: Support client animations and overlay of native elements” option in the Visualization Manager.

You can animate a movement, a rotation, and the visibility of a visualization element.

<table>
<thead>
<tr>
<th>“Animation duration”</th>
<th>Time that the element executes an animation (in milliseconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Example: 500</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>“Move to the foreground”</th>
<th>Property value (BOOL)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>TRUE: At runtime, the element is displayed in the foreground.</td>
</tr>
<tr>
<td></td>
<td>FALSE: At runtime, the element is displayed in the layer where it was inserted in the visualization editor.</td>
</tr>
</tbody>
</table>

See also

- Chapter 1.3.5.21.5 “Using client animation” on page 1962

Element property 'Input configuration'

The properties contain the configurations for the user input when using the mouse or keyboard. User input is a user event from the perspective of the element.

The input configuration refers to the text area of the element only, not the two buttons.

The “Configure” button opens the “Input configuration” dialog box for creating or modifying a user input configuration.

A configuration contains one or more input actions for the respective input event. Existing input actions are displayed below it.

Example: “Execute ST code”: \[ PLC\_PRG\_.i\_x := 0; \]

<table>
<thead>
<tr>
<th>“OnDialogClosed”</th>
<th>Input event: The user closes the dialog box.</th>
</tr>
</thead>
<tbody>
<tr>
<td>“OnMouseClick”</td>
<td>Input event: The user clicks the element completely. The mouse button is clicked and released.</td>
</tr>
<tr>
<td>“OnMouseDown”</td>
<td>Input event: The user clicks down on the element only.</td>
</tr>
<tr>
<td>“OnMouseEnter”</td>
<td>Input event: The user drags the mouse pointer to the element.</td>
</tr>
<tr>
<td>“OnMouseLeave”</td>
<td>Input event: The user drags the mouse pointer away from the element.</td>
</tr>
<tr>
<td>“OnMouseMove”</td>
<td>Input event: The user moves the mouse pointer over the element area.</td>
</tr>
<tr>
<td>“OnMouseUp”</td>
<td>Input event: The user releases the mouse button over the element area.</td>
</tr>
</tbody>
</table>

See also

- Chapter 1.3.5.19.3.6 “Dialog ‘Input Configuration’” on page 1586
<table>
<thead>
<tr>
<th><strong>“Tap”</strong></th>
<th>When a mouse click event occurs, the variable defined in “Variable” is described in the application. The coding depends on the options “Tap FALSE” and “Tap on enter if captured”.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Variable</strong></td>
<td>Variable (BOOL). Contains the information whether a mouse click event exists.</td>
</tr>
<tr>
<td></td>
<td>Example: PLC_PRG.bIsTapped</td>
</tr>
<tr>
<td></td>
<td>TRUE: A mouse click event exists. It lasts while the user presses the mouse button over the element. It ends when the button is released.</td>
</tr>
<tr>
<td></td>
<td>FALSE: A mouse click event does not exist.</td>
</tr>
<tr>
<td></td>
<td>Requirement: The “Tap FALSE” option is not activated.</td>
</tr>
<tr>
<td><strong>Tap FALSE</strong></td>
<td>☑: The mouse click event leads to a complementary value in “Variable”.</td>
</tr>
<tr>
<td></td>
<td>TRUE: A mouse click event does not exist.</td>
</tr>
<tr>
<td></td>
<td>FALSE: While the mouse click event exists.</td>
</tr>
<tr>
<td><strong>Tap on enter if captured</strong></td>
<td>☑: During user input, it is also taken into consideration whether the mouse pointer is dragged within the element area or not while the mouse button is pressed.</td>
</tr>
<tr>
<td></td>
<td>TRUE: While the mouse click event exists and the mouse pointer is moved over the element area.</td>
</tr>
<tr>
<td></td>
<td>FALSE: A mouse click event does not exist. Or the user moves the mouse pointer outside of the element area while the mouse button is pressed.</td>
</tr>
<tr>
<td></td>
<td>The value is TRUE again as soon as the user moves the pointer back to the element area. The mouse is then captured.</td>
</tr>
<tr>
<td><strong>Shift</strong></td>
<td>When a mouse click event occurs, the variable here is described in the application. When the mouse click event ends, its value is toggled with the “Toggle on up if captured” option.</td>
</tr>
<tr>
<td><strong>Variable</strong></td>
<td>Variable (BOOL). Its value toggled when the mouse click event is ended. This is when the user releases the mouse button while the mouse pointer is over the element area.</td>
</tr>
<tr>
<td></td>
<td>If the user releases the mouse button while the mouse pointer is outside of the element area, then the mouse click event is not ended and the value is not toggled.</td>
</tr>
<tr>
<td></td>
<td>Tip: The user can cancel a started toggle input by dragging the mouse pointer out of the element area.</td>
</tr>
<tr>
<td><strong>Toggle on up if captured</strong></td>
<td>☑: The value toggles regardless of where the mouse pointer is when the mouse button is released. The mouse is then captured.</td>
</tr>
<tr>
<td><strong>Hotkeys</strong></td>
<td>Keyboard shortcut on the element for triggering specific input actions.</td>
</tr>
<tr>
<td></td>
<td>When the keyboard shortcut event occurs, the input actions in the “Event(s)” property are triggered. In this way, it is not the input action itself that leads to this input action, but the mouse input action.</td>
</tr>
<tr>
<td><strong>Key</strong></td>
<td>Key pressed for input action.</td>
</tr>
<tr>
<td></td>
<td>Example: [T]</td>
</tr>
<tr>
<td></td>
<td>Note: The following properties appear when a key is selected.</td>
</tr>
</tbody>
</table>
"Event(s)"

- "None"
- "Mouse down": Pressing the key triggers the input actions that are configured in the "OnMouseDown" property.
- "Mouse up": Releasing the key triggers the input actions that are configured in the "OnMouseUp" property.
- "Mouse down/up": Pressing and releasing the key triggers the input actions that are configured in the "OnMouseDown" property and the "OnMouseUp" property.

"Shift"

☑ Combination with the Shift key
Example: [Shift]+[T].

"Control"

☑ Combination with the Ctrl key
Example: [Ctrl]+[T].

"Alt"

☑ Combination with the Alt key
Example: [Alt]+[T].

All keyboard shortcuts and their actions that are configured in the visualization are listed in the “Keyboard configuration” tab.

See also
- % Chapter 1.3.5.19.2.2 “Command 'Keyboard Configuration'” on page 1558
- % Chapter 1.3.5.19.3.6 “Dialog ‘Input Configuration’” on page 1586

Element property 'Access rights'

Requirement: User management is set up for the visualization.

"Access rights"

Opens the “Access rights” dialog. There you can edit the access privileges for the element.
Status messages:
- "Not set. Full rights.": Access rights for all user groups: “operable”
- "Rights are set: Limited rights": Access is restricted for at least one group.

See also
- % Chapter 1.3.5.19.3.1 “Dialog ‘Access Rights’” on page 1583

See also
- "Visibility" (Common controls)

Visualization element 'Invisible input'

Symbol:

Tag: “Common controls”

This element is displayed in the editor with a dashed line and is not visible in online mode. You define the behavior of the el in the input configuration.
Element properties

<table>
<thead>
<tr>
<th>“Element name”</th>
<th>Optional</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hint: Assign individual names for elements so that they are found faster in the element list.</td>
<td></td>
</tr>
<tr>
<td>Example: invisible_input_1</td>
<td></td>
</tr>
</tbody>
</table>

| “Type of element”     | “Invisible input”                             |

Element property 'Position'

The position defines the location and size of the element in the visualization window. These are based on the Cartesian coordinate system. The origin is located at the upper left corner of the window. The positive horizontal x-axis runs to the right. The positive vertical y-axis runs downwards.

<table>
<thead>
<tr>
<th>“X”</th>
<th>X coordinate of the upper left corner of the element</th>
</tr>
</thead>
<tbody>
<tr>
<td>Specified in pixels.</td>
<td></td>
</tr>
<tr>
<td>Example: 10.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>“Y”</th>
<th>Y coordinate of the upper left corner of the element</th>
</tr>
</thead>
<tbody>
<tr>
<td>Specified in pixels.</td>
<td></td>
</tr>
<tr>
<td>Example: 10.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>“Width”</th>
<th>Specified in pixels.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example: 150</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>“Height”</th>
<th>Specified in pixels.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example: 30</td>
<td></td>
</tr>
</tbody>
</table>

You can also change the values by dragging the box symbols (_drag) to other positions in the editor.

See also

- Chapter 1.3.5.3.2 “Positioning elements, adapting size and layer” on page 1113

Element property 'Center'

The properties contain fixed values for the coordinates of the point of rotation. This point of rotation is shown as the symbol. The point is used as the center for rotating and scaling.

<table>
<thead>
<tr>
<th>“X”</th>
<th>X-coordinate of the point of rotation</th>
</tr>
</thead>
</table>

| “Y”                  | Y-coordinate of the point of rotation                |

You can also change the values by dragging the symbols (drag) to other positions in the editor.

Element property 'Absolute movement'

The properties contain IEC variables for controlling the position of the element dynamically. The reference point is the upper left corner of the element. In runtime mode, the entire element is moved.
### “Movement”

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
<th>Example</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>“X”</td>
<td>Variable (numeric data type). Defines the X position (in pixels).</td>
<td>PLC_PRG.iPos_X.</td>
<td>Increasing this value in runtime mode moves the element to the right.</td>
</tr>
<tr>
<td>“Y”</td>
<td>Variable (numeric data type). Defines the Y position (in pixels).</td>
<td>PLC_PRG.iPos_Y.</td>
<td>Increasing this value in runtime mode moves the element downwards.</td>
</tr>
<tr>
<td>“Rotation”</td>
<td>Variable (numeric data type). Defines the angle of rotation (in degrees).</td>
<td>PLC_PRG.iAngle1.</td>
<td>The midpoint of the element rotates at the “Center” point. This rotation point is shown as the symbol. In runtime mode, the alignment of the element remains the same with respect to the coordinate system of the visualization. Increasing the value rotates the element to the right.</td>
</tr>
<tr>
<td>“Interior rotation”</td>
<td>Variable (numeric data type). Defines the angle of rotation (in degrees).</td>
<td>PLC_PRG.iAngle2.</td>
<td>In runtime mode, the element rotates about the point of rotation specified in “Center” according to the value of the variable. In addition, the alignment of the element rotates according to the coordinate system of the visualization. Increasing the value in the code rotates clockwise. The rotation point is shown as the symbol. Note: If a static angle of rotation is specified in the “Position ➔ Angle” property, then the static angle of rotation is added to the variable angle of rotation (offset) when the visualization is executed.</td>
</tr>
</tbody>
</table>

You can link the variables to a unit conversion.

The “X”, “Y”, “Rotation”, and “Interior rotation” properties are supported by the “Client Animation” functionality.

See also
- [Unit conversion](#)
The variables control the element behavior dynamically.

<table>
<thead>
<tr>
<th><strong>&quot;Deactivate inputs&quot;</strong></th>
<th>Variable (BOOL). Toggles the operability of the element. TRUE: User inputs do not have any effect in runtime more. The element is shown as deactivated.</th>
</tr>
</thead>
</table>

These properties are available only when you select the “Preview: Support client animations and overlay of native elements” option in the Visualization Manager.

You can animate a movement, a rotation, and the visibility of a visualization element.

| **"Animation duration"** | Time that the element executes an animation (in milliseconds)  
Example: 500 |
|-------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| **"Move to the foreground"** | Property value (BOOL)  
TRUE: At runtime, the element is displayed in the foreground.  
FALSE: At runtime, the element is displayed in the layer where it was inserted in the visualization editor. |
**“Tap”**

When a mouse click event occurs, the variable defined in “Variable” is described in the application. The coding depends on the options “Tap FALSE” and “Tap on enter if captured”.

**“Variable”**

Variable (BOOL) that is set on mouse click.

**Example:** PLC_PRG.bIsTapped

- **TRUE**: A mouse click event exists. It lasts while the user presses the mouse button over the element. It ends when the button is released.
- **FALSE**: A mouse click event does not exist.

**Requirement:** The “Tap FALSE” option is not activated.

**“Tap FALSE”**

- **☑**: The mouse click event leads to a complementary value in “Variable”.
  - **TRUE**: A mouse click event does not exist.
  - **FALSE**: While the mouse click event exists.

**“Tap on enter if captured”**

- **☑**: During user input, it is also taken into consideration whether the mouse pointer is dragged within the element area or not while the mouse button is pressed.
  - **TRUE**: While the mouse click event exists and the mouse pointer is moved over the element area.
  - **FALSE**: A mouse click event does not exist. Or the user moves the mouse pointer outside of the element area while the mouse button is pressed.

The value is **TRUE** again as soon as the user moves the pointer back to the element area. The mouse is then captured.

**“Shift”**

With the onset of a mouse click event, the variable is set; when the mouse click event is completed, the variable is reset.

**“Variable”**

Variable (BOOL). Its value toggled when the mouse click event is ended. This is when the user releases the mouse button while the mouse pointer is over the element area.

If the user releases the mouse button while the mouse pointer is outside of the element area, then the mouse click event is not ended and the value is not toggled.

**Tip:** The user can cancel a started toggle input by dragging the mouse pointer out of the element area.

**“Toggle on up if captured”**

- **☑**: The value toggles regardless of where the mouse pointer is when the mouse button is released. The mouse is then captured.

**“Keyboard shortcuts”**

Keyboard shortcut on the element for triggering specific input actions.

When the keyboard shortcut event occurs, the input actions in the “Event(s)” property are triggered. In this way, it is not the input action itself that leads to this input action, but the mouse input action.

**“Key”**

Key pressed for input action.

**Example:** [T]

**Note:** The following properties appear when a key is selected.
“Event(s)”

- “None”
- “Mouse down”: Pressing the key triggers the input actions that are configured in the “OnMouseDown” property.
- “Mouse up”: Releasing the key triggers the input actions that are configured in the “OnMouseUp” property.
- “Mouse down/up”: Pressing and releasing the key triggers the input actions that are configured in the “OnMouseDown” property and the “OnMouseUp” property.

“Shift”

- Combination with the Shift key
  Example: [Shift]+[T].

“Control”

- Combination with the Ctrl key
  Example: [Ctrl]+[T].

“Alt”

- Combination with the Alt key
  Example: [Alt]+[T].

All keyboard shortcuts and their actions that are configured in the visualization are listed in the “Keyboard configuration” tab.

See also
- ¶ Chapter 1.3.5.19.2.2 “Command ‘Keyboard Configuration’” on page 1558
- ¶ Chapter 1.3.5.19.3.6 “Dialog ‘Input Configuration’” on page 1586

Element property 'Access rights'

Requirement: User management is set up for the visualization.

“Access rights”

Opens the “Access rights” dialog. There you can edit the access privileges for the element.
Status messages:
- “Not set. Full rights.”: Access rights for all user groups: “operable”
- “Rights are set: Limited rights”: Access is restricted for at least one group.

See also
- ¶ Chapter 1.3.5.19.3.1 “Dialog ‘Access Rights’” on page 1583

Visualization element ‘Progress bar’

Symbol:

Tag: “Common controls”
The element represents the value of a variable as a progress bar.
**Element properties**

<table>
<thead>
<tr>
<th>“Element name”</th>
<th>Optional</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hint: Assign individual names for elements so that they are found faster in the element list.</td>
<td></td>
</tr>
<tr>
<td><strong>Example:</strong> Progress_Data_Transfer</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>“Type of element”</th>
<th>“Progress bar”</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>“Text ID”</th>
<th>Identification of the global text list</th>
</tr>
</thead>
<tbody>
<tr>
<td>Requirement: Text is configured in the property “Texts ➔ Text”.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>“Variable”</th>
<th>Variable (numeric data type). Represents the length of the progress bar.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>“Minimum value”</th>
<th>Value range of the variable</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>“Maximum value”</th>
<th></th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>“Style”</th>
<th>“Blocks”</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Bar”</td>
<td></td>
</tr>
</tbody>
</table>

**Element property 'Position'**  
The position defines the location and size of the element in the visualization window. These are based on the Cartesian coordinate system. The origin is located at the upper left corner of the window. The positive horizontal x-axis runs to the right. The positive vertical y-axis runs downwards.

<table>
<thead>
<tr>
<th>“X”</th>
<th>X coordinate of the upper left corner of the element</th>
</tr>
</thead>
<tbody>
<tr>
<td>Specified in pixels.</td>
<td></td>
</tr>
<tr>
<td><strong>Example:</strong> 10.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>“Y”</th>
<th>Y coordinate of the upper left corner of the element</th>
</tr>
</thead>
<tbody>
<tr>
<td>Specified in pixels.</td>
<td></td>
</tr>
<tr>
<td><strong>Example:</strong> 10.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>“Width”</th>
<th>Specified in pixels.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Example:</strong> 150</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>“Height”</th>
<th>Specified in pixels.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Example:</strong> 30</td>
<td></td>
</tr>
</tbody>
</table>

You can also change the values by dragging the box symbols (_drag) to other positions in the editor.

**See also**
- § Chapter 1.3.5.3.2 “Positioning elements, adapting size and layer” on page 1113

**Element property 'Center'**  
The properties contain fixed values for the coordinates of the point of rotation. This point of rotation is shown as the symbol. The point is used as the center for rotating and scaling.

<table>
<thead>
<tr>
<th>“X”</th>
<th>X-coordinate of the point of rotation</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Y”</td>
<td>Y-coordinate of the point of rotation</td>
</tr>
</tbody>
</table>
Element property 'Texts'

<table>
<thead>
<tr>
<th>&quot;Text&quot;</th>
<th>String label for the element.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example: Zoom</td>
<td></td>
</tr>
</tbody>
</table>

Element property 'Absolute movement'

The properties contain IEC variables for controlling the position of the element dynamically. The reference point is the upper left corner of the element. In runtime mode, the entire element is moved.

"Movement"

<table>
<thead>
<tr>
<th>&quot;X&quot;</th>
<th>Variable (numeric data type). Defines the X position (in pixels).</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example: PLC_PRG.iPos_X.</td>
<td>Increasing this value in runtime mode moves the element to the right.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>&quot;Y&quot;</th>
<th>Variable (numeric data type). Defines the Y position (in pixels).</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example: PLC_PRG.iPos_Y.</td>
<td>Increasing this value in runtime mode moves the element downwards.</td>
</tr>
</tbody>
</table>

"Rotation"

<table>
<thead>
<tr>
<th>&quot;Rotation&quot;</th>
<th>Variable (numeric data type). Defines the angle of rotation (in degrees).</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example: PLC_PRG.iAngle1.</td>
<td>The midpoint of the element rotates at the “Center” point. This rotation point is shown as the symbol.</td>
</tr>
<tr>
<td></td>
<td>In runtime mode, the alignment of the element remains the same with respect to the coordinate system of the visualization. Increasing the value rotates the element to the right.</td>
</tr>
</tbody>
</table>

"Interior rotation"

<table>
<thead>
<tr>
<th>&quot;Interior rotation&quot;</th>
<th>Variable (numeric data type). Defines the angle of rotation (in degrees).</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example: PLC_PRG.iAngle2.</td>
<td>In runtime mode, the element rotates about the point of rotation specified in “Center” according to the value of the variable. In addition, the alignment of the element rotates according to the coordinate system of the visualization. Increasing the value in the code rotates clockwise.</td>
</tr>
<tr>
<td></td>
<td>The rotation point is shown as the symbol.</td>
</tr>
<tr>
<td></td>
<td>Note: If a static angle of rotation is specified in the “Position Angle” property, then the static angle of rotation is added to the variable angle of rotation (offset) when the visualization is executed.</td>
</tr>
</tbody>
</table>
You can link the variables to a unit conversion.

The "X", "Y", "Rotation", and "Interior rotation" properties are supported by the "Client Animation" functionality.

See also
- Unit conversion

Element property 'State variables'

The variables control the element behavior dynamically.

<table>
<thead>
<tr>
<th><strong>&quot;Invisible&quot;</strong></th>
<th>Variable (BOOL). Toggles the visibility of the element.</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRUE: The element is not visible at runtime.</td>
<td></td>
</tr>
</tbody>
</table>

The "Invisible" property is supported by the "Client Animation" functionality.

See also
- 

These properties are available only when you select the "Preview: Support client animations and overlay of native elements" option in the Visualization Manager.

You can animate a movement, a rotation, and the visibility of a visualization element.

<table>
<thead>
<tr>
<th><strong>&quot;Animation duration&quot;</strong></th>
<th>Time that the element executes an animation (in milliseconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example: 500</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>&quot;Move to the foreground&quot;</strong></th>
<th>Property value (BOOL)</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRUE: At runtime, the element is displayed in the foreground.</td>
<td></td>
</tr>
<tr>
<td>FALSE: At runtime, the element is displayed in the layer where it was inserted in the visualization editor.</td>
<td></td>
</tr>
</tbody>
</table>

See also
- Chapter 1.3.5.21.5 “Using client animation” on page 1962

Element property 'Access rights'

Requirement: User management is set up for the visualization.
“Access rights” Opens the “Access rights” dialog. There you can edit the access privileges for the element.

Status messages:
- “Not set. Full rights.”: Access rights for all user groups: “operable”
- “Rights are set: Limited rights”: Access is restricted for at least one group.

See also
- Chapter 1.3.5.19.3.1 “Dialog ‘Access Rights’” on page 1583

Visualization element ‘Checkbox’

Symbol:

<table>
<thead>
<tr>
<th>Tag: “Common controls”</th>
</tr>
</thead>
</table>

The element is used for setting and resetting a Boolean variable. The set state is represented by a check mark.

Element properties

<table>
<thead>
<tr>
<th>“Element name”</th>
<th>Example: signal_tone_for_parts_deficit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Optional</td>
<td></td>
</tr>
</tbody>
</table>
| Hint: Assign individual names for elements so that they are found faster in the element list.

<table>
<thead>
<tr>
<th>“Type of element”</th>
<th>“Checkbox”</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>“Text ID”</th>
<th>ID for the text in the “GlobalTextList”</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example: 22</td>
<td></td>
</tr>
</tbody>
</table>

The text ID cannot be modified. As soon as you specify and save a text in “Texts” - “Text”, CODESYS automatically creates an entry in the “GlobalTextList” and displays the ID here.

Element property ‘Position’

The position defines the location and size of the element in the visualization window. These are based on the Cartesian coordinate system. The origin is located at the upper left corner of the window. The positive horizontal x-axis runs to the right. The positive vertical y-axis runs downwards.

<table>
<thead>
<tr>
<th>“X”</th>
<th>X coordinate of the upper left corner of the element</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Specified in pixels. Example: 10.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>“Y”</th>
<th>Y coordinate of the upper left corner of the element</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Specified in pixels. Example: 10.</td>
</tr>
</tbody>
</table>
### "Width"
Specified in pixels.
Example: 150

### "Height"
Specified in pixels.
Example: 30

You can also change the values by dragging the box symbols (quito) to other positions in the editor.

See also
-  Chapter 1.3.5.3.2 “Positioning elements, adapting size and layer” on page 1113

#### Element property 'Center'
The properties contain fixed values for the coordinates of the point of rotation. This point of rotation is shown as the (quito) symbol. The point is used as the center for rotating and scaling.

<table>
<thead>
<tr>
<th>“X”</th>
<th>X-coordinate of the point of rotation</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Y”</td>
<td>Y-coordinate of the point of rotation</td>
</tr>
</tbody>
</table>

You can also change the values by dragging the symbols (quito) to other positions in the editor.

#### "Variable"
Variable of type BOOL.
Example: "PLC_PRG.xIsTrue"

#### "Frame size"
Distance of the element to the edge
Example: “From style”

#### Element property 'Texts'
The properties contains character strings for labeling the element.
CODESYS accepts the specified texts automatically into the “GlobalTextList” text list. Therefore, these texts can be localized.

<table>
<thead>
<tr>
<th>“Text”</th>
<th>Character string (without single straight quotation marks) for the labeling the element.</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Tooltip”</td>
<td>Character string (without single straight quotation marks) that is displayed as the tooltip of an element.</td>
</tr>
</tbody>
</table>

Example:
- Text: Axis 1.
- Tooltip: Parameters of Axis 1.

See also
-  

#### Element property 'Text properties'
The properties contain fixed values for the text properties.
<table>
<thead>
<tr>
<th>“Usage of”</th>
<th>“Default style values”: The values of the visualization style are used.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>“Individual settings”: The “Individual text properties” property group is shown</td>
</tr>
<tr>
<td></td>
<td>The values can be customized here.</td>
</tr>
<tr>
<td>“Individual text properties”</td>
<td>Requirement: The “Individual settings” text property is defined.</td>
</tr>
<tr>
<td>“Horizontal alignment”</td>
<td>Horizontal alignment of the text within the element.</td>
</tr>
<tr>
<td>“Vertical alignment”</td>
<td>Vertical alignment of the text within the element.</td>
</tr>
<tr>
<td>“Text format”</td>
<td>Definition for displaying texts that are too long</td>
</tr>
<tr>
<td></td>
<td>“Default”: The long text is truncated.</td>
</tr>
<tr>
<td></td>
<td>“Line break”: The text is split into parts.</td>
</tr>
<tr>
<td></td>
<td>“Ellipsis”: The visible text ends with “…”, indicating that it is not complete.</td>
</tr>
<tr>
<td>“Font”</td>
<td>Example: “Default”</td>
</tr>
<tr>
<td></td>
<td>📔: The “Font” dialog box opens.</td>
</tr>
<tr>
<td></td>
<td>✶: Drop-down list with style fonts.</td>
</tr>
<tr>
<td>“Font color”</td>
<td>Example: “Black”</td>
</tr>
<tr>
<td></td>
<td>📔: The “Color” dialog box opens.</td>
</tr>
<tr>
<td></td>
<td>✶: Drop-down list with style colors.</td>
</tr>
<tr>
<td>“Transparency”</td>
<td>Whole number (value range from 0 to 255). This determines the transparency of the respective color.</td>
</tr>
<tr>
<td></td>
<td>Example: 255: The color is opaque.</td>
</tr>
<tr>
<td></td>
<td>0: The color is completely transparent.</td>
</tr>
<tr>
<td></td>
<td>Please note: If the color is a style color and already has a transparency value, then this property is write-protected.</td>
</tr>
</tbody>
</table>

**Element property 'Absolute movement'**

The properties contain IEC variables for controlling the position of the element dynamically. The reference point is the upper left corner of the element. In runtime mode, the entire element is moved.

<table>
<thead>
<tr>
<th>“Movement”</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>“X”</strong></td>
<td>Variable (numeric data type). Defines the X position (in pixels).</td>
</tr>
<tr>
<td></td>
<td>Example: PLC_PRG.iPos_X.</td>
</tr>
<tr>
<td></td>
<td>Increasing this value in runtime mode moves the element to the right.</td>
</tr>
<tr>
<td><strong>“Y”</strong></td>
<td>Variable (numeric data type). Defines the Y position (in pixels).</td>
</tr>
<tr>
<td></td>
<td>Example: PLC_PRG.iPos_Y.</td>
</tr>
<tr>
<td></td>
<td>Increasing this value in runtime mode moves the element downwards.</td>
</tr>
</tbody>
</table>
"Rotation" Variable (numeric data type). Defines the angle of rotation (in degrees).
Example: PLC_PRG.iAngle1.
The midpoint of the element rotates at the “Center” point. This rotation point is shown as the symbol.
In runtime mode, the alignment of the element remains the same with respect to the coordinate system of the visualization. Increasing the value rotates the element to the right.

"Interior rotation" Variable (numeric data type). Defines the angle of rotation (in degrees).
Example: PLC_PRG.iAngle2.
In runtime mode, the element rotates about the point of rotation specified in “Center” according to the value of the variable. In addition, the alignment of the element rotates according to the coordinate system of the visualization. Increasing the value in the code rotates clockwise.
The rotation point is shown as the symbol.
Note: If a static angle of rotation is specified in the "Position Angle" property, then the static angle of rotation is added to the variable angle of rotation (offset) when the visualization is executed.

You can link the variables to a unit conversion.

The “X”, “Y”, “Rotation”, and “Interior rotation” properties are supported by the "Client Animation" functionality.

See also
- Unit conversion

Element property ‘State variables’ The variables control the element behavior dynamically.

<table>
<thead>
<tr>
<th>“Invisible”</th>
<th>Variable (BOOL). Toggles the visibility of the element. TRUE: The element is not visible at runtime.</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Deactivate inputs”</td>
<td>Variable (BOOL). Toggles the operability of the element. TRUE: User inputs do not have any effect in runtime more. The element is shown as deactivated.</td>
</tr>
</tbody>
</table>
The “Invisible” property is supported by the “Client Animation” functionality.

See also

These properties are available only when you select the “Preview: Support client animations and overlay of native elements” option in the Visualization Manager.

You can animate a movement, a rotation, and the visibility of a visualization element.

<table>
<thead>
<tr>
<th>“Animation duration”</th>
<th>Time that the element executes an animation (in milliseconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example: 500</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>“Move to the foreground”</th>
<th>Property value (BOOL)</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRUE:</td>
<td>At runtime, the element is displayed in the foreground.</td>
</tr>
<tr>
<td>FALSE:</td>
<td>At runtime, the element is displayed in the layer where it was inserted in the visualization editor.</td>
</tr>
</tbody>
</table>

See also

- Chapter 1.3.5.21.5 “Using client animation” on page 1962

Element property ‘Access rights’

Requirement: User management is set up for the visualization.

<table>
<thead>
<tr>
<th>“Access rights”</th>
<th>Opens the “Access rights” dialog. There you can edit the access privileges for the element.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Status messages:</td>
<td>- “Not set. Full rights.”: Access rights for all user groups: “operable”</td>
</tr>
<tr>
<td></td>
<td>- “Rights are set: Limited rights” : Access is restricted for at least one group.</td>
</tr>
</tbody>
</table>

See also

- Chapter 1.3.5.19.3.1 “Dialog ‘Access Rights’” on page 1583

See also

Visualization element ‘Radio button’

Symbol:

Tag: “Common controls”

The element provides a field with any number of options.
## Element properties

<table>
<thead>
<tr>
<th>Element properties</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Element name&quot;</td>
<td>Optional: Hint: Assign individual names for elements so that they are found faster in the element list. Example: early_shift</td>
</tr>
<tr>
<td>&quot;Type of element&quot;</td>
<td>&quot;Radio button&quot;</td>
</tr>
</tbody>
</table>

### Element property 'Position'

The position defines the location and size of the element in the visualization window. These are based on the Cartesian coordinate system. The origin is located at the upper left corner of the window. The positive horizontal x-axis runs to the right. The positive vertical y-axis runs downwards.

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;X&quot;</td>
<td>X coordinate of the upper left corner of the element Specified in pixels. Example: 10.</td>
</tr>
<tr>
<td>&quot;Y&quot;</td>
<td>Y coordinate of the upper left corner of the element Specified in pixels. Example: 10.</td>
</tr>
<tr>
<td>&quot;Width&quot;</td>
<td>Specified in pixels. Example: 150</td>
</tr>
<tr>
<td>&quot;Height&quot;</td>
<td>Specified in pixels. Example: 30</td>
</tr>
</tbody>
</table>

You can also change the values by dragging the box symbols (_drag_ ) to other positions in the editor.

See also
- § Chapter 1.3.5.3.2 “Positioning elements, adapting size and layer” on page 1113

### Variables

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Variable&quot;</td>
<td>Variable (integer data type) that gives the index of the radio button that the visualization user has activated Example: PLC_PRG.iNrOfActivatedRadioButton</td>
</tr>
<tr>
<td>&quot;Number of columns&quot;</td>
<td>Definition of the number of list boxes displayed in a row Example: 2</td>
</tr>
<tr>
<td>&quot;Radio button order&quot;</td>
<td>&quot;Left to right&quot;: The radio buttons are aligned by rows until the number of columns is reached. &quot;Top to bottom&quot;: The radio buttons are aligned row by columns until the number of columns is reached.</td>
</tr>
<tr>
<td>&quot;Frame size&quot;</td>
<td>Defines the distance from the list boxes to the edge (in pixels).</td>
</tr>
<tr>
<td>&quot;Row height&quot;</td>
<td>Height of the row (in pixels). Modifying the height of the row also changes the size of the list box.</td>
</tr>
</tbody>
</table>
Element property 'Text properties'  
The properties contain fixed values for the text properties.

| “Usage of”                      | ● “Default style values”: The values of the visualization style are used.  
|                                | ● “Individual settings”: The “Individual text properties” property group is shown  
|                                | The values can be customized here.  

“Individual text properties”  
Requirement: The “Individual settings” text property is defined.

<table>
<thead>
<tr>
<th>“Horizontal alignment”</th>
<th>Horizontal alignment of the text within the element.</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Vertical alignment”</td>
<td>Vertical alignment of the text within the element.</td>
</tr>
</tbody>
</table>

| “Text format”          | Definition for displaying texts that are too long  
|                        | ● “Default”: The long text is truncated.  
|                        | ● “Line break”: The text is split into parts.  
|                        | ● “Ellipsis”: The visible text ends with “...” indicating that it is not complete.  

| “Font”                 | Example: “Default”  
|                        | : The “Font” dialog box opens.  
|                        | : Drop-down list with style fonts.  

| “Font color”           | Example: “Black”  
|                        | : The “Color” dialog box opens.  
|                        | : Drop-down list with style colors.  

| “Transparency”         | Whole number (value range from 0 to 255). This determines the transparency of the respective color.  
|                        | Example: 255: The color is opaque.  
|                        | 0: The color is completely transparent.  
|                        | Please note: If the color is a style color and already has a transparency value, then this property is write-protected.  

Element property 'Absolute movement'  
The properties contain IEC variables for controlling the position of the element dynamically. The reference point is the upper left corner of the element. In runtime mode, the entire element is moved.

| “Movement”             |  
|                       |  
| “X”                   | Variable (numeric data type). Defines the X position (in pixels).  
|                        | Example: PLC_PRG.iPos_X.  
|                        | Increasing this value in runtime mode moves the element to the right.  
| “Y”                   | Variable (numeric data type). Defines the Y position (in pixels).  
|                        | Example: PLC_PRG.iPos_Y.  
|                        | Increasing this value in runtime mode moves the element downwards.  

### "Rotation"

Variable (numeric data type). Defines the angle of rotation (in degrees).

**Example:** PLC_PRG.iAngle1.

The midpoint of the element rotates at the "Center" point. This rotation point is shown as the symbol.

In runtime mode, the alignment of the element remains the same with respect to the coordinate system of the visualization. Increasing the value rotates the element to the right.

### "Interior rotation"

Variable (numeric data type). Defines the angle of rotation (in degrees).

**Example:** PLC_PRG.iAngle2.

In runtime mode, the element rotates about the point of rotation specified in "Center" according to the value of the variable. In addition, the alignment of the element rotates according to the coordinate system of the visualization. Increasing the value in the code rotates clockwise.

The rotation point is shown as the symbol.

Note: If a static angle of rotation is specified in the "Position Angle" property, then the static angle of rotation is added to the variable angle of rotation (offset) when the visualization is executed.

---

You can link the variables to a unit conversion.

The "X", "Y", "Rotation", and "Interior rotation" properties are supported by the "Client Animation" functionality.

---

**Element property 'State variables'**

The variables control the element behavior dynamically.

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Invisible&quot;</td>
<td>Variable (BOOL). Toggles the visibility of the element. TRUE: The element is not visible at runtime.</td>
</tr>
<tr>
<td>&quot;Deactivate inputs&quot;</td>
<td>Variable (BOOL). Toggles the operability of the element. TRUE: User inputs do not have any effect in runtime more. The element is shown as deactivated.</td>
</tr>
</tbody>
</table>
The “Invisible” property is supported by the “Client Animation” functionality.

See also

- "Radio button"
  - “Areas”
    - “[<n>]”
  - “Create new”: Clicking this button creates a new selection button in the editor and lists an additional area in the properties editor. For each radio button, an area is visible that records the settings.
  - “[<n>]”
    - “[<n>]”: This number indicates the area. Clicking “Delete” will delete the associated radio button with its settings “Text”, “Tooltip”, and “Line spacing (in pixels)”.

Areas: [<n>]

<table>
<thead>
<tr>
<th>“Text”</th>
<th>The button name is specified here. Default value: “Radio_button”</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Tooltip”</td>
<td>Text is specified here that is displayed in a tooltip.</td>
</tr>
<tr>
<td>“Line spacing (in pixels)”</td>
<td>The distance (in pixels) to the upper button can be specified here.</td>
</tr>
</tbody>
</table>

These properties are available only when you select the “Preview: Support client animations and overlay of native elements” option in the Visualization Manager.

You can animate a movement, a rotation, and the visibility of a visualization element.

<table>
<thead>
<tr>
<th>“Animation duration”</th>
<th>Time that the element executes an animation (in milliseconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example: 500</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>“Move to the foreground”</th>
<th>Property value (BOOL)</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRUE: At runtime, the element is displayed in the foreground.</td>
<td></td>
</tr>
<tr>
<td>FALSE: At runtime, the element is displayed in the layer where it was inserted in the visualization editor.</td>
<td></td>
</tr>
</tbody>
</table>

See also

- \% Chapter 1.3.5.21.5 “Using client animation” on page 1962

Element property 'Access rights'

Requirement: User management is set up for the visualization.

<table>
<thead>
<tr>
<th>“Access rights”</th>
<th>Opens the “Access rights” dialog. There you can edit the access privileges for the element.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Status messages:</td>
<td>• “Not set. Full rights.”: Access rights for all user groups: “operable”</td>
</tr>
<tr>
<td></td>
<td>• “Rights are set: Limited rights”: Access is restricted for at least one group.</td>
</tr>
</tbody>
</table>

See also

- \% Chapter 1.3.5.19.3.1 “Dialog ‘Access Rights’” on page 1583
Visualization element 'Alarm table'

Symbol:

Tag: “Alarm manager”

The element displays alarms in a list. In the element properties, you specify which information is shown. You define the appearance of the element and the variables that control the element behavior.

In online mode, you can sort an alarm table by a specific column – even in the classic view. Click into the column header. A small triangle indicates the current sort order (ascending, descending). Clicking the symbol reverses the order.

Sorting inside the column depends on the type of the contained information. The "Priority" column is sorted numerically, and the "Message" and "Class" columns alphabetically. The "Value" and "Latch" columns may contain different value types. In this case, sorting is first by type (blank, Boolean, numeric value, character string) and then either numerically or alphabetically depending on the type.

Element properties

<table>
<thead>
<tr>
<th>“Element name”</th>
<th>Example: GenElemInst_1</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Type of element”</td>
<td>“Alarm Table”</td>
</tr>
<tr>
<td>“Data source”</td>
<td>Selection of the device and the application where the data to be visualized and the alarms are generated</td>
</tr>
<tr>
<td></td>
<td>• Remote data source which accesses a remote device, accesses a remote application, and then transfers the data to the alarm configuration</td>
</tr>
<tr>
<td></td>
<td>Example: DataSource_A</td>
</tr>
<tr>
<td></td>
<td>Below the (now visible) &quot;Application&quot; property, the remote application is displayed as configured in the data source.</td>
</tr>
<tr>
<td></td>
<td>Example: App_A</td>
</tr>
<tr>
<td></td>
<td>Note: If the data source is accessed symbolically by means of a symbol file (CODESYS symbolic), then the required symbol file and the corresponding project have to be saved in the same folder.</td>
</tr>
<tr>
<td></td>
<td>• Local application below which the alarm configuration is located</td>
</tr>
<tr>
<td></td>
<td>Example: &quot;&lt;local application&gt;&quot;</td>
</tr>
</tbody>
</table>

See also

- **Object 'Data source'**
## “Alarm groups”
Opens the “Select Alarm Group” dialog where you define the alarm groups that you want to display.

## “Priority from”
Lowest priority for alarm display. (0 to 255).

## “Priority to”
Highest priority for alarm display. (0 to 255).

## “Alarm classes”
Opens the “Select Class Group” dialog where you define the alarm classes that you want to display.

## “Filter criterion”
For the alarm banner element only
- **“Most important”**: The alarm with the highest priority (lowest value) is displayed.
- **“Newest”**: The most recent alarm is displayed.

## “Filter by latch 1”
The generated alarms (previous and current) can be filtered by the contents of “Latch variable 1”, which is specified in the configuration of the alarm group. In “Filter type”, you define whether or not the filtering is performed by a string value or a numerical value.

- **“Filter variable”**: Indicates what the alarms are filtered by. Possible entries: Application variable of data type STRING or WSTRING, or a literal value directly. Examples: `PLC_PRG.strFilterVariable`, `'STRING'`.
- **“Filter type”**: Integer value that determines by which criteria the latch variable value is used for filtering. Possible entries: Numerical variable from the application (example: `PLC_PRG.diFilterType`), or a value directly (example: 2).

Possible values:
- 0: No filtering
- 1: Filter by alarms whose latch variable 1 contains the string specified in “Filter variable”. Example: The filter variable contains ‘Error 1’ which is the latch variable 1 of different alarms of type STRING and has the value 'Error 1' ->. Only these alarms are displayed.
- 2: Filter by alarms whose latch variable 1 contains the typed literal specified in “Filter variable” according to IEC 61131-3. Examples: `T#1h2s`, `DINT#15`, `REAL#1.5`, `FALSE`
- 3: Filter by alarms whose latch variable 1 contains the LINT literal value specified in “Filter variable”. Therefore, the value of the latch variables has to be in the range of 9,223,372,036,854,775,808 to 9,223,372,036,854,775,807.
- All other values: The behavior is not defined and can change in the future.

See also
- § Chapter 1.3.5.19.3.17 “Dialog ‘Selected Alarm Group’” on page 1605
- § Chapter 1.3.5.19.3.16 “Dialog ‘Selected Alarm Class’” on page 1604
- § Chapter 1.3.5.7 “Visualizing alarm management” on page 1146

### Element property ‘General table configuration’

- **“Show row header”**: Display of the row number at the beginning of the row.
- **“Show column header”**: Display of the column heading as defined in “Column heading”.
- **“Row height”**: Height of the table rows (in pixels).
- **“Row header width”**: Width of the line header (in pixels).
Scrollbar size

Width of the scrollbar when it runs vertically. Width of the scrollbar when it runs horizontally. Specified in pixels.

Automatic line break for alarm message

☑️ The message text is truncated at the end of the line.
☐☐ The message text is truncated at the end of the column, if the text is too long.

Element property 'Columns: Column [<>]'

By default, columns [0] and [1] are configured: “Time stamp” and “Message text”. You can create more columns by clicking the “Create new”, and remove columns by clicking “Delete”.

Animations (dynamic text, font variables), text, and tooltip are not supported.

Column header

The standard header is set and changed here by specifying a new text.

Use text alignment in title

☑️ The text in the column header is aligned according to the current definition in “Text alignment”.
☐☐ The text in the column header is centered.

Width

Width of the column (in pixels).

Data type

Notice about time stamps: For use in a TargetVisu or WebVisu, you can control the date and time format with the help of the global string variables from the library Alarmmanager.library:AlarmGlobals.g_sDateFormat (example: AlarmGlobals.g_sDateFormat := 'MM.yyyy') and AlarmGlobals.g_sTimeFormat (example: AlarmGlobals.g_sTimeFormat := 'HH:mm').

Define the information to be displayed in the column.

- “Symbol”
- “Time stamp”: Date and time of the last status change of the alarm.
- “Time stamp active”: Date and time of the last activation of the alarm.
- “Time stamp inactive”: Date and time of the last deactivation of the alarm.
- “Time stamp acknowledge”: Date and time of the last acknowledgment.
- “Value”: Current value of the printout
- “Message text”: Output of the message text
- “Priority”: Alarm priority
- “Class”: Alarm class
- “State”: Alarm state
- “Latch Variable <n>”: Value of the selected latch variables

Text alignment

Alignment of the text in this column

- “Left”
- “Centered”
- “Right”

Color settings

- “Activate color settings”: Boolean variable for activating and deactivating the color settings defined here. Example: PLC_PRG.bColorSettings
- “Cell fill color”:
  - “Color variable”: Variable for the cell fill color, example: dwCellColor (hexadecimal color definition: 16#TTTRGGGBB)
  - “Use color also for column header”: ☑️ The color defined via “Color variable” is used in the column header as well.
- “Text color”:
  - “Color variable”: Variable for the definition of the text color in the column, example: dwTextColor (hexadecimal color definition: 16#TTTRGGGBB)
  - “Use color also for column header”: ☐☐ The color defined via “Color variable” is used in the column header as well.

See also

- ☸ Chapter 1.3.5.8.3 “Animating a color display” on page 1152
The position defines the location and size of the element in the visualization window. These are based on the Cartesian coordinate system. The origin is located at the upper left corner of the window. The positive horizontal x-axis runs to the right. The positive vertical y-axis runs downwards.

### Element property 'Position'

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;X&quot;</td>
<td>X coordinate of the upper left corner of the element</td>
<td>10</td>
</tr>
<tr>
<td>&quot;Y&quot;</td>
<td>Y coordinate of the upper left corner of the element</td>
<td>10</td>
</tr>
<tr>
<td>&quot;Width&quot;</td>
<td>Specified in pixels.</td>
<td>150</td>
</tr>
<tr>
<td>&quot;Height&quot;</td>
<td>Specified in pixels.</td>
<td>30</td>
</tr>
</tbody>
</table>

You can also change the values by dragging the box symbols ( realloc ) to other positions in the editor.

See also
- " Chapter 1.3.5.2 “Positioning elements, adapting size and layer” on page 1113

### Element property 'Center'

The properties contain fixed values for the coordinates of the point of rotation. This point of rotation is shown as the symbol. The point is used as the center for rotating and scaling.

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;X&quot;</td>
<td>X-coordinate of the point of rotation</td>
</tr>
<tr>
<td>&quot;Y&quot;</td>
<td>Y-coordinate of the point of rotation</td>
</tr>
</tbody>
</table>

You can also change the values by dragging the symbols ( realloc ) to other positions in the editor.

### Element property 'Text properties'

The properties contain fixed values for the text properties.
### “Font”
Example: “Default”
- The “Font” dialog box opens.
- Drop-down list with style fonts.

### “Font color”
Example: “Black”
- The “Color” dialog box opens.
- Drop-down list with style colors.

### “Transparency”
Whole number (value range from 0 to 255). This determines the transparency of the respective color.
- Example: 255: The color is opaque.
- 0: The color is completely transparent.
Please note: If the color is a style color and already has a transparency value, then this property is write-protected.

### Element property ‘Selection’

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Background color on selection”</td>
<td>Fill color of the selected row.</td>
</tr>
<tr>
<td>“Selection font color”</td>
<td>Font color of the selected row.</td>
</tr>
<tr>
<td>“Frame around selected cells”</td>
<td>A frame is drawn around the selected cells at runtime.</td>
</tr>
<tr>
<td>“Variable for selected alarm group”</td>
<td>Name of the affected alarm group; type: STRING, WSTRING.</td>
</tr>
<tr>
<td>“Variable for selected alarm ID”</td>
<td>Alarm ID of the affected alarm group; type: STRING, WSTRING.</td>
</tr>
<tr>
<td>“Variable for selected line”</td>
<td>Index of the selected alarm line (0-based). The index can be read and written; integer data type</td>
</tr>
<tr>
<td>“Variable for valid selection”</td>
<td>TRUE: An alarm line is selected. FALSE: The selection is invalid. For example, for an empty alarm table or when an alarm is not selected yet.</td>
</tr>
</tbody>
</table>

### Element property ‘Control variables’

Boolean variables are defined here for executing specific actions in the table can be executed at runtime.

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Acknowledge selected”</td>
<td>Variable (BOOL)</td>
</tr>
<tr>
<td>Example: PLC_PRG.bAckSelectedAlarms</td>
<td>If the assigned variable is TRUE, then the selected alarm is acknowledged.</td>
</tr>
<tr>
<td>“Acknowledge all visible”</td>
<td>Variable (BOOL)</td>
</tr>
<tr>
<td>Example: PLC_PRG.bAckVisibleAlarms</td>
<td>If the assigned variable is TRUE, then all alarms are acknowledged that are visible in the alarm table.</td>
</tr>
<tr>
<td>“History”</td>
<td>Variable (BOOL)</td>
</tr>
<tr>
<td>Example: PLC_PRG.bShowHistory</td>
<td>If the assigned variable is TRUE, then the history alarms are displayed in addition to the active alarms. In the classic view, the same sort options apply as in normal mode. Note: Acknowledgment is not possible in this view.</td>
</tr>
</tbody>
</table>
### “Freeze scroll position”

Variable (BOOL)

**Example:** PLC_PRG.bFreezeScrollPosition

If the assigned variable is TRUE, then the scroll position set in the “History” view is retained, even if a new alarm is active. If not, then the scroll position jumps to the first table row (the newest alarm).

### “Count alarms”

Variable (integer data type)

**Example:** PLC_PRG.iNumberOfAlarms.

Number of alarms that are currently displayed in the alarm table. Defined by the alarm table.

### “Count visible rows”

Variable (integer data type)

**Example:** PLC_PRG.iNumberVisibleLines

Number of alarms that can be displayed on one page of the alarm table. Defined by the alarm table.

### “Current scroll index”

Variable (integer data type)

**Example:** PLC_PRG.iScrollIndex

The index of the first visible row if the alarm table (0-based). The variable can be read and written.

### “Current column sorting”

Variable (integer data type)

**Example:** PLC_PRG.iColSort

The variable contains a value of the enumeration "VisuElemsAlarm.VisuEnumAlarmDataType". This value determines the column that sorts the alarm table.

### “Variable for sorting direction”

Variable (BOOL)

**Example:** PLC_PRG.xSortAscending

The variable determines the sort order for the entries in the alarm table (TRUE: ascending; FALSE: descending).

---

You can also use the “Insert elements for acknowledging alarms” command to define buttons with predefined control variables.

See also

- ☞ Chapter 1.3.5.19.2.23 “Command ‘Add Elements for Alarm Acknowledgement’” on page 1582

---

### Element property 'Absolute movement'

The properties contain IEC variables for controlling the position of the element dynamically. The reference point is the upper left corner of the element. In runtime mode, the entire element is moved.

<table>
<thead>
<tr>
<th>“Movement”</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>“X”</strong></td>
</tr>
<tr>
<td>Variable (numeric data type). Defines the X position (in pixels).</td>
</tr>
<tr>
<td><strong>Example:</strong> PLC_PRG.iPos_X.</td>
</tr>
<tr>
<td>Increasing this value in runtime mode moves the element to the right.</td>
</tr>
<tr>
<td><strong>“Y”</strong></td>
</tr>
<tr>
<td>Variable (numeric data type). Defines the Y position (in pixels).</td>
</tr>
<tr>
<td><strong>Example:</strong> PLC_PRG.iPos_Y.</td>
</tr>
<tr>
<td>Increasing this value in runtime mode moves the element downwards.</td>
</tr>
</tbody>
</table>
### “Rotation”
Variable (numeric data type). Defines the angle of rotation (in degrees).

**Example:** PLC_PRG.iAngle1.

The midpoint of the element rotates at the “Center” point. This rotation point is shown as the symbol.

In runtime mode, the alignment of the element remains the same with respect to the coordinate system of the visualization. Increasing the value rotates the element to the right.

![Diagram of rotation](image)

### “Interior rotation”
Variable (numeric data type). Defines the angle of rotation (in degrees).

**Example:** PLC_PRG.iAngle2.

In runtime mode, the element rotates about the point of rotation specified in “Center” according to the value of the variable. In addition, the alignment of the element rotates according to the coordinate system of the visualization. Increasing the value in the code rotates clockwise.

The rotation point is shown as the symbol.

**Note:** If a static angle of rotation is specified in the “Position Angle” property, then the static angle of rotation is added to the variable angle of rotation (offset) when the visualization is executed.

![Diagram of interior rotation](image)

You can link the variables to a unit conversion.

The “X”, “Y”, “Rotation”, and “Interior rotation” properties are supported by the “Client Animation” functionality.

**See also**
- Unit conversion

### Element property ‘State variables’
The variables control the element behavior dynamically.

<table>
<thead>
<tr>
<th>“Invisible”</th>
<th>Variable (BOOL). Toggles the visibility of the element.</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRUE: The element is not visible at runtime.</td>
<td></td>
</tr>
</tbody>
</table>

The “Invisible” property is supported by the “Client Animation” functionality.
See also

These properties are available only when you select the “Preview: Support client animations and overlay of native elements” option in the Visualization Manager.

You can animate a movement, a rotation, and the visibility of a visualization element.

<table>
<thead>
<tr>
<th>“Animation duration”</th>
<th>Time that the element executes an animation (in milliseconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Example: 500</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>“Move to the foreground”</th>
<th>Property value (BOOL)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>TRUE: At runtime, the element is displayed in the foreground.</td>
</tr>
<tr>
<td></td>
<td>FALSE: At runtime, the element is displayed in the layer where it was inserted in the visualization editor.</td>
</tr>
</tbody>
</table>

See also

- Chapter 1.3.5.21.5 “Using client animation” on page 1962

Element property 'Access rights'

Requirement: User management is set up for the visualization.

<table>
<thead>
<tr>
<th>“Access rights”</th>
<th>Opens the “Access rights” dialog. There you can edit the access privileges for the element.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Status messages:</td>
</tr>
<tr>
<td></td>
<td>“Not set. Full rights.”: Access rights for all user groups: “operable”</td>
</tr>
<tr>
<td></td>
<td>“Rights are set: Limited rights”: Access is restricted for at least one group.</td>
</tr>
</tbody>
</table>

See also

- Chapter 1.3.5.19.3.1 “Dialog ‘Access Rights’” on page 1583

See also

- Chapter 1.3.5.7 “Visualizing alarm management” on page 1146
- Chapter 1.3.5.8.3 “Animating a color display” on page 1152

Visualization element ‘Alarm banner’

Symbol:

Tag: “Alarm manager”

The element is a simplified version of the alarm table. It visualizes a single alarm only. In the element properties, you specify which information is shown. You define the appearance of the element and the variables that control the element behavior.
The alarm banner displays active alarms only. If the alarm is acknowledged, then it disappears from the alarm banner.

### Element properties

<table>
<thead>
<tr>
<th><strong>“Element name”</strong></th>
<th><strong>Example:</strong> GenElemInst_1</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>“Type of element”</strong></td>
<td><strong>“Alarm banner”</strong></td>
</tr>
<tr>
<td><strong>“Data source”</strong></td>
<td>If you intend to use a remote alarm configuration, then you have to specify the name of the remote application here. If you do not specify anything, the alarm configuration will be located locally.</td>
</tr>
</tbody>
</table>

### Element property ‘Alarm configuration’

<table>
<thead>
<tr>
<th><strong>“Alarm groups”</strong></th>
<th>Opens the “Select Alarm Group” dialog where you define the alarm groups that you want to display.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>“Priority from”</strong></td>
<td>Lowest priority for alarm display. (0 to 255).</td>
</tr>
<tr>
<td><strong>“Priority to”</strong></td>
<td>Highest priority for alarm display. (0 to 255).</td>
</tr>
<tr>
<td><strong>“Alarm classes”</strong></td>
<td>Opens the “Select Class Group” dialog where you define the alarm classes that you want to display.</td>
</tr>
</tbody>
</table>
### “Filter criterion”

For the alarm banner element only

- **“Most important”**: The alarm with the highest priority (lowest value) is displayed.
- **“Newest”**: The most recent alarm is displayed.

### “Filter by latch 1”

The generated alarms (previous and current) can be filtered by the contents of “Latch variable 1”, which is specified in the configuration of the alarm group. In “Filter type”, you define whether or not the filtering is performed by a string value or a numerical value.

- **“Filter variable”**: Indicates what the alarms are filtered by. Possible entries: Application variable of data type STRING or WSTRING, or a literal value directly. Examples: PLC_PRG.strFilterVariable, 'STRING'.
- **“Filter type”**: Integer value that determines by which criteria the latch variable value is used for filtering. Possible entries: Numerical variable from the application (example: PLC_PRG.diFilterType, or a value directly (example: 2).

Possible values:
- 0: No filtering
- 1: Filter by alarms whose latch variable 1 contains the string specified in “Filter variable”. Example: The filter variable contains ‘Error 1’ which is the latch variable 1 of different alarms of type STRING and has the value ‘Error 1’ ->. Only these alarms are displayed.
- 2: Filter by alarms whose latch variable 1 contains the typed literal specified in “Filter variable” according to IEC 61131-3. Examples: T#1h2s, DINT#15, REAL#1.5, FALSE
- 3: Filter by alarms whose latch variable 1 contains the LINT literal value specified in “Filter variable”. Therefore, the value of the latch variables has to be in the range of 9,223,372,036,854,775,808 to 9,223,372,036,854,775,807.
- All other values: The behavior is not defined and can change in the future.

See also

- ¶ Chapter 1.3.5.19.3.17 “Dialog ‘Selected Alarm Group’” on page 1605
- ¶ Chapter 1.3.5.19.3.16 “Dialog ‘Selected Alarm Class’” on page 1604
- ¶ Chapter 1.3.5.7 “Visualizing alarm management” on page 1146

**Element property 'Columns: Column [<n>]':**

By default, columns [0] and [1] are preconfigured: “Time stamp” and “Message text”. You create more columns by clicking “Create new”. You remove columns by clicking “Delete”.

Animations (dynamic text, font variables), texts, and tooltips are not supported.
<table>
<thead>
<tr>
<th>“Width”</th>
<th>Width of the column (in pixels)</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Type of data”</td>
<td>About time stamps: When used in a TargetVisu or WebVisu, you can control the</td>
</tr>
<tr>
<td></td>
<td>date and time format by means of the global string variables from the library</td>
</tr>
<tr>
<td></td>
<td>Alarmmanager.library:AlarmGlobals.g_sDateFormat (example: AlarmGlobals.g_sDateFormat := 'MM.yyyy') and AlarmGlobals.g_sTimeFormat (example: AlarmGlobals.g_sTimeFormat := 'HH:mm').</td>
</tr>
<tr>
<td></td>
<td>Here you define the information to be displayed in the column.</td>
</tr>
<tr>
<td></td>
<td>● “Bitmap”</td>
</tr>
<tr>
<td></td>
<td>● “Time stamp”: Date and time of the last status change of the alarm</td>
</tr>
<tr>
<td></td>
<td>● “Time stamp active”: Date and time of the last activation of the alarm</td>
</tr>
<tr>
<td></td>
<td>● “Time stamp inactive”: Date and time of the last deactivation of the alarm</td>
</tr>
<tr>
<td></td>
<td>● “Time stamp acknowledge”: Date and time of the last acknowledgement</td>
</tr>
<tr>
<td></td>
<td>● “Value”: Actual value of the expression</td>
</tr>
<tr>
<td></td>
<td>● “Message”: Output of the message text</td>
</tr>
<tr>
<td></td>
<td>● “Priority”: Alarm priority</td>
</tr>
<tr>
<td></td>
<td>● “Class”: Alarm class</td>
</tr>
<tr>
<td></td>
<td>● “State”: Alarm state</td>
</tr>
<tr>
<td></td>
<td>● “Latch Variable &lt;n&gt;”: Value of the selected latch variables</td>
</tr>
<tr>
<td>“Text alignment”</td>
<td>Alignment of the contents in the column</td>
</tr>
<tr>
<td></td>
<td>● “Left”</td>
</tr>
<tr>
<td></td>
<td>● “Centered”</td>
</tr>
<tr>
<td></td>
<td>● “Right”</td>
</tr>
</tbody>
</table>

**Element property 'Position'**

The position defines the location and size of the element in the visualization window. These are based on the Cartesian coordinate system. The origin is located at the upper left corner of the window. The positive horizontal x-axis runs to the right. The positive vertical y-axis runs downwards.

<table>
<thead>
<tr>
<th>“X”</th>
<th>X coordinate of the upper left corner of the element</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Specified in pixels.</td>
</tr>
<tr>
<td></td>
<td>Example: 10.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>“Y”</th>
<th>Y coordinate of the upper left corner of the element</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Specified in pixels.</td>
</tr>
<tr>
<td></td>
<td>Example: 10.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>“Width”</th>
<th>Specified in pixels.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Example: 150</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>“Height”</th>
<th>Specified in pixels.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Example: 30</td>
</tr>
</tbody>
</table>

*You can also change the values by dragging the box symbols (_drag_2915) to other positions in the editor.*

See also

- § Chapter 1.3.5.3.2 “Positioning elements, adapting size and layer” on page 1113
The properties contain fixed values for the coordinates of the point of rotation. This point of rotation is shown as the symbol. The point is used as the center for rotating and scaling.

<table>
<thead>
<tr>
<th>&quot;X&quot;</th>
<th>X-coordinate of the point of rotation</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Y&quot;</td>
<td>Y-coordinate of the point of rotation</td>
</tr>
</tbody>
</table>

You can also change the values by dragging the symbols ( symbol) to other positions in the editor.

The properties contain fixed values for the text properties.

<table>
<thead>
<tr>
<th>&quot;Font&quot;</th>
<th>Example: &quot;Default&quot;</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>☐: The &quot;Font&quot; dialog box opens.</td>
</tr>
<tr>
<td></td>
<td>☐: Drop-down list with style fonts.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>&quot;Font color&quot;</th>
<th>Example: &quot;Black&quot;</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>☐: The &quot;Color&quot; dialog box opens.</td>
</tr>
<tr>
<td></td>
<td>☐: Drop-down list with style colors.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>&quot;Transparency&quot;</th>
<th>Whole number (value range from 0 to 255). This determines the transparency of the respective color.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Example: 255: The color is opaque.</td>
</tr>
<tr>
<td></td>
<td>0: The color is completely transparent.</td>
</tr>
<tr>
<td></td>
<td>Please note: If the color is a style color and already has a transparency value, then this property is write-protected.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>&quot;Acknowledge variable&quot;</th>
<th>A rising edge of this variable acknowledges the alarm.</th>
</tr>
</thead>
</table>

Handling of multiple active alarms

<table>
<thead>
<tr>
<th>&quot;Automatic switch&quot;</th>
<th>☑: The display in the alarm banner is switched automatically according to the time to the next alarm as configured in &quot;Every N second&quot;.</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Every N second&quot;</td>
<td>Time period until the next switching. Available only if &quot;Automatic switch&quot; is selected.</td>
</tr>
<tr>
<td>&quot;Next alarm&quot;</td>
<td>Variable for switching to the next alarm. Available only if &quot;Automatic switch&quot; is not selected.</td>
</tr>
<tr>
<td>&quot;Previous alarm&quot;</td>
<td>Variable for switching to the previous alarm. Available only if &quot;Automatic switch&quot; is not selected.</td>
</tr>
<tr>
<td>&quot;Multiple alarms active&quot;</td>
<td>Variable that has the value TRUE if multiple alarms are active.</td>
</tr>
</tbody>
</table>

The properties contain IEC variables for controlling the position of the element dynamically. The reference point is the upper left corner of the element. In runtime mode, the entire element is moved.

| Element property 'Absolute movement' | The properties contain IEC variables for controlling the position of the element dynamically. The reference point is the upper left corner of the element. In runtime mode, the entire element is moved. |
**“Movement”**

| **“X”** | Variable (numeric data type). Defines the X position (in pixels).  
Example: PLC_PRG.iPos_X.  
Increasing this value in runtime mode moves the element to the right. |
| **“Y”** | Variable (numeric data type). Defines the Y position (in pixels).  
Example: PLC_PRG.iPos_Y.  
Increasing this value in runtime mode moves the element downwards. |
| **“Rotation”** | Variable (numeric data type). Defines the angle of rotation (in degrees).  
Example: PLC_PRG.iAngle1.  
The midpoint of the element rotates at the “Center” point. This rotation point is shown as the symbol.  
In runtime mode, the alignment of the element remains the same with respect to the coordinate system of the visualization. Increasing the value rotates the element to the right. |
| **“Interior rotation”** | Variable (numeric data type). Defines the angle of rotation (in degrees).  
Example: PLC_PRG.iAngle2.  
In runtime mode, the element rotates about the point of rotation specified in “Center” according to the value of the variable. In addition, the alignment of the element rotates according to the coordinate system of the visualization. Increasing the value in the code rotates clockwise.  
The rotation point is shown as the symbol.  
Note: If a static angle of rotation is specified in the “Position ➔ Angle” property, then the static angle of rotation is added to the variable angle of rotation (offset) when the visualization is executed. |

---

You can link the variables to a unit conversion.

The “X”, “Y”, “Rotation”, and “Interior rotation” properties are supported by the “Client Animation” functionality.

See also
- [Unit conversion](#)
Element property 'State variables'

The variables control the element behavior dynamically.

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Invisible&quot;</td>
<td>Variable (BOOL). Toggles the visibility of the element. TRUE: The element is not visible at runtime.</td>
</tr>
</tbody>
</table>

The "Invisible" property is supported by the "Client Animation" functionality.

See also

These properties are available only when you select the “Preview: Support client animations and overlay of native elements” option in the Visualization Manager.

You can animate a movement, a rotation, and the visibility of a visualization element.

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Animation duration&quot;</td>
<td>Time that the element executes an animation (in milliseconds) Example: 500</td>
</tr>
<tr>
<td>&quot;Move to the foreground&quot;</td>
<td>Property value (BOOL) TRUE: At runtime, the element is displayed in the foreground. FALSE: At runtime, the element is displayed in the layer where it was inserted in the visualization editor.</td>
</tr>
</tbody>
</table>

See also

- Chapter 1.3.5.21.5 “Using client animation” on page 1962

Element property 'Access rights'

Requirement: User management is set up for the visualization.

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
</table>
| "Access rights" | Opens the “Access rights” dialog. There you can edit the access privileges for the element. Status messages:
  - “Not set. Full rights.”: Access rights for all user groups: “operable”
  - “Rights are set: Limited rights”: Access is restricted for at least one group. |

See also

- Chapter 1.3.5.19.3.1 “Dialog 'Access Rights'” on page 1583

See also

- Chapter 1.3.5.7 “Visualizing alarm management” on page 1146

Visualization element 'Bar display'

Symbol:
Tag: “Measurement controls”
The element displays the value of a variable.

See also

Element properties

<table>
<thead>
<tr>
<th>“Element name”</th>
<th>Example: GenElemInst_2</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Type of element”</td>
<td>“Bar display”</td>
</tr>
<tr>
<td>“Value”</td>
<td>Variable (numeric data type)</td>
</tr>
<tr>
<td></td>
<td>The value of the variable is displayed as a bar length.</td>
</tr>
</tbody>
</table>

Element property ‘Center’
The properties contain fixed values for the coordinates of the point of rotation. This point of rotation is shown as the symbol. The point is used as the center for rotating and scaling.

| “X” | X-coordinate of the point of rotation |
| “Y” | Y-coordinate of the point of rotation |

You can also change the values by dragging the symbols to other positions in the editor.

These properties are available only when you select the “Preview: Support client animations and overlay of native elements” option in the Visualization Manager.

You can animate a movement, a rotation, and the visibility of a visualization element.

<table>
<thead>
<tr>
<th>“Animation duration”</th>
<th>Time that the element executes an animation (in milliseconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example: 500</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>“Move to the foreground”</th>
<th>Property value (BOOL)</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRUE: At runtime, the element is displayed in the foreground.</td>
<td></td>
</tr>
<tr>
<td>FALSE: At runtime, the element is displayed in the layer where it was inserted in the visualization editor.</td>
<td></td>
</tr>
</tbody>
</table>

See also

- Chapter 1.3.5.21.5 “Using client animation” on page 1962

Element property ‘Position’
The position defines the location and size of the element in the visualization window. These are based on the Cartesian coordinate system. The origin is located at the upper left corner of the window. The positive horizontal x-axis runs to the right. The positive vertical y-axis runs downwards.
### Element property 'Background'

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>“Background color”</strong></td>
<td>Drop-down list with background colors</td>
</tr>
<tr>
<td></td>
<td>Note: This property depends on the style. For example, there are no heterochromatic background images for “FlatStyle” and “Whitestyle”.</td>
</tr>
<tr>
<td><strong>“Own image”</strong></td>
<td>• “image”: Image ID of the background image. You select the background image from an image pool by clicking the symbol. Info: If you specify the “&lt;default&gt;” value or select the image from the “Default” category in the input assistant, then the original element background image is used.</td>
</tr>
<tr>
<td></td>
<td>• “Transparent color”: Color of pixels that are displayed as transparent. Selection from drop-down list or input assistant.</td>
</tr>
<tr>
<td><strong>“Optimized drawing”</strong></td>
<td>✔️: The background image is drawn one time. If there is a change in the foreground, then only the affected part of the image is redrawn.</td>
</tr>
<tr>
<td></td>
<td>☐: The background image is redrawn in cycles. Note: Deactivating this option is sensible only in certain exceptional cases.</td>
</tr>
</tbody>
</table>

### Element property 'Bar'

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>“Diagram type”</strong></td>
<td>Position of the scale</td>
</tr>
<tr>
<td></td>
<td>• “Scale besides bar”</td>
</tr>
<tr>
<td></td>
<td>• “Scale in bar”</td>
</tr>
<tr>
<td></td>
<td>• “Bar in scale”</td>
</tr>
<tr>
<td></td>
<td>• “No scale”</td>
</tr>
<tr>
<td><strong>“Orientation”</strong></td>
<td>Orientation depending on the ratio of width to height of the bar display:</td>
</tr>
<tr>
<td></td>
<td>• “Horizontal”</td>
</tr>
<tr>
<td></td>
<td>• “Vertical”</td>
</tr>
</tbody>
</table>
"Running direction" | Direction the values are increased.  
| Drop-down list for "Orientation Horizontal":  
| ● "Left to right"  
| ● "Right to left"  
| Drop-down list for "Orientation Vertical":  
| ● "Bottom to top"  
| ● "Top to bottom"

"Optimum size for bar" | ✓ The bar width requires the majority of the element surface.  
| Note: This property depends on the style. It is not provided for "FlatStyle" or "WhiteStyle".

### Element property 'Scale'

| "Scale start" | Least value of the scale and the lower limit of the value range for the element.  
| Example: 0  
| Note: The property “Variable” is shown below.

| "Variable" | Variable (integer data type). Contains the scale start.  
| Example: PLC_PRG.iScaleStart  
| Declaration:  
| PROGRAM PLC_PRG  
| VAR   
| iScaleStart : INT := 0;  
| END_VAR

| "Scale end" | Greatest value of the scale and the upper limit of the value range for the element.  
| Example: 100  
| Note: The property “Variable” is shown below.

| "Variable" | Variable (integer data type). Contains the scale end.  
| Example: PLC_PRG.iScaleEnd  
| Declaration:  
| PROGRAM PLC_PRG  
| VAR   
| iScaleEnd : INT := 120;  
| END_VAR

| "Main scale" | Distance between 2 values on the rough scale.  
| Example: 10  
| Note: The property “Variable” is shown below.

| "Variable" | Variable (integer data type). Contains the distance.  
| Example: PLC_PRG.iMainScale  
| Declaration:  
| PROGRAM PLC_PRG  
| VAR   
| iMainScale : INT := 20;  
| END_VAR
### "Subscale"

Distance between 2 values on the fine scale.
You can hide the fine scale by setting the value to 0.

**Example:** 2

*: The property "Variable" is shown below.

### "Variable"

Variable (integer data type). Contains the spacing.

**Example:** PLC_PRG.iSubScale

**Declaration:**

```plaintext
PROGRAM PLC_PRG
VAR
  iSubScale : INT := 5;
END_VAR
```

### "Scale line width"

Specified in pixels.

**Example:** 3

### "Scale color"

Color of scale lines

- ★★★: The "Color" dialog box opens.
- ▼▼▼: A drop-down list with color names opens.

### "Scale in 3D"

* ✓: Tick marks are displayed with slight 3D shadowing.
  Note: This property depends on the style. Not available for "FlatStyle".

### "Element frame"

* ✓: A frame is drawn around the element.

---

#### Element property 'Label'

### "Unit"

Text that is displayed in the element.

**Example:** Units displayed in m/s.

### "Font"

Font for labels (example: scale numbering).

Selection from the drop-down list or by clicking the "" button.

### "Scale format (C Syntax)"

Values scaled in "printf" syntax

**Examples:** %d, %5.2f

### "Max. text width of labels"

(optional) Value that redefines the maximum width of the scale label. The correct value is normally set automatically.

**Note:** Change this value only if the automatic adjustment does not yield the expected result.

### "Text height of labels"

(optional) Value that redefines the maximum height of the scale label. The correct value is normally set automatically.

**Note:** Change this value only if the automatic adjustment does not yield the expected result.

### "Font color"

Selection from the drop-down list or by clicking the "" button.
<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Horizontal offset</strong></td>
<td>Distance from the scale (bar) to the horizontal element frame</td>
</tr>
<tr>
<td></td>
<td>Specified in pixels.</td>
</tr>
<tr>
<td></td>
<td>Used for achieving the exact position relative to the background image.</td>
</tr>
<tr>
<td><strong>Vertical offset</strong></td>
<td>Distance from the scale (bar) to the vertical element frame</td>
</tr>
<tr>
<td></td>
<td>Specified in pixels.</td>
</tr>
<tr>
<td></td>
<td>Used for achieving the exact position relative to the background image.</td>
</tr>
<tr>
<td><strong>Horizontal scaling</strong></td>
<td>Horizontal division of the scale</td>
</tr>
<tr>
<td></td>
<td>Specified in pixels.</td>
</tr>
<tr>
<td></td>
<td>Used for achieving the exact positioning relative to the background image.</td>
</tr>
<tr>
<td><strong>Vertical scaling</strong></td>
<td>Vertical division of the scale</td>
</tr>
<tr>
<td></td>
<td>Specified in pixels.</td>
</tr>
<tr>
<td></td>
<td>Used for achieving the exact positioning relative to the background image.</td>
</tr>
</tbody>
</table>

### Element property 'Colors'###

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Graph color</strong></td>
<td>Color of the bar</td>
</tr>
<tr>
<td><strong>Bar background</strong></td>
<td>![Black] The background of the bar is black.</td>
</tr>
<tr>
<td></td>
<td>![White] The background of the bar is white.</td>
</tr>
<tr>
<td><strong>Frame color</strong></td>
<td>Color that the frames are drawn.</td>
</tr>
<tr>
<td><strong>Switch whole color</strong></td>
<td>![Switch] The total color of the bar is switched to the color of the color area of the current value.</td>
</tr>
<tr>
<td><strong>Use gradient color for bar</strong></td>
<td>![Gradient] Bar is displayed with a gradient.</td>
</tr>
<tr>
<td><strong>Color range markers</strong></td>
<td>The color areas can be separated from each other inside the bar with a vertical mark.</td>
</tr>
<tr>
<td></td>
<td>- <em>No markers</em>: No display.</td>
</tr>
<tr>
<td></td>
<td>- <em>Marker forwards</em>: The color of the vertical mark corresponds to the color of the previous color area.</td>
</tr>
<tr>
<td></td>
<td>- <em>Marker backwards</em>: The color of the vertical mark corresponds to the color of the next color area.</td>
</tr>
<tr>
<td><strong>Color areas</strong></td>
<td>![Create new] A new color area is added.</td>
</tr>
<tr>
<td></td>
<td>![Delete] The color area is removed from the list.</td>
</tr>
<tr>
<td><strong>Begin of area</strong></td>
<td>Start value of the color area</td>
</tr>
<tr>
<td><strong>End of area</strong></td>
<td>End value of the color area</td>
</tr>
<tr>
<td><strong>Color</strong></td>
<td>Color that is used for displaying the area.</td>
</tr>
</tbody>
</table>

### Element property 'Absolute movement'###

The properties contain IEC variables for controlling the position of the element dynamically. The reference point is the upper left corner of the element. In runtime mode, the entire element is moved.
### Movement

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
<th>Example</th>
<th>Behavior</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>X</strong></td>
<td>Variable (numeric data type). Defines the X position (in pixels).</td>
<td>PLC_PRG.iPos_X</td>
<td>Increasing this value in runtime mode moves the element to the right.</td>
</tr>
<tr>
<td><strong>Y</strong></td>
<td>Variable (numeric data type). Defines the Y position (in pixels).</td>
<td>PLC_PRG.iPos_Y</td>
<td>Increasing this value in runtime mode moves the element downwards.</td>
</tr>
<tr>
<td><strong>Rotation</strong></td>
<td>Variable (numeric data type). Defines the angle of rotation (in degrees).</td>
<td>PLC_PRG.iAngle1</td>
<td>The midpoint of the element rotates at the “Center” point. This rotation point is shown as the symbol. In runtime mode, the alignment of the element remains the same with respect to the coordinate system of the visualization. Increasing the value rotates the element to the right.</td>
</tr>
<tr>
<td><strong>Interior rotation</strong></td>
<td>Variable (numeric data type). Defines the angle of rotation (in degrees).</td>
<td>PLC_PRG.iAngle2</td>
<td>In runtime mode, the element rotates about the point of rotation specified in “Center” according to the value of the variable. In addition, the alignment of the element rotates according to the coordinate system of the visualization. Increasing the value in the code rotates clockwise. The rotation point is shown as the symbol. Note: If a static angle of rotation is specified in the “Position ➔ Angle” property, then the static angle of rotation is added to the variable angle of rotation (offset) when the visualization is executed.</td>
</tr>
</tbody>
</table>

You can link the variables to a unit conversion.

The “X”, “Y”, “Rotation”, and “Interior rotation” properties are supported by the “Client Animation” functionality.

See also
- Unit conversion
The variables control the element behavior dynamically.

| "Deactivate inputs" | Variable (BOOL). Toggles the operability of the element. TRUE: User inputs do not have any effect in runtime more. The element is shown as deactivated. |

Element property 'Access rights'

Requirement: User management is set up for the visualization.

| "Access rights" | Opens the "Access rights" dialog. There you can edit the access privileges for the element. Status messages: ● "Not set. Full rights.": Access rights for all user groups: "operable" ● "Rights are set: Limited rights": Access is restricted for at least one group. |

See also

- % Chapter 1.3.5.19.3.1 “Dialog ‘Access Rights’” on page 1583

Visualization element 'Meter 90°'

Symbol:

Tag: “Measurement controls”

The element displays the value of a variable. The needle is positioned according to the value of the assigned variable. A meter is used to represent a tachometer, for example.

Element properties

| "Element name" | Example: GenElemInst_1 |
| "Type of element" | "Meter 90°" |
| "Value" | Variable (numeric data type) The variable value determines the pointer movement of the element. |

These properties are available only when you select the “Preview: Support client animations and overlay of native elements” option in the Visualization Manager.

You can animate a movement, a rotation, and the visibility of a visualization element.
<table>
<thead>
<tr>
<th><strong>“Animation duration”</strong></th>
<th>Time that the element executes an animation (in milliseconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Example: 500</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>“Move to the foreground”</strong></th>
<th>Property value (BOOL)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>TRUE: At runtime, the element is displayed in the foreground.</td>
</tr>
<tr>
<td></td>
<td>FALSE: At runtime, the element is displayed in the layer where it was inserted in the visualization editor.</td>
</tr>
</tbody>
</table>

See also
- Chapter 1.3.5.21.5 “Using client animation” on page 1962

**Element property ‘Position’**
The position defines the location and size of the element in the visualization window. These are based on the Cartesian coordinate system. The origin is located at the upper left corner of the window. The positive horizontal x-axis runs to the right. The positive vertical y-axis runs downwards.

<table>
<thead>
<tr>
<th><strong>“X”</strong></th>
<th>X coordinate of the upper left corner of the element</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Specified in pixels.</td>
</tr>
<tr>
<td></td>
<td>Example: 10.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>“Y”</strong></th>
<th>Y coordinate of the upper left corner of the element</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Specified in pixels.</td>
</tr>
<tr>
<td></td>
<td>Example: 10.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>“Width”</strong></th>
<th>Specified in pixels.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Example: 150</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>“Height”</strong></th>
<th>Specified in pixels.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Example: 30</td>
</tr>
</tbody>
</table>

You can also change the values by dragging the box symbols (_drag_ ) to other positions in the editor.

See also
- Chapter 1.3.5.3.2 “Positioning elements, adapting size and layer” on page 1113

**Element property ‘Background’**

<table>
<thead>
<tr>
<th><strong>“Image color”</strong></th>
<th>List box containing background colors</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>“Own image”</strong></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>“Image”: ID of the background image.</td>
</tr>
<tr>
<td></td>
<td>You select the background image from an image pool by clicking [<em>drag</em>].</td>
</tr>
<tr>
<td></td>
<td>Info: If you specify the value “&lt;default&gt;” or select the image from the “Default” category in the Input Assistant, then the original element background image is used.</td>
</tr>
<tr>
<td></td>
<td>“Transparency color”: Selection from list box or Input Assistant.</td>
</tr>
</tbody>
</table>


## Element property 'Arrow'

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Hand style”</td>
<td>Drop-down list with different arrow types</td>
</tr>
<tr>
<td>“Color”</td>
<td>- [ ]: The “Color” dialog box opens.</td>
</tr>
<tr>
<td></td>
<td>- ▼: Drop-down list with color names</td>
</tr>
<tr>
<td>“Angle range”</td>
<td>Drop-down list for the alignment of the element</td>
</tr>
<tr>
<td>“Additional arrow”</td>
<td>[ ]: An additional arrow is shown inside the scale.</td>
</tr>
</tbody>
</table>

## Element property 'Scale'

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Sub scale position”</td>
<td>- “Outside”: The subscale is displayed on the outer scale ring. (“Frame outside”)</td>
</tr>
<tr>
<td></td>
<td>- “Inside”: The subscale is displayed on the inner scale ring. (“Frame inside”)</td>
</tr>
<tr>
<td>“Scale type”</td>
<td>Type of scale</td>
</tr>
<tr>
<td></td>
<td>- “Lines”</td>
</tr>
<tr>
<td></td>
<td>- “Dots”</td>
</tr>
<tr>
<td></td>
<td>- “Squares”</td>
</tr>
<tr>
<td>“Scale start”</td>
<td>Least value of the scale and the lower limit of the value range for the element</td>
</tr>
<tr>
<td></td>
<td>Example: 0</td>
</tr>
<tr>
<td></td>
<td>[ ]: The “Variable” property is displayed in the line below this.</td>
</tr>
<tr>
<td>“Variable”</td>
<td>Variable (integer data type). Contains the scale start</td>
</tr>
<tr>
<td></td>
<td>Example: PLC_PRG.iScaleStart</td>
</tr>
<tr>
<td></td>
<td>Declaration:</td>
</tr>
<tr>
<td></td>
<td>PROGRAM PLC_PRG</td>
</tr>
<tr>
<td></td>
<td>VAR iScaleStart : INT := 0;</td>
</tr>
<tr>
<td></td>
<td>END_VAR</td>
</tr>
<tr>
<td>“Scale end”</td>
<td>Greatest value of the scale and the upper limit of the value range for the element</td>
</tr>
<tr>
<td></td>
<td>Example: 100</td>
</tr>
<tr>
<td></td>
<td>[ ]: The “Variable” property is shown below this.</td>
</tr>
<tr>
<td>“Variable”</td>
<td>Variable (integer data type). Contains the scale end</td>
</tr>
<tr>
<td></td>
<td>Example: PLC_PRG.iScaleEnd</td>
</tr>
<tr>
<td></td>
<td>Declaration:</td>
</tr>
<tr>
<td></td>
<td>PROGRAM PLC_PRG</td>
</tr>
<tr>
<td></td>
<td>VAR iScaleEnd : INT := 120;</td>
</tr>
<tr>
<td></td>
<td>END_VAR</td>
</tr>
<tr>
<td>“Main scale”</td>
<td>Distance between two values on the main scale</td>
</tr>
<tr>
<td></td>
<td>Example: 10</td>
</tr>
<tr>
<td></td>
<td>[ ]: The “Variable” property is shown below.</td>
</tr>
<tr>
<td>Property</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>Variable</strong></td>
<td>Variable (integer data type) Contains the distance between two values on the main scale</td>
</tr>
<tr>
<td>Example:</td>
<td>PLC_PRG.iMainScale</td>
</tr>
<tr>
<td>Declaration:</td>
<td>PROGRAM PLC_PRG</td>
</tr>
<tr>
<td></td>
<td>VAR</td>
</tr>
<tr>
<td></td>
<td>iMainScale : INT := 20;</td>
</tr>
<tr>
<td></td>
<td>END_VAR</td>
</tr>
<tr>
<td><strong>Sub scale</strong></td>
<td>Distance between two values on the fine scale</td>
</tr>
<tr>
<td>Example:</td>
<td>2</td>
</tr>
<tr>
<td>Note:</td>
<td>The “Variable” property is shown below this.</td>
</tr>
<tr>
<td><strong>Variable</strong></td>
<td>Variable (integer data type) Contains the distance between two values on the fine scale</td>
</tr>
<tr>
<td>Example:</td>
<td>PLC_PRG.iSubScale</td>
</tr>
<tr>
<td>Declaration:</td>
<td>PROGRAM PLC_PRG</td>
</tr>
<tr>
<td></td>
<td>VAR</td>
</tr>
<tr>
<td></td>
<td>iSubScale : INT := 5;</td>
</tr>
<tr>
<td></td>
<td>END_VAR</td>
</tr>
<tr>
<td><strong>Scale line width</strong></td>
<td>Specified in pixels</td>
</tr>
<tr>
<td>Example:</td>
<td>3</td>
</tr>
<tr>
<td><strong>Scale color</strong></td>
<td>Color of scale lines</td>
</tr>
<tr>
<td></td>
<td>● [ ]: The “Color” dialog opens.</td>
</tr>
<tr>
<td></td>
<td>● ▼: A list box with style colors opens.</td>
</tr>
<tr>
<td><strong>Scale in 3D</strong></td>
<td>[ ]: Scale lines are displayed with soft 3D shadowing.</td>
</tr>
<tr>
<td>Note:</td>
<td>This property is not displayed in “FlatStyle”.</td>
</tr>
<tr>
<td><strong>Show scale</strong></td>
<td>[ ]: The scale is displayed.</td>
</tr>
<tr>
<td><strong>Frame inside</strong></td>
<td>[ ]: A frame is drawn at the inner end of the scale.</td>
</tr>
<tr>
<td><strong>Frame outside</strong></td>
<td>[ ]: A frame is drawn at the outer end of the scale.</td>
</tr>
</tbody>
</table>

**Element property 'Label'**

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Label</strong></td>
<td>Selection list</td>
</tr>
<tr>
<td></td>
<td>● “Outside”: Scale values are placed outside of the scale.</td>
</tr>
<tr>
<td></td>
<td>● “Inside”: Scale values are placed inside of the scale.</td>
</tr>
<tr>
<td><strong>Unit</strong></td>
<td>Text that is displayed in the element.</td>
</tr>
<tr>
<td>Example:</td>
<td>Units displayed in m/s.</td>
</tr>
<tr>
<td><strong>Font</strong></td>
<td>Font for labels (example: scale numbering).</td>
</tr>
<tr>
<td>Selection:</td>
<td>from the drop-down list or by clicking the “” button.</td>
</tr>
<tr>
<td><strong>Scale format (C Syntax)</strong></td>
<td>Values scaled in &quot;printf&quot; syntax</td>
</tr>
<tr>
<td>Example:</td>
<td>%d, %5.2f</td>
</tr>
</tbody>
</table>
**Max. text width of labels**  
(optional) Value that redefines the maximum width of the scale label. The correct value is normally set automatically.  
Note: Change this value only if the automatic adjustment does not yield the expected result.

**Text height of labels**  
(optional) Value that redefines the maximum height of the scale label. The correct value is normally set automatically.  
Note: Change this value only if the automatic adjustment does not yield the expected result.

**Font color**  
Selection from the drop-down list or by clicking the button.

---

### Element property ‘Positioning’

| Usage of          | "Preset style values": Values from the current style  
| "User-defined settings": The subnode “Positioning” appears. |

**Positioning**  
Requirement: “User-defined settings” is selected as “Usage of”.  
The displayed positioning settings depend on the type of needle instrument and potentiometer, and partially on whether a custom background image is selected. The following settings are used for achieving the exact position relative to the background image.

| Needle movement | Length of the needle (in pixels) |
| Scale movement  | Distance from the tick marks to the center (in pixels)  
| Requirement: A customer image is selected as “Background”. |
| Scale length    | Length of the tick marks (in pixels)  
| Requirement: A customer image is selected as “Background”. |
| Label offset    | Distance from the labels to the tick marks (in pixels) |
| Unit offset     | Distance of the unit text “Label ➔ Unit” from the upper scale edge (in pixels) |
| Origin offset   | Offset of the element (in pixels)  
| Requirement: For the elements “Meter 180°” and “Meter 90°”, this property is displayed only if a custom image is selected as “Background”. |

---

### Element property ‘Colors’

| Durable color areas | □: All color areas are visible, regardless of the current value.  
| ▮: Only the color area is visible that includes the current value. |

| Use colors for scale | ▮: Colors in the color area are used only for the scale and frame. |

| Create new | A new color area is added to the “Elements” view. |

| Delete | The color area is removed from the list and the list is refreshed. |

| Begin of area | Start value of the color area  
<p>| Example: 20 | ▮: The property “Variable” is shown below. |</p>
<table>
<thead>
<tr>
<th><strong>“Variable”</strong></th>
<th>Variable (integer data type). Contains the start value. Example: PLC_PRG.iColorAreaStart0</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Declaration</strong></td>
<td>PROGRAM PLC_PRG&lt;br&gt;VAR&lt;br&gt;   icolorAreaStart0 : INT := 80; END_VAR</td>
</tr>
<tr>
<td><strong>“End of area”</strong></td>
<td>End value of the color area Example: 120</td>
</tr>
<tr>
<td><strong>Variable</strong></td>
<td>Variable (integer data type). Contains the end value. Example: iColorAreaEnd0</td>
</tr>
<tr>
<td><strong>Declaration</strong></td>
<td>PROGRAM PLC_PRG&lt;br&gt;VAR&lt;br&gt;   icolorAreaEnd0 : INT := 100; END_VAR</td>
</tr>
<tr>
<td><strong>“Color”</strong></td>
<td>Color that is used for displaying the area.</td>
</tr>
</tbody>
</table>

**Element property 'Absolute movement'**

The properties contain IEC variables for controlling the position of the element dynamically. The reference point is the upper left corner of the element. In runtime mode, the entire element is moved.

<table>
<thead>
<tr>
<th><strong>“Movement”</strong></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>“X”</strong></td>
<td>Variable (numeric data type). Defines the X position (in pixels). Example: PLC_PRG.iPos_X. Increasing this value in runtime mode moves the element to the right.</td>
</tr>
<tr>
<td><strong>“Y”</strong></td>
<td>Variable (numeric data type). Defines the Y position (in pixels). Example: PLC_PRG.iPos_Y. Increasing this value in runtime mode moves the element downwards.</td>
</tr>
<tr>
<td><strong>“Rotation”</strong></td>
<td>Variable (numeric data type). Defines the angle of rotation (in degrees). Example: PLC_PRG.iAngle1. The midpoint of the element rotates at the “Center” point. This rotation point is shown as the symbol. In runtime mode, the alignment of the element remains the same with respect to the coordinate system of the visualization. Increasing the value rotates the element to the right.</td>
</tr>
</tbody>
</table>
“Interior rotation” Variable (numeric data type). Defines the angle of rotation (in degrees).

**Example:** PLC_PRG.iAngle2.

In runtime mode, the element rotates about the point of rotation specified in “Center” according to the value of the variable. In addition, the alignment of the element rotates according to the coordinate system of the visualization. Increasing the value in the code rotates clockwise.

The rotation point is shown as the symbol.

Note: If a static angle of rotation is specified in the “Position ➔ Angle” property, then the static angle of rotation is added to the variable angle of rotation (offset) when the visualization is executed.

---

You can link the variables to a unit conversion.

---

The “X”, “Y”, “Rotation”, and “Interior rotation” properties are supported by the "Client Animation" functionality.

See also
- Unit conversion

---

**Element property 'Access rights'**

Requirement: User management is set up for the visualization.

“Access rights” Opens the “Access rights” dialog. There you can edit the access privileges for the element.

Status messages:
- “Not set. Full rights.”: Access rights for all user groups: “operable”
- “Rights are set: Limited rights”: Access is restricted for at least one group.

See also
- § Chapter 1.3.5.19.3.1 “Dialog 'Access Rights'” on page 1583

---

**Visualization element 'Meter 180°’**

Symbol:

Tag: “Measurement controls”
The element displays the value of a variable. The needle is positioned according to the value of the assigned variable on a scale. A meter is used to represent a tachometer, for example.

### Element properties

<table>
<thead>
<tr>
<th>“Element name”</th>
<th>Example: GenElemInst_1</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Type of element”</td>
<td>“Meter 180°”</td>
</tr>
</tbody>
</table>
| “Value” | Variable (numeric data type)  
The variable value determines the pointer movement of the element. |

### Element property 'Center'

The properties contain fixed values for the coordinates of the point of rotation. This point of rotation is shown as the symbol. The point is used as the center for rotating and scaling.

| “X” | X-coordinate of the point of rotation |
| “Y” | Y-coordinate of the point of rotation |

You can also change the values by dragging the symbols to other positions in the editor.

These properties are available only when you select the “Preview: Support client animations and overlay of native elements” option in the Visualization Manager.

You can animate a movement, a rotation, and the visibility of a visualization element.

| “Animation duration” | Time that the element executes an animation (in milliseconds)  
Example: 500 |
| “Move to the foreground” | Property value (BOOL)  
TRUE: At runtime, the element is displayed in the foreground.  
FALSE: At runtime, the element is displayed in the layer where it was inserted in the visualization editor. |

See also

- ☞ Chapter 1.3.5.21.5 “Using client animation” on page 1962

### Element property 'Position'

The position defines the location and size of the element in the visualization window. These are based on the Cartesian coordinate system. The origin is located at the upper left corner of the window. The positive horizontal x-axis runs to the right. The positive vertical y-axis runs downwards.
“X”   
X coordinate of the upper left corner of the element
Specified in pixels.
Example: 10.

“Y”   
Y coordinate of the upper left corner of the element
Specified in pixels.
Example: 10.

“Width”   
Specified in pixels.
Example: 150

“Height”   
Specified in pixels.
Example: 30

You can also change the values by dragging the box symbols ( Freel) to other positions in the editor.

See also
●  § Chapter 1.3.5.3.2 “Positioning elements, adapting size and layer” on page 1113

Element property ‘Background’

“Image color”   
List box containing background colors

“Own image”   
●  “Image”: ID of the background image.
    You select the background image from an image pool by clicking .
    Info: If you specify the value “<default>” or select the image from the “Default” category in the Input Assistant, then the original element background image is used.

●  “Transparency color”: Selection from list box or Input Assistant.

Element property ‘Arrow’

“Hand style”   
Drop-down list with different arrow types

“Color”   
●  ▼: The “Color” dialog box opens.
●  ▼: Drop-down list with color names

“Angle range”   
Drop-down list for the alignment of the element

“Additional arrow”   
☑: An additional arrow is shown inside the scale.

Element property ‘Scale’
| **Sub scale position** | ● **Outside**: The subscale is displayed on the outer scale ring. ("Frame outside")
  ● **Inside**: The subscale is displayed on the inner scale ring. ("Frame inside") |
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Scale type</strong></td>
<td>Type of scale</td>
</tr>
</tbody>
</table>
|                        | ● **Lines**
  ● **Dots**
  ● **Squares** |
| **Scale start**        | Least value of the scale and the lower limit of the value range for the element                                                  |
|                        | Example: 0.  
  🌟: The "Variable" property is displayed in the line below this.                                                               |
| **Variable**           | Variable (integer data type). Contains the scale start                                                                           |
|                        | Example: PLC_PRG.iScaleStart                                                                                                    |
|                        | Declaration:                                                                                                                     |
|                        | ```                                                                                                                             |
|                        | PROGRAM PLC_PRG
|                        | VAR
|                        | iScaleStart : INT := 0;                                                                                                         |
|                        | ```                                                                                                                             |
| **Scale end**          | Greatest value of the scale and the upper limit of the value range for the element                                               |
|                        | Example: 100.  
  🌟: The "Variable" property is shown below this.                                                                              |
| **Variable**           | Variable (integer data type). Contains the scale end                                                                           |
|                        | Example: PLC_PRG.iScaleEnd                                                                                                       |
|                        | Declaration:                                                                                                                     |
|                        | ```                                                                                                                             |
|                        | PROGRAM PLC_PRG
|                        | VAR
|                        | iScaleEnd : INT := 120;                                                                                                          |
|                        | ```                                                                                                                             |
| **Main scale**         | Distance between two values on the main scale                                                                                   |
|                        | Example: 10.  
  🌟: The "Variable" property is shown below.                                                                                     |
| **Variable**           | Variable (integer data type) Contains the distance between two values on the main scale                                         |
|                        | Example: PLC_PRG.iMainScale                                                                                                      |
|                        | Declaration:                                                                                                                     |
|                        | ```                                                                                                                             |
|                        | PROGRAM PLC_PRG
|                        | VAR
|                        | iMainScale : INT := 20;                                                                                                          |
|                        | ```                                                                                                                             |
| **Sub scale**          | Distance between two values on the fine scale                                                                                   |
|                        | You can hide the fine scale by setting the value to 0.                                                                             |
|                        | Example: 2.  
  🌟: The "Variable" property is shown below this.                                                                                     |
| **“Variable”** | Variable (integer data type) Contains the distance between two values on the fine scale  
Example: PLC_PRG.iSubScale  
Declaration:  
PROGRAM PLC_PRG  
VAR  
iSubScale : INT := 5;  
END_VAR |
|---|---|
| **“Scale line width”** | Specified in pixels  
Example: 3 |
| **“Scale color”** | Color of scale lines  
- [ ]: The “Color” dialog opens.  
- ▼: A list box with style colors opens. |
| **“Scale in 3D”** | [ ]: Scale lines are displayed with soft 3D shadowing.  
Note: This property is not displayed in “FlatStyle”. |
| **“Show scale”** | [ ]: The scale is displayed. |
| **“Frame inside”** | [ ]: A frame is drawn at the inner end of the scale. |
| **“Frame outside”** | [ ]: A frame is drawn at the outer end of the scale. |

**Element property 'Label'**

| **“Label”** | Selection list  
- “Outside”: Scale values are placed outside of the scale.  
- “Inside”: Scale values are placed inside of the scale. |
| **“Unit”** | Text that is displayed in the element.  
Example: Units displayed in m/s. |
| **“Font”** | Font for labels (example: scale numbering).  
Selection from the drop-down list or by clicking the “” button. |
| **“Scale format (C Syntax)”** | Values scaled in “printf” syntax  
Examples: %d, %5.2f |
| **“Max. text width of labels”** | (optional) Value that redefines the maximum width of the scale label. The correct value is normally set automatically.  
Note: Change this value only if the automatic adjustment does not yield the expected result. |
| **“Text height of labels”** | (optional) Value that redefines the maximum height of the scale label. The correct value is normally set automatically.  
Note: Change this value only if the automatic adjustment does not yield the expected result. |
| **“Font color”** | Selection from the drop-down list or by clicking the “” button. |
“Usage of”
- “Preset style values”: Values from the current style
- “User-defined settings”: The subnode “Positioning” appears.

“Positioning”
Requirement: “User-defined settings” is selected as “Usage of”.

The displayed positioning settings depend on the type of needle instrument and potentiometer, and partially on whether a custom background image is selected. The following settings are used for achieving the exact position relative to the background image.

<table>
<thead>
<tr>
<th>Setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Needle movement”</td>
<td>Length of the needle (in pixels)</td>
</tr>
<tr>
<td>“Scale movement”</td>
<td>Distance from the tick marks to the center (in pixels)</td>
</tr>
<tr>
<td>Requirement: A customer image is selected as “Background”.</td>
<td></td>
</tr>
<tr>
<td>“Scale length”</td>
<td>Length of the tick marks (in pixels)</td>
</tr>
<tr>
<td>Requirement: A customer image is selected as “Background”.</td>
<td></td>
</tr>
<tr>
<td>“Label offset”</td>
<td>Distance from the labels to the tick marks (in pixels)</td>
</tr>
<tr>
<td>“Unit offset”</td>
<td>Distance of the unit text “Label ➔ Unit” from the upper scale edge (in pixels)</td>
</tr>
<tr>
<td>“Origin offset”</td>
<td>Offset of the element (in pixels)</td>
</tr>
<tr>
<td>Requirement: For the elements “Meter 180°” and “Meter 90°”, this property is displayed only if a custom image is selected as “Background”.</td>
<td></td>
</tr>
</tbody>
</table>

“Color areas”

<table>
<thead>
<tr>
<th>Setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Durable color areas”</td>
<td>☐: All color areas are visible, regardless of the current value.</td>
</tr>
<tr>
<td></td>
<td>☑: Only the color area is visible that includes the current value.</td>
</tr>
<tr>
<td>“Use colors for scale”</td>
<td>☑: Colors in the color area are used only for the scale and frame.</td>
</tr>
</tbody>
</table>

“Color areas”

<table>
<thead>
<tr>
<th>Setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Create new”</td>
<td>A new color area is added to the “Elements” view.</td>
</tr>
<tr>
<td>“Delete”</td>
<td>The color area is removed from the list and the list is refreshed.</td>
</tr>
</tbody>
</table>

“Begin of area”
Start value of the color area
Example: 20

嵎: The property “Variable” is shown below.

“Variable”
Variable (integer data type). Contains the start value.
Example: PLC_PRG.iColorAreaStart0

Declaration:

```plaintext
PROGRAM PLC_PRG
VAR
   iColorAreaStart0 : INT := 80;
END_VAR
```

“End of area”
End value of the color area
Example: 120

嵎: The property “Variable” is shown below.
### Variable

Variable (integer data type). Contains the end value.

**Example:** iColorAreaEnd0

**Declaration:**

```plaintext
PROGRAM PLC_PRG
VAR
  iColorAreaEnd0 : INT := 100;
END_VAR
```

### Color

Color that is used for displaying the area.

**Element property 'Absolute movement'**

The properties contain IEC variables for controlling the position of the element dynamically. The reference point is the upper left corner of the element. In runtime mode, the entire element is moved.

### Movement

#### X

Variable (numeric data type). Defines the X position (in pixels).

**Example:** PLC_PRG.iPos_X.

Increasing this value in runtime mode moves the element to the right.

#### Y

Variable (numeric data type). Defines the Y position (in pixels).

**Example:** PLC_PRG.iPos_Y.

Increasing this value in runtime mode moves the element downwards.

#### Rotation

Variable (numeric data type). Defines the angle of rotation (in degrees).

**Example:** PLC_PRG.iAngle1.

The midpoint of the element rotates at the “Center” point. This rotation point is shown as the symbol.

In runtime mode, the alignment of the element remains the same with respect to the coordinate system of the visualization. Increasing the value rotates the element to the right.

#### Interior rotation

Variable (numeric data type). Defines the angle of rotation (in degrees).

**Example:** PLC_PRG.iAngle2.

In runtime mode, the element rotates about the point of rotation specified in “Center” according to the value of the variable. In addition, the alignment of the element rotates according to the coordinate system of the visualization. Increasing the value in the code rotates clockwise.

The rotation point is shown as the symbol.

Note: If a static angle of rotation is specified in the “Position Angle” property, then the static angle of rotation is added to the variable angle of rotation (offset) when the visualization is executed.
You can link the variables to a unit conversion.

The "X", "Y", "Rotation", and "Interior rotation" properties are supported by the "Client Animation" functionality.

See also
- Unit conversion
- ...

**Element property 'State variables'**

The variables control the element behavior dynamically.

| "Invisible" | Variable (BOOL). Toggles the visibility of the element. 
| TRUE: The element is not visible at runtime. |

The "Invisible" property is supported by the "Client Animation" functionality.

See also
- ...

**Element property 'Access rights'**

Requirement: User management is set up for the visualization.

| "Access rights" | Opens the "Access rights" dialog. There you can edit the access privileges for the element. 
| Status messages: 
| - "Not set. Full rights.": Access rights for all user groups: "operable" 
| - "Rights are set: Limited rights": Access is restricted for at least one group. |

See also
- ☀ Chapter 1.3.5.19.3.1 "Dialog 'Access Rights" on page 1583

See also
- ...

**Visualization element 'Meter'**

Symbol:
Tag: “Measurement controls”

The element displays the value of a variable. The needle is positioned according to the value of the assigned variable. A meter is used to represent a tachometer, for example.

**Element properties**

<table>
<thead>
<tr>
<th>“Element name”</th>
<th>Example: GenElemInst_1</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Type of element”</td>
<td>“Meter”</td>
</tr>
<tr>
<td>“Value”</td>
<td>Variable (numeric data type). The variable value determines the pointer movement of the element.</td>
</tr>
</tbody>
</table>

**Element property 'Center’**

The properties contain fixed values for the coordinates of the point of rotation. This point of rotation is shown as the symbol. The point is used as the center for rotating and scaling.

| “X” | X-coordinate of the point of rotation |
| “Y” | Y-coordinate of the point of rotation |

You can also change the values by dragging the symbols to other positions in the editor.

These properties are available only when you select the “Preview: Support client animations and overlay of native elements” option in the Visualization Manager.

You can animate a movement, a rotation, and the visibility of a visualization element.

<table>
<thead>
<tr>
<th>“Animation duration”</th>
<th>Time that the element executes an animation (in milliseconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example: 500</td>
<td></td>
</tr>
<tr>
<td>“Move to the foreground”</td>
<td>Property value (BOOL)</td>
</tr>
<tr>
<td>TRUE: At runtime, the element is displayed in the foreground.</td>
<td></td>
</tr>
<tr>
<td>FALSE: At runtime, the element is displayed in the layer where it was inserted in the visualization editor.</td>
<td></td>
</tr>
</tbody>
</table>

See also

- Chapter 1.3.5.21.5 “Using client animation” on page 1962

**Element property ‘Position’**

The position defines the location and size of the element in the visualization window. These are based on the Cartesian coordinate system. The origin is located at the upper left corner of the window. The positive horizontal x-axis runs to the right. The positive vertical y-axis runs downwards.
| **“X”**  | X coordinate of the upper left corner of the element  
Specified in pixels.  
Example: 10. |
|**“Y”**  | Y coordinate of the upper left corner of the element  
Specified in pixels.  
Example: 10. |
|**“Width”**  | Specified in pixels.  
Example: 150 |
|**“Height”**  | Specified in pixels.  
Example: 30 |

You can also change the values by dragging the box symbols ((formatter-icons) to other positions in the editor.

See also
- ♦ Chapter 1.3.5.3.2 “Positioning elements, adapting size and layer” on page 1113

### Element property 'Background'

| **“Image color”**  | List box containing background colors |
|**“Own image”**  |  
- **“Image”**: ID of the background image.  
You select the background image from an image pool by clicking [formatter-icons].  
Info: If you specify the value “<default>” or select the image from the “Default” category in the Input Assistant, then the original element background image is used.  
- **“Transparency color”**: Selection from list box or Input Assistant. |

### Element property 'Arrow'

| **“Hand style”**  | Drop-down list with different arrow types |
|**“Color”**  |  
- [formatter-icons]: The “Color” dialog box opens.  
- [formatter-icons]: Drop-down list with color names |
|**“Arrow start”**  | Angle (in degrees) between the scale start and the horizontal axis |
|**“Arrow end”**  | Angle (in degrees) between the right edge of the pointer instrument and the horizontal axis |
|**“Additional arrow”**  | [formatter-icons]: An additional arrow is shown inside the scale. |
### “Sub scale position”
- **“Outside”**: The subscale is displayed on the outer scale ring. ("Frame outside")
- **“Inside”**: The subscale is displayed on the inner scale ring. ("Frame inside")

### “Scale type”
Type of scale
- **“Lines”**
- **“Dots”**
- **“Squares”**

### “Scale start”
Least value of the scale and the lower limit of the value range for the element
Example: 0
⇒ The “Variable” property is displayed in the line below this.

### “Variable”
Variable (integer data type). Contains the scale start
Example: PLC_PRG.iScaleStart
Declaration:

```plaintext
PROGRAM PLC_PRG
VAR
  iScaleStart : INT := 0;
END_VAR
```

### “Scale end”
Greatest value of the scale and the upper limit of the value range for the element
Example: 100
⇒ The “Variable” property is shown below this.

### “Variable”
Variable (integer data type). Contains the scale end
Example: PLC_PRG.iScaleEnd
Declaration:

```plaintext
PROGRAM PLC_PRG
VAR
  iScaleEnd : INT := 120;
END_VAR
```

### “Main scale”
Distance between two values on the main scale
Example: 10
⇒ The “Variable” property is shown below.

### “Variable”
Variable (integer data type) Contains the distance between two values on the main scale
Example: PLC_PRG.iMainScale
Declaration:

```plaintext
PROGRAM PLC_PRG
VAR
  iMainScale : INT := 20;
END_VAR
```

### “Sub scale”
Distance between two values on the fine scale
You can hide the fine scale by setting the value to 0.
Example: 2
⇒ The “Variable” property is shown below this.
### “Variable”

Variable (integer data type) Contains the distance between two values on the fine scale  
**Example:** PLC_PRG.iSubScale  
**Declaration:**

```plaintext
PROGRAM PLC_PRG  
VAR  
iSubScale : INT := 5;  
END_VAR
```

### “Scale line width”

Specified in pixels  
**Example:** 3

### “Scale color”

Color of scale lines  
- ▶️: The “Color” dialog opens.  
- ▼️: A list box with style colors opens.

### “Scale in 3D”

☑️: Scale lines are displayed with soft 3D shadowing.  
**Note:** This property is not displayed in “FlatStyle”.

### “Show scale”

☑️: The scale is displayed.

### “Frame inside”

☑️: A frame is drawn at the inner end of the scale.

### “Frame outside”

☑️: A frame is drawn at the outer end of the scale.

---

**Element property 'Label'**

### “Label”

Selection list  
- “Outside”: Scale values are placed outside of the scale.  
- “Inside”: Scale values are placed inside of the scale.

### “Unit”

Text that is displayed in the element.  
**Example:** Units displayed in m/s.

### “Font”

Font for labels (example: scale numbering).  
Selection from the drop-down list or by clicking the “” button.

### “Scale format (C Syntax)”

Values scaled in "printf" syntax  
**Examples:** %d, %5.2f

### “Max. text width of labels”

(optional) Value that redefines the maximum width of the scale label. The correct value is normally set automatically.  
**Note:** Change this value only if the automatic adjustment does not yield the expected result.

### “Text height of labels”

(optional) Value that redefines the maximum height of the scale label. The correct value is normally set automatically.  
**Note:** Change this value only if the automatic adjustment does not yield the expected result.

### “Font color”

Selection from the drop-down list or by clicking the “” button.
**Usage of**

- “Preset style values”: Values from the current style
- “User-defined settings”: The subnode “Positioning” appears.

**Positioning**

Requirement: “User-defined settings” is selected as “Usage of”.

The displayed positioning settings depend on the type of needle instrument and potentiometer, and partially on whether a custom background image is selected. The following settings are used for achieving the exact position relative to the background image.

<table>
<thead>
<tr>
<th>Setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Needle movement”</td>
<td>Length of the needle (in pixels)</td>
</tr>
<tr>
<td>“Scale movement”</td>
<td>Distance from the tick marks to the center (in pixels)</td>
</tr>
<tr>
<td></td>
<td>Requirement: A customer image is selected as “Background”.</td>
</tr>
<tr>
<td>“Scale length”</td>
<td>Length of the tick marks (in pixels)</td>
</tr>
<tr>
<td></td>
<td>Requirement: A customer image is selected as “Background”.</td>
</tr>
<tr>
<td>“Label offset”:</td>
<td>Distance from the labels to the tick marks (in pixels)</td>
</tr>
<tr>
<td>“Unit offset”:</td>
<td>Distance of the unit text “Label Unit” from the upper scale edge (in pixels)</td>
</tr>
<tr>
<td>“Origin offset”</td>
<td>Offset of the element (in pixels)</td>
</tr>
<tr>
<td></td>
<td>Requirement: For the elements “Meter 180°” and “Meter 90°”, this property is displayed only if a custom image is selected as “Background”.</td>
</tr>
</tbody>
</table>

**Element property ‘Colors’**

**“Color areas”**

- **“Durable color areas”**: □: All color areas are visible, regardless of the current value. □: Only the color area is visible that includes the current value.
- **“Use colors for scale”**: □: Colors in the color area are used only for the scale and frame.
- **“Color areas”**

  - **Create new**: A new color area is added to the “Elements” view.
  - **Delete**: The color area is removed from the list and the list is refreshed.

**“Begin of area”**

Start value of the color area

Example: 20

- The property “Variable” is shown below.

**“Variable”**

Variable (integer data type). Contains the start value.

Example: PLC_PRG.iColorAreaStart0

Declaration:

```plaintext
PROGRAM PLC_PRG
VAR
  iColorAreaStart0 : INT := 80;
END_VAR
```

**“End of area”**

End value of the color area

Example: 120

- The property “Variable” is shown below.
**“Variable”**

Variable (integer data type). Contains the end value.

**Example:** `IColorAreaEnd0`

**Declaration:**

```plaintext
PROGRAM PLC_PRG
VAR
  iColorAreaEnd0 : INT := 100;
END_VAR
```

**“Color”**

Color that is used for displaying the area.

---

**Element property ‘Absolute movement’**

The properties contain IEC variables for controlling the position of the element dynamically. The reference point is the upper left corner of the element. In runtime mode, the entire element is moved.

---

**“Movement”**

**“X”**

Variable (numeric data type). Defines the X position (in pixels).

**Example:** `PLC_PRG.iPos_X`

Increasing this value in runtime mode moves the element to the right.

**“Y”**

Variable (numeric data type). Defines the Y position (in pixels).

**Example:** `PLC_PRG.iPos_Y`

Increasing this value in runtime mode moves the element downwards.

**“Rotation”**

Variable (numeric data type). Defines the angle of rotation (in degrees).

**Example:** `PLC_PRG.iAngle1`

The midpoint of the element rotates at the “Center” point. This rotation point is shown as the symbol.

In runtime mode, the alignment of the element remains the same with respect to the coordinate system of the visualization. Increasing the value rotates the element to the right.

**“Interior rotation”**

Variable (numeric data type). Defines the angle of rotation (in degrees).

**Example:** `PLC_PRG.iAngle2`

In runtime mode, the element rotates about the point of rotation specified in “Center” according to the value of the variable. In addition, the alignment of the element rotates according to the coordinate system of the visualization. Increasing the value in the code rotates clockwise.

The rotation point is shown as the symbol.

Note: If a static angle of rotation is specified in the “Position ➔ Angle” property, then the static angle of rotation is added to the variable angle of rotation (offset) when the visualization is executed.
You can link the variables to a unit conversion.

The "X", "Y", "Rotation", and "Interior rotation" properties are supported by the "Client Animation" functionality.

See also
- Unit conversion

Element property 'State variables'

<table>
<thead>
<tr>
<th>&quot;Invisible&quot;</th>
<th>Variable (BOOL). Toggles the visibility of the element.</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRUE: The element is not visible at runtime.</td>
<td></td>
</tr>
</tbody>
</table>

The "Invisible" property is supported by the "Client Animation" functionality.

See also
- 

Element property 'Access rights'

Requirement: User management is set up for the visualization.

<table>
<thead>
<tr>
<th>&quot;Access rights&quot;</th>
<th>Opens the &quot;Access rights&quot; dialog. There you can edit the access privileges for the element.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Status messages:</td>
<td>- &quot;Not set. Full rights.&quot;: Access rights for all user groups: &quot;operable&quot;</td>
</tr>
<tr>
<td></td>
<td>- &quot;Rights are set: Limited rights&quot;: Access is restricted for at least one group.</td>
</tr>
</tbody>
</table>

See also
- ≈ Chapter 1.3.5.19.3.1 "Dialog 'Access Rights'" on page 1583

See also
- 

Visualization element 'Potentiometer'

Symbol: Tag: "Measurement controls"
The element displays the value of a variable as a setting on the potentiometer. A visualization user can modify the value by dragging the pointer to another position.

See also

Element properties

<table>
<thead>
<tr>
<th>“Element name”</th>
<th>Example: GenElemInst_1</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Type of element”</td>
<td>“Potentiometer”</td>
</tr>
<tr>
<td>“Variable”</td>
<td>Variable (numeric data type). Contains the position of the pointer for the potentiometer. A visualization user can modify the value by dragging the pointer to another position.</td>
</tr>
</tbody>
</table>

Element property ‘Center’

The properties contain fixed values for the coordinates of the point of rotation. This point of rotation is shown as the symbol. The point is used as the center for rotating and scaling.

| “X” | X-coordinate of the point of rotation |
| “Y” | Y-coordinate of the point of rotation |

You can also change the values by dragging the symbols (—at runtime in the editor.

These properties are available only when you select the “Preview: Support client animations and overlay of native elements” option in the Visualization Manager.

You can animate a movement, a rotation, and the visibility of a visualization element.

<table>
<thead>
<tr>
<th>“Animation duration”</th>
<th>Time that the element executes an animation (in milliseconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example: 500</td>
<td></td>
</tr>
<tr>
<td>“Move to the foreground”</td>
<td>Property value (BOOL)</td>
</tr>
<tr>
<td>TRUE: At runtime, the element is displayed in the foreground.</td>
<td></td>
</tr>
<tr>
<td>FALSE: At runtime, the element is displayed in the layer where it was inserted in the visualization editor.</td>
<td></td>
</tr>
</tbody>
</table>

See also

- Chapter 1.3.5.21.5 “Using client animation” on page 1962
Element property 'Position'

The position defines the location and size of the element in the visualization window. These are based on the Cartesian coordinate system. The origin is located at the upper left corner of the window. The positive horizontal x-axis runs to the right. The positive vertical y-axis runs downwards.

<table>
<thead>
<tr>
<th>Element property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;X&quot;</td>
<td>X coordinate of the upper left corner of the element. Specified in pixels. Example: 10.</td>
</tr>
<tr>
<td>&quot;Y&quot;</td>
<td>Y coordinate of the upper left corner of the element. Specified in pixels. Example: 10.</td>
</tr>
<tr>
<td>&quot;Width&quot;</td>
<td>Specified in pixels. Example: 150</td>
</tr>
<tr>
<td>&quot;Height&quot;</td>
<td>Specified in pixels. Example: 30</td>
</tr>
</tbody>
</table>

You can also change the values by dragging the box symbols (_drag_to_position) to other positions in the editor.

See also
- ☀ Chapter 1.3.5.3.2 “Positioning elements, adapting size and layer” on page 1113

Element property 'Background'

<table>
<thead>
<tr>
<th>Element property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Image color&quot;</td>
<td>List box containing background colors</td>
</tr>
<tr>
<td>&quot;Own image&quot;</td>
<td>• &quot;Image&quot;: ID of the background image. You select the background image from an image pool by clicking_drag_to_image. Info: If you specify the value &quot;&lt;default&gt;&quot; or select the image from the &quot;Default&quot; category in the Input Assistant, then the original element background image is used. • &quot;Transparency color&quot;: Selection from list box or Input Assistant.</td>
</tr>
</tbody>
</table>

Element property 'Arrow'

<table>
<thead>
<tr>
<th>Element property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Hand style&quot;</td>
<td>Drop-down list with different arrow types</td>
</tr>
<tr>
<td>&quot;Color&quot;</td>
<td>• <em>drag_to_color</em>: The “Color” dialog box opens. • ^: Drop-down list with color names</td>
</tr>
<tr>
<td>&quot;Arrow start&quot;</td>
<td>Angle (in degrees) between the left edge of the element and the horizontal axis</td>
</tr>
<tr>
<td>&quot;Arrow end&quot;</td>
<td>Angle (in degrees) between the right edge of the element and the horizontal axis</td>
</tr>
</tbody>
</table>

Element property 'Scale'
### Sub scale position
- **Outside**: The subscale is displayed on the outer scale ring. ("Frame outside")
- **Inside**: The subscale is displayed on the inner scale ring. ("Frame inside")

### Scale type
Type of scale
- **Lines**
- **Dots**
- **Squares**

### Scale start
Least value of the scale and the lower limit of the value range for the element
Example: 0
*: The "Variable" property is displayed in the line below this.

### Variable
Variable (integer data type). Contains the scale start
Example: PLC_PRG.iScaleStart
Declaration:

```
PROGRAM PLC_PRG
VAR
  iScaleStart : INT := 0;
END_VAR
```

### Scale end
Greatest value of the scale and the upper limit of the value range for the element
Example: 100
*: The "Variable" property is shown below this.

### Main scale
Distance between two values on the main scale
Example: 10
*: The "Variable" property is shown below.

### Variable
Variable (integer data type) Contains the distance between two values on the main scale
Example: PLC_PRG.iMainScale
Declaration:

```
PROGRAM PLC_PRG
VAR
  iMainScale : INT := 20;
END_VAR
```

### Sub scale
Distance between two values on the fine scale
You can hide the fine scale by setting the value to 0.
Example: 2
*: The "Variable" property is shown below this.
"Variable" Variable (integer data type) Contains the distance between two values on the fine scale
Example: PLC_PRG.iSubScale
Declaration:

PROGRAM PLC_PRG
VAR
  iSubScale : INT := 5;
END_VAR

"Scale line width" Specified in pixels
Example: 3

"Scale color" Color of scale lines
- ▉: The "Color" dialog opens.
- ▼: A list box with style colors opens.

"Scale in 3D" ▽: Scale lines are displayed with soft 3D shadowing.
Note: This property is not displayed in "FlatStyle".

"Show scale" ▽: The scale is displayed.

"Frame inside" ▽: A frame is drawn at the inner end of the scale.

"Frame outside" ▽: A frame is drawn at the outer end of the scale.

Element property 'Label'

"Label" Selection list
- "Outside": Scale values are placed outside of the scale.
- "Inside": Scale values are placed inside of the scale.

"Unit" Text that is displayed in the element.
Example: Units displayed in m/s.

"Font" Font for labels (example: scale numbering).
Selection from the drop-down list or by clicking the " " button.

"Scale format (C Syntax)" Values scaled in "printf" syntax
Examples: %d, %5.2f

"Max. text width of labels" (optional) Value that redefines the maximum width of the scale label. The correct value is normally set automatically.
Note: Change this value only if the automatic adjustment does not yield the expected result.

"Text height of labels" (optional) Value that redefines the maximum height of the scale label. The correct value is normally set automatically.
Note: Change this value only if the automatic adjustment does not yield the expected result.

"Font color" Selection from the drop-down list or by clicking the " " button.

Element property 'Positioning'
“Usage of”
- “Preset style values”: Values from the current style
- “User-defined settings”: The subnode “Positioning” appears.

“Positioning”
Requirement: “User-defined settings” is selected as “Usage of”.

The displayed positioning settings depend on the type of needle instrument and potentiometer, and partially on whether a custom background image is selected. The following settings are used for achieving the exact position relative to the background image.

| “Needle movement”               | Length of the needle (in pixels) |
| “Scale movement”               | Distance from the tick marks to the center (in pixels) |
|                               | Requirement: A customer image is selected as “Background” |
| “Scale length”                 | Length of the tick marks (in pixels) |
|                               | Requirement: A customer image is selected as “Background” |
| “Label offset”                 | Distance from the labels to the tick marks (in pixels) |
| “Unit offset”                  | Distance of the unit text “Label ➔ Unit” from the upper scale edge (in pixels) |
| “Origin offset”                | Offset of the element (in pixels) |
|                               | Requirement: For the elements “Meter 180°” and “Meter 90°”, this property is displayed only if a custom image is selected as “Background” |

Element property ‘Colors’

“Color areas”

“Durable color areas”
- ☐: All color areas are visible, regardless of the current value.
- ☑: Only the color area is visible that includes the current value.

“Use colors for scale”
- ☑: Colors in the color area are used only for the scale and frame.

“Color areas”

“Create new”
A new color area is added to the “Elements” view.

“Delete”
The color area is removed from the list and the list is refreshed.

“Begin of area”
Start value of the color area
Example: 20
- The property “Variable” is shown below.

“Variable”
Variable (integer data type). Contains the start value.
Example: PLC_PRG.iColorAreaStart0
Declaration:

```
PROGRAM PLC_PRG
VAR
  iColorAreaStart0 : INT := 80;
END_VAR
```

“End of area”
End value of the color area
Example: 120
- The property “Variable” is shown below.
**“Variable”**

Variable (integer data type). Contains the end value.

**Example:** `iColorAreaEnd0`  
**Declaration:**

```plaintext
PROGRAM PLC_PRG
VAR
  iColorAreaEnd0 : INT := 100;
END_VAR
```

**“Color”**

Color that is used for displaying the area.

---

**Element property ‘State variables’**

The variables control the element behavior dynamically.

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
</table>
| **“Invisible”** | Variable (BOOL). Toggles the visibility of the element.  
TRUE: The element is not visible at runtime. |
| **“Deactivate inputs”** | Variable (BOOL). Toggles the operability of the element.  
TRUE: User inputs do not have any effect in runtime more. The element is shown as deactivated. |

See also

-  

**Element property ‘Absolute movement’**

The properties contain IEC variables for controlling the position of the element dynamically. The reference point is the upper left corner of the element. In runtime mode, the entire element is moved.

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>“Movement”</strong></td>
<td></td>
</tr>
</tbody>
</table>
| **“X”** | Variable (numeric data type). Defines the X position (in pixels).  
**Example:** `PLC_PRG.iPos_X`.  
Increasing this value in runtime mode moves the element to the right. |
| **“Y”** | Variable (numeric data type). Defines the Y position (in pixels).  
**Example:** `PLC_PRG.iPos_Y`.  
Increasing this value in runtime mode moves the element downwards. |
**"Rotation"**

Variable (numeric data type). Defines the angle of rotation (in degrees).

**Example:** PLC_PRG.iAngle1.

The midpoint of the element rotates at the "Center" point. This rotation point is shown as the symbol.

In runtime mode, the alignment of the element remains the same with respect to the coordinate system of the visualization. Increasing the value rotates the element to the right.

---

**"Interior rotation"**

Variable (numeric data type). Defines the angle of rotation (in degrees).

**Example:** PLC_PRG.iAngle2.

In runtime mode, the element rotates about the point of rotation specified in "Center" according to the value of the variable. In addition, the alignment of the element rotates according to the coordinate system of the visualization. Increasing the value in the code rotates clockwise.

The rotation point is shown as the symbol.

Note: If a static angle of rotation is specified in the "Position Angle" property, then the static angle of rotation is added to the variable angle of rotation (offset) when the visualization is executed.

---

You can link the variables to a unit conversion.

---

The "X", "Y", "Rotation", and "Interior rotation" properties are supported by the "Client Animation" functionality.

---

See also

- Unit conversion

---

**Element property 'Access rights'**

Requirement: User management is set up for the visualization.

**"Access rights"**

Opens the "Access rights" dialog. There you can edit the access privileges for the element.

Status messages:

- "Not set. Full rights.": Access rights for all user groups : "operable"
- "Rights are set: Limited rights": Access is restricted for at least one group.

See also

- Chapter 1.3.5.19.3.1 “Dialog ‘Access Rights’” on page 1583
Visualization element 'Histogram'

Symbol:

Tag: “Measurement controls”

The element displays the data of a one-dimensional array as a histogram. You can define specific colors for certain value ranges.

See also

● Chapter 1.3.5.21.2 “Displaying array data in histograms” on page 1958

Element properties

<table>
<thead>
<tr>
<th>&quot;Element name&quot;</th>
<th>Example: GenElemInst_35</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Type of element&quot;</td>
<td>“Histogram”</td>
</tr>
<tr>
<td>&quot;Data array&quot;</td>
<td>One-dimensional array with data displayed in this histogram. Example: PLC_PRG.arr1</td>
</tr>
</tbody>
</table>

Element property 'Subrange of array'

| "Use subrange" | ✔: Only part of the array is displayed in the histogram. |
| "Start index" | First array index with a displayed value. Requirement: "Use subrange" is activated. |
| "End index" | Last array index with a displayed value. Requirement: "Use subrange" is activated. |

| "Display type" | ● "Bars": Data is displayed as bars. ● "Lines": Data is displayed as lines. ● "Curve": Interpolation of data into a curve. |
| "Line width" | Specified in pixels Requirement: "Curve" is selected as the "Display type". |
| "Show horizontal lines" | ✔: Horizontal lines are drawn on the main scale. Note: Not all visualization styles have this property. This element property is not available for visualization styles that have striped backgrounds (example: “Flat style”). |
| "Relative bar width" | Integer value between 1 and 100 ● 1: The bars are drawn as lines. ● 100: The complete width of the histogram is filled with the bars. |

Element property ‘Position’

The position defines the location and size of the element in the visualization window. These are based on the Cartesian coordinate system. The origin is located at the upper left corner of the window. The positive horizontal x-axis runs to the right. The positive vertical y-axis runs downwards.
<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;X&quot;</td>
<td>X coordinate of the upper left corner of the element. Specified in pixels. Example: 10.</td>
</tr>
<tr>
<td>&quot;Y&quot;</td>
<td>Y coordinate of the upper left corner of the element. Specified in pixels. Example: 10.</td>
</tr>
<tr>
<td>&quot;Width&quot;</td>
<td>Specified in pixels. Example: 150</td>
</tr>
<tr>
<td>&quot;Height&quot;</td>
<td>Specified in pixels. Example: 30</td>
</tr>
</tbody>
</table>

You can also change the values by dragging the box symbols ( draggable) to other positions in the editor.

See also
- Chapter 1.3.5.3.2 “Positioning elements, adapting size and layer” on page 1113

Element property 'Center'
The properties contain fixed values for the coordinates of the point of rotation. This point of rotation is shown as the symbol. The point is used as the center for rotating and scaling.

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;X&quot;</td>
<td>X-coordinate of the point of rotation</td>
</tr>
<tr>
<td>&quot;Y&quot;</td>
<td>Y-coordinate of the point of rotation</td>
</tr>
</tbody>
</table>

You can also change the values by dragging the symbols ( draggable) to other positions in the editor.

Element property 'Scale'

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Scale start&quot;</td>
<td>Least value of the scale and the lower limit of the value range for the element. Example: 0</td>
</tr>
<tr>
<td>&quot;Variable&quot;</td>
<td>Variable (integer data type). Contains the scale start. Example: PLC_PRG.iScaleStart Declaration:</td>
</tr>
</tbody>
</table>

```plaintext
PROGRAM PLC_PRG
VAR
   iScaleStart : INT := 0;
END_VAR
```
<table>
<thead>
<tr>
<th><strong>Property</strong></th>
<th>Description</th>
</tr>
</thead>
</table>
| **“Scale end”** | Greatest value of the scale and the upper limit of the value range for the element.  
Example: 100 |
| **“Variable”** | Variable (integer data type). Contains the scale end.  
Example: PLC_PRG.iScaleEnd |
| Declaration: |  
PROGRAM PLC_PRG  
VAR  
iScaleEnd : INT := 120;  
END_VAR |
| **“Main scale”** | Distance between 2 values on the rough scale.  
Example: 10 |
| **“Variable”** | Variable (integer data type). Contains the distance.  
Example: PLC_PRG.iMainScale |
| Declaration: |  
PROGRAM PLC_PRG  
VAR  
iMainScale : INT := 20;  
END_VAR |
| **“Subscale”** | Distance between 2 values on the fine scale.  
You can hide the fine scale by setting the value to 0.  
Example: 2 |
| **“Variable”** | Variable (integer data type). Contains the spacing.  
Example: PLC_PRG.iSubScale |
| Declaration: |  
PROGRAM PLC_PRG  
VAR  
iSubScale : INT := 5;  
END_VAR |
| **“Scale color”** | Color of scale lines  
- ▲: The “Color” dialog box opens.  
- ▼: A drop-down list with color names opens. |
| **“Base line”** | Value of the main scale where the horizontal base line of the histogram is located.  
The drawing of the bar starts at the base line. |

**Element property ‘Label’**
<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Unit&quot;</td>
<td>Text that is displayed in the element. Example: Units displayed in m/s.</td>
</tr>
<tr>
<td>&quot;Font&quot;</td>
<td>Font for labels (example: scale numbering). Selection from the drop-down list or by clicking the &quot;...&quot; button.</td>
</tr>
<tr>
<td>&quot;Scale format (C Syntax)&quot;</td>
<td>Values scaled in &quot;printf&quot; syntax. Examples: %d, %5.2f.</td>
</tr>
<tr>
<td>&quot;Max. text width of labels&quot;</td>
<td>Optional value that defines the maximum width of the scale label. Note: Change this value only if the automatic adjustment does not yield the expected result.</td>
</tr>
<tr>
<td>&quot;Text height of labels&quot;</td>
<td>Optional value that defines the maximum height of the scale label. Note: Change this value only if the automatic adjustment does not yield the expected result.</td>
</tr>
<tr>
<td>&quot;Font color&quot;</td>
<td>Selection from the drop-down list or by clicking the &quot;...&quot; button.</td>
</tr>
</tbody>
</table>

**Element property 'Colors'**

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Graph color&quot;</td>
<td>Color of the bar in normal state. Note: The normal state is in effect when the current value of the array component does not fulfill the alarm condition.</td>
</tr>
<tr>
<td>&quot;Alarm value&quot;</td>
<td>Threshold for the alarm</td>
</tr>
<tr>
<td>&quot;Alarm condition&quot;</td>
<td>If the current value of the array component fulfills the alarm condition, then the alarm condition is set.</td>
</tr>
<tr>
<td></td>
<td>● &quot;Less&quot;: The current value is less than the “Alarm value”</td>
</tr>
<tr>
<td></td>
<td>● &quot;More&quot;: The current value is greater than the “Alarm value”</td>
</tr>
<tr>
<td>&quot;Alarm color&quot;</td>
<td>Color of the bar in alarm state.</td>
</tr>
<tr>
<td>&quot;Use color areas&quot;</td>
<td>☑: The color areas defined in this element are used.</td>
</tr>
<tr>
<td>&quot;Color areas&quot;</td>
<td></td>
</tr>
<tr>
<td>&quot;Create new&quot;</td>
<td>A new color area is added.</td>
</tr>
<tr>
<td>&quot;Delete&quot;</td>
<td>The color area is removed from the list.</td>
</tr>
<tr>
<td>&quot;Begin of area&quot;</td>
<td>The start value on the “Scale” of the histogram where the color area begins.</td>
</tr>
<tr>
<td>&quot;End of area&quot;</td>
<td>The end value on the “Scale” of the histogram where the color area ends.</td>
</tr>
<tr>
<td>&quot;Color&quot;</td>
<td>Color that is used for displaying the area.</td>
</tr>
</tbody>
</table>

**Element property 'Absolute movement'**

The properties contain IEC variables for controlling the position of the element dynamically. The reference point is the upper left corner of the element. In runtime mode, the entire element is moved.

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Movement&quot;</td>
<td></td>
</tr>
<tr>
<td>&quot;X&quot;</td>
<td>Variable (numeric data type). Defines the X position (in pixels). Example: PLC_PRG.iPos_X. Increase this value in runtime mode moves the element to the right.</td>
</tr>
</tbody>
</table>
**“Y”**

Variable (numeric data type). Defines the Y position (in pixels).

**Example:** PLC_PRG.iPos_Y.

Increasing this value in runtime mode moves the element downwards.

**“Rotation”**

Variable (numeric data type). Defines the angle of rotation (in degrees).

**Example:** PLC_PRG.iAngle1.

The midpoint of the element rotates at the “Center” point. This rotation point is shown as the symbol.

In runtime mode, the alignment of the element remains the same with respect to the coordinate system of the visualization. Increasing the value rotates the element to the right.

**“Interior rotation”**

Variable (numeric data type). Defines the angle of rotation (in degrees).

**Example:** PLC_PRG.iAngle2.

In runtime mode, the element rotates about the point of rotation specified in “Center” according to the value of the variable. In addition, the alignment of the element rotates according to the coordinate system of the visualization. Increasing the value in the code rotates clockwise.

The rotation point is shown as the symbol.

Note: If a static angle of rotation is specified in the “Position ➔ Angle” property, then the static angle of rotation is added to the variable angle of rotation (offset) when the visualization is executed.

---

You can link the variables to a unit conversion.

The “X”, “Y”, “Rotation”, and “Interior rotation” properties are supported by the “Client Animation” functionality.

See also

- [Unit conversion](#)

These properties are available only when you select the “Preview: Support client animations and overlay of native elements” option in the Visualization Manager.

You can animate a movement, a rotation, and the visibility of a visualization element.
“Animation duration” | Time that the element executes an animation (in milliseconds)
---|---
Example: | 500

“Move to the foreground” | Property value (BOOL)
---|---
Property value | TRUE: At runtime, the element is displayed in the foreground.
FALSE: At runtime, the element is displayed in the layer where it was inserted in the visualization editor.

See also
- Chapter 1.3.5.21.5 “Using client animation” on page 1962

Element property ‘Access rights’

Requirement: User management is set up for the visualization.

“Access rights” | Opens the “Access rights” dialog. There you can edit the access privileges for the element.
---|---
Status messages:
- “Not set. Full rights.”: Access rights for all user groups: “operable”
- “Rights are set: Limited rights”: Access is restricted for at least one group.

See also
- Chapter 1.3.5.19.3.1 “Dialog ‘Access Rights’” on page 1583

Visualization element ‘Image switcher’

Symbol:

Tag: “Lamps/Switches/Bitmaps”
The element displays one of three referenced images. Mouse actions change the displayed image. The images are defined in the “Image settings” element properties. The effects of mouse clicks are defined in the “Element behavior” property.

Element properties

“Element name” | Optional
---|---
Hint: Assign individual names for elements so that they are found faster in the element list.
Example: ImageSwitcher_1

“Type of element” | “Image switcher”

Element property ‘Position’
The position defines the location and size of the element in the visualization window. These are based on the Cartesian coordinate system. The origin is located at the upper left corner of the window. The positive horizontal x-axis runs to the right. The positive vertical y-axis runs downwards.
### Positioning Elements

**“X”**  
X coordinate of the upper left corner of the element  
Specified in pixels.  
Example: 10.

**“Y”**  
Y coordinate of the upper left corner of the element  
Specified in pixels.  
Example: 10.

**“Width”**  
Specified in pixels.  
Example: 150

**“Height”**  
Specified in pixels.  
Example: 30

You can also change the values by dragging the box symbols (낭) to other positions in the editor.

See also
- ☞ Chapter 1.3.5.3.2 “Positioning elements, adapting size and layer” on page 1113

### Variable

**“Variable”**  
Variable (BOOL).  
The value of the variable changes according to user input and it is independent of the “Element behavior” element property.

### Image Settings

**“Image “on””**  
Image ID from an image pool. The image can be selected using the input assistant.  
The image is used if the variable of the “Variable” property has the value TRUE.

**“Image “off””**  
Image ID from an image pool. The image can be selected using the input assistant.  
The image is used if the variable of the “Variable” property has the value FALSE.

**“Image “clicked””**  
Image ID from an image pool. The image is selected using the input assistant.  
In runtime mode, the visualization displays the referenced image when the element is clicked (and the mouse button is held down).  
Requirement: The “Element behavior” is “Image toggler”.

**“Transparency”**  
☑️ The “Transparent color” is selected.

**“Transparent color”**  
The image pixels that have the transparent color are displayed as transparent.  
Requirement: “Transparency” is activated.  
- ➕ The “Color” dialog box opens.  
- ⤴ A drop-down list with color names opens.
### “Scaling type”

Defines how an image fits in the element frame.

- **“Fixed”**: The original size of the image is retained, regardless of the dimensions of the element.
- **“Isotropic”**: The entire image is shown in the element frame, either larger or smaller. As a result, the proportion of height and width are retained.
- **“Anisotropic”**: The image resizes automatically to the dimensions of the element frame, filling the entire element frame. As a result, the proportions are not retained.

### “Horizontal alignment”

Horizontal alignment of the image within the element frame or element

- Left
- Centered
- Right

Requirement: “Scaling type” is “Isotropic”.

### “Vertical alignment”

Vertical alignment of the image within the element frame or element

- Top
- Centered
- Bottom

Requirement: “Scaling type” is “Isotropic”.

### “Element behavior”

- **“Image toggler”**: Every mouse click switches the image.
- **“Image tapper”**: While a visualization user holds down the mouse button, the image of the “Image on” property is displayed. At the same time, the value TRUE is assigned to the “Variable” property.

### “Tap FALSE”

✅: While the mouse button is pressed, the image of the “Image” property is displayed and the “Variable” property gets the value FALSE instead of the value TRUE, and back.

Requirement: “Image tapper” is selected in the “Element behavior” property.

### Element property 'Center'

The properties contain fixed values for the coordinates of the point of rotation. This point of rotation is shown as the symbol. The point is used as the center for rotating and scaling.

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>X-coordinate of the point of rotation</td>
</tr>
<tr>
<td>Y</td>
<td>Y-coordinate of the point of rotation</td>
</tr>
</tbody>
</table>

You can also change the values by dragging the symbols to other positions in the editor.

### Element property 'Texts'

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Tooltip”</td>
<td>String display as tooltip for the element</td>
</tr>
<tr>
<td>Example</td>
<td>Valid access.</td>
</tr>
</tbody>
</table>
The properties contain IEC variables for controlling the position of the element dynamically. The reference point is the upper left corner of the element. In runtime mode, the entire element is moved.

**“Movement”**

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>“X”</td>
<td>Variable (numeric data type). Defines the X position (in pixels).</td>
<td><code>PLC_PRG.iPos_X</code>.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Increasing this value in runtime mode moves the element to the right.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Increasing this value in runtime mode moves the element downwards.</td>
</tr>
<tr>
<td>“Rotation”</td>
<td>Variable (numeric data type). Defines the angle of rotation (in degrees).</td>
<td><code>PLC_PRG.iAngle1</code>.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The midpoint of the element rotates at the “Center” point. This rotation point is shown as the symbol.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>In runtime mode, the alignment of the element remains the same with respect to the coordinate system of the visualization. Increasing the value rotates the element to the right.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>In runtime mode, the element rotates about the point of rotation specified in “Center” according to the value of the variable. In addition, the alignment of the element rotates according to the coordinate system of the visualization. Increasing the value in the code rotates clockwise.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The rotation point is shown as the symbol.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Note: If a static angle of rotation is specified in the “Position ➔ Angle” property, then the static angle of rotation is added to the variable angle of rotation (offset) when the visualization is executed.</td>
</tr>
</tbody>
</table>

You can link the variables to a unit conversion.

The “X”, “Y”, “Rotation”, and “Interior rotation” properties are supported by the “Client Animation” functionality.
The variables control the element behavior dynamically.

<table>
<thead>
<tr>
<th>&quot;Invisible&quot;</th>
<th>Variable (BOOL). Toggles the visibility of the element.</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRUE: The element is not visible at runtime.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>&quot;Deactivate inputs&quot;</th>
<th>Variable (BOOL). Toggles the operability of the element.</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRUE: User inputs do not have any effect in runtime more. The element is shown as deactivated.</td>
<td></td>
</tr>
</tbody>
</table>

The "Invisible" property is supported by the "Client Animation" functionality.

These properties are available only when you select the "Preview: Support client animations and overlay of native elements" option in the Visualization Manager.

You can animate a movement, a rotation, and the visibility of a visualization element.

<table>
<thead>
<tr>
<th>&quot;Animation duration&quot;</th>
<th>Time that the element executes an animation (in milliseconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example: 500</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>&quot;Move to the foreground&quot;</th>
<th>Property value (BOOL)</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRUE: At runtime, the element is displayed in the foreground.</td>
<td></td>
</tr>
<tr>
<td>FALSE: At runtime, the element is displayed in the layer where it was inserted in the visualization editor.</td>
<td></td>
</tr>
</tbody>
</table>

Requirement: User management is set up for the visualization.

<table>
<thead>
<tr>
<th>&quot;Access rights&quot;</th>
<th>Opens the &quot;Access rights&quot; dialog. There you can edit the access privileges for the element.</th>
</tr>
</thead>
</table>
| Status messages:| • "Not set. Full rights.
|                 | • "Rights are set: Limited rights": Access is restricted for at least one group. |

See also

- Chapter 1.3.5.21.5 “Using client animation” on page 1962
- Chapter 1.3.5.19.3.1 “Dialog ‘Access Rights’” on page 1583
Visualization element 'Lamp'

Symbol: 

Tag: “Lamps/Switches/Bitmaps”
The element shows the value of a variable, and the element is displayed as illuminated or not.

Element properties

<table>
<thead>
<tr>
<th>“Element name”</th>
<th>Optional</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hint: Assign individual names for elements so that they are found faster in the element list.</td>
<td></td>
</tr>
<tr>
<td>Example: Lamp_green</td>
<td></td>
</tr>
</tbody>
</table>

| “Type of element” | “Lamp” |

Element property ‘Position’
The position defines the location and size of the element in the visualization window. These are based on the Cartesian coordinate system. The origin is located at the upper left corner of the window. The positive horizontal x-axis runs to the right. The positive vertical y-axis runs downwards.

<table>
<thead>
<tr>
<th>“X”</th>
<th>X coordinate of the upper left corner of the element Specified in pixels.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Example: 10.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>“Y”</th>
<th>Y coordinate of the upper left corner of the element Specified in pixels.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Example: 10.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>“Width”</th>
<th>Specified in pixels.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Example: 150</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>“Height”</th>
<th>Specified in pixels.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Example: 30</td>
</tr>
</tbody>
</table>

You can also change the values by dragging the box symbols (_drag-) to other positions in the editor.

See also

- ☮ Chapter 1.3.5.3.2 “Positioning elements, adapting size and layer” on page 1113
“Variable”

Variable (BOOL).
The variable value is displayed as a lamp that goes on (TRUE) or off (FALSE).

Image settings

“Transparency” ☑: The “Transparent color” property is selected.

“Transparent color”

Pixels in this color are displayed as transparent.
Requirement: “Transparency” is activated.
- The “Color” dialog box opens.
- A drop-down list with style colors opens.

“Scaling type”

Reaction of the element when the dimension of the “Frame” element is changed:
- “Isotropic”: The height and width of the image are resized proportionally to the “Frame”.
  Please note: To retain the alignment of elements also within a scaled “Frame” element, define the “Horizontal alignment” or “Vertical alignment” explicitly with “Centered”.
- “Anisotropic”: The image fills the entire “Frame” regardless of its proportions.

“Horizontal alignment”

Horizontal alignment of the image within the element frame or element
- Left
- Centered
- Right
Requirement: “Scaling type” is “Isotropic”.

“Vertical alignment”

Vertical alignment of the image within the element frame or element
- Top
- Centered
- Bottom
Requirement: “Scaling type” is “Isotropic”.

Element property 'Center'

The properties contain fixed values for the coordinates of the point of rotation. This point of rotation is shown as the symbol. The point is used as the center for rotating and scaling.

“X”

X-coordinate of the point of rotation

“Y”

Y-coordinate of the point of rotation

You can also change the values by dragging the symbols to other positions in the editor.

Element property 'Texts'

“Tooltip”

String display as tooltip for the element
Example: Valid access.
The properties contain IEC variables for controlling the position of the element dynamically. The reference point is the upper left corner of the element. In runtime mode, the entire element is moved.

**“Movement”**

| “X”   | Variable (numeric data type). Defines the X position (in pixels).  
Example: PLC_PRG.iPos_X.  
Increasing this value in runtime mode moves the element to the right. |
|-------|------------------------------------------------------------------|
| “Y”   | Variable (numeric data type). Defines the Y position (in pixels).  
Example: PLC_PRG.iPos_Y.  
Increasing this value in runtime mode moves the element downwards. |

**“Rotation”**

Variable (numeric data type). Defines the angle of rotation (in degrees).  
Example: PLC_PRG.iAngle1.

The midpoint of the element rotates at the “Center” point. This rotation point is shown as the symbol.

In runtime mode, the alignment of the element remains the same with respect to the coordinate system of the visualization. Increasing the value rotates the element to the right.

**“Interior rotation”**

Variable (numeric data type). Defines the angle of rotation (in degrees).  
Example: PLC_PRG.iAngle2.

In runtime mode, the element rotates about the point of rotation specified in “Center” according to the value of the variable. In addition, the alignment of the element rotates according to the coordinate system of the visualization. Increasing the value in the code rotates clockwise.

The rotation point is shown as the symbol.

Note: If a static angle of rotation is specified in the “Position ➔ Angle” property, then the static angle of rotation is added to the variable angle of rotation (offset) when the visualization is executed.

You can link the variables to a unit conversion.

The “X”, “Y”, “Rotation”, and “Interior rotation” properties are supported by the “Client Animation” functionality.
The variables control the element behavior dynamically.

| **“Invisible”** | Variable (BOOL). Toggles the visibility of the element. 
| **TRUE** | The element is not visible at runtime. |

The “Invisible” property is supported by the "Client Animation" functionality.

These properties are available only when you select the “Preview: Support client animations and overlay of native elements” option in the Visualization Manager.

You can animate a movement, a rotation, and the visibility of a visualization element.

| **“Animation duration”** | Time that the element executes an animation (in milliseconds) 
| **Example: 500** |

| **“Move to the foreground”** | Property value (BOOL) 
| **TRUE** | At runtime, the element is displayed in the foreground. 
| **FALSE** | At runtime, the element is displayed in the layer where it was inserted in the visualization editor. |

| **“Image”** | Drop-down list with background colors 
| Depends on the visualization style |

| **“Access rights”** | Opens the “Access rights” dialog. There you can edit the access privileges for the element. 
| Status messages: 
| **“Not set. Full rights.”**: Access rights for all user groups: “operable” 
| **“Rights are set: Limited rights”**: Access is restricted for at least one group. |
Visualization element 'Dip switch', 'Power switch', 'Push switch', 'Push switch LED', 'Rocker switch'

Symbols:

Tag: “Lamps/Switches/Bitmaps”

The element assigns a value to a Boolean variable. The switch position "on" the value TRUE to the variable, and the switch position "off" assigns the value FALSE. Use the mouse to change the switch position.

Element properties

<table>
<thead>
<tr>
<th>&quot;Element name&quot;</th>
<th>Example: Operating_Switch</th>
</tr>
</thead>
<tbody>
<tr>
<td>Optional</td>
<td></td>
</tr>
<tr>
<td>Hint: Assign individual names for elements so that they are found faster in the element list.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>&quot;Type of element&quot;</th>
<th>Depending on the element: “Dip switch”, “Power switch”, “Push switch”, “Push switch LED”, or “Rocker switch”</th>
</tr>
</thead>
</table>

Element property 'Position'

The position defines the location and size of the element in the visualization window. These are based on the Cartesian coordinate system. The origin is located at the upper left corner of the window. The positive horizontal x-axis runs to the right. The positive vertical y-axis runs downwards.

| "X" | X coordinate of the upper left corner of the element  
Specified in pixels.  
Example: 10. |
|-----|---------------------------------------------------|
| "Y" | Y coordinate of the upper left corner of the element  
Specified in pixels.  
Example: 10. |
| "Width" | Specified in pixels.  
Example: 150 |
| "Height" | Specified in pixels.  
Example: 30 |
You can also change the values by dragging the box symbols (_drag_•) to other positions in the editor.

See also
- § Chapter 1.3.5.3.2 “Positioning elements, adapting size and layer” on page 1113

<table>
<thead>
<tr>
<th>“Variable”</th>
<th>Variable (BOOL)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>The value of the variables TRUE and FALSE indicates the switch position on/off.</td>
</tr>
</tbody>
</table>

### Image settings

| “Transparency” | [☐]: The “Transparent color” property is selected. |
| “Transparent color” | Pixels in this color are displayed as transparent. |
| Requirement: “Transparency” is activated. |
| ● [☐]: The “Color” dialog box opens. |
| ● [☐]: A drop-down list with style colors opens. |

| “Scaling type” | Reaction of the element when the dimension of the “Frame” element is changed: |
| “Isotropic”: The height and width of the image are resized proportionally to the “Frame”. |
| Please note: To retain the alignment of elements also within a scaled “Frame” element, define the “Horizontal alignment” or “Vertical alignment” explicitly with “Centered”. |
| “Anisotropic”: The image fills the entire “Frame” regardless of its proportions. |

| “Horizontal alignment” | Horizontal alignment of the image within the element frame or element |
| ● Left |
| ● Centered |
| ● Right |
| Requirement: “Scaling type” is “Isotropic”. |

| “Vertical alignment” | Vertical alignment of the image within the element frame or element |
| ● Top |
| ● Centered |
| ● Bottom |
| Requirement: “Scaling type” is “Isotropic”. |

| “Element behavior” | ● “Image toggler”: Every mouse click changes the switch and the “Variable” value. |
| ● “Image tapper”: The switch is "on" and the “Variable” value is TRUE while the mouse button is pressed. |

| “Tap FALSE” | [☑]: The value TRUE is assigned to the “Variable” property instead of the value FALSE, and back. |
| Requirement: “Image tapper” is selected in the “Element behavior” property. |

**Element property ‘Center’** The properties contain fixed values for the coordinates of the point of rotation. This point of rotation is shown as the ⬇️ symbol. The point is used as the center for rotating and scaling.
Element property 'Texts'

| "Tooltip" | String display as tooltip for the element  
Example: Valid access.

Element property 'Absolute movement'

The properties contain IEC variables for controlling the position of the element dynamically. The reference point is the upper left corner of the element. In runtime mode, the entire element is moved.

| "Movement"        | Variable (numeric data type). Defines the X position (in pixels).  
Example: PLC_PRG.iPos_X.  
Increasing this value in runtime mode moves the element to the right.

| "X"                | Variable (numeric data type). Defines the Y position (in pixels).  
Example: PLC_PRG.iPos_Y.  
Increasing this value in runtime mode moves the element downwards.

| "Rotation"        | Variable (numeric data type). Defines the angle of rotation (in degrees).  
Example: PLC_PRG.iAngle1.  
The midpoint of the element rotates at the "Center" point. This rotation point is shown as the symbol.  
In runtime mode, the alignment of the element remains the same with respect to the coordinate system of the visualization. Increasing the value rotates the element to the right.  

You can also change the values by dragging the symbols (♀) to other positions in the editor.
**“Interior rotation”**

Variable (numeric data type). Defines the angle of rotation (in degrees).

**Example:** PLC_PRG.iAngle2.

In runtime mode, the element rotates about the point of rotation specified in “Center” according to the value of the variable. In addition, the alignment of the element rotates according to the coordinate system of the visualization. Increasing the value in the code rotates clockwise.

The rotation point is shown as the symbol.

Note: If a static angle of rotation is specified in the “Position ➔ Angle” property, then the static angle of rotation is added to the variable angle of rotation (offset) when the visualization is executed.

---

You can link the variables to a unit conversion.

The “X”, “Y”, “Rotation”, and “Interior rotation” properties are supported by the “Client Animation” functionality.

See also
- Unit conversion

Element property 'State variables'

The variables control the element behavior dynamically.

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>“Invisible”</strong></td>
<td>Variable (BOOL). Toggles the visibility of the element. <strong>TRUE:</strong> The element is not visible at runtime.</td>
</tr>
<tr>
<td><strong>“Deactivate inputs”</strong></td>
<td>Variable (BOOL). Toggles the operability of the element. <strong>TRUE:</strong> User inputs do not have any effect in runtime more. The element is shown as deactivated.</td>
</tr>
</tbody>
</table>

The “Invisible” property is supported by the "Client Animation" functionality.

See also
- 

These properties are available only when you select the “Preview: Support client animations and overlay of native elements” option in the Visualization Manager.

You can animate a movement, a rotation, and the visibility of a visualization element.
| “Animation duration” | Time that the element executes an animation (in milliseconds)  
Example: 500 |
|-----------------------|---------------------------------------------------------------|
| “Move to the foreground” | Property value (BOOL)  
**TRUE**: At runtime, the element is displayed in the foreground.  
**FALSE**: At runtime, the element is displayed in the layer where it was inserted in the visualization editor. |

See also
- Chapter 1.3.5.21.5 “Using client animation” on page 1962

### Element property 'Background'

| “Image” | Drop-down list with background colors  
Depends on the visualization style |

### Element property 'Access rights'

Requirement: User management is set up for the visualization.

| “Access rights” | Opens the “Access rights” dialog. There you can edit the access privileges for the element.  
Status messages:  
- “Not set. Full rights.”: Access rights for all user groups: “operable”  
- “Rights are set: Limited rights”: Access is restricted for at least one group. |

See also
- Chapter 1.3.5.19.3.1 “Dialog ‘Access Rights’” on page 1583

See also
- 

### Visualization element ‘Rotary switch’

Symbol:

Tag: “Lamps/Switches/Bitmaps”

The element assigns a value to a Boolean variable. The switch position "on" the value **TRUE** to the variable, and the switch position "off" assigns the value **FALSE**. Use the mouse to change the switch position.

### Element properties
<table>
<thead>
<tr>
<th>“Element name”</th>
<th>Example: Operating_Switch</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Optional</td>
</tr>
<tr>
<td></td>
<td>Hint: Assign individual names for elements so that they are found faster in the element list.</td>
</tr>
</tbody>
</table>

| “Type of element”       | “Rotary switch”           |

**Element property 'Position'**

The position defines the location and size of the element in the visualization window. These are based on the Cartesian coordinate system. The origin is located at the upper left corner of the window. The positive horizontal x-axis runs to the right. The positive vertical y-axis runs downwards.

<table>
<thead>
<tr>
<th>“X”</th>
<th>X coordinate of the upper left corner of the element</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Specified in pixels.</td>
</tr>
<tr>
<td></td>
<td>Example: 10.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>“Y”</th>
<th>Y coordinate of the upper left corner of the element</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Specified in pixels.</td>
</tr>
<tr>
<td></td>
<td>Example: 10.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>“Width”</th>
<th>Specified in pixels.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Example: 150</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>“Height”</th>
<th>Specified in pixels.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Example: 30</td>
</tr>
</tbody>
</table>

You can also change the values by dragging the box symbols (_drag_ _) to other positions in the editor.

See also

- ☝ Chapter 1.3.5.3.2 “Positioning elements, adapting size and layer” on page 1113

<table>
<thead>
<tr>
<th>“Variable”</th>
<th>Variable (BOOL).</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>The value of the variables TRUE and FALSE indicates the switch position on/off.</td>
</tr>
</tbody>
</table>

**Image settings**

<table>
<thead>
<tr>
<th>“Transparency”</th>
<th>☑: The “Transparent color” property is selected.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>“Transparent color”</th>
<th>Pixels in this color are displayed as transparent.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Requirement: “Transparency” is activated.</td>
</tr>
<tr>
<td></td>
<td>☝ The “Color” dialog box opens.</td>
</tr>
<tr>
<td></td>
<td>☰: A drop-down list with style colors opens.</td>
</tr>
</tbody>
</table>
“Scaling type”

Reaction of the element when the dimension of the “Frame” element is changed:

- **“Isotropic”**: The height and width of the image are resized proportionally to the “Frame”.

  Please note: To retain the alignment of elements also within a scaled “Frame” element, define the “Horizontal alignment” or “Vertical alignment” explicitly with “Centered”.

- **“Anisotropic”**: The image fills the entire “Frame” regardless of its proportions.

“Horizontal alignment”

Horizontal alignment of the image within the element frame or element

- Left
- Centered
- Right

Requirement: “Scaling type” is “Isotropic”.

“Vertical alignment”

Vertical alignment of the image within the element frame or element

- Top
- Centered
- Bottom

Requirement: “Scaling type” is “Isotropic”.

“Element behavior”

- **“Image toggler”**: Every mouse click changes the switch and the “Variable” value.

- **“Image tapper”**: The switch is “on” and the “Variable” value is TRUE while the mouse button is pressed.

“Orientation”

- **“At top”**: The rotary switch turns from the top right to the top left.

- **“At side”**: The rotary switch turns from the top right to the bottom right.

“Color change”

☑: The element changes in color when “Variable” is TRUE.

Element property ‘Center’

The properties contain fixed values for the coordinates of the point of rotation. This point of rotation is shown as the symbol. The point is used as the center for rotating and scaling.

“X”

X-coordinate of the point of rotation

“Y”

Y-coordinate of the point of rotation

You can also change the values by dragging the symbols to other positions in the editor.

Element property ‘Texts’

“Tooltip”

String display as tooltip for the element

Example: Valid access.

Element property ‘Absolute movement’

The properties contain IEC variables for controlling the position of the element dynamically. The reference point is the upper left corner of the element. In runtime mode, the entire element is moved.
### Movement

<table>
<thead>
<tr>
<th><strong>“X”</strong></th>
<th>Variable (numeric data type). Defines the X position (in pixels).</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Example:</strong> PLC_PRG.iPos_X.</td>
<td>Increasing this value in runtime mode moves the element to the right.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>“Y”</strong></th>
<th>Variable (numeric data type). Defines the Y position (in pixels).</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Example:</strong> PLC_PRG.iPos_Y.</td>
<td>Increasing this value in runtime mode moves the element downwards.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>“Rotation”</strong></th>
<th>Variable (numeric data type). Defines the angle of rotation (in degrees).</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Example:</strong> PLC_PRG.iAngle1.</td>
<td>The midpoint of the element rotates at the “Center” point. This rotation point is shown as the symbol.</td>
</tr>
<tr>
<td></td>
<td>In runtime mode, the alignment of the element remains the same with respect to the coordinate system of the visualization. Increasing the value rotates the element to the right.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>“Interior rotation”</strong></th>
<th>Variable (numeric data type). Defines the angle of rotation (in degrees).</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Example:</strong> PLC_PRG.iAngle2.</td>
<td>In runtime mode, the element rotates about the point of rotation specified in “Center” according to the value of the variable. In addition, the alignment of the element rotates according to the coordinate system of the visualization. Increasing the value in the code rotates clockwise.</td>
</tr>
<tr>
<td></td>
<td>The rotation point is shown as the symbol.</td>
</tr>
<tr>
<td></td>
<td>Note: If a static angle of rotation is specified in the “Position ➔ Angle” property, then the static angle of rotation is added to the variable angle of rotation (offset) when the visualization is executed.</td>
</tr>
</tbody>
</table>

You can link the variables to a unit conversion.

The “X”, “Y”, “Rotation”, and “Interior rotation” properties are supported by the “Client Animation” functionality.

See also
- Unit conversion
The variables control the element behavior dynamically.

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Invisible&quot;</td>
<td>Variable (BOOL). Toggles the visibility of the element. TRUE: The element is not visible at runtime.</td>
</tr>
<tr>
<td>&quot;Deactivate inputs&quot;</td>
<td>Variable (BOOL). Toggles the operability of the element. TRUE: User inputs do not have any effect in runtime more. The element is shown as deactivated.</td>
</tr>
</tbody>
</table>

The "Invisible" property is supported by the "Client Animation" functionality.

See also
-  

These properties are available only when you select the "Preview: Support client animations and overlay of native elements" option in the Visualization Manager.

You can animate a movement, a rotation, and the visibility of a visualization element.

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Animation duration&quot;</td>
<td>Time that the element executes an animation (in milliseconds) Example: 500</td>
</tr>
<tr>
<td>&quot;Move to the foreground&quot;</td>
<td>Property value (BOOL) TRUE: At runtime, the element is displayed in the foreground. FALSE: At runtime, the element is displayed in the layer where it was inserted in the visualization editor.</td>
</tr>
</tbody>
</table>

See also
-  |

Element property 'Background'

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Image&quot;</td>
<td>Drop-down list with background colors Depends on the visualization style</td>
</tr>
</tbody>
</table>

Element property 'Access rights'

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Access rights&quot;</td>
<td>Opens the &quot;Access rights&quot; dialog. There you can edit the access privileges for the element. Status messages: - &quot;Not set. Full rights.&quot;: Access rights for all user groups: &quot;operable&quot; - &quot;Rights are set: Limited rights&quot;: Access is restricted for at least one group.</td>
</tr>
</tbody>
</table>

See also
-  |

Requirement: User management is set up for the visualization.
Visualization element ‘Trace’

Symbol:

Tag: “Special controls”

The element displays the graphical curve of variable values. In addition, variables can be configured for controlling the view.

See also
- Chapter 1.3.5.10 “Displaying data curve with trace” on page 1162
- Chapter 1.3.5.19.2.13 “Command ‘Configure Trace’” on page 1572

<table>
<thead>
<tr>
<th>“Element name”</th>
<th>Example: Velocity</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Datasource”</td>
<td>Location where the trace data is buffered.</td>
</tr>
<tr>
<td></td>
<td>- “&lt;local application&gt;”: The trace record is listed below the local application. The visualization that contains the trace is located below this application. When the application is downloaded, the trace configuration is downloaded to the local device. During execution, the data is stored locally in the trace buffer.</td>
</tr>
<tr>
<td></td>
<td>- “&lt;data source name&gt;”: Data source that identifies the remote device where the trace record is created. When the local application is downloaded with the visualization, the trace configuration is downloaded to the remote device. During execution, the trace buffer is filled, and the trace data is transferred and then displayed in the local visualization as HMI. Example: DataSource_PLC_A</td>
</tr>
<tr>
<td></td>
<td>Note: The trace buffer is filled only if the remote application is being executed. The data recording is started when the local visualization is started.</td>
</tr>
<tr>
<td>“Application”</td>
<td>Application where data was recorded.</td>
</tr>
<tr>
<td></td>
<td>▼: Lists all applications that are present below the data source.</td>
</tr>
<tr>
<td></td>
<td>Requirement: A remote data source (not “&lt;local application&gt;”) is referenced in the “Datasource” property.</td>
</tr>
<tr>
<td>“Type of element”</td>
<td>“Trace”</td>
</tr>
<tr>
<td>“Trace”</td>
<td>@“&lt;name of trace configuration&gt;”: Opens the “Trace Configuration” dialog where you can modify the trace configuration.</td>
</tr>
</tbody>
</table>

See also
- Chapter 1.3.5.19.2.13 “Command ‘Configure Trace’” on page 1572
- Data Source Manager
The position defines the location and size of the element in the visualization window. This is based on the Cartesian coordinate system. The origin is located at the upper left corner of the window. The positive horizontal x-axis runs to the right. The positive vertical y-axis runs downwards.

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>“X”</td>
<td>The x-coordinate of the upper left corner of the element</td>
</tr>
<tr>
<td></td>
<td>Specified in pixels</td>
</tr>
<tr>
<td></td>
<td><strong>Example:</strong> 10</td>
</tr>
<tr>
<td>“Y”</td>
<td>The y-coordinate of the upper left corner of the element</td>
</tr>
<tr>
<td></td>
<td>Specified in pixels</td>
</tr>
<tr>
<td></td>
<td><strong>Example:</strong> 10</td>
</tr>
<tr>
<td>“Width”</td>
<td>Specified in pixels</td>
</tr>
<tr>
<td></td>
<td><strong>Example:</strong> 150</td>
</tr>
<tr>
<td>“Height”</td>
<td>Specified in pixels</td>
</tr>
<tr>
<td></td>
<td><strong>Example:</strong> 30</td>
</tr>
<tr>
<td>Tip:</td>
<td>You can change the values in “X”, “Y”, “Width”, and “Height” by dragging</td>
</tr>
<tr>
<td></td>
<td>the corresponding symbols to another position in the editor.</td>
</tr>
<tr>
<td>“Angle”</td>
<td>Static angle of rotation (in degrees)</td>
</tr>
<tr>
<td></td>
<td><strong>Example:</strong> 35</td>
</tr>
<tr>
<td></td>
<td>The element is displayed rotated in the editor. The point of rotation is the</td>
</tr>
<tr>
<td></td>
<td>center of the element. A positive value rotates clockwise.</td>
</tr>
<tr>
<td>Tip:</td>
<td>You can change the value in the editor by focusing the element to the</td>
</tr>
<tr>
<td></td>
<td>handle. When the cursor is displayed as a rotating arrow, you can rotate the</td>
</tr>
<tr>
<td></td>
<td>element about its center as a handle.</td>
</tr>
</tbody>
</table>

![Diagram](image)

(1): Handle

Note: If a dynamic angle of rotation is also configured in the property “Absolute movement ➔ Internal rotation”, then the static and dynamic angles of rotation are added in runtime mode. The static angle of rotation acts as an offset.

See also
- ☰ Chapter 1.3.5.3.2 “Positioning elements, adapting size and layer” on page 1113

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Show cursor”</td>
<td>☐: A cursor (vertical line) is displayed at the mouse position. The trigger</td>
</tr>
<tr>
<td></td>
<td>and variable values where the cursor points are displayed as a tooltip.</td>
</tr>
<tr>
<td>“Overwrite existing trace on PLC”</td>
<td>☑: If a trace with the same name is on the PLC, then it is overwritten at down-</td>
</tr>
<tr>
<td></td>
<td>load with the configuration that is defined here.</td>
</tr>
<tr>
<td>“Number format”</td>
<td>Number format of values in the tooltip in printf syntax (example: %d, %5.2f).</td>
</tr>
</tbody>
</table>

The control variables are assigned automatically when you click “Insert elements for controlling trace”.

2020/12/10
| **“Reset Trigger”** | Variable (BOOL). | Standard control variable: bResetTrigger | TRUE: Resets the triggering. After the action is executed, the variable is set automatically to FALSE. |
| **“Start Trace”** | Variable (BOOL). | Standard control variable: bStart | TRUE: Starts the trace. After the action is executed, the variable is set automatically to FALSE. |
| **“Stop Trace”** | Variable (BOOL). | Standard control variable: bStop | TRUE: Stops the trace. After the action is executed, the variable is set automatically to FALSE. |
| **“Save trace to a file”** | Variable (BOOL). | Standard control variable: bStore | TRUE: Saves the current trace configuration and the data that is stored in the development system to a file. When the action is ended, the variable is set automatically to FALSE. |
| **“File name”** | Variable (STRING) that contains the file name of the file to be saved. | Standard control variable: sStoreFilename |
| **“Load trace from file”** | Variable (BOOL). | Standard control variable: bRestore | TRUE: Reads the file specified below and loads its contents into the trace editor. The file contains a trace configuration and possibly also trace data. To do this, the stored trace configuration must match the application where the trace configuration is located. When the action is ended, the variable is set automatically to FALSE. Note: A trace configuration can be loaded from a file only under special circumstances. The file must have been created with exactly the same (running) application with which it will then be loaded. The consequence of changing the running application (for example by downloading again) is that a file which was previously created from the application cannot no longer be read into the application. Even external manual changes to the file can cause this. You should edit only those configuration settings that have an effect on displaying the variables. If you change variable definitions directly in the file (for example by replacing variable x with v y), then the file cannot be loaded. |
| **“File name”** | Variable (STRING) that contains the file name of the file to be read. | Standard control variable: sRestoreFilename |

See also
- § Chapter 1.3.5.19.2.15 “Command ‘Insert Elements for Controlling Trace’” on page 1575

**Element property 'Center'**
The properties contain fixed values for the coordinates of the point of rotation. This point of rotation is shown as the symbol. The point is used as the center for rotating and scaling.
**Element property 'Absolute movement'**

The properties contain IEC variables for controlling the position of the element dynamically. The reference point is the upper left corner of the element. In runtime mode, the entire element is moved.

<table>
<thead>
<tr>
<th><strong>Movement</strong></th>
</tr>
</thead>
</table>
| **“X”** | Variable (numeric data type). Defines the X position (in pixels).  
Example: PLC_PRG.iPos_X.  
Increasing this value in runtime mode moves the element to the right. |
| **“Y”** | Variable (numeric data type). Defines the Y position (in pixels).  
Example: PLC_PRG.iPos_Y.  
Increasing this value in runtime mode moves the element downwards. |

| **Rotation”** | Variable (numeric data type). Defines the angle of rotation (in degrees).  
Example: PLC_PRG.iAngle1.  
The midpoint of the element rotates at the “Center” point. This rotation point is shown as the symbol.  
In runtime mode, the alignment of the element remains the same with respect to the coordinate system of the visualization. Increasing the value rotates the element to the right. |

| **“Interior rotation”** | Variable (numeric data type). Defines the angle of rotation (in degrees).  
Example: PLC_PRG.iAngle2.  
In runtime mode, the element rotates about the point of rotation specified in “Center” according to the value of the variable. In addition, the alignment of the element rotates according to the coordinate system of the visualization. Increasing the value in the code rotates clockwise.  
The rotation point is shown as the symbol.  
Note: If a static angle of rotation is specified in the “Position ➔ Angle” property, then the static angle of rotation is added to the variable angle of rotation (offset) when the visualization is executed. |
You can link the variables to a unit conversion.

The “X”, “Y”, “Rotation”, and “Interior rotation” properties are supported by the “Client Animation” functionality.

See also

- Unit conversion

**Element property 'State variables'**

<table>
<thead>
<tr>
<th>“Invisible”</th>
<th>Variable (BOOL). Toggles the visibility of the element.</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRUE: The element is not visible at runtime.</td>
<td></td>
</tr>
</tbody>
</table>

The “Invisible” property is supported by the "Client Animation" functionality.

See also

- 

These properties are available only when you select the “Preview: Support client animations and overlay of native elements” option in the Visualization Manager.

You can animate a movement, a rotation, and the visibility of a visualization element.

<table>
<thead>
<tr>
<th>“Animation duration”</th>
<th>Time that the element executes an animation (in milliseconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example: 500</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>“Move to the foreground”</th>
<th>Property value (BOOL)</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRUE: At runtime, the element is displayed in the foreground.</td>
<td></td>
</tr>
<tr>
<td>FALSE: At runtime, the element is displayed in the layer where it was inserted in the visualization editor.</td>
<td></td>
</tr>
</tbody>
</table>

See also

- § Chapter 1.3.5.21.5 “Using client animation” on page 1962

**Element property 'Access rights'**

Requirement: User management is set up for the visualization.
“Access rights”
Opens the “Access rights” dialog. There you can edit the access privileges for the element.
Status messages:
- “Not set. Full rights.”: Access rights for all user groups: “operable”
- “Rights are set: Limited rights”: Access is restricted for at least one group.

See also
- Chapter 1.3.5.19.3.1 “Dialog ‘Access Rights’” on page 1583

Visualization element 'Trend'
Symbol:

Tag: “Special controls”
The element displays the graphical curve of variable values over a longer period of time. It provides additional controls for the view.

Element properties

<table>
<thead>
<tr>
<th>“Element name”</th>
<th>Example: Velocity</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Data source”</td>
<td>Selection of the device and the application where the data to be visualized and the trend recording are generated</td>
</tr>
</tbody>
</table>
  - Remote data source which accesses a remote device, accesses a remote application, and then transfers the data from there
    Example: DataSource_A
    Below the (now visible) “Application” property, the remote application is displayed as configured in the data source.
    Example: App_A
    Note: If the data source is accessed symbolically by means of a symbol file (CODESYS symbolic), then the required symbol file and the corresponding project have to be saved in the same folder.
  - Local application below which the alarm configuration is located
    Example: `<local application>`
    Example: Device_A.App_A
| “Type of element” | “Trend” |
| “Trend recording” | 📡: Trend recording which is displayed in the visualization and which is located on the device as specified in “Data source” |

See also
- Chapter 1.3.5.19.2.12 “Command ‘Configure Display Settings of the Trend’” on page 1570
- Object ’Data source’

Element property 'Position'
The position defines the location and size of the element in the visualization window. This is based on the Cartesian coordinate system. The origin is located at the upper left corner of the window. The positive horizontal x-axis runs to the right. The positive vertical y-axis runs downwards.
The x-coordinate of the upper left corner of the element

<table>
<thead>
<tr>
<th>X</th>
<th>The x-coordinate of the upper left corner of the element</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Specified in pixels</td>
</tr>
<tr>
<td></td>
<td>Example: 10</td>
</tr>
</tbody>
</table>

The y-coordinate of the upper left corner of the element

<table>
<thead>
<tr>
<th>Y</th>
<th>The y-coordinate of the upper left corner of the element</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Specified in pixels</td>
</tr>
<tr>
<td></td>
<td>Example: 10</td>
</tr>
</tbody>
</table>

Width

<table>
<thead>
<tr>
<th>Width</th>
<th>Specified in pixels</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Example: 150</td>
</tr>
</tbody>
</table>

Height

<table>
<thead>
<tr>
<th>Height</th>
<th>Specified in pixels</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Example: 30</td>
</tr>
</tbody>
</table>

Tip: You can change the values in “X”, “Y”, “Width”, and “Height” by dragging the corresponding symbols to another position in the editor.

Static angle of rotation (in degrees)

<table>
<thead>
<tr>
<th>Angle</th>
<th>Static angle of rotation (in degrees)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Example: 35</td>
</tr>
</tbody>
</table>

The element is displayed rotated in the editor. The point of rotation is the center of the element. A positive value rotates clockwise.

Tip: You can change the value in the editor by focusing the element to the handle. When the cursor is displayed as a rotating arrow, you can rotate the element about its center as a handle.

Note: If a dynamic angle of rotation is also configured in the property “Absolute movement ➔ Internal rotation”, then the static and dynamic angles of rotation are added in runtime mode. The static angle of rotation acts as an offset.

See also

- § Chapter 1.3.5.3.2 “Positioning elements, adapting size and layer” on page 1113

<table>
<thead>
<tr>
<th>Show cursor</th>
<th>A cursor (vertical line) is displayed at the mouse position.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Show tooltip</th>
<th>Requirement: “Show cursor” is activated. The information about the variable is displayed in a tooltip at the cursor position.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Show frame</th>
<th>The visualization element is drawn with a frame.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Number format</th>
<th>Number format of values in the tooltip and in the legend in printf syntax (example: %d, %5.2f).</th>
</tr>
</thead>
</table>

Element property 'Tick mark labels'
### “Time stamp”

The values of the time stamps are displayed in absolute time. Example: 18.03.2016, 14:05:30:000

The display formatting can be adjusted in the property “Internationalization (format strings)”.

### “Two-line labelling”

- ☑: The time stamps are displayed in two lines. The date is displayed in the first line and the time is displayed in the second line.
- ☐: Time stamp is displayed in one line. The date and time can also be displayed in one line depending on the formatting.

### “Omit irrelevant information in time stamp”

- ☑: The time stamp has a shorter form. For example, the date is displayed only for the first tick mark, and only the time for the following tick marks. The settings in “Internationalization (format strings)” are ignored for this setting.
- ☐: All information is displayed for each time stamp.

### “Internationalization (format strings)”

Only active when the parameter “Omit irrelevant information in timestamps” is deactivated.

### “Date”

Definition of the date format. The default setting is taken from the Windows control panel.

### “Time”

Definition of the time format. The default setting is taken from the Windows control panel.

---

**Element property ‘Assigned control elements’**

These elements are created automatically when the control elements are added with the command “Insert elements for controlling trend”.

### “Date range picker”

Control element for changing the date and time of the displayed data sets. With all elements are provided that have implemented the interface IDateRangeSelector. By default, instances of the “Date range picker” visualization element are available.

### “Time picker”

Control element for changing the time of the displayed data sets. With all elements are provided that have implemented the interface ITimeSelector. By default, instances of the “Time range picker” visualization element are available.

### “Legend”

Control element for displaying a legend for the graphs. With all elements are provided that have implemented the interface ILegendDisplayer.

---

**See also**

- § Chapter 1.3.5.19.2.18 “Command ‘Insert Elements for Controlling the Trend’” on page 1577

---

**Element property ‘Center’**

The properties contain fixed values for the coordinates of the point of rotation. This point of rotation is shown as the symbol. The point is used as the center for rotating and scaling.

### “X”

X-coordinate of the point of rotation

### “Y”

Y-coordinate of the point of rotation

---

You can also change the values by dragging the symbols ( 중심) to other positions in the editor.
Element property 'Absolute movement'
The properties contain IEC variables for controlling the position of the element dynamically. The reference point is the upper left corner of the element. In runtime mode, the entire element is moved.

### “Movement”

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>“X”</td>
<td>Variable (numeric data type). Defines the X position (in pixels). Increasing this value in runtime mode moves the element to the right.</td>
<td>PLC_PRG.iPos_X.</td>
</tr>
<tr>
<td>“Y”</td>
<td>Variable (numeric data type). Defines the Y position (in pixels). Increasing this value in runtime mode moves the element downwards.</td>
<td>PLC_PRG.iPos_Y.</td>
</tr>
</tbody>
</table>

### “Rotation”

- The midpoint of the element rotates at the “Center” point. This rotation point is shown as the symbol.
- In runtime mode, the alignment of the element remains the same with respect to the coordinate system of the visualization. Increasing the value rotates the element to the right.

### “Interior rotation”

- In runtime mode, the element rotates about the point of rotation specified in “Center” according to the value of the variable. In addition, the alignment of the element rotates according to the coordinate system of the visualization. Increasing the value in the code rotates clockwise.
- The rotation point is shown as the symbol.
- Note: If a static angle of rotation is specified in the “Position ➔ Angle” property, then the static angle of rotation is added to the variable angle of rotation (offset) when the visualization is executed.

---

**You can link the variables to a unit conversion.**

**The “X”, “Y”, “Rotation”, and “Interior rotation” properties are supported by the “Client Animation” functionality.**
The variables control the element behavior dynamically.

**Element property 'State variables'**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Invisible&quot;</td>
<td>Variable (BOOL). Toggles the visibility of the element. TRUE: The element is not visible at runtime.</td>
</tr>
</tbody>
</table>

The "Invisible" property is supported by the "Client Animation" functionality.

See also

- **Unit conversion**

These properties are available only when you select the "Preview: Support client animations and overlay of native elements" option in the Visualization Manager.

You can animate a movement, a rotation, and the visibility of a visualization element.

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Animation duration&quot;</td>
<td>Time that the element executes an animation (in milliseconds) Example: 500</td>
</tr>
<tr>
<td>&quot;Move to the foreground&quot;</td>
<td>Property value (BOOL) TRUE: At runtime, the element is displayed in the foreground. FALSE: At runtime, the element is displayed in the layer where it was inserted in the visualization editor.</td>
</tr>
</tbody>
</table>

See also

- ☞ Chapter 1.3.5.21.5 “Using client animation” on page 1962

**Element property 'Access rights'**

Requirement: User management is set up for the visualization.

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Access rights&quot;</td>
<td>Opens the “Access rights” dialog. There you can edit the access privileges for the element. Status messages: • &quot;Not set. Full rights.&quot;: Access rights for all user groups: &quot;operable&quot; • &quot;Rights are set: Limited rights&quot;: Access is restricted for at least one group.</td>
</tr>
</tbody>
</table>

See also

- ☞ Chapter 1.3.5.19.3.1 “Dialog ‘Access Rights’” on page 1583

**Visualization element 'Legend'**

Symbol:
Tag: “Special controls”

The element serves as a legend for another element, for example a trend. The legend is assigned in the properties of the other element.

See also

Element properties

<table>
<thead>
<tr>
<th>“Element name”</th>
<th>Example: LegendOfTrendA</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Optional</td>
</tr>
<tr>
<td></td>
<td>Hint: Assign individual names for elements so that they are found faster in the element list.</td>
</tr>
</tbody>
</table>

| “Type of element” | “Legend” |

<table>
<thead>
<tr>
<th>Element property ‘Position’</th>
<th>The position defines the location and size of the element in the visualization window. These are based on the Cartesian coordinate system. The origin is located at the upper left corner of the window. The positive horizontal x-axis runs to the right. The positive vertical y-axis runs downwards.</th>
</tr>
</thead>
<tbody>
<tr>
<td>“X”</td>
<td>X coordinate of the upper left corner of the element</td>
</tr>
<tr>
<td></td>
<td>Specified in pixels.</td>
</tr>
<tr>
<td></td>
<td>Example: 10.</td>
</tr>
<tr>
<td>“Y”</td>
<td>Y coordinate of the upper left corner of the element</td>
</tr>
<tr>
<td></td>
<td>Specified in pixels.</td>
</tr>
<tr>
<td></td>
<td>Example: 10.</td>
</tr>
<tr>
<td>“Width”</td>
<td>Specified in pixels.</td>
</tr>
<tr>
<td></td>
<td>Example: 150</td>
</tr>
<tr>
<td>“Height”</td>
<td>Specified in pixels.</td>
</tr>
<tr>
<td></td>
<td>Example: 30</td>
</tr>
</tbody>
</table>

You can also change the values by dragging the box symbols (_drag_1) to other positions in the editor.

See also

- Chapter 1.3.5.3.2 “Positioning elements, adapting size and layer” on page 1113

Element property ‘Center’

The properties contain fixed values for the coordinates of the point of rotation. This point of rotation is shown as the _drag_2 symbol. The point is used as the center for rotating and scaling.

| “X” | X-coordinate of the point of rotation |
| “Y” | Y-coordinate of the point of rotation |
You can also change the values by dragging the symbols ((nullable) to other positions in the editor.

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Orientation”</td>
<td>Orientation of the element. The value is configured in the assigned element.</td>
</tr>
<tr>
<td></td>
<td>● “Horizontal”</td>
</tr>
<tr>
<td></td>
<td>● “Vertical”</td>
</tr>
<tr>
<td>“Attached element instance”</td>
<td>Example: Element_A</td>
</tr>
<tr>
<td>“Show frame”</td>
<td>![Frame Icon]: The element is shown in the class diagram.</td>
</tr>
<tr>
<td>“Number format”</td>
<td>The format of the value in printf syntax (example: %d, %5.2f)</td>
</tr>
</tbody>
</table>

### Element Property 'Layout'

Defines how many variables can be displayed at a maximum and is calculated from the row and column number.

<table>
<thead>
<tr>
<th>Property</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Max. number of rows”</td>
<td>3</td>
</tr>
<tr>
<td>“Max. number of columns”</td>
<td>2</td>
</tr>
</tbody>
</table>

### Element Property 'Text properties'

The property affects the text configured in the associated element.

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Text format”</td>
<td>“Default”: The text will be cut and displayed in only the part that fits into the visualization element.</td>
</tr>
<tr>
<td></td>
<td>“Linebreak”: The text will be wrapped in rows.</td>
</tr>
<tr>
<td></td>
<td>“Ellipsis”: The text is cut and ellipsis . . . are added to indicate that something is missing.</td>
</tr>
<tr>
<td>“Font”</td>
<td>Font of the text. The entries of the selection list are defined in the visualization style.</td>
</tr>
<tr>
<td>“Font color”</td>
<td>Text color, for example Grey. The entries of the selection list are defined in the visualization style.</td>
</tr>
<tr>
<td>“Transparency”</td>
<td>Transparency value (255 to 0), which defines the transparency of the corresponding color.</td>
</tr>
<tr>
<td></td>
<td>Example: 255: The color is opaque. 0: The color is fully transparent.</td>
</tr>
</tbody>
</table>

### Element property 'Absolute movement'

The properties contain IEC variables for controlling the position of the element dynamically. The reference point is the upper left corner of the element. In runtime mode, the entire element is moved.

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Movement”</td>
<td>“X” Variable (numeric data type). Defines the X position (in pixels).</td>
</tr>
<tr>
<td></td>
<td>Example: PLC_PRG.iPos.X.</td>
</tr>
<tr>
<td></td>
<td>Increasing this value in runtime mode moves the element to the right.</td>
</tr>
<tr>
<td>Variable</td>
<td>Description</td>
</tr>
<tr>
<td>------------------</td>
<td>----------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>“Y”</td>
<td>Variable (numeric data type). Defines the Y position (in pixels).</td>
</tr>
<tr>
<td>“Rotation”</td>
<td>Variable (numeric data type). Defines the angle of rotation (in degrees).</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>“Interior rotation”</td>
<td>Variable (numeric data type). Defines the angle of rotation (in degrees).</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

You can link the variables to a unit conversion.

The “X”, “Y”, “Rotation”, and “Interior rotation” properties are supported by the “Client Animation” functionality.

See also
- **Unit conversion**

These properties are available only when you select the “Preview: Support client animations and overlay of native elements” option in the Visualization Manager.

You can animate a movement, a rotation, and the visibility of a visualization element.
**“Animation duration”**

Time that the element executes an animation (in milliseconds)

Example: 500

**“Move to the foreground”**

Property value (**BOOL**)

TRUE: At runtime, the element is displayed in the foreground.
FALSE: At runtime, the element is displayed in the layer where it was inserted in the visualization editor.

See also

- § Chapter 1.3.5.21.5 “Using client animation” on page 1962

**Element property 'Access rights’**

Requirement: User management is set up for the visualization.

**“Access rights”**

Opens the “Access rights” dialog. There you can edit the access privileges for the element.

Status messages:

- “Not set. Full rights.”: Access rights for all user groups: “operable”
- “Rights are set: Limited rights”: Access is restricted for at least one group.

See also

- § Chapter 1.3.5.19.3.1 “Dialog ‘Access Rights’” on page 1583

**Visualization element 'ActiveX’**

Symbol: ![Visual Element]

Tag: “Special controls”

The element is used for linking an existing ActiveX element in the visualization. In the element properties of the ActiveX element, you can configure the method calls and their parameters.

*In the CODESYS Store, you will find a sample project for using ActiveX components in visualizations.*

See also

- CODESYS Store

**Element properties**

<table>
<thead>
<tr>
<th>“Element name”</th>
<th>Example: GenElemInst_1</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Type of element”</td>
<td>“ActiveX element”</td>
</tr>
<tr>
<td>“Element”</td>
<td>Installed ActiveX component that is linked to the visualization.</td>
</tr>
</tbody>
</table>

Hint: To avoid typing errors, select the required ActiveX component with the input assistant.
Element property 'Position' The position defines the location and size of the element in the visualization window. These are based on the Cartesian coordinate system. The origin is located at the upper left corner of the window. The positive horizontal x-axis runs to the right. The positive vertical y-axis runs downwards.

<table>
<thead>
<tr>
<th>“X”</th>
<th>X coordinate of the upper left corner of the element</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Specified in pixels.</td>
</tr>
<tr>
<td></td>
<td>Example: 10.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>“Y”</th>
<th>Y coordinate of the upper left corner of the element</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Specified in pixels.</td>
</tr>
<tr>
<td></td>
<td>Example: 10.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>“Width”</th>
<th>Specified in pixels.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Example: 150</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>“Height”</th>
<th>Specified in pixels.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Example: 30</td>
</tr>
</tbody>
</table>

You can also change the values by dragging the box symbols (_drag_box_) to other positions in the editor.

See also
- § Chapter 1.3.5.3.2 “Positioning elements, adapting size and layer” on page 1113

Element property 'Center' The properties contain fixed values for the coordinates of the point of rotation. This point of rotation is shown as therotate_symbol_. The point is used as the center for rotating and scaling.

<table>
<thead>
<tr>
<th>“X”</th>
<th>X-coordinate of the point of rotation</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>“Y”</th>
<th>Y-coordinate of the point of rotation</th>
</tr>
</thead>
</table>

You can also change the values by dragging the symbols (rotate_symbol_) to other positions in the editor.

Element property 'Absolute movement' The properties contain IEC variables for controlling the position of the element dynamically. The reference point is the upper left corner of the element. In runtime mode, the entire element is moved.

<table>
<thead>
<tr>
<th>“Movement”</th>
<th>Variable (numeric data type). Defines the X position (in pixels).</th>
</tr>
</thead>
<tbody>
<tr>
<td>“X”</td>
<td>Example: PLC_PRG.iPos_X.</td>
</tr>
<tr>
<td></td>
<td>Increasing this value in runtime mode moves the element to the right.</td>
</tr>
<tr>
<td>Variable</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>-------------</td>
</tr>
<tr>
<td>&quot;Y&quot;</td>
<td>Variable (numeric data type). Defines the Y position (in pixels).&lt;br&gt;Example: PLC_PRG.iPos_Y.&lt;br&gt;Increasing this value in runtime mode moves the element downwards.</td>
</tr>
<tr>
<td>&quot;Rotation&quot;</td>
<td>Variable (numeric data type). Defines the angle of rotation (in degrees).&lt;br&gt;Example: PLC_PRG.iAngle1.&lt;br&gt;The midpoint of the element rotates at the &quot;Center&quot; point. This rotation point is shown as the symbol. &lt;br&gt;In runtime mode, the alignment of the element remains the same with respect to the coordinate system of the visualization. Increasing the value rotates the element to the right.</td>
</tr>
<tr>
<td>&quot;Scaling&quot;</td>
<td>Variable (integer data type). Causes centric stretching.&lt;br&gt;Example: PLC_PRG.iScaling.&lt;br&gt;The reference point is the &quot;Center&quot; property.&lt;br&gt;The value 1 shrinks the element by a factor of 0.001. The value 1000 returns the element to its original size.</td>
</tr>
<tr>
<td>&quot;Interior rotation&quot;</td>
<td>Variable (numeric data type). Defines the angle of rotation (in degrees).&lt;br&gt;Example: PLC_PRG.iAngle2.&lt;br&gt;In runtime mode, the element rotates about the point of rotation specified in &quot;Center&quot; according to the value of the variable. In addition, the alignment of the element rotates according to the coordinate system of the visualization. Increasing the value in the code rotates clockwise. &lt;br&gt;The rotation point is shown as the symbol. &lt;br&gt;Note: If a static angle of rotation is specified in the property &quot;Position ➔ Angle&quot;, then the static angle of rotation is added to the variable angle of rotation (offset) when the visualization is executed.</td>
</tr>
</tbody>
</table>

You can link the variables to a unit conversion.

The properties "X", "Y", "Rotation", and "Interior rotation" are supported by the "Client Animation" functionality.

See also
- Unit conversion
The variables control the element behavior dynamically.

| Visible Variable | "Invisible" (BOOL). Toggles the visibility of the element. TRUE: The element is not visible at runtime. |

The "Invisible" property is supported by the "Client Animation" functionality.

See also

Element property 'Initial calls'

These method calls are executed during initialization. They are executed in the first cycle only.

| Method calls | Button "Create new" Creates a subnode below "Methods" with parameters for the method call. |

<table>
<thead>
<tr>
<th>Methods</th>
<th>&quot;[&lt;number&gt;]&quot;</th>
</tr>
</thead>
<tbody>
<tr>
<td>Method Call</td>
<td>Name of the method</td>
</tr>
<tr>
<td>Parameter</td>
<td>Parameter passed at the method call</td>
</tr>
<tr>
<td>Result parameter</td>
<td>Optional variable for the return value of the method</td>
</tr>
</tbody>
</table>

Element property 'Cyclic calls'

These method calls are executed in every cycle. They are executed in the refresh rate of the visualization.

| Method calls | Button "Create new" Creates a subnode below "Methods" for a method call and its parameters. |

<table>
<thead>
<tr>
<th>Methods</th>
<th>&quot;[&lt;number&gt;]&quot;</th>
</tr>
</thead>
<tbody>
<tr>
<td>Method Call</td>
<td>Name of the method</td>
</tr>
<tr>
<td>Parameter</td>
<td>Parameter passed at the method call</td>
</tr>
<tr>
<td>Result parameter</td>
<td>Optional variable for the return value of the method</td>
</tr>
</tbody>
</table>

Element property 'Conditional calls'

These method calls are executed in the refresh rate of the visualization. You define the call condition in the property "Methods [ <number> ] Call condition".

| Method calls | Button "Create new" Creates a subnode below "Methods" with a call condition and parameters for the method call. |

<table>
<thead>
<tr>
<th>Methods</th>
<th>&quot;[&lt;number&gt;]&quot;</th>
</tr>
</thead>
<tbody>
<tr>
<td>Method Call</td>
<td>Name of the method</td>
</tr>
<tr>
<td>Call condition</td>
<td>Variable (BOOL). A rising edge of this variable triggers the call of this method.</td>
</tr>
<tr>
<td>Parameter</td>
<td>Parameter passed at the method call</td>
</tr>
<tr>
<td>Result parameter</td>
<td>Optional variable for the return value of the method</td>
</tr>
</tbody>
</table>
These properties are available only when you select the “Preview: Support client animations and overlay of native elements” option in the Visualization Manager.
You can animate a movement, a rotation, and the visibility of a visualization element.

<table>
<thead>
<tr>
<th><strong>“Animation duration”</strong></th>
<th>Time that the element executes an animation (in milliseconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Example: 500</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>“Move to the foreground”</strong></th>
<th>Property value (BOOL)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>TRUE: At runtime, the element is displayed in the foreground.</td>
</tr>
<tr>
<td></td>
<td>FALSE: At runtime, the element is displayed in the layer where it was inserted in the visualization editor.</td>
</tr>
</tbody>
</table>

See also
- § Chapter 1.3.5.21.5 “Using client animation” on page 1962

**Element property ‘Access rights’**

Requirement: User management is set up for the visualization.

<table>
<thead>
<tr>
<th><strong>“Access rights”</strong></th>
<th>Opens the “Access rights” dialog. There you can edit the access privileges for the element.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Status messages:</td>
</tr>
<tr>
<td></td>
<td>• “Not set. Full rights.”: Access rights for all user groups: “operable”</td>
</tr>
<tr>
<td></td>
<td>• “Rights are set: Limited rights”: Access is restricted for at least one group.</td>
</tr>
</tbody>
</table>

See also
- § Chapter 1.3.5.19.3.1 “Dialog ‘Access Rights’” on page 1583

**Visualization element ‘Webbrowser’**

Symbol:

![Symbol](image)

Tag: “Special controls”
The element shows a website, PDF file, or video that has a URL.

**NOTICE!**
The display options of the “Webbrowser” element depend on the operating system and the display variant of the visualization.

Requirement: The software components of the web browser are available in the runtime system and configured accordingly (example: videos to be shown on Linux).

See also
- § Chapter 1.3.5.21.4 “Displaying web content” on page 1961

**Element properties**
The position defines the location and size of the element in the visualization window. These are based on the Cartesian coordinate system. The origin is located at the upper left corner of the window. The positive horizontal x-axis runs to the right. The positive vertical y-axis runs downwards.

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>X</strong></td>
<td>X coordinate of the upper left corner of the element Specified in pixels. Example: 10.</td>
</tr>
<tr>
<td><strong>Y</strong></td>
<td>Y coordinate of the upper left corner of the element Specified in pixels. Example: 10.</td>
</tr>
<tr>
<td><strong>Width</strong></td>
<td>Specified in pixels. Example: 150</td>
</tr>
<tr>
<td><strong>Height</strong></td>
<td>Specified in pixels. Example: 30</td>
</tr>
</tbody>
</table>

You can also change the values by dragging the box symbols (■) to other positions in the editor.

See also
- % Chapter 1.3.5.3.2 “Positioning elements, adapting size and layer” on page 1113

The properties contain fixed values for the coordinates of the point of rotation. This point of rotation is shown as the symbol. The point is used as the center for rotating and scaling.

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>X</strong></td>
<td>X-coordinate of the point of rotation</td>
</tr>
<tr>
<td><strong>Y</strong></td>
<td>Y-coordinate of the point of rotation</td>
</tr>
</tbody>
</table>

You can also change the values by dragging the symbols (●) to other positions in the editor.

The properties contain IEC variables for controlling the position of the element dynamically. The reference point is the upper left corner of the element. In runtime mode, the entire element is moved.
### "Movement"

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>&quot;X&quot;</strong></td>
<td>X position (in pixels). Increasing this value in runtime mode moves the element to the right.</td>
<td><code>PLC_PRG.iPos_X</code></td>
</tr>
<tr>
<td><strong>&quot;Y&quot;</strong></td>
<td>Y position (in pixels). Increasing this value in runtime mode moves the element downwards.</td>
<td><code>PLC_PRG.iPos_Y</code></td>
</tr>
<tr>
<td><strong>&quot;Rotation&quot;</strong></td>
<td>Angle of rotation (in degrees). In runtime mode, the alignment of the element rotates according to the coordinate system of the visualization. Increasing the value rotates the element to the right.</td>
<td><code>PLC_PRG.iAngle1</code></td>
</tr>
<tr>
<td><strong>&quot;Scaling&quot;</strong></td>
<td>Causes centric stretching. The value 1 shrinks the element by a factor of 0.001. The value 1000 returns the element to its original size.</td>
<td><code>PLC_PRG.iScaling</code></td>
</tr>
<tr>
<td><strong>&quot;Interior rotation&quot;</strong></td>
<td>Angle of rotation (in degrees). In runtime mode, the element rotates about the point of rotation specified in &quot;Center&quot; according to the value of the variable. Increasing the value rotates the element clockwise.</td>
<td><code>PLC_PRG.iAngle2</code></td>
</tr>
</tbody>
</table>

Note: If a static angle of rotation is specified in the property "Position ➔ Angle", then the static angle of rotation is added to the variable angle of rotation (offset) when the visualization is executed.

---

*You can link the variables to a unit conversion.*

*The properties "X", "Y", "Rotation", and "Interior rotation" are supported by the "Client Animation" functionality.*
Element property 'State variables'

| “Invisible” | Variable (BOOL). Toggles the visibility of the element.  
| TRUE: The element is not visible at runtime. |

The “Invisible” property is supported by the "Client Animation" functionality.

See also

Element property 'Control variables'

| “URL” | URL of the web page that is displayed in the visualization.  
| Variable (STRING or WSTRING)  
| Example: PLC_PRG.stURL  
| Literal in single straight quotation marks  
| Example: ‘http://de.wikipedia.org’ |

| “Show” | Variable (BOOL).  
| Example: PLC_PRG.bSetURL  
| Controls the display of the “Web browser” element.  
| If the variable contains a rising edge, then the visualization calls the web page given in “URL” and displays its contents in the 'Web browser' visualization element. |

| “Back” | Variable (BOOL).  
| Example: PLC_PRG.bGoBack  
| Controls the back navigation in the “Web browser”. If the variable has a rising edge, then the visualization displays the contents of the previously displayed page. |

| “Forward” | Variable (BOOL).  
| Example: PLC_PRG.bGoForward  
| Controls the forward navigation in the “Web browser”. If the variable has a rising edge, then the visualization displays the contents of the previously displayed page. |

These properties are available only when you select the “Preview: Support client animations and overlay of native elements” option in the Visualization Manager.

You can animate a movement, a rotation, and the visibility of a visualization element.

See also

- Unit conversion
| "Animation duration" | Time that the element executes an animation (in milliseconds)  
| Example: 500 |
| "Move to the foreground" | Property value (BOOL)  
| TRUE: At runtime, the element is displayed in the foreground.  
| FALSE: At runtime, the element is displayed in the layer where it was inserted in the visualization editor. |

See also  
- § Chapter 1.3.5.21.5 “Using client animation” on page 1962

Element property 'Access rights'  
Requirement: User management is set up for the visualization.

| "Access rights" | Opens the “Access rights“ dialog. There you can edit the access privileges for the element.  
| Status messages:  
| "Not set. Full rights." : Access rights for all user groups : “operable”  
| "Rights are set: Limited rights" : Access is restricted for at least one group. |

See also  
- § Chapter 1.3.5.19.3.1 “Dialog ‘Access Rights’” on page 1583

Visualization element ‘Waiting symbol cube’

Symbol:

Tag: “Special controls”

In runtime mode, this element indicates automatically that the runtime system is busy or waiting for data.

Element properties

| "Element name" | Example: Data_Transfer  
| Optional  
| Hint: Assign individual names for elements so that they are found faster in the element list. |
| "Type of element" | Waiting symbol cube

Element property 'Position'  
The position defines the location and size of the element in the visualization window. These are based on the Cartesian coordinate system. The origin is located at the upper left corner of the window. The positive horizontal x-axis runs to the right. The positive vertical y-axis runs downwards.
### X

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>X coordinate of the upper left corner of the element</td>
<td>Specified in pixels. Example: 10.</td>
</tr>
</tbody>
</table>

### Y

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Y</td>
<td>Y coordinate of the upper left corner of the element</td>
<td>Specified in pixels. Example: 10.</td>
</tr>
</tbody>
</table>

### Width

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Width</td>
<td>Specified in pixels.</td>
<td>Example: 150</td>
</tr>
</tbody>
</table>

### Height

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Height</td>
<td>Specified in pixels.</td>
<td>Example: 30</td>
</tr>
</tbody>
</table>

You can also change the values by dragging the box symbols ((Constructor) to other positions in the editor.

See also

- Chapter 1.3.5.3.2 “Positioning elements, adapting size and layer” on page 1113

### Element property 'Center'

The properties contain fixed values for the coordinates of the point of rotation. This point of rotation is shown as the symbol. The point is used as the center for rotating and scaling.

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>X-coordinate of the point of rotation</td>
</tr>
<tr>
<td>Y</td>
<td>Y-coordinate of the point of rotation</td>
</tr>
</tbody>
</table>

You can also change the values by dragging the symbols (Constructor) to other positions in the editor.

### Element property 'Absolute movement'

The properties contain IEC variables for controlling the position of the element dynamically. The reference point is the upper left corner of the element. In runtime mode, the entire element is moved.

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Movement</td>
<td>Variable (numeric data type). Defines the X position (in pixels). Example: PLC_PRG.iPos_X. Increasing this value in runtime mode moves the element to the right.</td>
</tr>
<tr>
<td></td>
<td>Variable (numeric data type). Defines the Y position (in pixels). Example: PLC_PRG.iPos_Y. Increasing this value in runtime mode moves the element downwards.</td>
</tr>
</tbody>
</table>
"Rotation" Variable (numeric data type). Defines the angle of rotation (in degrees).

Example: PLC_PRG.iAngle1.

The midpoint of the element rotates at the “Center” point. This rotation point is shown as the symbol.

In runtime mode, the alignment of the element remains the same with respect to the coordinate system of the visualization. Increasing the value rotates the element to the right.

"Interior rotation" Variable (numeric data type). Defines the angle of rotation (in degrees).

Example: PLC_PRG.iAngle2.

In runtime mode, the element rotates about the point of rotation specified in “Center” according to the value of the variable. In addition, the alignment of the element rotates according to the coordinate system of the visualization. Increasing the value in the code rotates clockwise.

The rotation point is shown as the symbol.

Note: If a static angle of rotation is specified in the “Position Angle” property, then the static angle of rotation is added to the variable angle of rotation (offset) when the visualization is executed.

You can link the variables to a unit conversion.

The “X”, “Y”, “Rotation”, and “Interior rotation” properties are supported by the "Client Animation" functionality.

See also
- Unit conversion
- 

Element property 'State variables'
The variables control the element behavior dynamically.

"Invisible" Variable (BOOL). Toggles the visibility of the element.

TRUE: The element is not visible at runtime.

The “Invisible” property is supported by the "Client Animation" functionality.
These properties are available only when you select the “Preview: Support client animations and overlay of native elements” option in the Visualization Manager.

You can animate a movement, a rotation, and the visibility of a visualization element.

<table>
<thead>
<tr>
<th>“Animation duration”</th>
<th>Time that the element executes an animation (in milliseconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Example: 500</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>“Move to the foreground”</th>
<th>Property value (BOOL)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>TRUE: At runtime, the element is displayed in the foreground.</td>
</tr>
<tr>
<td></td>
<td>FALSE: At runtime, the element is displayed in the layer where it was inserted in the visualization editor.</td>
</tr>
</tbody>
</table>

See also

- Chapter 1.3.5.21.5 “Using client animation” on page 1962

Element property 'Access rights'

Requirement: User management is set up for the visualization.

<table>
<thead>
<tr>
<th>“Access rights”</th>
<th>Opens the “Access rights” dialog. There you can edit the access privileges for the element.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Status messages:</td>
</tr>
<tr>
<td></td>
<td>- “Not set. Full rights.”: Access rights for all user groups: “operable”</td>
</tr>
<tr>
<td></td>
<td>- “Rights are set: Limited rights.”: Access is restricted for at least one group.</td>
</tr>
</tbody>
</table>

See also

- Chapter 1.3.5.19.3.1 “Dialog ‘Access Rights’” on page 1583

See also

- 

Visualization element 'Waiting symbol flower'

Symbol:

Tag: “Special controls”

The element indicates that the system is busy or waiting for data.
### Element property 'Position'

The position defines the location and size of the element in the visualization window. These are based on the Cartesian coordinate system. The origin is located at the upper left corner of the window. The positive horizontal x-axis runs to the right. The positive vertical y-axis runs downwards.

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;X&quot;</td>
<td>X coordinate of the upper left corner of the element</td>
</tr>
<tr>
<td></td>
<td>Specified in pixels.</td>
</tr>
<tr>
<td>Example:</td>
<td>10.</td>
</tr>
<tr>
<td>&quot;Y&quot;</td>
<td>Y coordinate of the upper left corner of the element</td>
</tr>
<tr>
<td></td>
<td>Specified in pixels.</td>
</tr>
<tr>
<td>Example:</td>
<td>10.</td>
</tr>
<tr>
<td>&quot;Width&quot;</td>
<td>Specified in pixels.</td>
</tr>
<tr>
<td>Example:</td>
<td>150</td>
</tr>
<tr>
<td>&quot;Height&quot;</td>
<td>Specified in pixels.</td>
</tr>
<tr>
<td>Example:</td>
<td>30</td>
</tr>
</tbody>
</table>

You can also change the values by dragging the box symbols (_drag_box) to other positions in the editor.

See also
- Chapter 1.3.5.3.2 “Positioning elements, adapting size and layer” on page 1113

### Element property 'Center'

The properties contain fixed values for the coordinates of the point of rotation. This point of rotation is shown as the _rotate_ symbol. The point is used as the center for rotating and scaling.

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;X&quot;</td>
<td>X-coordinate of the point of rotation</td>
</tr>
<tr>
<td>&quot;Y&quot;</td>
<td>Y-coordinate of the point of rotation</td>
</tr>
</tbody>
</table>

You can also change the values by dragging the symbols (_rotate) to other positions in the editor.

### Element property 'Colors'

The properties contain fixed values for setting colors.
### “Frame color”

### “Fill color”

### “Transparency”
Value (0 to 255) for defining the transparency of the selected color.
Example 255: The color is opaque. 0: The color is completely transparent.

See also
- Chapter 1.3.5.19.3.5 “Dialog ‘Gradient Editor’” on page 1586

### Element property ‘Appearance’

The properties contain fixed values for setting the look of the element.

<table>
<thead>
<tr>
<th>“Line width”</th>
<th>Value in pixels</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example: 2</td>
<td></td>
</tr>
</tbody>
</table>

Note: The values 0 and 1 both result in a line weight of 1 pixel. If no line should be displayed, then the “Line style” property must be set to the option “Invisible”.

### “Fill attributes”
The way in which the element is filled.
- “Filled”: The element is filled with the color from property “Colors ➔ Fill color”.
- “Invisible”: The fill color is invisible.

### “Line style”
Type of line representation
- “Solid”
- “Dashes”
- “Dots”
- “Dash Dot”
- “Dash Dot Dot”
- “not visible”

You can assign variables in the “Appearance variables” property for controlling the appearance dynamically. The fixed values here are overwritten.

See also
- “Element property ‘Appearance variables’” on page 1285

### “Symbol color”
Selection of a color for the flower symbol.

### “Line”
Stroke width of the lines (in pixels).

### Element property ‘Absolute movement’
The properties contain IEC variables for controlling the position of the element dynamically. The reference point is the upper left corner of the element. In runtime mode, the entire element is moved.

<table>
<thead>
<tr>
<th>“Movement”</th>
<th></th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>“X”</th>
<th>Variable (numeric data type). Defines the X position (in pixels).</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Example: PLC_PRG.iPos_X.</td>
</tr>
<tr>
<td></td>
<td>Increasing this value in runtime mode moves the element to the right.</td>
</tr>
<tr>
<td>Property</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>-------------</td>
</tr>
<tr>
<td>“Y”</td>
<td>Variable (numeric data type). Defines the Y position (in pixels).</td>
</tr>
<tr>
<td>“Rotation”</td>
<td>Variable (numeric data type). Defines the angle of rotation (in degrees).</td>
</tr>
<tr>
<td>“Interior rotation”</td>
<td>Variable (numeric data type). Defines the angle of rotation (in degrees).</td>
</tr>
</tbody>
</table>

You can link the variables to a unit conversion.

The “X”, “Y”, “Rotation”, and “Interior rotation” properties are supported by the “Client Animation” functionality.

See also
- Unit conversion

Element property 'State variables'

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Invisible”</td>
<td>Variable (BOOL). Toggles the visibility of the element.</td>
<td>TRUE: The element is not visible at runtime.</td>
</tr>
</tbody>
</table>

The variables control the element behavior dynamically.
The “Invisible” property is supported by the “Client Animation” functionality.

See also

●

These properties are available only when you select the “Preview: Support client animations and overlay of native elements” option in the Visualization Manager.

You can animate a movement, a rotation, and the visibility of a visualization element.

**“Animation duration”**

| Time that the element executes an animation (in milliseconds) |
| Example: 500 |

**“Move to the foreground”**

| Property value (BOOL) |
| TRUE: At runtime, the element is displayed in the foreground. |
| FALSE: At runtime, the element is displayed in the layer where it was inserted in the visualization editor. |

See also

● † Chapter 1.3.5.21.5 “Using client animation” on page 1962

**Element property ‘Access rights’**

Requirement: User management is set up for the visualization.

**“Access rights”**

Opens the “Access rights” dialog. There you can edit the access privileges for the element.

Status messages:

● “Not set. Full rights.”: Access rights for all user groups: “operable”

● “Rights are set: Limited rights”: Access is restricted for at least one group.

See also

● † Chapter 1.3.5.19.3.1 “Dialog ‘Access Rights’” on page 1583

See also

●

**Visualization element ‘Text editor’**

Symbol:

Tag: “Special controls”

The element shows the contents of text files that are saved on the controller. Files can be encoded in ASCII or Unicode formats.

A visualization user can also edit the text.
Element properties

<table>
<thead>
<tr>
<th>“Element name”</th>
<th>Example: GenElemInst_1</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Type of element”</td>
<td>“Text editor”</td>
</tr>
</tbody>
</table>

**Element property 'Position'**

The position defines the location and size of the element in the visualization window. These are based on the Cartesian coordinate system. The origin is located at the upper left corner of the window. The positive horizontal x-axis runs to the right. The positive vertical y-axis runs downwards.

| “X” | X coordinate of the upper left corner of the element  
Specified in pixels.  
Example: 10. |
| “Y” | Y coordinate of the upper left corner of the element  
Specified in pixels.  
Example: 10. |
| “Width” | Specified in pixels.  
Example: 150 |
| “Height” | Specified in pixels.  
Example: 30 |

You can also change the values by dragging the box symbols (_drag_ ) to other positions in the editor.

See also

- Chapter 1.3.5.3.2 “Positioning elements, adapting size and layer” on page 1113

**Element property 'Font'**

| “Font name” | Non-proportional font used by the visualization to display the contents of the file  
Example: “Courier New” |
| “Size” | Font size  
Example: 12 |

**Element property 'Control variables'**
### Table 261: Element property “Control variables --> File”

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
</table>
| **Variable** | Variable (STRING). Contains the file names and optionally the location of the file. It is located in the file system of the controller.  
Example: FLC_PRG.strFile: STRING := '/Documentation/Info.txt'; |
| **Open** | Variable (BOOL). Controls opening the file which is defined in the “Variable” property  
Example: bOpen: BOOL;  
TRUE: The file is opened. |
| **Close** | Variable (BOOL). Controls closing the file which is defined in the “Variable” property  
Example: bClose: BOOL;  
TRUE: The file is closed. |
| **Save** | Variable (BOOL). Controls saving the file which is defined in the “Variable” property  
Example: bStore: BOOL;  
TRUE: The file is saved. |
| **New** | Variable (BOOL). Controls creating a new file. The name is defined in the “Variable” property.  
Example: bCreate: BOOL;  
TRUE: A file is created and opened. |

### Table 262: Element property “Control variables --> Edit”

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
</table>
| **Variable** | Variable (STRING). Contains the string to search for in the file  
Example: strFind: STRING := 'abc'; |
| **Find** | Variable (BOOL). Controls executing the search for the string in the “Variable” property  
Example: bFind: BOOL;  
TRUE: The search is performed. The variable is automatically reset to FALSE. |
| **Find next** | Variable (BOOL). Controls the location to begin the search in the file  
Example: bFindNext: BOOL;  
TRUE: The search begins at the last search result location.  
FALSE: The search begins at the beginning of the file. |

### Table 263: Element property “Control variables --> Cursor position”

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
</table>
| **Line** | Variable (integer data type). Contains the line of the cursor  
Example: iRowCursor: INT; |
| **Column** | Variable (integer data type). Contains the column of the cursor  
Example: iColumnCursor: INT; |
**“Position”**
Output variable (integer data type). Shows the absolute cursor position in the text.
Example: iPosCursor: INT;

**“Set cursor”**
Variable (BOOL). Controls the setting of the cursor at a specific location
Example: iSetCursor: INT;
TRUE: The cursor is moved. The new position is defined in the “Line” and “Column” properties.
FALSE: The “Line”, “Column”, and “Position” properties contain the actual values.
Note: The variable is used as the control variable for an input event triggered by a visualization user.

**Table 264: Element property “Control variables --> Selection”**

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Start position”</td>
<td>Output variable (integer data type). Shows the absolute position for starting the text selection</td>
<td>iPosSelection: INT;</td>
</tr>
<tr>
<td>“End position”</td>
<td>Output variable (integer data type). Shows the absolute position for ending the text selection</td>
<td>iPosEndSelection: INT;</td>
</tr>
<tr>
<td>“Start line number”</td>
<td>Output variable (integer data type). Shows the line where the text selection begins</td>
<td>iRowSelection: INT;</td>
</tr>
<tr>
<td>“Start column index”</td>
<td>Output variable (integer data type). Shows the column where the text selection begins</td>
<td>iColumnSelection: INT;</td>
</tr>
<tr>
<td>“End line number”</td>
<td>Output variable (integer data type). Shows the line where the text selection ends</td>
<td>iRowEndSelection: INT;</td>
</tr>
<tr>
<td>“End column index”</td>
<td>Output variable (integer data type). Shows the column where the text selection ends</td>
<td>iColumnEndSelection: INT;</td>
</tr>
<tr>
<td>“Line to select”</td>
<td>Variable (integer data type). Contains the line number that is selected</td>
<td>bSetSelection: BOOL;</td>
</tr>
<tr>
<td></td>
<td>Note: The selection is controlled by the variables in the “Trigger selection” property.</td>
<td></td>
</tr>
<tr>
<td>“Set selection”</td>
<td>Variable (BOOL). Controls the selection of a line.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Example: bSetSelection: BOOL;</td>
<td></td>
</tr>
<tr>
<td></td>
<td>TRUE: The line from the “Line to select” property is selected and highlighted in the text editor.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>if the line is not in the current text segment of the text editor, then the text segment is moved to this line.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Note: The variable is used as the control variable for an input event triggered by a visualization user. The control variable is not reset automatically. You are responsible for this to occur in the visualization.</td>
<td></td>
</tr>
</tbody>
</table>
### Table 265: Element property “Control variables --> Error handling”

<table>
<thead>
<tr>
<th>Variable for error code</th>
<th>Variable (integer data type). Contains the error code when an error occurs.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Example: <code>iError: INT;</code></td>
</tr>
<tr>
<td></td>
<td>The error codes are declared in <code>GVL_ErrorCodes</code> in the <code>VisuElemTextEditor</code> library. To display the error text, the <code>VisuFctTextEditorGetErrorText()</code> function of the library must be called.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Variable for content changed</th>
<th>Variable (BOOL). Shows whether the contents have changed</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Example: <code>bIsContentEdited: BOOL;</code></td>
</tr>
<tr>
<td></td>
<td><code>TRUE</code>: The contents of the text editor have changed.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Variable for access mode</th>
<th>Variable (BOOL). Controls the access privileges to the file</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Example: <code>bIsReadOnly: BOOL;</code></td>
</tr>
<tr>
<td></td>
<td><code>TRUE</code>: A visualization user has read-only permission. At runtime, the file contents are highlighted in gray in the text editor.</td>
</tr>
<tr>
<td></td>
<td><code>FALSE</code>: A visualization user has read/write permission.</td>
</tr>
<tr>
<td></td>
<td>Note: The variable overwrites the setting in the “Editor mode” property.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Maximum line length</th>
<th>Maximum number of characters per line</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Editor mode</th>
<th>“Read-only”: A visualization user has read-only permissions to the file. At runtime, the file contents are highlighted in gray in the text editor.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>“Read/write”: A visualization user has read-write permissions.</td>
</tr>
</tbody>
</table>

### Element property ‘New files’

<table>
<thead>
<tr>
<th>Encoding</th>
<th>Character encoding of the new file:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>“ASCII”</td>
</tr>
<tr>
<td></td>
<td>“Unicode (Little endian)”</td>
</tr>
<tr>
<td></td>
<td>“Unicode (Big endian)”</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>New line character sequence</th>
<th>End of line character of the new file:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>“CR/LF”: Normal for Windows systems</td>
</tr>
<tr>
<td></td>
<td>“LF”: Normal for UNIX systems</td>
</tr>
<tr>
<td></td>
<td>Please note: When a visualization user opens an existing file, the end-of-line character of the file is detected and used automatically.</td>
</tr>
</tbody>
</table>

These properties are available only when you select the “Preview: Support client animations and overlay of native elements” option in the Visualization Manager.

You can animate a movement, a rotation, and the visibility of a visualization element.
### Animation duration

Time that the element executes an animation (in milliseconds)

**Example:** 500

### Move to the foreground

**Property value** *(BOOL)*

- **TRUE:** At runtime, the element is displayed in the foreground.
- **FALSE:** At runtime, the element is displayed in the layer where it was inserted in the visualization editor.

**See also**

- Chapter 1.3.5.21.5 “Using client animation” on page 1962

#### Requirement: User management is set up for the visualization.

### Access rights

Opens the “Access rights” dialog. There you can edit the access privileges for the element.

**Status messages:**

- **“Not set. Full rights.”**: Access rights for all user groups: “operable”
- **“Rights are set: Limited rights.”**: Access is restricted for at least one group.

**See also**

- Chapter 1.3.5.19.3.1 “Dialog ‘Access Rights’” on page 1583

**See also**

- [Visualization element ‘Path3D’](#)

#### Symbol:

![Path3D Symbol](image)

**Tag:** “Special controls”

The visualization element “Path3D” graphically displays the curves of two independent records as a 3D path. It is specially designed for use with CODESYS SoftMotion CNC in order to display the trajectory of a machine tool or a robot. The programmed path (path) and the path actually traveled (track) is visualized.

Although the visualization element is designed for use with CODESYS SoftMotion CNC, it can also be used to display any other record. In this case the application must provide the path data. The sample application 3D Path Generator, which is available in the CODESYS Store, shows how these data can be generated.

If the element is used together with SoftMotion CNC, then function blocks from the library SM3_CNC_Visu help to generate the data from the path and track. These function blocks are used by the sample project CNC_File_3DPath, which is stored in the installation directory of CODESYS.

- SMC_PathCopier
- SMC_PathCopierCompleteQueue
- SMC_PathCopierFile
- SMC_PositionTracker
A description of the function blocks can be found in the Library Manager in the SM3_CNC_Visu library.

The element does not work with the CODESYS HMI display variant.

See also
- CNC Example 6: Using Path3D with SoftMotion CNC

Element properties

<table>
<thead>
<tr>
<th>&quot;Element name&quot;</th>
<th>Example: GenElemInst_1</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Type of element&quot;</td>
<td>&quot;Path3D&quot;</td>
</tr>
</tbody>
</table>

Element property 'Position'

The position defines the location and size of the element in the visualization window. These are based on the Cartesian coordinate system. The origin is located at the upper left corner of the window. The positive horizontal x-axis runs to the right. The positive vertical y-axis runs downwards.

<table>
<thead>
<tr>
<th>&quot;X&quot;</th>
<th>X coordinate of the upper left corner of the element Specified in pixels. Example: 10.</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Y&quot;</td>
<td>Y coordinate of the upper left corner of the element Specified in pixels. Example: 10.</td>
</tr>
<tr>
<td>&quot;Width&quot;</td>
<td>Specified in pixels. Example: 150</td>
</tr>
<tr>
<td>&quot;Height&quot;</td>
<td>Specified in pixels. Example: 30</td>
</tr>
</tbody>
</table>

You can also change the values by dragging the box symbols (_drag-) to other positions in the editor.

See also
- Chapter 1.3.5.3.2 “Positioning elements, adapting size and layer” on page 1113

Path description
### Path data (VisuStruct3DTrack)
Variable of the type VisuStruct3DTrack, which is declared in the IEC code. Example: PLC_PRG.pc.vs3dt. A description of the structure can be found in the library manager in the library VisuElem3DPath.library.

The data structure describes a path or track through a certain number of points. The points are determined and buffered by the application. The track typically displays the last n positions, so that only a certain part of them is ever displayed at any one time. VisuStruct3DTrack.pProjection is a variable that is set by the visualization element and contains information about the path/track projection. It can be read (only) by the application. In addition, the methods Projection.Apply or .ApplyV can be used in order to see whether the transformed position lies inside or outside the visualization display area, which is defined by Projection.ElementRect.

#### Path color
Color of the path drawn

#### Path line width
Path line width in pixels, e.g.: "2"

#### Style of boundary points
Display of the points between two successive objects in the path
- End points are not displayed
- End points are marked with a circle
- End points are marked with a cross
- End points are marked with a plus

### Track description
The track data are structured in exactly the same way as the path data: VisuStruct3DTrack

#### Track data (VisuStruct3DTrack)
Variable of the type VisuStruct3DTrack, which is declared in the IEC code. Example: PLC_PRG.pc.vs3dt. A description of the structure can be found in the library manager in the library VisuElem3DPath.library.

#### Track color
Color of the track drawn

#### Track line width
Track line width in pixels, e.g.: "2"

### Camera control
The camera position for the 3D mode is controlled with a reference to the external data structure. This structure allows the following operations:
- Shifting to the left/to the right/upwards/downwards
- Rotation around the X/Y/Z axis
- Resetting of the view at X/Y, Y/Z or Z/X level, so that the path and the track are completely visible.

#### Control data (VisuStruct3DControl)
Variable of the type VisuStruct3DControl, which is declared in the IEC code. Example: PLC_PRG.pc.vs3dc. A description of the structure can be found in the library manager in the library VisuElem3DPath library.

The values can be set via the application itself or via the visualization element “ControlPanel”. The library VisuElem3DPath contains ready-to-use visualization frames that provide a possible user interface for these data.
Highlighting

Individual parts of the path can be visually highlighted. This is typically used to mark the already processed part of a track with a different color. Each point in the path is given a unique ID, which in the case of a CNC editor is linked with the object ID on which the point lies. This ID ("highlight ID") can be specified via the application so that dynamic elements/parts of the track can be highlighted.

<table>
<thead>
<tr>
<th>&quot;Highlight mode&quot;</th>
<th>Select one of the following highlight modes:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Only the element whose ID corresponds to the value of the variable is highlighted.</td>
<td></td>
</tr>
<tr>
<td>All elements whose ID (linked with the object ID in the case of a CNC editor) is smaller than or equal to the value in &quot;Variable&quot; are highlighted.</td>
<td></td>
</tr>
</tbody>
</table>

| "Variable" | Project variable that specifies the ID of an element. Example: PLC_PRG.iVarElementID. This "highlight ID" is taken into account for the setting of the "highlight mode". The variable must be set in the IEC application. |

| "Highlight color" | Color of the grid lines |

Element look

| "Frame line width" | Width of the frame around the element, in pixels, for example: "1" |
| "Frame line style" | Select one of these style types for the frame line: |
| Solid | Dashes | Dots | Dash Dot | Dash Dot Dot | Hollow |

| "Transparent background" | The background of the element is displayed transparently. |
| "Background color" | The background of the element is displayed in the defined background color. |

These properties are available only when you select the "Preview: Support client animations and overlay of native elements" option in the Visualization Manager.

You can animate a movement, a rotation, and the visibility of a visualization element.

<table>
<thead>
<tr>
<th>&quot;Animation duration&quot;</th>
<th>Time that the element executes an animation (in milliseconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example: 500</td>
<td></td>
</tr>
</tbody>
</table>

| "Move to the foreground" | Property value (BOOL) |
| TRUE: At runtime, the element is displayed in the foreground. |
| FALSE: At runtime, the element is displayed in the layer where it was inserted in the visualization editor. |
Element property 'Access rights'

<table>
<thead>
<tr>
<th>“Access rights”</th>
<th>Opens the “Access rights” dialog. There you can edit the access privileges for the element.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Status messages:</td>
</tr>
<tr>
<td></td>
<td>● “Not set. Full rights.”: Access rights for all user groups: “operable”</td>
</tr>
<tr>
<td></td>
<td>● “Rights are set: Limited rights”: Access is restricted for at least one group.</td>
</tr>
</tbody>
</table>

See also

● § Chapter 1.3.5.19.3.1 “Dialog ‘Access Rights’” on page 1583

Visualization element 'Control Panel'

Symbol:

Tag: “Special controls”

This visualization element is used in connection with the “Path3D” visualization element. It is used for changing the position and orientation to the CNC path shown with “Path3D”.

See also

● § Chapter 1.3.5.18.1.42 “Visualization element ‘Path3D’” on page 1499

Element properties

<table>
<thead>
<tr>
<th>“Element name”</th>
<th>Optional.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Hint: Assign individual names for elements so that they are found faster in the element list.</td>
</tr>
<tr>
<td></td>
<td>Example: Camera_Path_1</td>
</tr>
<tr>
<td>“Type of element”</td>
<td>“Frame”</td>
</tr>
<tr>
<td>“Clipping”</td>
<td>☑️ If you have set the “Scaling type” to “Fixed”, then only that part of the visualization is displayed that fits in the frame.</td>
</tr>
<tr>
<td>“Show frame”</td>
<td>Displays the frame</td>
</tr>
<tr>
<td></td>
<td>● “No frame”: The displayed area of the frame does not have borders.</td>
</tr>
<tr>
<td></td>
<td>● “Frame”: The displayed area of the frame has borders.</td>
</tr>
<tr>
<td></td>
<td>● “No frame with offset”: The displayed area of the frame does not have a border and the displayed area of the referenced visualization is reduced downwards by one pixel as compared to the frame area. The resulting gap prevents the referenced visualization from touching any adjacent elements.</td>
</tr>
</tbody>
</table>
"Scaling type" Describes how the frame reacts when the visualization is resized:

- "Isotropic": The frame retains its proportions. This allows the ratio of height to width to be preserved, even if the height and width of the visualization have been changed separately.
- "Anisotropic": The frame depends on the size of the visualization, so that height and width of the referenced visualization can be changed separately.
- "Fixed": The original size of the frame is retained, regardless of the visualization size. If you have also selected the "Clipping" option, then only the fitting part is displayed.
- "Fixed and scrollable": The referenced visualization is displayed without scaling. If it is larger than the window area of the frame, then scrollbars are included on the frame. To set the position of the scrollbar with a variable, use the "Scroll position variable horizontal" or "Scroll position variable vertical" property.

"Deactivation of the background drawing" To optimize the performance of the visualization, the non-animated elements of the frame element are drawn as a background bitmap. This could result in the elements not being displayed in the expected order.

- Deactivation of the background drawing. This can prevent the behavior described above.

Element property 'References'
Contains the currently configured visualization references as a subnode

"References" Clicking “Configure” opens the “Frame Configuration” dialog. This is used to manage the referenced visualizations.

Caution: Visualizations can be nested at any depth by means of frame elements. In order to use the “Switch to any visualization” frame selection type without any problems, a frame must not contain more than 21 referenced visualizations. For more information, see also the description for the “Input configuration” of an element: Action “Switch frame visualization”.

List of the currently referenced visualizations Visualizations that have a button also have this displayed as a subnode. Each interface variable is listed with the currently assigned transfer parameters.

Example:
vis_FormA
- iDataToDisplay_1: PLC_PRG.iVar1
- iDataToDisplay_2: PLC_PRG.iVar2

Hint: You can change the assignment of the variables to an interface variable here and edit the value field. Or click the “Configure” button instead.

See also
- Chapter 1.3.5.19.2.1 “Command 'Interface Editor'” on page 1556
- Chapter 1.3.5.19.3.6 “Dialog 'Input Configuration'” on page 1586

Element property 'Position'
The position defines the location and size of the element in the visualization window. These are based on the Cartesian coordinate system. The origin is located at the upper left corner of the window. The positive horizontal x-axis runs to the right. The positive vertical y-axis runs downwards.
<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>“X”</td>
<td>X coordinate of the upper left corner of the element&lt;br&gt;Specified in pixels.&lt;br&gt;Example: 10.</td>
</tr>
<tr>
<td>“Y”</td>
<td>Y coordinate of the upper left corner of the element&lt;br&gt;Specified in pixels.&lt;br&gt;Example: 10.</td>
</tr>
<tr>
<td>“Width”</td>
<td>Specified in pixels.&lt;br&gt;Example: 150</td>
</tr>
<tr>
<td>“Height”</td>
<td>Specified in pixels.&lt;br&gt;Example: 30</td>
</tr>
</tbody>
</table>

You can also change the values by dragging the box symbols (_drag) to other positions in the editor.

See also
- Chapter 1.3.5.3.2 “Positioning elements, adapting size and layer” on page 1113

**Element property ‘Center’**

The properties contain fixed values for the coordinates of the point of rotation. This point of rotation is shown as the symbol. The point is used as the center for rotating and scaling.

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>“X”</td>
<td>X-coordinate of the point of rotation</td>
</tr>
<tr>
<td>“Y”</td>
<td>Y-coordinate of the point of rotation</td>
</tr>
</tbody>
</table>

You can also change the values by dragging the symbols (drag) to other positions in the editor.

**Element property ‘Colors’**

The properties contain fixed values for setting colors.

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Color”</td>
<td>Color for the element in its normal state.&lt;br&gt;Please note that the normal state is in effect if the expression in the “Color variables ➔ Toggle color” property is not defined or it has the value FALSE.</td>
</tr>
<tr>
<td>“Alarm color”</td>
<td>Color for the element in alarm state.&lt;br&gt;Please note that the alarm state is in effect if the expression in the “Color variables ➔ Toggle color” property has the value TRUE.</td>
</tr>
<tr>
<td>“Transparency”</td>
<td>Value (0 to 255) for defining the transparency of the selected color.&lt;br&gt;Example 255: The color is opaque. 0: The color is completely transparent.</td>
</tr>
<tr>
<td>“Use gradient color”</td>
<td>: The element is displayed with a color gradient.</td>
</tr>
<tr>
<td>“Gradient setting”</td>
<td>The “Color gradient editor” dialog box opens.</td>
</tr>
<tr>
<td>“Frame color”</td>
<td>Example: “Black”</td>
</tr>
<tr>
<td>“Fill color”</td>
<td>Example: “Light gray”</td>
</tr>
</tbody>
</table>
The properties contain fixed values for setting the look of the element.

**Element property ‘Appearance’**

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>“Line width”</strong></td>
<td>Value in pixels</td>
</tr>
<tr>
<td></td>
<td>Example: 2</td>
</tr>
<tr>
<td></td>
<td>Note: The values 0 and 1 both result in a line weight of one pixel. If no line</td>
</tr>
<tr>
<td></td>
<td>should be displayed, then the “Line style” property must be set to the option</td>
</tr>
<tr>
<td></td>
<td>“Invisible”.</td>
</tr>
<tr>
<td><strong>“Line style”</strong></td>
<td>Type of line representation</td>
</tr>
<tr>
<td></td>
<td>● “Solid”</td>
</tr>
<tr>
<td></td>
<td>● “Dashes”</td>
</tr>
<tr>
<td></td>
<td>● “Dots”</td>
</tr>
<tr>
<td></td>
<td>● “Dash Dot”</td>
</tr>
<tr>
<td></td>
<td>● “Dash Dot Dot”</td>
</tr>
<tr>
<td></td>
<td>● “not visible”</td>
</tr>
</tbody>
</table>

You can assign variables in the “Appearance variables” property for controlling the appearance dynamically. The fixed values are defined here.

**Element property ‘Texts’**

The properties contains character strings for labeling the element. The character string can also contain a placeholder with a format definition. In runtime mode, the placeholder is replaced by the current value in the specified format.

CODESYS accepts the specified texts automatically into the “GlobalTextList” text list. Therefore, these texts can be localized.

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>“Text”</strong></td>
<td>Character string (without single straight quotation marks) for the labeling the element. Add a line break by pressing the keyboard shortcut [Ctrl] + [Enter].</td>
</tr>
<tr>
<td></td>
<td>Example: Accesses: %i</td>
</tr>
<tr>
<td></td>
<td>The variable that contains the current value for the placeholder is specified in the property “Text variable ➔ Text”.</td>
</tr>
<tr>
<td><strong>“Tooltip”</strong></td>
<td>Character string (without single straight quotation marks) that is displayed as the tooltip of an element.</td>
</tr>
<tr>
<td></td>
<td>Example: Number of valid accesses.</td>
</tr>
<tr>
<td></td>
<td>The variable that contains the current value for the placeholder is specified in the property “Text variable ➔ Tooltip”.</td>
</tr>
</tbody>
</table>

See also

- “Element property ‘Appearance variables’” on page 1285
- “Element property ‘Text variables’” on page 1229
The properties contain fixed values for the text properties.

<table>
<thead>
<tr>
<th><strong>Horizontal alignment</strong></th>
<th>Horizontal alignment of the text within the element.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Vertical alignment</strong></td>
<td>Vertical alignment of the text within the element.</td>
</tr>
</tbody>
</table>

**Text format**
- **Default**: The long text is truncated.
- **Line break**: The text is split into parts.
- **Ellipsis**: The visible text ends with "..." indicating that it is not complete.

**Font**
Example: **Default**
- The “Font” dialog box opens.
- DROP-DOWN LIST WITH STYLE FONTS.

**Font color**
Example: **Black**
- The “Color” dialog box opens.
- DROP-DOWN LIST WITH STYLE COLORS.

**Transparency**
Whole number (value range from 0 to 255). This determines the transparency of the respective color.
Example: 255: The color is opaque.
0: The color is completely transparent.
Please note: If the color is a style color and already has a transparency value, then this property is write-protected.

The properties contain IEC variables for controlling the position of the element dynamically. The reference point is the upper left corner of the element. In runtime mode, the entire element is moved.

**Movement**

**X**
Variable (numeric data type). Defines the X position (in pixels).
Example: PLC_PRG.iPos_X.
Increasing this value in runtime mode moves the element to the right.

**Y**
Variable (numeric data type). Defines the Y position (in pixels).
Example: PLC_PRG.iPos_Y.
Increasing this value in runtime mode moves the element downwards.

**Rotation**
Variable (numeric data type). Defines the angle of rotation (in degrees).
Example: PLC_PRG.iAngle1.
The midpoint of the element rotates at the “Center” point. This rotation point is shown as the symbol.
In runtime mode, the alignment of the element remains the same with respect to the coordinate system of the visualization. Increasing the value rotates the element to the right.
**“Scaling”**

Variable (integer data type). Causes centric stretching.

Example: PLC_PRG.iScaling.

The reference point is the “Center” property.

The value 1 shrinks the element by a factor of 0.001. The value 1000 returns the element to its original size.

**“Interior rotation”**

Variable (numeric data type). Defines the angle of rotation (in degrees).

Example: PLC_PRG.iAngle2.

In runtime mode, the element rotates about the point of rotation specified in “Center” according to the value of the variable. In addition, the alignment of the element rotates according to the coordinate system of the visualization. Increasing the value in the code rotates clockwise.

The rotation point is shown as the symbol.

Note: If a static angle of rotation is specified in the property “Position ➔ Angle”, then the static angle of rotation is added to the variable angle of rotation (offset) when the visualization is executed.

---

You can link the variables to a unit conversion.

The properties “X”, “Y”, “Rotation”, and “Interior rotation” are supported by the “Client Animation” functionality.

---

See also

- *Unit conversion*

---

**Element property 'Relative movement'**

The properties contains variables for moving the element. The reference point is the position of the element (“Position” property). The shape of the element can change.

<table>
<thead>
<tr>
<th>“Movement top-left”</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>“X”</strong></td>
<td>Variable (integer data type). It contains the number (in pixels) that the left edge is moved horizontally. Incrementing the value moves the element to the right.</td>
</tr>
<tr>
<td>Example: PLC_PRG.iDeltaX</td>
<td></td>
</tr>
<tr>
<td><strong>“Y”</strong></td>
<td>Variable (integer data type). It contains the number (in pixels) that the top edge is moved vertically. Incrementing the value moves the element to the down.</td>
</tr>
<tr>
<td>Example: PLC_PRG.iDeltaY</td>
<td></td>
</tr>
<tr>
<td><strong>“Movement bottom-right”</strong></td>
<td></td>
</tr>
<tr>
<td><strong>X</strong></td>
<td>Variable (integer data type). It contains the number (in pixels) that the right edge is moved horizontally. Incrementing the value moves the element to the right. <strong>Example:</strong> PLC_PRG.iDeltaWidth</td>
</tr>
<tr>
<td><strong>Y</strong></td>
<td>Variable (integer data type). It contains the number (in pixels) that the bottom edge is moved vertically. Incrementing the value moves the element to the down. <strong>Example:</strong> PLC_PRG.iDeltaHeight</td>
</tr>
</tbody>
</table>

See also

- “Element property ‘Absolute movement’” on page 1227

**Element property ‘Text variables’**

These properties are variables with contents that replace a format definition.

| “Text variable” | Variable (data type compliant with the format definition). It contains what is printed instead of the format definition. **Example:** PLC_PRG.iAccesses |
| “Tooltip variable” | Variable (data type compliant with the format definition). It contains what is printed instead of the format definition. **Example:** PLC_PRG.iAccessesInTooltip |

See also

- Chapter 1.3.5.18.2 “Placeholders with format definition in the output text” on page 1546
- “Element property ‘Texts’” on page 1226
- Enumerations

**Element property ‘Dynamic texts’**

Dynamic texts are variably indexed texts of a text list. At runtime, the text is displayed that is currently indexed in the variable.
| **“Text list”** | Variable (string) or name of the text list as a fixed string in single straight quotation marks.  
Example: 'Errorlist'  
▼: Drop-down list with the dialogs available in the text lists. |
| **“Text index”** | Text list ID. This refers to the desired output text.  
- As fixed string with the ID in single straight quotation marks.  
  Example: '1'  
- As a variable (STRING) for dynamically controlling the text output.  
  Example: strTextID  
  Sample assignment: PLC_PRG.strTextID := '1'; |
| **“Tooltip index”** | Text list ID. This refers to the desired output text.  
- As fixed string with the ID in single straight quotation marks.  
  Example: '2'  
- As a variable (STRING) for dynamically controlling the text output.  
  Example: strToolTipID  
  Sample assignment: PLC_PRG.strToolTipID := '2'; |

See also  
- Text list

**Element property ’Font variables’**  
The variables allow for dynamic control of the text display.

| **“Font name”** | Variable (STRING). Includes the font of the text.  
Example: PLC_PRG.stFontVar := 'Arial';  
The selection of fonts corresponds to the default “Font” dialog. |
| **“Size”** | Variable (numeric data type). Contains the font size (in pixels or points). The applied unit is specified in brackets after the variable name.  
- <pt>: Points (default)  
  Example: PLC_PRG.iFontHeight <pt>  
  Code: iFontHeight : INT := 12;  
- <px>: Pixels  
  Example: PLC_PRG.iFontHeight <px>  
  Code: iFontHeight : INT := 19;  

If you click in the value field, a drop-down list opens on the right for setting the unit.  

Hint: The font size is specified in points (example: Arial 12). Use points when the variable font size should match a font, for example if a font is set in the property “Text property ➔ Font”.

| pt | px |
### Flags

Variable (DWORD). Contains the flags for displaying fonts.

**Flags:**
- 1: Italics
- 2: Bold
- 4: Underline
- 8: Strikethrough

Note: You can combine the font displays by adding the coding of the flags. For example, a bold and underlined text:

```plaintext
PLC_PRG.dwFontType := 6;
```

### Character set

Variable (DWORD). Contains a character set number for the font.

The selection of character set numbers corresponds to the “Script” setting of the standard “Font” dialog.

### Color

Variable (DWORD). Includes the color of the text.

Example:

```plaintext
PLC_PRG.dwColorFont := 16#FF000000;
```

### Flags for text alignment

Variable (integer data type). Contains the coding for text alignment.

**Coding:**
- 0: Top left
- 1: Horizontal center
- 2: Right
- 4: Vertical center
- 8: Bottom

Note: You can combine the text alignments by adding the coding of the flags. For example, a vertical and horizontal centered text:

```plaintext
PLC_PRG.dwFontType := 5;
```

---

*Fixed values for displaying texts are set in “Text properties”.*

See also
- § “Element property ‘Text properties’” on page 1227

**Element property ‘Colorvariables’**

The Element property is used as an interface for project variables to dynamically control colors at runtime.
### “Toggle color”

The property controls the toggled color at runtime.

**Value assignment:**
- **FALSE**: The element is displayed with the color specified in the “Color” property.
- **TRUE**: The element is displayed with the color specified in the “Alarm color” property.

**Assignment options:**
- Placeholder for the user input variable
  - “<toggle/tap variable>”
  - “<NOT toggle/tap variable>”

  The color change is not controlled by its own variable, but by a user input variable.

  Note: Specify a variable for the mouse events “Tap” or “Toggle” in the input configuration of the element. Only then is the placeholder set. If you configure a variable in both “Toggle” and “Tap”, then the variable specified in “Tap” is used.

  Hint: Click the symbol to insert the placeholder “<toggle/tap variable>”. When you activate the “Inputconfiguration”, “Tap FALSE” property, then the “<NOT toggle/tap variable>” placeholder is displayed.

- Instance path of a project variable (BOOL)
  Example: PLC_PRG.xColorIsToggeled

  Note: In the code, declare and implement the variable specified here. Its value assignment determines when the color changes.

---

### “Normal state”

**“Alarm state”**

The properties listed below control the color depending on the state. The normal state is in effect if the variable in “Color variables”, “Toggle color” is not defined or it has the value FALSE. The alarm state is in effect if the variable in “Colorvariables”, “Toggle color” has the value TRUE.

---

### “Frame color”

**Assignment options:**
- Variable (DWORD) for the frame color
  Example: PLC_PRG.dwBorderColor
- Color literal
  Example of green and opaque: 16#FF00FF00

---

### “Filling color”

**Assignment options:**
- Variable (DWORD) for the fill color
  Example: PLC_PRG.dwFillColor
- Color literal
  Example of gray and opaque: 16#FF888888

---

*The transparency part of the color value is evaluated only if the “Activate semi-transparent drawing” option of the visualization manager is selected.*

---

*Select the “Advanced” option in the toolbar of the properties view. Then all element properties are visible.*

---

**See also**
- [Chapter 1.3.5.8.3 “Animating a color display” on page 1152](#)
**Element property 'Appearance variables'**

The properties contain variables for controlling the appearance of the element dynamically.

| “Line width” | Variable (integer data type). Contains the line weight (in pixels).  
|              | Note: The values 0 and 1 both result in a line weight of one pixel. If no line should be displayed, then the “Line style” property must be set to the option “Invisible”. |

| “Line style” | Variable (DWORD). Controls the line style.  
|             | Coding:  
|             | ● 0: Solid line  
|             | ● 1: Dashed line  
|             | ● 2: Dotted line  
|             | ● 3: Line type "Dash Dot"  
|             | ● 3: Line type "Dash Dot Dot"  
|             | ● 8: Invisible: The line is not drawn. |

*Fixed values can be set in the “Appearance” property. These values can be overwritten by dynamic variables at runtime.*

See also

- ® “Element property ‘Appearance’” on page 1238

**Element property 'Switch frame variable'**

The variable controls the switching of the referenced visualizations. This variable indexes one of the referenced frame visualizations and this is displayed in the frame. When the value of the variable changes, it switches to the recently indexed visualization.

| “Variable” | Variable (integer data type) that contains the index of the active visualization  
|           | Example: PLC_PRG.uiIndexVisu  
|           | Hint: The “Frame Configuration” dialog includes a list of referenced visualizations. The visualizations are automatically numerically indexed via the order in the list.  
|           | Note: This variant of switching usually affects all connected display variants.  
|           | ● Array element (integer data type) for index access via CURRENTCLIENTID  
|           | Example: PLC_PRG.aIndexVisu[CURRENTCLIENTID]  
|           | Note: This variant of switching applies to the current client only, and therefore only on one display variant. That is the display variant where the value change was triggered (for example, by means of user input). |

See also

- ® Chapter 1.3.5.19.2.9 “Command ‘Frame Selection’” on page 1564

**Element property 'State variables'**

The variables control the element behavior dynamically.
"Invisible" Variable (BOOL). Toggles the visibility of the element.
TRUE: The element is not visible at runtime.

"Deactivate inputs" Variable (BOOL). Toggles the operability of the element.
TRUE: User inputs do not have any effect in runtime more. The element is shown as deactivated.

The "Invisible" property is supported by the "Client Animation" functionality.

See also

These properties are available only when you select the "Preview: Support client animations and overlay of native elements" option in the Visualization Manager.

You can animate a movement, a rotation, and the visibility of a visualization element.

"Animation duration" Time that the element executes an animation (in milliseconds)
Example: 500

"Move to the foreground" Property value (BOOL)
TRUE: At runtime, the element is displayed in the foreground.
FALSE: At runtime, the element is displayed in the layer where it was inserted in the visualization editor.

See also

- Chapter 1.3.5.21.5 “Using client animation” on page 1962

Element property 'Input configuration'
The properties contain the configurations for the user input when using the mouse or keyboard. A user input defines an event and one or more actions that are executed when an event occurs.

The "Configure" button opens the "Input configuration" dialog box for creating or modifying a user input.

Configured user inputs are briefly listed below the events. They each include the action that is triggered and the setting in short form.

Example: "Execute ST code": # PLC_PRG.i_x := 0;

"OnDialogClosed" Input event: The user closes the dialog box.
"OnMouseClick" Input event: The user clicks the mouse button completely. The mouse button is clicked and released.
"OnMouseDown" Input event: The user clicks down on the mouse button.
"OnMouseEnter" Input event: The user drags the mouse pointer to the element.
"OnMouseLeave" Input event: The user drags the mouse pointer away from the element.
"OnMouseMove" Input event: The user moves the mouse pointer over the element area.
"OnMouseUp" Input event: The user releases the mouse button over the element area.
### "Tap"

When a mouse click event occurs, the variable defined in "Variable" is described in the application. The coding depends on the options "Tap FALSE" and "Tap on enter if captured".

### "Variable"

Variable (BOOL) that is set on mouse click.

**Example:** PLC_PRG.bIsTapped

- **TRUE:** A mouse click event exists. It lasts while the user presses the mouse button over the element. It ends when the button is released.
- **FALSE:** A mouse click event does not exist.

**Requirement:** The "Tap FALSE" option is not activated.

### "Tap FALSE"

- ✔️: The mouse click event leads to a complementary value in "Variable".
- **TRUE:** A mouse click event does not exist.
- **FALSE:** While the mouse click event exists.

### "Tap on enter if captured"

- ✔️: During user input, it is also taken into consideration whether the mouse pointer is dragged within the element area or not while the mouse button is pressed.
- **TRUE:** While the mouse click event exists and the mouse pointer is moved over the element area.
- **FALSE:** A mouse click event does not exist. Or the user moves the mouse pointer outside of the element area while the mouse button is pressed.

The value is **TRUE** again as soon as the user moves the pointer back to the element area. The mouse is then captured.

### "Shift"

With the onset of a mouse click event, the variable is set; when the mouse click event is completed, the variable is reset.

### "Variable"

Variable (BOOL). Its value toggled when the mouse click event is ended. This is when the user releases the mouse button while the mouse pointer is over the element area.

- If the user releases the mouse button while the mouse pointer is outside of the element area, then the mouse click event is not ended and the value is not toggled.

**Tip:** The user can cancel a started toggle input by dragging the mouse pointer out of the element area.

### "Toggle on up if captured"

- ✔️: The value toggles regardless of where the mouse pointer is when the mouse button is released. The mouse is then captured.

### "Keyboard shortcuts"

Keyboard shortcut on the element for triggering specific input actions.

When the keyboard shortcut event occurs, the input actions in the "Event(s)" property are triggered. In this way, it is not the input action itself that leads to this input action, but the mouse input action.

### "Key"

Key pressed for input action.

**Example:** [T]

**Note:** The following properties appear when a key is selected.
### Event(s)
- **None**: 
- **Mouse down**: Pressing the key triggers the input actions that are configured in the “OnMouseDown” property.
- **Mouse up**: Releasing the key triggers the input actions that are configured in the “OnMouseUp” property.
- **Mouse down/up**: Pressing and releasing the key triggers the input actions that are configured in the “OnMouseDown” property and the “OnMouseUp” property.

### Shift
- **Shift**: Combination with the Shift key
- Example: [Shift]+[T].

### Control
- **Control**: Combination with the Ctrl key
- Example: [Ctrl]+[T].

### Alt
- **Alt**: Combination with the Alt key
- Example: [Alt]+[T].

All keyboard shortcuts and their actions that are configured in the visualization are listed in the “Keyboard configuration” tab.

See also
- Chapter 1.3.5.19.2.2 “Command ‘Keyboard Configuration’” on page 1558
- Chapter 1.3.5.19.3.6 “Dialog ‘Input Configuration’” on page 1586

### Visualization element ‘Cartesian XY Chart’

#### Symbol:

![Cartesian XY Chart Symbol](image)

#### Tag: “Special controls”

The element displays the curve of array values graphically as a line or bar chart in the Cartesian coordinate system. The chart can display multiple curves at one time.

---

**NOTICE!**

**Restriction**
The element can be used with controller with V3.5 SP11 and later.

---

In the CODESYS Store, you will find a sample project for using XY chart elements in visualizations.
Element properties

| “Element name”               | Example: Velocity chart |
| “Type of element”            | “XY Chart”              |
| “Cartesian XY Chart”         | XYChart: Opens the “XY Chart Configuration” dialog. This is where the chart is configured. |

Element property 'Position'

The position defines the location and size of the element in the visualization window. These are based on the Cartesian coordinate system. The origin is located at the upper left corner of the window. The positive horizontal x-axis runs to the right. The positive vertical y-axis runs downwards.

| “X”                          | X coordinate of the upper left corner of the element Specified in pixels. Example: 10. |
| “Y”                          | Y coordinate of the upper left corner of the element Specified in pixels. Example: 10. |
| “Width”                      | Specified in pixels. Example: 150 |
| “Height”                     | Specified in pixels. Example: 30 |

You can also change the values by dragging the box symbols ( ) to other positions in the editor.

See also

- § Chapter 1.3.5.3.2 “Positioning elements, adapting size and layer” on page 1113

Element property 'Element look'

The properties contain fixed values for defining the look of the element.

| “Border line width”          | Value in pixels Example: 2 Note: The values 0 and 1 both result in a line weight of one pixel. If no line should be displayed, then the “Border line style” property must be set to the option “Invisible”. |
| “Frame line color”           | Style color from the drop-down list. Example: Black Fixed value that is selected in the color dialog. Example: 0; 0; 0 |
### Element property 'Axis font'

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Font</strong></td>
<td>Example: “Default”</td>
</tr>
</tbody>
</table>

- ![opens button]: Opens the “Font” dialog.
- 📦: Drop-down list with style fonts

### Element property 'Control variables'

#### Table 266: “Zoom”

- Zooming the displayed curve is done by means of the mouse, or the pinch gesture on a multitouch device. It also applies to all axes.

  - At runtime when “Enable” is TRUE, you can draw a box with the mouse by holding down the left mouse button. When you release the mouse button, the display zooms in on the box and the curve is magnified. To zoom in and out on a multitouch device, move two fingers together or away from each other, respectively.

  - Zooming and panning can work together.

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Enable</strong></td>
<td>Variable (BOOL) that enables or disables zooming.</td>
</tr>
</tbody>
</table>

  - TRUE: Enables zooming

  - Example: PLC_PRG.xZoomEnable

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Home</strong></td>
<td>Variable (BOOL).</td>
</tr>
</tbody>
</table>

  - Rising edge: Reset the displayed curve to the initial state after the display has changed due to zooming.

  - Example: PLC_PRG.xZoomHome

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Undo</strong></td>
<td>Variable (BOOL).</td>
</tr>
</tbody>
</table>

  - Rising edge: Reset the displayed curve to the previous position after the display has changed due to zooming.

  - Example: PLC_PRG.xZoomUndo

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Is zoomed</strong></td>
<td>Variable (BOOL) that indicates whether or not the displayed curve was modified due to zooming.</td>
</tr>
</tbody>
</table>

  - TRUE: Curve setting was zoomed.

  - Example: PLC_PRG.xIsZoomed

#### Table 267: “Pan”

- Panning the displayed curve is done by means of the mouse or the pinch gesture on a multitouch device. It also applies to all axes.

  - At runtime if “Enable” is TRUE, then you can drag the displayed curve to another position by holding down the left mouse button. To pan the displayed curve on a multitouch device, drag it with one finger to another position.

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Enable</strong></td>
<td>Variable (BOOL) to enable or disable panning.</td>
</tr>
</tbody>
</table>

  - TRUE: Enables panning

  - Example: PLC_PRG.xPanEnable
### "Home"

**Variable (BOOL).**
Rising edge: Reset the displayed curve to the initial position after the display has changed due to panning.

**Example:** PLC_PRG.xPanHome

### "Is panned"

**Variable (BOOL) whose state indicates whether or not the displayed curve was modified due to zooming.**

**TRUE:** Curve setting was panned.

**Example:** PLC_PRG.xIsPanned

---

**Element property 'Absolute movement'**

The properties contain IEC variables for controlling the position of the element dynamically. The reference point is the upper left corner of the element. In runtime mode, the entire element is moved.

---

### "Movement"

| **"X"**   | Variable (numeric data type). Defines the X position (in pixels).
|-----------|--------------------------------------------------------------------------
| Example:  | PLC_PRG.iPos_X.                                                           |
|           | Increasing this value in runtime mode moves the element to the right.     |
| **"Y"**   | Variable (numeric data type). Defines the Y position (in pixels).
| Example:  | PLC_PRG.iPos_Y.                                                           |
|           | Increasing this value in runtime mode moves the element downwards.         |

### "Rotation"

**Variable (numeric data type). Defines the angle of rotation (in degrees).**

**Example:** PLC_PRG.iAngle1.

The midpoint of the element rotates at the "Center" point. This rotation point is shown as the symbol.

In runtime mode, the alignment of the element remains the same with respect to the coordinate system of the visualization. Increasing the value rotates the element to the right.

### "Interior rotation"

**Variable (numeric data type). Defines the angle of rotation (in degrees).**

**Example:** PLC_PRG.iAngle2.

In runtime mode, the element rotates about the point of rotation specified in "Center" according to the value of the variable. In addition, the alignment of the element rotates according to the coordinate system of the visualization. Increasing the value in the code rotates clockwise.

The rotation point is shown as the symbol.

**Note:** If a static angle of rotation is specified in the "Position ➔ Angle" property, then the static angle of rotation is added to the variable angle of rotation (offset) when the visualization is executed.
You can link the variables to a unit conversion.

The “X”, “Y”, “Rotation”, and “Interior rotation” properties are supported by the “Client Animation” functionality.

See also
- Unit conversion

## Element property 'State variables'

The variables control the element behavior dynamically.

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>“Invisible”</strong></td>
<td>Variable (BOOL). Toggles the visibility of the element.</td>
</tr>
<tr>
<td></td>
<td>TRUE: The element is not visible at runtime.</td>
</tr>
<tr>
<td><strong>“Deactivate inputs”</strong></td>
<td>Variable (BOOL). Toggles the operability of the element.</td>
</tr>
<tr>
<td></td>
<td>TRUE: User inputs do not have any effect in runtime more. The element is shown as deactivated.</td>
</tr>
</tbody>
</table>

The “Invisible” property is supported by the "Client Animation" functionality.

See also
- 

These properties are available only when you select the “Preview: Support client animations and overlay of native elements” option in the Visualization Manager.

You can animate a movement, a rotation, and the visibility of a visualization element.

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>“Animation duration”</strong></td>
<td>Time that the element executes an animation (in milliseconds)</td>
</tr>
<tr>
<td>Example: 500</td>
<td></td>
</tr>
<tr>
<td><strong>“Move to the foreground”</strong></td>
<td>Property value (BOOL)</td>
</tr>
<tr>
<td></td>
<td>TRUE: At runtime, the element is displayed in the foreground.</td>
</tr>
<tr>
<td></td>
<td>FALSE: At runtime, the element is displayed in the layer where it was inserted in the visualization editor.</td>
</tr>
</tbody>
</table>

See also
- % Chapter 1.3.5.21.5 “Using client animation” on page 1962

## Element property 'Access rights'

Requirement: User management is set up for the visualization.
“Access rights” opens the “Access rights” dialog. There you can edit the access privileges for the element.

Status messages:
- “Not set. Full rights.”: Access rights for all user groups: “operable”
- “Rights are set: Limited rights”: Access is restricted for at least one group.

See also
- Chapter 1.3.5.19.3.1 “Dialog ‘Access Rights’” on page 1583

Visualisation element ‘Date range picker’

Symbol:

Tag: “Date/time controls”

The element provides the capability of selecting the date and time range of a saved data set. The element is used with the “Trend” visualization element.

Element properties

<table>
<thead>
<tr>
<th>“Element name”</th>
<th>Example: DateTrend1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Option</td>
<td></td>
</tr>
<tr>
<td>Hint: Assign individual names for elements so that they are found faster in the element list.</td>
<td></td>
</tr>
</tbody>
</table>

| “Type of element” | “Date range picker” |

Element property ‘Position’

The position defines the location and size of the element in the visualization window. These are based on the Cartesian coordinate system. The origin is located at the upper left corner of the window. The positive horizontal x-axis runs to the right. The positive vertical y-axis runs downwards.

<table>
<thead>
<tr>
<th>“X”</th>
<th>X coordinate of the upper left corner of the element</th>
</tr>
</thead>
<tbody>
<tr>
<td>Specified in pixels</td>
<td>Example: 10.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>“Y”</th>
<th>Y coordinate of the upper left corner of the element</th>
</tr>
</thead>
<tbody>
<tr>
<td>Specified in pixels</td>
<td>Example: 10.</td>
</tr>
</tbody>
</table>
"Width"  Specified in pixels.
Example: 150

"Height"  Specified in pixels.
Example: 30

You can also change the values by dragging the box symbols (_drag) to other positions in the editor.

See also
●  Chapter 1.3.5.3.2 “Positioning elements, adapting size and layer” on page 1113

Element property 'Center'
The properties contain fixed values for the coordinates of the point of rotation. This point of rotation is shown as the symbol. The point is used as the center for rotating and scaling.

"X"  X-coordinate of the point of rotation

"Y"  Y-coordinate of the point of rotation

You can also change the values by dragging the symbols (drag) to other positions in the editor.

"Show frame"  : The visualization element is displayed with a frame.

"Resolution"  Resolution saved for the time stamp: "Millisecond" or "Microsecond"

"Attached element instance"  The element can be assigned to a "Trend" visualization element. In this way, the time range of the trend element can be changed. The available visual elements are selected with the help of the input assistant (select).

Element property 'Tick mark labels'

"Two-line labelling"  : The time stamps are displayed in two lines. The date is displayed in the first line and the time is displayed in the second line.
☐: Time stamp is displayed in one line. The date and time can also be displayed in one line depending on the formatting.

"Omit irrelevant information in time stamp"  : The time stamp has a shorter form. For example, the date is displayed only for the first tick mark, and only the time for the following tick marks. The settings in "Internationalization (format strings)" are ignored for this setting.
☐: All information is displayed for all time stamps.

"Internationalization (format strings)"  Only active when the parameter "Omit irrelevant information in timestamps" is deactivated.

"Date"  Definition of the date format. The default setting is taken from the Windows control panel.

"Time"  Definition of the time format. The default setting is taken from the Windows control panel.
Element property 'Text properties'  The properties contain fixed values for the text properties.

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Horizontal alignment&quot;</td>
<td>Horizontal alignment of the text within the element.</td>
</tr>
<tr>
<td>&quot;Vertical alignment&quot;</td>
<td>Vertical alignment of the text within the element.</td>
</tr>
<tr>
<td>&quot;Font&quot;</td>
<td>Example: &quot;Default&quot;&lt;br&gt;촉: The “Font” dialog box opens.</td>
</tr>
<tr>
<td>&quot;Font color&quot;</td>
<td>Example: &quot;Black&quot;&lt;br&gt;촉: The “Color” dialog box opens.</td>
</tr>
<tr>
<td>&quot;Transparency&quot;</td>
<td>Whole number (value range from 0 to 255). This determines the transparency of the respective color. Example: 255: The color is opaque. 0: The color is completely transparent. Please note: If the color is a style color and already has a transparency value, then this property is write-protected.</td>
</tr>
</tbody>
</table>

Element property 'Additional buttons'  The properties contain IEC variables for controlling the position of the element dynamically. The reference point is the upper left corner of the element. In runtime mode, the entire element is moved.

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Jump to the largest possible time stamp&quot;</td>
<td>총: An additional button ( sensit) is displayed for jumping to the last time stamp.</td>
</tr>
<tr>
<td>&quot;Jump to the smallest possible time stamp&quot;</td>
<td>총: An additional button ( sensit) is displayed for jumping to the first time stamp.</td>
</tr>
<tr>
<td>&quot;Zoom out&quot;</td>
<td>총: An additional button ( sensit) is displayed for setting the current min./max. range to the maximum range. The selected range is left.</td>
</tr>
</tbody>
</table>

Element property 'Absolute movement'  The properties contain IEC variables for controlling the position of the element dynamically. The reference point is the upper left corner of the element. In runtime mode, the entire element is moved.

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Movement&quot;</td>
<td>Variable (numeric data type). Defines the X position (in pixels). Example: PLC_PRG.iPos_X. Increasing this value in runtime mode moves the element to the right.</td>
</tr>
<tr>
<td>&quot;Y&quot;</td>
<td>Variable (numeric data type). Defines the Y position (in pixels). Example: PLC_PRG.iPos_Y. Increasing this value in runtime mode moves the element downwards.</td>
</tr>
</tbody>
</table>
### Rotation

Variable (numeric data type). Defines the angle of rotation (in degrees).

**Example:** PLC_PRG.iAngle1.

The midpoint of the element rotates at the “Center” point. This rotation point is shown as the symbol.

In runtime mode, the alignment of the element remains the same with respect to the coordinate system of the visualization. Increasing the value rotates the element to the right.

### Interior rotation

Variable (numeric data type). Defines the angle of rotation (in degrees).

**Example:** PLC_PRG.iAngle2.

In runtime mode, the element rotates about the point of rotation specified in “Center” according to the value of the variable. In addition, the alignment of the element rotates according to the coordinate system of the visualization. Increasing the value in the code rotates clockwise.

The rotation point is shown as the symbol.

Note: If a static angle of rotation is specified in the “Position Angle” property, then the static angle of rotation is added to the variable angle of rotation (offset) when the visualization is executed.

---

You can link the variables to a unit conversion.

The “X”, “Y”, “Rotation”, and “Interior rotation” properties are supported by the “Client Animation” functionality.

See also

- Unit conversion

---

**Element property 'State variables'**

The variables control the element behavior dynamically.

<table>
<thead>
<tr>
<th>&quot;Invisible&quot;</th>
</tr>
</thead>
<tbody>
<tr>
<td>Variable (BOOL). Toggles the visibility of the element.</td>
</tr>
<tr>
<td>TRUE: The element is not visible at runtime.</td>
</tr>
</tbody>
</table>

The “Invisible” property is supported by the “Client Animation” functionality.
These properties are available only when you select the “Preview: Support client animations and overlay of native elements” option in the Visualization Manager.

You can animate a movement, a rotation, and the visibility of a visualization element.

<table>
<thead>
<tr>
<th>“Animation duration”</th>
<th>Time that the element executes an animation (in milliseconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Example: 500</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>“Move to the foreground”</th>
<th>Property value (BOOL)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>TRUE: At runtime, the element is displayed in the foreground.</td>
</tr>
<tr>
<td></td>
<td>FALSE: At runtime, the element is displayed in the layer where it was inserted in the visualization editor.</td>
</tr>
</tbody>
</table>

Requirement: User management is set up for the visualization.

<table>
<thead>
<tr>
<th>“Access rights”</th>
<th>Opens the “Access rights” dialog. There you can edit the access privileges for the element.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Status messages:</td>
</tr>
<tr>
<td></td>
<td>• “Not set. Full rights.”: Access rights for all user groups: “operable”</td>
</tr>
<tr>
<td></td>
<td>• “Rights are set: Limited rights”: Access is restricted for at least one group.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>“Element name”</th>
<th>Optional</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Hint: Assign individual names for elements so that they are found faster in the element list.</td>
</tr>
<tr>
<td></td>
<td>Example: TimeRangeTemperature</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>“Type of element”</th>
<th>“Time range picker”</th>
</tr>
</thead>
</table>

See also

- § Chapter 1.3.5.21.5 “Using client animation” on page 1962

Visualization element 'Time range picker'

Symbol: 🕒

Tag: “Date/time controls”

The element provides configurable buttons for setting the time range of a trend display to a defined time. In the process the end time of the previous display is left unchanged and the start time is adapted.
**Element property 'Position'**

The position defines the location and size of the element in the visualization window. These are based on the Cartesian coordinate system. The origin is located at the upper left corner of the window. The positive horizontal x-axis runs to the right. The positive vertical y-axis runs downwards.

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>X coordinate of the upper left corner of the element. Specified in pixels. Example: 10.</td>
</tr>
<tr>
<td>Y</td>
<td>Y coordinate of the upper left corner of the element. Specified in pixels. Example: 10.</td>
</tr>
<tr>
<td>Width</td>
<td>Specified in pixels. Example: 150</td>
</tr>
<tr>
<td>Height</td>
<td>Specified in pixels. Example: 30</td>
</tr>
</tbody>
</table>

You can also change the values by dragging the box symbols (_drag_•_•_•_) to other positions in the editor.

See also
- Chapter 1.3.5.3.2 “Positioning elements, adapting size and layer” on page 1113

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Orientation</td>
<td>Specifies whether the time picker element is aligned horizontally or vertically in the editor. Hint: Change the width to height ratio of the element in the editor.</td>
</tr>
<tr>
<td>Show frame</td>
<td><em>✓</em>: The visualization element is drawn with a frame.</td>
</tr>
<tr>
<td>Resolution</td>
<td>Resolution saved for the time stamp: “Millisecond” or “Microsecond”</td>
</tr>
<tr>
<td>Attached element instance</td>
<td>Assignment to the element that processes the time picker. The element can be assigned for example to a “Trend” visualization element. Then the time range of the trend element can be changed. The available visual elements are selected with the help of the input assistance (input assistance). Example: GenElemInst_1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Element property 'Texts'</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Text</td>
<td>String label for the element. Example: Zoom</td>
</tr>
</tbody>
</table>

**Element property 'Text properties'**

The properties contain fixed values for the text properties.
<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
</table>
| **“Font”** | Example: “Default”  
- The “Font” dialog box opens.  
- Drop-down list with style fonts. |
| **“Font color”** | Example: “Black”  
- The “Color” dialog box opens.  
- Drop-down list with style colors. |
| **“Transparency”** | Whole number (value range from 0 to 255). This determines the transparency of the respective color.  
Example: 255: The color is opaque.  
0: The color is completely transparent.  
Please note: If the color is a style color and already has a transparency value, then this property is write-protected. |

**Property ’Times’**  
In “Times”, the buttons that the element provides at runtime are defined and configured in an array.

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>“Provide &quot;All&quot; selection”</strong></td>
<td>☑: Time selector bar extended by &quot;All&quot; button. The diagram represents a time interval that covers all time stamps.</td>
</tr>
<tr>
<td><strong>“Times”</strong></td>
<td>☑: Adds another button to the time selection bar and increases the array by one entry. An additional index is present in the property “Times Æ Times Æ [&lt;new&gt;]”. “Time” is located under this index. The configuration of the button is to be entered there.</td>
</tr>
</tbody>
</table>
| **“Times”** | Array of all buttons in the time selection bar. Index corresponds to the number of buttons.  
- “[Index]”  
- with index € {0, 1, 2,...}  
- The associated button is removed from the time picker bar. The configuration entry is deleted from the “Times” property list. |
| **“[Index]”** | ☑: Time interval in standardized notation. Example: 3M for 3 months; 30m for 30 minutes. If a time interval is indicated in the field, then the button is labelled with it. If a user clicks on the button at runtime, the command is executed to switch the diagram to this time interval. The default is empty. |

**Element property ’Control variables’**

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
</table>
| **“Time”** | Displays which time is currently selected.  
Variable (STRING)  
Example: PLC_PRG.strSelectedTime |
| **“‘All’ selected”** | Displays the state of the “All” button  
Variable (BOOL)  
Example: PLC_PRG.AllTimesAreSelected |

**Element property ’Absolute movement’**  
The properties contain IEC variables for controlling the position of the element dynamically. The reference point is the upper left corner of the element. In runtime mode, the entire element is moved.
### Movement

<table>
<thead>
<tr>
<th>Variable</th>
<th>Definition</th>
<th>Example</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>X</strong></td>
<td>Variable (numeric data type). Defines the X position (in pixels).&lt;br&gt;Example: PLC_PRG.iPos_X. Increasing this value in runtime mode moves the element to the right.</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Y</strong></td>
<td>Variable (numeric data type). Defines the Y position (in pixels).&lt;br&gt;Example: PLC_PRG.iPos_Y. Increasing this value in runtime mode moves the element downwards.</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Rotation</strong></td>
<td>Variable (numeric data type). Defines the angle of rotation (in degrees).&lt;br&gt;Example: PLC_PRG.iAngle1. The midpoint of the element rotates at the “Center” point. This rotation point is shown as the symbol. In runtime mode, the alignment of the element remains the same with respect to the coordinate system of the visualization. Increasing the value rotates the element to the right.</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Interior rotation</strong></td>
<td>Variable (numeric data type). Defines the angle of rotation (in degrees).&lt;br&gt;Example: PLC_PRG.iAngle2. In runtime mode, the element rotates about the point of rotation specified in “Center” according to the value of the variable. In addition, the alignment of the element rotates according to the coordinate system of the visualization. Increasing the value in the code rotates clockwise. The rotation point is shown as the symbol. Note: If a static angle of rotation is specified in the “Position Angle” property, then the static angle of rotation is added to the variable angle of rotation (offset) when the visualization is executed.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

You can link the variables to a unit conversion.

The “X”, “Y”, “Rotation”, and “Interior rotation” properties are supported by the “Client Animation” functionality.

See also
- [Unit conversion](#)
Element property 'State variables'

The variables control the element behavior dynamically.

<table>
<thead>
<tr>
<th>Variable (BOOL)</th>
<th>Toggles the visibility of the element.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>&quot;Invisible&quot;</strong></td>
<td>TRUE: The element is not visible at runtime.</td>
</tr>
</tbody>
</table>

The "Invisible" property is supported by the "Client Animation" functionality.

See also

* These properties are available only when you select the “Preview: Support client animations and overlay of native elements” option in the Visualization Manager.
You can animate a movement, a rotation, and the visibility of a visualization element.

<table>
<thead>
<tr>
<th>Time that the element executes an animation (in milliseconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>&quot;Animation duration&quot;</strong></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Property value (BOOL)</th>
<th>At runtime, the element is displayed in the foreground.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>&quot;Move to the foreground&quot;</strong></td>
<td>FALSE: At runtime, the element is displayed in the layer where it was inserted in the visualization editor.</td>
</tr>
</tbody>
</table>

See also

* Chapter 1.3.5.21.5 “Using client animation” on page 1962

Element property 'Access rights'

Requirement: User management is set up for the visualization.

<table>
<thead>
<tr>
<th>Opens the “Access rights” dialog. There you can edit the access privileges for the element.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>&quot;Access rights&quot;</strong></td>
</tr>
</tbody>
</table>

Status messages:

* "Not set. Full rights.": Access rights for all user groups: "operable"
* "Rights are set: Limited rights": Access is restricted for at least one group.

See also

* Chapter 1.3.5.19.3.1 “Dialog 'Access Rights'” on page 1583

Visualization element 'Date picker'

Symbol:

Tag: “Date/time controls”
The element is a calendar that displays the current date. A user can click a tag to select a date, which is saved to a variable. In addition, it can customize the time interval that the calendar displays. Clicking the calendar header changes the year. Clicking the arrows in the calendar header changes the month.

The element contains language-dependent texts that are managed in the System text list. This deals with the names of the month and the days of the week written out completely or abbreviated. When the date picker is added to a visualization, CODESYS generates the text list automatically below the POU view. The IDs correspond to the default text and therefore English terms. The text list makes it possible to translate these texts.

### System text list

<table>
<thead>
<tr>
<th>ID</th>
<th>Standard</th>
</tr>
</thead>
<tbody>
<tr>
<td>Apr</td>
<td>Apr</td>
</tr>
<tr>
<td>April</td>
<td>April</td>
</tr>
</tbody>
</table>

See also

- [Chapter 1.3.5.6 “Setting up multiple languages” on page 1143](#)

### Element properties

**“Element name”** | Example: DueDateCalendar
---|---

**“Type of element”** | “Date picker”

### Element property 'Position'

The position defines the location and size of the element in the visualization window. These are based on the Cartesian coordinate system. The origin is located at the upper left corner of the window. The positive horizontal x-axis runs to the right. The positive vertical y-axis runs downwards.

<table>
<thead>
<tr>
<th>“X”</th>
<th>X coordinate of the upper left corner of the element</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Specified in pixels.</td>
</tr>
<tr>
<td></td>
<td>Example: 10.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>“Y”</th>
<th>Y coordinate of the upper left corner of the element</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Specified in pixels.</td>
</tr>
<tr>
<td></td>
<td>Example: 10.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>“Width”</th>
<th>Specified in pixels.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Example: 150</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>“Height”</th>
<th>Specified in pixels.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Example: 30</td>
</tr>
</tbody>
</table>

You can also change the values by dragging the box symbols (_drag) to other positions in the editor.
See also

- Chapter 1.3.5.3.2 “Positioning elements, adapting size and layer” on page 1113

**Element property 'Center'**

The properties contain fixed values for the coordinates of the point of rotation. This point of rotation is shown as the symbol. The point is used as the center for rotating and scaling.

| “X” | X-coordinate of the point of rotation |
| “Y” | Y-coordinate of the point of rotation |

You can also change the values by dragging the symbols (_symbol) to other positions in the editor.

| “Variable” | Input variable (DATE). Contains the date that a user selects in the calendar. Example: PLC_PRG.dtDueDate |
| “Design” | - “From style”: All settings are preconfigured according to the style.  
- “Explicit”: The “Design settings” property appears. Here you can customize the calendar. |

**Design settings**

Requirement: This property is visible only if the “Design” property is set to “Explicit”.

The values of the property can be predefined in the style. Then they are available in the drop-down list.

Table 268: “Header of date picker”

<table>
<thead>
<tr>
<th>Design of the header</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>“Font”</td>
<td>Style font or user-defined font</td>
</tr>
<tr>
<td>“Font color”</td>
<td>Style color or user-defined color</td>
</tr>
<tr>
<td>“Arrows”</td>
<td></td>
</tr>
<tr>
<td>“Arrow color”</td>
<td>Style color or user-defined color</td>
</tr>
<tr>
<td>“Color of printed arrow”</td>
<td></td>
</tr>
<tr>
<td>“Background”</td>
<td></td>
</tr>
<tr>
<td>“Draw background”</td>
<td>“From style”: The style defines whether and how a background is drawn. “Yes”: The background is filled with the color in the “Background color” property. “No”: The background is not filled with a color.</td>
</tr>
<tr>
<td>“Fill color”</td>
<td>Style color or user-defined color</td>
</tr>
</tbody>
</table>

Table 269: Design of the main display area

<table>
<thead>
<tr>
<th>Design of the main display area</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>“Today”</td>
<td>Design of today</td>
</tr>
<tr>
<td>“Font”</td>
<td>Style font or user-defined font</td>
</tr>
<tr>
<td>“Font color”</td>
<td>Style color or user-defined color</td>
</tr>
<tr>
<td>Setting</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
</tbody>
</table>
| **Draw background** | "From style": The style defines whether and which background is drawn.  
|                 | "Yes": The background is filled with the color in the "Background color" property.  
|                 | "No": The background is not filled with a color.                           |
| **Background color** | Style color or user-defined color. Used if "Yes" is selected in "Draw background". |
| **Show frame**  | "From style": The style defines whether and how a frame is drawn.          
|                 | "Yes": The frame is displayed with the following properties.                
|                 | "No": A frame is not displayed.                                            |
| **Frame color** | Used if "Yes" is selected in "Show frame".                                |
| **Rectangle type** |                                                                 |
| **Line width**  |                                                                 |

<table>
<thead>
<tr>
<th>Setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Selected day</strong></td>
<td>Design of the selected day</td>
</tr>
<tr>
<td><strong>Font</strong></td>
<td>Style font or user-defined font</td>
</tr>
<tr>
<td><strong>Font color</strong></td>
<td>Style color or user-defined color</td>
</tr>
</tbody>
</table>
| **Draw background** | "From style": The style defines whether and how a background is drawn.  
|                 | "Yes": The background is filled with the color in the "Background color" property.  
|                 | "No": The background is not filled with a color.                           |
| **Background color** | Style color or user-defined color                                           |
| **Show frame**  | "From style": The style defines whether and how a background is drawn.  
|                 | "Yes": The frame is displayed with the following properties.                
|                 | "No": A frame is not displayed.                                            |
| **Frame color** | Used if "Yes" is selected in "Show frame".                                |
| **Rectangle type** |                                                                 |
| **Line width**  |                                                                 |

<table>
<thead>
<tr>
<th>Setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Current month</strong></td>
<td>Design of the current month</td>
</tr>
<tr>
<td><strong>Font</strong></td>
<td>Style font or user-defined font</td>
</tr>
<tr>
<td><strong>Font color</strong></td>
<td>Style color or user-defined color</td>
</tr>
</tbody>
</table>
| **Draw background** | "From style": The style defines whether and how a background is drawn.  
|                 | "Yes": The background is filled with the color in the "Background color" property.  
|                 | "No": The background is not filled with a color.                           |
| **Background color** | Style color or user-defined color                                           |
| **Show frame**  | "From style": The style defines whether and how a background is drawn.  
|                 | "Yes": The frame is displayed with the following properties.                
|                 | "No": A frame is not displayed.                                            |
| **Frame color** | Used if "Yes" is selected in "Show frame".                                |
| **Rectangle type** |                                                                 |
| **Line width**  |                                                                 |
### Other months
Design of the previous and subsequent months

### Font
Style font or user-defined font

### Font color
Style color or user-defined color

### Display other month
Design of the previous and subsequent months

### Draw background
*From style*: The style defines whether and how a background is drawn.  
*Yes*: The background is filled with the color in the “Background color” property.  
*No*: The background is not filled with a color.

### Background color

### Show frame
*From style*: The style defines whether and how a frame is drawn.  
*Yes*: The frame is displayed with the following properties.  
*No*: A frame is not displayed.

### Frame color
Used if “Yes” is selected in “Show frame”.

### Rectangle type

### Line width

### Day of week heading
Design of the heading with the days of the week

### Font
Style font or user-defined font

### Font color
Style color or user-defined color

### Draw background
*From style*: The style defines whether and how a background is drawn.  
*Yes*: The background is filled with the style color “From style”. The style defines whether and how a background is drawn.  
*No*: The background is not filled with a color.

### Background color

### Show frame
*From style*: The style defines whether and how a frame is drawn.  
*Yes*: The frame is displayed with the following properties.  
*No*: A frame is not displayed.

### Frame color
Used if “Yes” is selected in “Show frame”.

### Rectangle type

### Line width

### Display separator line
*From style*: The style defines whether and how a separator line is drawn.  
*Yes*: Display with the following properties.  
*No*: A separator line is not displayed.

### Color of the separator line
Used if “Yes” is selected in “Display separator line”.

### Width of separator line

### Background
Design of the calendar days

### Draw background
*From style*: The style defines whether and how a background is drawn.  
*Yes*: The background is filled with the color in the “Fill color” property and framed in the “Frame color”.  
*No*: The background is not filled with a color.
<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Fill color&quot;</td>
<td>Style color or user-defined color</td>
</tr>
<tr>
<td>&quot;Frame color&quot;</td>
<td></td>
</tr>
</tbody>
</table>

Element property 'Display type'

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Rows&quot;</td>
<td>Number of month calendars per row (preset: 1)</td>
</tr>
<tr>
<td>&quot;Columns&quot;</td>
<td>Number of month calendars per column (preset: 1)</td>
</tr>
</tbody>
</table>

Element property 'Absolute movement'

The properties contain IEC variables for controlling the position of the element dynamically. The reference point is the upper left corner of the element. In runtime mode, the entire element is moved.

"Movement"

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;X&quot;</td>
<td>Variable (numeric data type). Defines the X position (in pixels).</td>
</tr>
<tr>
<td></td>
<td>Example: PLC_PRG.iPos_X</td>
</tr>
<tr>
<td></td>
<td>Increasing this value in runtime mode moves the element to the right.</td>
</tr>
<tr>
<td>&quot;Y&quot;</td>
<td>Variable (numeric data type). Defines the Y position (in pixels).</td>
</tr>
<tr>
<td></td>
<td>Example: PLC_PRG.iPos_Y</td>
</tr>
<tr>
<td></td>
<td>Increasing this value in runtime mode moves the element downwards.</td>
</tr>
<tr>
<td>&quot;Rotation&quot;</td>
<td>Variable (numeric data type). Defines the angle of rotation (in degrees).</td>
</tr>
<tr>
<td></td>
<td>Example: PLC_PRG.iAngle1</td>
</tr>
<tr>
<td></td>
<td>The midpoint of the element rotates at the &quot;Center&quot; point. This rotation point is shown as the symbol.</td>
</tr>
<tr>
<td></td>
<td>In runtime mode, the alignment of the element remains the same with respect to the coordinate system of the visualization. Increasing the value rotates the element to the right.</td>
</tr>
<tr>
<td>&quot;Interior rotation&quot;</td>
<td>Variable (numeric data type). Defines the angle of rotation (in degrees).</td>
</tr>
<tr>
<td></td>
<td>Example: PLC_PRG.iAngle2</td>
</tr>
<tr>
<td></td>
<td>In runtime mode, the element rotates about the point of rotation specified in &quot;Center&quot; according to the value of the variable. In addition, the alignment of the element rotates according to the coordinate system of the visualization. Increasing the value in the code rotates clockwise.</td>
</tr>
<tr>
<td></td>
<td>The rotation point is shown as the symbol.</td>
</tr>
<tr>
<td></td>
<td>Note: If a static angle of rotation is specified in the &quot;Position ➔ Angle&quot; property, then the static angle of rotation is added to the variable angle of rotation (offset) when the visualization is executed.</td>
</tr>
</tbody>
</table>
You can link the variables to a unit conversion.

The "X", "Y", "Rotation", and "Interior rotation" properties are supported by the "Client Animation" functionality.

See also
- Unit conversion

Element property ‘State variables’

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Invisible”</td>
<td>Variable (BOOL). Toggles the visibility of the element. TRUE: The element is not visible at runtime.</td>
</tr>
<tr>
<td>“Deactivate inputs”</td>
<td>Variable (BOOL). Toggles the operability of the element. TRUE: User inputs do not have any effect in runtime more. The element is shown as deactivated.</td>
</tr>
</tbody>
</table>

The “Invisible” property is supported by the "Client Animation" functionality.

See also
- 

These properties are available only when you select the “Preview: Support client animations and overlay of native elements” option in the Visualization Manager.

You can animate a movement, a rotation, and the visibility of a visualization element.

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Animation duration”</td>
<td>Time that the element executes an animation (in milliseconds) Example: 500</td>
</tr>
<tr>
<td>“Move to the foreground”</td>
<td>Property value (BOOL) TRUE: At runtime, the element is displayed in the foreground. FALSE: At runtime, the element is displayed in the layer where it was inserted in the visualization editor.</td>
</tr>
</tbody>
</table>

See also
- Chapter 1.3.5.21.5 “Using client animation” on page 1962

Element property ‘Access rights’

Requirement: User management is set up for the visualization.
“Access rights” Opens the “Access rights” dialog. There you can edit the access privileges for the element.

Status messages:
- “Not set. Full rights.”: Access rights for all user groups: “operable”
- “Rights are set: Limited rights”: Access is restricted for at least one group.

See also
- Chapter 1.3.5.19.3.1 “Dialog ‘Access Rights’” on page 1583

Visualization Element ‘Analog clock’

Symbol:

Tag: “Date/time controls”
The element is a clock that displays the current time of day. The clock can also display any time.

Element properties

<table>
<thead>
<tr>
<th>“Element name”</th>
<th>Example: GenElemInst_1</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Type of element”</td>
<td>“Analog clock”</td>
</tr>
</tbody>
</table>

Element property 'Position' The position defines the location and size of the element in the visualization window. These are based on the Cartesian coordinate system. The origin is located at the upper left corner of the window. The positive horizontal x-axis runs to the right. The positive vertical y-axis runs downwards.

<table>
<thead>
<tr>
<th>“X”</th>
<th>X coordinate of the upper left corner of the element</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Specified in pixels.</td>
</tr>
<tr>
<td>Example: 10</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>“Y”</th>
<th>Y coordinate of the upper left corner of the element</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Specified in pixels.</td>
</tr>
<tr>
<td>Example: 10</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>“Width”</th>
<th>Specified in pixels.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example: 150</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>“Height”</th>
<th>Specified in pixels.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example: 30</td>
<td></td>
</tr>
</tbody>
</table>
You can also change the values by dragging the box symbols (ற) to other positions in the editor.

See also
- Chapter 1.3.5.3.2 “Positioning elements, adapting size and layer” on page 1113

**Element property 'Center'**
The properties contain fixed values for the coordinates of the point of rotation. This point of rotation is shown as the symbol. The point is used as the center for rotating and scaling.

<table>
<thead>
<tr>
<th>“X”</th>
<th>X-coordinate of the point of rotation</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Y”</td>
<td>Y-coordinate of the point of rotation</td>
</tr>
</tbody>
</table>

You can also change the values by dragging the symbols (❑) to other positions in the editor.

**Element property 'Time Display'**

| “Use system time” | ☑: The system time of the PLC is displayed. |
| “Variable” | Variable (time data type TOD, TIME_OF_DAY). This receives the time of day that is not the system time. Example: PLC_PRG.todTimeTokio |

Requirement: The "Use system time" property is not activated.

See also
- Time/date type TOD

| “Design” | ● “From style”: All settings are preconfigured according to the style. ● “Explicit”: The “Settings” property appears. Here you can customize the analog clock. |

**Element property 'Absolute movement'**
The properties contain IEC variables for controlling the position of the element dynamically. The reference point is the upper left corner of the element. In runtime mode, the entire element is moved.

| “Movement” | | |
| “X” | Variable (numeric data type). Defines the X position (in pixels). Example: PLC_PRG.iPos_X. Increasing this value in runtime mode moves the element to the right. |
| “Y” | Variable (numeric data type). Defines the Y position (in pixels). Example: PLC_PRG.iPos_Y. Increasing this value in runtime mode moves the element downwards. |
### "Rotation"

Variable (numeric data type). Defines the angle of rotation (in degrees).

**Example:** PLC_PRG.iAngle1.

The midpoint of the element rotates at the "Center" point. This rotation point is shown as the symbol. In runtime mode, the alignment of the element remains the same with respect to the coordinate system of the visualization. Increasing the value rotates the element to the right.

### "Scaling"

Variable (integer data type). Causes centric stretching.

**Example:** PLC_PRG.iScaling.

The reference point is the "Center" property.

The value 1 shrinks the element by a factor of 0.001. The value 1000 returns the element to its original size.

### "Interior rotation"

Variable (numeric data type). Defines the angle of rotation (in degrees).

**Example:** PLC_PRG.iAngle2.

In runtime mode, the element rotates about the point of rotation specified in "Center" according to the value of the variable. In addition, the alignment of the element rotates according to the coordinate system of the visualization. Increasing the value in the code rotates clockwise.

The rotation point is shown as the symbol.

Note: If a static angle of rotation is specified in the property "Position ➝ Angle", then the static angle of rotation is added to the variable angle of rotation (offset) when the visualization is executed.

### "Use REAL values"

Note: Only available if the device supports the use of REAL coordinates.

- The properties of the absolute movement are interpreted as REAL values. The values are not rounded.

- The option allows for the individual fine-tuning of drawing the element, for example for the visualization of a smoother rotation.

- Hint: If a horizontal or vertical line is drawn blurry on a specific visualization platform, then this can be corrected by an offset of 0.5px in the direction of the line thickness.

---

You can link the variables to a unit conversion.
The properties “X”, “Y”, “Rotation”, and “Interior rotation” are supported by the "Client Animation" functionality.

See also

- Unit conversion

Element property 'Settings'

Requirement: The “Property” is “Explicit”. Only then is the “Clock Settings” category visible.

Table 270: “Background”

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Background color”</td>
<td>Color variants of the default background image</td>
</tr>
<tr>
<td>“Yellow”</td>
<td></td>
</tr>
<tr>
<td>“Red”</td>
<td></td>
</tr>
<tr>
<td>“Blue”</td>
<td></td>
</tr>
<tr>
<td>“Green”</td>
<td></td>
</tr>
<tr>
<td>“Black”</td>
<td></td>
</tr>
<tr>
<td>“Own background”</td>
<td>Background display with the specific “Image”. Replaces the default background image.</td>
</tr>
<tr>
<td>“Image”</td>
<td>Image from an image pool or library</td>
</tr>
<tr>
<td>“Transparency color”</td>
<td>The transparent color in the image representation.</td>
</tr>
<tr>
<td>“Use background color”</td>
<td>The image background is displayed using the color defined in the “Background color” property.</td>
</tr>
<tr>
<td>“Background color”</td>
<td>Style color or color</td>
</tr>
</tbody>
</table>

Table 271: “Hands”

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Hand style”</td>
<td>Example: “Thin arrow”</td>
</tr>
<tr>
<td>“Color hour hand”</td>
<td>Style color or color for the hands</td>
</tr>
<tr>
<td>“Color minute hand”</td>
<td></td>
</tr>
<tr>
<td>“Color second hand”</td>
<td></td>
</tr>
</tbody>
</table>

Table 272: “Lines”

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Lines style”</td>
<td>Clock face graduation</td>
</tr>
<tr>
<td>“None”</td>
<td></td>
</tr>
<tr>
<td>“Line”: Graduation lines by hour</td>
<td></td>
</tr>
<tr>
<td>“Hours and minutes”: Graduation lines by hours and minutes</td>
<td></td>
</tr>
<tr>
<td>“Dots”: Graduation dots by hour</td>
<td></td>
</tr>
<tr>
<td>“Color”</td>
<td>Color of the clock face graduation</td>
</tr>
<tr>
<td>“Line width”</td>
<td>Line weight of the clock face graduation</td>
</tr>
<tr>
<td>“Scale in 3D”</td>
<td>Representation of the clock face with 3D effect</td>
</tr>
</tbody>
</table>
### Table 273: “Numerics”

<table>
<thead>
<tr>
<th>“Style of numerics”</th>
<th>Digits on the clock face</th>
</tr>
</thead>
<tbody>
<tr>
<td>● “None”</td>
<td></td>
</tr>
<tr>
<td>● “Quarter”</td>
<td></td>
</tr>
<tr>
<td>● “All”</td>
<td></td>
</tr>
<tr>
<td>“Font”</td>
<td>Font for displaying the digits</td>
</tr>
<tr>
<td>“Font color”</td>
<td>Font for displaying the digits</td>
</tr>
</tbody>
</table>

### Table 274: “Center point”

| “Color” | Color of the center of the clock |

### Table 275: “Positioning”

| “Usage of” | ● “Default style values”: Presetting of the style values  
| “Positioning” | Requirement: Visible when the “Usage or” property is set to “Individual settings”. |
| “Numerics movement” | Value (in pixels) for shifting the digits.  
Example: 80 |
| “Line movement” | Value (in pixels) for shifting the hour lines.  
Example: 100 |
| “Hands scaling” | Factor for scaling the length of the hour hand. You can customize the exact position of the hour hand relative to the background image.  
Example: 100 |

### “Scaling type”

| Defines the scaling of the height and width of the element. |
| ● “Anisotropic”: The background image is scaled to the size of the element The height and width are scaled independently of each other. |
| ● “Isotropic”: The background image is scaled to the size of the element, retaining its proportion. The proportion of height and width is fixed. |

### “Optimized drawing”

| ☑: The background image is drawn one time. When the hour hand moves, only the affected part of the image is redrawn. |
| ☐: The background image is redrawn in cycles.  
Hint: Deactivate this option only for extreme exceptions. |

**Element property 'Absolute movement'**

The properties contain IEC variables for controlling the position of the element dynamically. The reference point is the upper left corner of the element. In runtime mode, the entire element is moved.

### “Movement”

| “X” | Variable (numeric data type). Defines the X position (in pixels).  
Example: PLC_PRG.iPos_X. |

Increasing this value in runtime mode moves the element to the right.
### “Y”
Variable (numeric data type). Defines the Y position (in pixels).

**Example:** PLC_PRG.iPos_Y.

Increasing this value in runtime mode moves the element downwards.

### “Rotation”
Variable (numeric data type). Defines the angle of rotation (in degrees).

**Example:** PLC_PRG.iAngle1.

The midpoint of the element rotates at the “Center” point. This rotation point is shown as the ◆ symbol.

In runtime mode, the alignment of the element remains the same with respect to the coordinate system of the visualization. Increasing the value rotates the element to the right.

### “Interior rotation”
Variable (numeric data type). Defines the angle of rotation (in degrees).

**Example:** PLC_PRG.iAngle2.

In runtime mode, the element rotates about the point of rotation specified in “Center” according to the value of the variable. In addition, the alignment of the element rotates according to the coordinate system of the visualization. Increasing the value in the code rotates clockwise.

The rotation point is shown as the ◆ symbol.

Note: If a static angle of rotation is specified in the “Position ➔ Angle” property, then the static angle of rotation is added to the variable angle of rotation (offset) when the visualization is executed.

---

**You can link the variables to a unit conversion.**

**The “X”, “Y”, “Rotation”, and “Interior rotation” properties are supported by the “Client Animation” functionality.**

See also
- Unit conversion
- 

---

**Element property ‘State variables’**

The variables control the element behavior dynamically.

### “Invisible”
Variable (BOOL). Toggles the visibility of the element.

TRUE: The element is not visible at runtime.
The “Invisible” property is supported by the “Client Animation” functionality.

See also

These properties are available only when you select the “Preview: Support client animations and overlay of native elements” option in the Visualization Manager.

You can animate a movement, a rotation, and the visibility of a visualization element.

<table>
<thead>
<tr>
<th>“Animation duration”</th>
<th>Time that the element executes an animation (in milliseconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Example: 500</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>“Move to the foreground”</th>
<th>Property value (BOOL)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>TRUE: At runtime, the element is displayed in the foreground.</td>
</tr>
<tr>
<td></td>
<td>FALSE: At runtime, the element is displayed in the layer where it was inserted in the visualization editor.</td>
</tr>
</tbody>
</table>

See also

- Chapter 1.3.5.21.5 “Using client animation” on page 1962

Visualization element “Date and time picker”

Symbol:

Tag: “Date/time controls”

The element provides the capability of selecting the date or time. The value can be changed by means of the keyboard or arrow keys. The date can be selected from a calendar.

Element properties

<table>
<thead>
<tr>
<th>“Element name”</th>
<th>Optional</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hint: Assign individual names for elements so that they are found faster in the element list.</td>
<td></td>
</tr>
<tr>
<td>Example: StartDateAndTime</td>
<td></td>
</tr>
</tbody>
</table>

| “Type of element” | “Date and time picker” |
Element property 'Position'

The position defines the location and size of the element in the visualization window. These are based on the Cartesian coordinate system. The origin is located at the upper left corner of the window. The positive horizontal x-axis runs to the right. The positive vertical y-axis runs downwards.

<table>
<thead>
<tr>
<th>“X”</th>
<th>X coordinate of the upper left corner of the element</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Specified in pixels.</td>
</tr>
<tr>
<td></td>
<td>Example: 10.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>“Y”</th>
<th>Y coordinate of the upper left corner of the element</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Specified in pixels.</td>
</tr>
<tr>
<td></td>
<td>Example: 10.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>“Width”</th>
<th>Specified in pixels.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Example: 150</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>“Height”</th>
<th>Specified in pixels.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Example: 30</td>
</tr>
</tbody>
</table>

You can also change the values by dragging the box symbols to other positions in the editor.

See also

- Chapter 1.3.5.3.2 “Positioning elements, adapting size and layer” on page 1113

Element property 'Center'

The properties contain fixed values for the coordinates of the point of rotation. This point of rotation is shown as the symbol. The point is used as the center for rotating and scaling.

<table>
<thead>
<tr>
<th>“X”</th>
<th>X-coordinate of the point of rotation</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>“Y”</th>
<th>Y-coordinate of the point of rotation</th>
</tr>
</thead>
</table>

You can also change the values by dragging the symbols to other positions in the editor.
"Variable"  
Variable (DATE, DT, TIME, LTIME, TOD)  
The value of the value of the variable is shown by the element and can be modified by the element.

The data type automatically determines the displayed value units:
- **TIME**: Day, hour, minute, second (by default, milliseconds are not shown)
- **DATE**: Year, month, day
- **DT**: Year, month, day, hour, minute, second
- **TOD**: Hour, minute, second (by default, milliseconds are not shown)
- **LTIME**: Day, hour, minute, second (by default, milliseconds, microseconds, and nanoseconds are not shown)

"Format string"  
The format can restrict the output to individual values.

Example for **LTIME**: Format: HH:mm:ss.ms.us.ns --> displayed: 08:15:12.780.150.360

Basically, all usual formats available for %t are also supported.

"Design date time picker"  
- "From style": All settings are preconfigured according to the style.
- "Explicit": The property "Designsettings" appears. Here you can customize the calendar.

"Design date picker"  
- "From style": All settings are preconfigured according to the style.
- "Explicit": The property "Designsettings" appears. Here you can customize the calendar.

"Positioning date picker"  
- "Dynamic": The calendar is adapted and positioned automatically.
- "Manual": The "Position settings" property appears. Here you can customize the calendar.

See also  
- § Chapter 1.3.5.18.2 “Placeholders with format definition in the output text” on page 1546

Element property 'Absolute movement'  
The properties contain IEC variables for controlling the position of the element dynamically. The reference point is the upper left corner of the element. In runtime mode, the entire element is moved.

"Movement"  
**X**  
Variable (numeric data type). Defines the X position (in pixels).
Example: PLC_PRG.iPos_X.
Increasing this value in runtime mode moves the element to the right.

**Y**  
Variable (numeric data type). Defines the Y position (in pixels).
Example: PLC_PRG.iPos_Y.
Increasing this value in runtime mode moves the element downwards.
### “Rotation”

Variable (numeric data type). Defines the angle of rotation (in degrees).

**Example:** PLC_PRG.iAngle1.

The midpoint of the element rotates at the “Center” point. This rotation point is shown as the symbol.

In runtime mode, the alignment of the element remains the same with respect to the coordinate system of the visualization. Increasing the value rotates the element to the right.

### “Scaling”

Variable (integer data type). Causes centric stretching.

**Example:** PLC_PRG.iScaling.

The reference point is the “Center” property.

The value 1 shrinks the element by a factor of 0.001. The value 1000 returns the element to its original size.

### “Interior rotation”

Variable (numeric data type). Defines the angle of rotation (in degrees).

**Example:** PLC_PRG.iAngle2.

In runtime mode, the element rotates about the point of rotation specified in “Center” according to the value of the variable. In addition, the alignment of the element rotates according to the coordinate system of the visualization. Increasing the value in the code rotates clockwise.

The rotation point is shown as the symbol.

Note: If a static angle of rotation is specified in the property “Position ➔ Angle”, then the static angle of rotation is added to the variable angle of rotation (offset) when the visualization is executed.

---

You can link the variables to a unit conversion.

---

The properties “X”, “Y”, “Rotation”, and “Interior rotation” are supported by the “Client Animation” functionality.

---

**Element property 'State variables’**

The variables control the element behavior dynamically.

---

---

---
"Invisible" Variable (BOOL). Toggles the visibility of the element.
  TRUE: The element is not visible at runtime.

"Deactivate inputs" Variable (BOOL). Toggles the operability of the element.
  TRUE: User inputs do not have any effect in runtime more. The element is shown as deactivated.

The "Invisible" property is supported by the "Client Animation" functionality.

See also

- These properties are available only when you select the "Preview: Support client animations and overlay of native elements" option in the Visualization Manager.
  You can animate a movement, a rotation, and the visibility of a visualization element.

"Animation duration" Time that the element executes an animation (in milliseconds)
  Example: 500

"Move to the foreground" Property value (BOOL)
  TRUE: At runtime, the element is displayed in the foreground.
  FALSE: At runtime, the element is displayed in the layer where it was inserted in the visualization editor.

See also

- Chapter 1.3.5.21.5 “Using client animation” on page 1962

Element property 'Access rights'

Requirement: User management is set up for the visualization.

"Access rights" Opens the "Access rights" dialog. There you can edit the access privileges for the element.
  Status messages:
  - "Not set. Full rights.": Access rights for all user groups: "operable"
  - "Rights are set: Limited rights": Access is restricted for at least one group.

See also

- Chapter 1.3.5.19.3.1 “Dialog 'Access Rights'” on page 1583

1.3.5.18.2 Placeholders with format definition in the output text

A character string that is output in the visualization can include the placeholder % for a variable.
At runtime, the placeholder is replaced by the actual value of the variable in the defined format.
The data type in the format definition and of the variable must be identical. A character string can contain a maximum of one placeholder.

Character strings for output are listed in the "Text" property. The associated variable is listed in the "Text variable" property.
### Printing integers

<table>
<thead>
<tr>
<th>%d</th>
<th>Printing a variable (integer data type) as a decimal number</th>
<th>Code: iCounter : INT := 12; Property “Text”: Value: %i Property “Text variable”: PLC_PRG.iCounter Output: Value: 12</th>
</tr>
</thead>
<tbody>
<tr>
<td>%i</td>
<td>Printing a variable (integer data type) as a binary number</td>
<td>Code: byCode : BYTE := 255; Property “Text”: Coding: %b Property “Text variable”: PLC_PRG.byCode Output: Coding: 11111111</td>
</tr>
<tr>
<td>%b</td>
<td>Printing a variable (integer data type) as an unsigned octal number without a preceding zero</td>
<td>Code: byCode : BYTE := 8#377; Property “Text”: Coding: %o Property “Text variable”: PLC_PRG.byCode Output: Coding: 377</td>
</tr>
<tr>
<td>%o</td>
<td>Printing a variable (integer data type) as an unsigned hexadecimal number without a preceding &quot;0x&quot;</td>
<td>Code: dwCode : INT := 16#FFFFFFF; Property “Text”: Coding: %x Property “Text variable”: PLC_PRG.dwCode Output: Coding: ffffffff</td>
</tr>
<tr>
<td>%x</td>
<td>Printing a variable (integer data type with maximum 32 bits) as an unsigned hexadecimal number without a preceding &quot;0x&quot;</td>
<td>Code: lwCode : LWORD := 16#4FFF_3FFF_2FFF_1FFF; Property “Text”: Coding: %llx Property “Text variable”: PLC_PRG.lwCode Output: Coding: 4fffff2ff1ff</td>
</tr>
<tr>
<td>%llX</td>
<td>Output of a 64-bit variable (LWORD, LINT, ULLINT) as a hexadecimal number. Note: llx means “long long hexadecimal”</td>
<td>Code: uiNumber : UINT := 1234; Property “Text”: Number: %u Property “Text variable”: PLC_PRG.uiNumber Output: Number: 1234</td>
</tr>
</tbody>
</table>

### Floating-point numbers

Floating-point numbers have the data type **REAL** or **LREAL**.
### Printing text

<table>
<thead>
<tr>
<th>Short format</th>
<th>Description</th>
<th>Code</th>
<th>Property “Text”</th>
<th>Property “Text variable”</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>%f</code></td>
<td>In decimal form with decimal point in format 1.6</td>
<td><code>rWeight : REAL := 1.123456789;</code></td>
<td>Weight: %f</td>
<td>PLC_PRG.rWeight</td>
<td>Weight: 1.123456</td>
</tr>
</tbody>
</table>
| `<alignment><minimum width>.<accuracy>f` | As decimal number in user-defined format  
  ● `<alignment>`: - or +, optional  
  - Left-aligned  
  + Right-aligned  
  ● `<minimum width>`: Number of places to the left of the decimal point  
  ● `<accuracy>`: Number of places to the right of the decimal point | `rWeight : REAL := 12.1` | Weight: %2.3f | Output: Weight: 12,100 |
| `%e`         | Printing a floating-point number (REAL or LREAL) in exponential notation of base 10 | `rValue : REAL := 1.234567e-003;` | Value: %E | PLC_PRG.rValue | Value: 1.23E-6 |
| `%E`         | | `rValue : REAL := 1.234567e-003;` | Value: %e | PLC_PRG.rValue | Value: 1.23e-6 |
| `%c`         | Printing a single character in ASCII | `bChar := 16#41;` | Key: %c | PLC_PRG.bChar | Key: A |
| `%s`         | Printing a character string | `strName := 'Paul Smith';` | Name: %s | PLC_PRG.strName | Name: Paul Smith |
Printing the percent sign

<table>
<thead>
<tr>
<th>%%</th>
<th>Printing the percent sign in a character string</th>
<th>Property “Text”: Valid until 90%</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Output: Valid until 90%</td>
<td>Code: iPercentage : INT := 80;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Property “Text”: Valid until %d%.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Property “Text variable”:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>PLC_PRG.iPercentage := 80;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Output: Valid until 80%</td>
</tr>
</tbody>
</table>

Printing the date and time

If the output text in the element property “Text” contains the placeholder “%t”, then a date and/or time is printed. If a variable is not specified in the property “Text variable”, then the system time is printed; otherwise it is the value of the variable.

The names of the days and months are displayed in English by default. If localized texts are used, then the text list System has to be supplemented. This text list is created automatically in the “POUs” view when the placeholder %t is used. The English terms have to be used as the ID here. The localization can be done for both the abbreviated and full names.

The localization can be done for both the abbreviated and full names.

Time data types include LTIME, TIME, TIME_OF_DAY, TOD, DATE, DATE_AND_TIME, and DT.

### Date and time formats

<table>
<thead>
<tr>
<th>%t[yyyy]</th>
<th>Years with century (4 digits)</th>
<th>Code: dateBy : DATE := DATE#2020-1-1;</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Property “Text”: By the year %t[yyyy]</td>
<td>Property “Text variable”: PLC_PRG.dateBy</td>
</tr>
<tr>
<td></td>
<td>Property “Text variable”: PLC_PRG.dateBy</td>
<td>Property “Text variable”: PLC_PRG.dateSince</td>
</tr>
<tr>
<td></td>
<td>Output: By the year 2020</td>
<td>Output: Since: 00</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>%t[yy]</th>
<th>Years without century (00-99)</th>
<th>Code: dateSince : DATE := DATE#2000-1-1;</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Property “Text”: Since: %t[yy]</td>
<td>Property “Text variable”: PLC_PRG.dateSince</td>
</tr>
<tr>
<td></td>
<td>Property “Text variable”: PLC_PRG.dateSince</td>
<td>Property “Text variable”: PLC_PRG.dateSince</td>
</tr>
<tr>
<td></td>
<td>Output: Since: 00</td>
<td>Output: Since: 0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>%t[y]</th>
<th>Years without century (0-99)</th>
<th>Code: dateSince : DATE := DATE#2000-1-1;</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Property “Text”: Since: %t[y]</td>
<td>Property “Text variable”: PLC_PRG.dateSince</td>
</tr>
<tr>
<td></td>
<td>Property “Text variable”: PLC_PRG.dateSince</td>
<td>Property “Text variable”: PLC_PRG.dateSince</td>
</tr>
<tr>
<td></td>
<td>Output: Since: 0</td>
<td>Output: Since: 0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>%t[MMMM]</th>
<th>Months as a full name</th>
<th>Code: dateMonth : DATE := DATE#2016-1-1;</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Property “Text”: Month: %t[MMMM]</td>
<td>Property “Text variable”: PLC_PRG.dateMonth</td>
</tr>
<tr>
<td></td>
<td>Property “Text variable”: PLC_PRG.dateMonth</td>
<td>Property “Text variable”: PLC_PRG.dateMonth</td>
</tr>
<tr>
<td></td>
<td>Output: Month: January</td>
<td>Output: Month: January</td>
</tr>
</tbody>
</table>
| `%t[MMM]` | Months as an abbreviated name | Code: `dateMonth : DATE := DATE#2016-1-1;`  
Property “Text”: Month: `%t[MMM]`  
Property “Text variable”: PLC_PRG.dateMonth  
Output: Month: Jan |
| `%t[MM]` | Months as a number (01 – 12) | Code: `dateMonth : DATE := DATE#2016-1-1;`  
Property “Text”: Month: `%t[MM]`  
Property “Text variable”: PLC_PRG.dateMonth  
Output: Month: 01 |
| `%t[M]` | Months as a number (1 – 12) | Code: `dateMonth : DATE := DATE#2016-1-1;`  
Property “Text”: Month: `%t[M]`  
Property “Text variable”: PLC_PRG.dateMonth  
Output: Month: 1 |
| `%t[ddddd]` | Days of week as a number (1=Monday to 7=Sunday) | Code: `iDay : INT := 7;`  
Property “Text”: Day: `%t[ddddd]`  
Property “Text variable”: PLC_PRG.iDay  
Output: Day: 7 |
| `%t[dddd]` | Days of week as a full name | Code: `iDay : INT := 7;`  
Property “Text”: Day: `%t[dddd]`  
Property “Text variable”: PLC_PRG.iDay  
Output: Day: Sunday |
| `%t[ddd]` | Days of week as an abbreviated name | Code: `iDay : INT := 7;`  
Property “Text”: Day: `%t[ddd]`  
Property “Text variable”: PLC_PRG.iDay  
Output: Day: Sun |
| `%t[dd]` | Days in month as a number (01 – 31) | Code: `iDay : INT := 1;`  
Property “Text”: Day: `%t[dd]`  
Property “Text variable”: PLC_PRG.iDay  
Output: Day: 01 |
| `%t[d]` | Days in month as a number (1 – 31) | Code: `iDay : INT := 1;`  
Property “Text”: Day: `%t[d]`  
Property “Text variable”: PLC_PRG.iDay  
Output: Day: 1 |
| %t[jjj] | Days in year as a number (001-366) | Code: dateOfNoReturn : DATE := DATE#2016-09-01;
Property “Text”: Day of no return: %t[jjj]
Property “Text variable”: PLC_PRG.dateOfNoReturn
Output: Day of no return: 245 |
| %t[HH] | Hours in 24-hour format (00-23) | Code: todEnd : TOD := TIME_OF_DAY#17:0:0;
Property “Text”: Ends at: %t[HH]:00
Property “Text variable”: PLC_PRG.todEnd
Output: Ends at 17:00 |
| %t[hh] | Hours in 12-hour format (01-12) | Code: todEnd : TOD := TIME_OF_DAY#17:0:0;
Property “Text”: Ends at: %t[hh]:00 o'clock
Property “Text variable”: PLC_PRG.todEnd
Output: Ends at 17:00 o'clock |
| %t[mm] | Minutes with leading zeros (00-59) | Code: tPeriod : TIME := T#5M;
Property “Text”: Period: %t[mm]m
Property “Text variable”: PLC_PRG.tPeriod
Output: Period: 05m |
| %t[m] | Minutes without leading zeros (0-59) | Code: tPeriod : TIME := T#5m;
Property “Text”: Period: %t[m 'm']
Property “Text variable”: PLC_PRG.tPeriod
Output: Period: 5 m |
| %t[ss] | Seconds with leading zeros (00-59) | Code: tPeriod : TIME := T#5m3s;
Property “Text”: Period: %t[mm'm's's's']
Property “Text variable”: PLC_PRG.tPeriod
Output: Period: 05m03s |
| %t[s] | Seconds without leading zeros (0-59) | Code: tPeriod : TIME := T#5m3s;
Property “Text”: Period: %t[m'm's's']
Property “Text variable”: PLC_PRG.tPeriod
Output: Period: 5m3s |
| %t[ms] | Milliseconds without leading zeros (0-999) | Code: tPeriod : TIME := T#500ms;
Property “Text”: Period: %t[ms'm's']
Property “Text variable”: PLC_PRG.tPeriod
Output: Period: 500ms |
For LTIME variables only: microsecond definition (0-999)

Code:
```plaintext
ltPeriod : LTIME := LTIME#1000D23H44M12S34MS2US44NS;
```

Property “Text”: Period: %t[dd.HH.m.s.ms.us.ns]

Property “Text variable”: PLC_PRG.ltPeriod

Output: Period: 1000.23.44.12.34.2.44

Hint: Overflow is permitted in the greatest time unit of a definition.

For LTIME variables only: nanosecond definition (0-999)

Code:
```plaintext
tClosed : TOD := TOD#17:17:17.17;
```

Property “Text”: Closed at %t[hht]

Property “Text variable”: PLC_PRG.tClosed

Output: Closed at 5P

If the value is a time < 12h, then AM is printed; otherwise PM is printed.

Code:
```plaintext
tClosed : TOD := TOD#17:17:17.17;
```

Property “Text”: Closed at %t[hh tt]

Property “Text variable”: PLC_PRG.tClosed

Output: Closed at 5 PM

If character strings should be printed that correspond to a format definition, then these must be represented in single straight quotation marks.

The format definitions can be represented in a series.

Printing the complete time

Code:
```plaintext
dwTime : DWORD := 4294967295;
```

Property “Text”: Time: %t[HH:mm:ss:ms]

Property “Text variable”: PLC_PRG.dwTime


Printing the date and day of the week

Code:
```plaintext
dateSet : DATE := DATE#2016-02-12;
```

Property “Text”: Date: %t[yyyy-MM-dd dddd]

Property “Text variable”: PLC_PRG.dateSet

Output: Date: 2016-02-12 Friday

See also

- **Time data types**

### 1.3.5.18.3 Methods of the dialog manager

Visualizations that have the visualization type “Dialog” and serve as input requests are automatically instanced and managed by the internal dialog manager.
The dialog manager can be accessed in the application via the similarly internal visualization manager by calling the method `GetDialogManager`.

The dialog manager has the following methods for handling a dialog.

**NOTICE!**

You can program the method calls in function blocks or functions that are themselves called from the visualization or via the action `Execute ST code`.

In addition, you can program the method calls in the application code. Make sure that the call runs in the VISU_TASK. The behavior is undefined if this is not the case.

See also

- Method 'GetDialog'

  Supplies the instance (IVisualisationDialog) of the dialog whose name is transferred.

  **Table 276: Inputs (VAR_INPUT)**

<table>
<thead>
<tr>
<th>Name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>stName</td>
<td>STRING</td>
<td>Name of the dialog</td>
</tr>
</tbody>
</table>

  **Table 277: Outputs (VAR_OUTPUT)**

<table>
<thead>
<tr>
<th>Name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetDialog</td>
<td>VisuElems.IVisualisationDialog</td>
<td>Instance (IVisualisationDialog) of the dialog</td>
</tr>
</tbody>
</table>

Method 'GetClientInterface'

Supplies a pointer to the dialog structure.

*Dialog data are held for each display variant.*

**Table 278: Inputs (VAR_INPUT)**

<table>
<thead>
<tr>
<th>Name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dialog</td>
<td>VisuElems.IVisualisationDialog</td>
<td>Name of the visualization</td>
</tr>
<tr>
<td>pClient</td>
<td>POINTER TO VisuElems.IVisualisationDialogVisuStructClientData</td>
<td>Pointer to the display variants</td>
</tr>
</tbody>
</table>

**Table 279: Outputs (VAR_OUTPUT)**

<table>
<thead>
<tr>
<th>Name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetClientInterface</td>
<td>Example: POINTER TO Login_VISU_STRUCT</td>
<td>Pointer to the dialog structure.</td>
</tr>
</tbody>
</table>
Method 'Open-Dialog'

Opens the dialog for the client.

In addition there is the extended method 'OpenDialog(Number)'.

Table 280: Inputs (VAR_INPUT)

<table>
<thead>
<tr>
<th>Name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dialog</td>
<td>VisuElems.IVisualisationDialog</td>
<td>Name of the visualization</td>
</tr>
<tr>
<td>pClient</td>
<td>POINTER TO VisuElems.VisuStructClientData</td>
<td></td>
</tr>
<tr>
<td>bModal</td>
<td>BOOL</td>
<td></td>
</tr>
<tr>
<td>pRect</td>
<td>POINTER TO</td>
<td></td>
</tr>
</tbody>
</table>

Table 281: Outputs (VAR_OUTPUT)

<table>
<thead>
<tr>
<th>Name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OpenDialog</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Method 'Close-Dialog'

Closes the dialog for the client.

Table 282: Inputs (VAR_INPUT)

<table>
<thead>
<tr>
<th>Name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dialog</td>
<td>VisuElems.IVisualisationDialog</td>
<td>Name of the visualization</td>
</tr>
<tr>
<td>pClient</td>
<td>POINTER TO VisuElems.VisuStructClientData</td>
<td></td>
</tr>
</tbody>
</table>

Table 283: Outputs (VAR_OUTPUT)

<table>
<thead>
<tr>
<th>Name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CloseDialog</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

1.3.5.18.4 Attribute 'VAR_IN_OUT_AS_POINTER'

Function: The pragma (attribute 'VAR_IN_OUT_AS_POINTER') allows for the passing of a reference to a data object to the interface variable of a visualization.

Requirement: The referenced visualization must be used as a dialog.

Syntax: (attribute 'VAR_IN_OUT_AS_POINTER')
1.3.5.18.5 Attribute 'parameterstringof'

The pragma \{attribute 'parameterstringof'\} allows that the instance name of the specified parameter is made accessible for the referenced visualization. An interface variable (STRING) will contain the instance name of the specified parameter. The interface variable is visible within the referenced visualization and can for example be used in a text output.

Syntax:

\{attribute 'parameterstringof' := '<variable>'\}

Example: declaration of an interface

```plaintext
VAR_IN_OUT
  \{attribute 'VAR_IN_OUT_AS_POINTER'\}
  iftController : ControlFB;
END_VAR

See also
- § Chapter 1.3.5.15.4 “Call dialog with interfaces” on page 1199
- § Chapter 1.3.5.19.2.1 “Command 'Interface Editor’” on page 1556
```

1.3.5.19 Reference, user interface

1.3.5.19.1 Keyboard Shortcuts for Default Keyboard Action

Requirement: The “Activate default keyboard handling” option is activated in the “Visualization Manager” object.

The keyboard shortcuts for default keyboard action make it possible for users to operate the visualization with the keyboard only. Elements that respond to user input can process a keyboard event instead of a mouse event. You do not have to change their input configuration for this purpose. The universal keyboard shortcuts are supported by all devices and are available on all display variants when needed.

```
Example: declaration of an interface

VAR_INPUT
  \{attribute 'parameterstringof' := 'iftDut_A'\}
  sItfNameDut_A: STRING;
END_VAR
VAR_IN_OUT
  iftDut_A : DUT_A;
END_VAR

See also
- § Chapter 1.3.5.15.2 “Calling a visualization with an interface” on page 1188
- § Chapter 1.3.5.19.2.1 “Command 'Interface Editor’” on page 1556
```
### Keyboard shortcuts

**[Tab]**  
Focus jumps to the next element.  
The next element that responds to a configured or preconfigured user input receives the focus. The order of elements corresponds to the order that the elements were added to the editor.  
If the focused element is a table, then the upper left cell in the table is the next focus. After that, each next cell until all cells have been focused. It also applies here that only cells that require input are focused.  
If the focused element is a frame, then an element of the referenced visualization is set next in focus in the frame. After that, each next element until all elements have been focused. It also applies here that only elements that require input are focused.

**[Shift]+[Tab]**  
Focus jumps to the previous element.  
The element is focused that is before the currently focused element in the added order. Therefore, the order is the opposite as for “Tab”.

**[Arrow]**  
The focus jumps to the element that is in the direction as indicated by the arrow.

**[Input]**  
The visualization detects the input at the focused element and triggers the input action.

### 1.3.5.19.2 Commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.3.5.19.2.1</td>
<td>Command 'Interface Editor'</td>
<td>1556</td>
</tr>
<tr>
<td>1.3.5.19.2.2</td>
<td>Command 'Keyboard Configuration'</td>
<td>1558</td>
</tr>
<tr>
<td>1.3.5.19.2.3</td>
<td>Command 'Element List'</td>
<td>1559</td>
</tr>
<tr>
<td>1.3.5.19.2.4</td>
<td>Command 'Activate Keyboard Usage'</td>
<td>1560</td>
</tr>
<tr>
<td>1.3.5.19.2.5</td>
<td>Command 'Order'</td>
<td>1560</td>
</tr>
<tr>
<td>1.3.5.19.2.6</td>
<td>Command 'Alignment'</td>
<td>1560</td>
</tr>
<tr>
<td>1.3.5.19.2.7</td>
<td>Command 'Group'</td>
<td>1564</td>
</tr>
<tr>
<td>1.3.5.19.2.8</td>
<td>Command 'Ungroup'</td>
<td>1564</td>
</tr>
<tr>
<td>1.3.5.19.2.9</td>
<td>Command 'Frame Selection'</td>
<td>1564</td>
</tr>
<tr>
<td>1.3.5.19.2.10</td>
<td>Command 'Background'</td>
<td>1565</td>
</tr>
<tr>
<td>1.3.5.19.2.11</td>
<td>Command 'Multiply Visu Element'</td>
<td>1566</td>
</tr>
<tr>
<td>1.3.5.19.2.12</td>
<td>Command 'Configure Display Settings of the Trend'</td>
<td>1570</td>
</tr>
<tr>
<td>1.3.5.19.2.13</td>
<td>Command 'Configure Trace'</td>
<td>1572</td>
</tr>
<tr>
<td>1.3.5.19.2.14</td>
<td>Command 'Export Trace Configuration'</td>
<td>1574</td>
</tr>
<tr>
<td>1.3.5.19.2.15</td>
<td>Command 'Insert Elements for Controlling Trace'</td>
<td>1575</td>
</tr>
<tr>
<td>1.3.5.19.2.16</td>
<td>Command 'Configure Display Settings of the Trend'</td>
<td>1576</td>
</tr>
<tr>
<td>1.3.5.19.2.17</td>
<td>Command 'Edit Trend Recording'</td>
<td>1577</td>
</tr>
<tr>
<td>1.3.5.19.2.18</td>
<td>Command 'Insert Elements for Controlling the Trend'</td>
<td>1577</td>
</tr>
<tr>
<td>1.3.5.19.2.19</td>
<td>Command 'Visualization Element Repository'</td>
<td>1578</td>
</tr>
<tr>
<td>1.3.5.19.2.20</td>
<td>Command 'Visualization Style Repository'</td>
<td>1580</td>
</tr>
<tr>
<td>1.3.5.19.2.21</td>
<td>Command 'Add Visual Element'</td>
<td>1581</td>
</tr>
<tr>
<td>1.3.5.19.2.22</td>
<td>Command 'Select None'</td>
<td>1581</td>
</tr>
<tr>
<td>1.3.5.19.2.23</td>
<td>Command 'Add Elements for Alarm Acknowledgement'</td>
<td>1582</td>
</tr>
</tbody>
</table>

**Command 'Interface Editor'**

Symbol: 🔄; keyboard shortcut: `[Alt]+[F6]`.

**Function:** The command opens and closes the “Interface Editor” tab above the visualization editor.

**Call:** Menu bar: “Visualization ➔ Interface Editor” Also by clicking on the small down arrow at the top of the visualization editor.
Tab 'Interface Editor'

The tab contains an editor for the declaration of interface variables. The editor behaves in a similar way to the declaration editor of a function block, however interface variables are not initialized.

Syntax

```plaintext
<scope>
    {attribute '<attribute name>' ( := '<expression>' )? }?
    <identifier> : <data type>;
END_VAR
```

```plaintext
<scope> : VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT // (...)? : Optional
```

Example

Declaration in the interface editor

```plaintext
VAR_INPUT
    {attribute 'parameterstringof'}
    sIdentifier : STRING; // String for instance name
    iCounter : INT;
END_VAR

VAR_IN_OUT
    {attribute 'VAR_IN_OUT_AS_POINTER'}
    fbController: FB_Controller;
END_VAR
```

Scopes

Possible scopes for interfaces of visualizations or dialogs

<table>
<thead>
<tr>
<th>Scope</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>VAR_INPUT</td>
<td>When transferring data that will only be read</td>
</tr>
<tr>
<td>Pragma</td>
<td></td>
</tr>
<tr>
<td></td>
<td>When transferring a variable (data type STRING) for the instance name of the transfer parameter specified in the attribute</td>
</tr>
<tr>
<td>VAR_IN_OUT</td>
<td>When transferring a structure</td>
</tr>
<tr>
<td></td>
<td>When the visualization is instanced, it gets a reference to the current application data.</td>
</tr>
<tr>
<td></td>
<td>When transferring a control variable, if the variable is written to when a user input is made. Only then can the visualization write to it.</td>
</tr>
<tr>
<td>Pragma</td>
<td></td>
</tr>
<tr>
<td></td>
<td>When transferring a pointer to a data object</td>
</tr>
<tr>
<td></td>
<td>In contrast to the VAR_IN_OUT scope (without an attribute), the variable changes are effective immediately and not just when the dialog is closed.</td>
</tr>
<tr>
<td></td>
<td>Note: Use this scope only if the visualization implements a Dialog.</td>
</tr>
<tr>
<td>VAR_IN_OUT</td>
<td>When transferring a pointer to a data object</td>
</tr>
<tr>
<td>Pragma</td>
<td></td>
</tr>
<tr>
<td></td>
<td>In contrast to the VAR_IN_OUT scope (without an attribute), the variable changes are effective immediately and not just when the dialog is closed.</td>
</tr>
<tr>
<td></td>
<td>Note: Use this scope only if the visualization implements a Dialog.</td>
</tr>
<tr>
<td>VAR_INPUT</td>
<td>When transferring data that will only be read</td>
</tr>
<tr>
<td>Pragma</td>
<td></td>
</tr>
<tr>
<td></td>
<td>When transferring a variable (data type STRING) for the instance name of the transfer parameter specified in the attribute</td>
</tr>
<tr>
<td>VAR_INPUT</td>
<td>When transferring a pointer to a data object</td>
</tr>
<tr>
<td>Pragma</td>
<td></td>
</tr>
<tr>
<td></td>
<td>In contrast to the VAR_IN_OUT scope (without an attribute), the variable changes are effective immediately and not just when the dialog is closed.</td>
</tr>
<tr>
<td></td>
<td>Note: Use this scope only if the visualization implements a Dialog.</td>
</tr>
</tbody>
</table>
See also

- Declaration of Variables
- Declaration Editor
- Chapter 1.3.5.15.2 “Calling a visualization with an interface” on page 1188
- Chapter 1.3.5.15.4 "Call dialog with interfaces" on page 1199
- Chapter 1.3.5.18.5 “Attribute ‘parameterstringof’” on page 1555
- Chapter 1.3.5.18.4 “Attribute ‘VAR_IN_OUT_AS_POINTER’” on page 1554

Command 'Keyboard Configuration'  

Symbol: ; keyboard shortcut: [Alt]+[F6].

Function: This command opens and closes the “Keyboard Configuration” tab above the visualization editor.

Call: Menu bar: “Visualization”.

Requirement: A visualization is open and active in the visualization editor.

See also

Tab 'Keyboard configuration'

This tab contains a list of keyboard shortcuts with an editing option.

A keyboard shortcut can refer specifically to an element. Then the configuration appears here and in the “Input configuration” property of the associated element.

A keyboard shortcut can also have several configurations. If a keyboard shortcut has multiple keyboard configurations, then its input actions are executed in the order listed here.

Keyboard shortcuts of the default keyboard action are not listed here.

| “Key” | Key that a keyboard configuration is defined. Example: [M] 
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Note:</td>
<td>You can combine the key with [Ctrl], [Alt], and/or [Shift].</td>
</tr>
</tbody>
</table>

| “Key down” | ☑: The input action is executed when the user presses the key. 
|------------|------------------------------------------------------------------|
|            | ☐: The input action is executed when the user releases the key.  
|            | Double-click: Drop-down list of all keys.                       |
| Note:      | If the input action should be executed for both pressing the key (KeyDown) and releasing the key (KeyUp), then you must define a keyboard configuration for both input actions. |

<table>
<thead>
<tr>
<th>“Shift”</th>
<th>☑: The input event is triggered for [Shift]+[key].</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Ctrl”</td>
<td>☑: The input event is triggered for [Ctrl]+[key].</td>
</tr>
<tr>
<td>“Alt”</td>
<td>☑: The input event is triggered for [Alt]+[key].</td>
</tr>
</tbody>
</table>

| “Action type” | input action 
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Double-click: Drop-down list of input actions.</td>
</tr>
<tr>
<td></td>
<td>Tip: For a description of input actions, refer to the “Input configuration” dialog box.</td>
</tr>
</tbody>
</table>

| “Action” | Configuration of the input action that was selected next. 
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Double-click</td>
<td>A dialog box opens that varies according to the input action. It allows the user-prompted customization of the settings.</td>
</tr>
<tr>
<td>Tip: For a description of dialog boxes, refer to the “Input configuration” dialog box. The input action is configured in the same way here.</td>
<td></td>
</tr>
</tbody>
</table>
"Element ID" | ID of the visualization element where the user can execute the key event. The ID is relevant only if the event is also assigned to an element.
---|---
| Tip: The assignment of ID to element name is listed in the “Element list”.

"Access rights" | Access privileges of the action per user group.
---|---
| Requirement: The visualization has a user management.

↓ | Clicking the symbol on the right of the list moves the selected row one line down.
↑ | Clicking the symbol on the right of the list moves the selected row one line up.
Blank line | Allows adding a new keyboard configuration.

See also
- ☞ Chapter 1.3.5.19.2.3 “Command ‘Element List’” on page 1559
- ☞ Chapter 1.3.5.19.3.6 “Dialog ‘Input Configuration’” on page 1586

Command 'Element List'
Symbol: ![element_list]

**Function:** The command opens the “Element List” tab for the current visualization. It is displayed in the upper part of the visualization editor.

**Call:** Menu “Visualization”

**Requirement:** A visualization is open in the editor.

Tab 'Elementlist'
This view provides a list of the visualization elements in the currently opened visualization. The selection is always synchronized with the selection in the main window of the visualization editor. Grouped elements are displayed in a tree structure. The order of the elements in this list from top to bottom is the same as the order of their positions in the display levels of the visualization from back to front. If an element is shifted forwards or backwards in the editor window with the commands from the menu “Visualization ➔ Order”, the element list is automatically updated.

<table>
<thead>
<tr>
<th>&quot;Type&quot;</th>
<th>Element type and symbol as used in the view “Tools”, as well as the element number that initially arises from the order of insertion.</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;X&quot;, &quot;Y&quot;</td>
<td>Position of the top left corner of the element (0,0 = top left corner of the visualization area).</td>
</tr>
<tr>
<td>&quot;Width&quot;</td>
<td>Dimensions of the element in pixels.</td>
</tr>
<tr>
<td>&quot;Height&quot;</td>
<td></td>
</tr>
<tr>
<td>&quot;ID&quot;</td>
<td>Internally assigned element identifier</td>
</tr>
<tr>
<td>&quot;Name&quot;</td>
<td>Element name as defined in “Properties ➔ Element name”.</td>
</tr>
<tr>
<td>&quot;Access rights&quot;</td>
<td>If the behavior of an element is restricted for some user groups, this is marked by a padlock symbol 🗝️.</td>
</tr>
</tbody>
</table>

See also
- ☞ Chapter 1.3.5.19.4.1.1 “Visualization editor” on page 1608
- ☞ Chapter 1.3.5.19.2.5 “Command ‘Order’” on page 1560
- ☞ Chapter 1.3.5.3.2 “Positioning elements, adapting size and layer” on page 1113
Command 'Activate Keyboard Usage'

Symbol: 

**Function:** This command activates and deactivates the keyboard usage when a visualization is executed in online mode (integrated in CODESYS).

**Call:** Menu bar: “Visualization”, context menu.

**Requirement:** A visualization is open.

When this command is active, the visualization executes the keyboard events that you specified as a visualization user.

When the command is inactive, CODESYS executes the keyboard events that you specify.

See also

- § Chapter 1.3.5.4.4 “Configuring keyboard shortcuts” on page 1131

Command 'Order'

**Function:** The command makes further commands available. They are for specifying the order of the elements in levels, since elements in the rear levels are concealed by those in the front levels.

**Call:** Menu “Visualization”, context menu

**Requirement:** The visualization elements are positioned behind one another.

See also

- § Chapter 1.3.5.3.2 “Positioning elements, adapting size and layer” on page 1113

Command 'Bring to Front'

Symbol: 

**Function:** The command positions the selected visualization element in the front level. The element becomes completely visible.

**Call:** Menu “Visualization ➔ Order”, context menu

Command 'Bring One to Front'

**Function:** The command positions the selected visualization element one level further forwards.

**Call:** Menu “Visualization ➔ Order”, context menu

Command 'Send to Back'

Symbol: 

**Function:** The command positions the selected visualization element in the back level.

**Call:** Menu “Visualization ➔ Order”, context menu

Command 'Send One to Back'

Symbol: 

**Function:** The command positions the selected visualization element one level further backwards.

**Call:** Menu “Visualization ➔ Order”, context menu

Command 'Alignment'

**Function:** The command makes further commands available. It is used for the alignment of visualization elements in the window area of the visualization.

**Call:** Menu “Visualization”, context menu
See also

- Chapter 1.3.5.3.2 “Positioning elements, adapting size and layer” on page 1113

**Command 'Align Left'**

Symbol: 

Function: the command aligns the selected visualization elements along a line through the left-hand edge of the element that is positioned furthest left.

Call: Menu “Visualization → Alignment”, context menu

Requirement: Several elements are selected.

**Command 'Align Top'**

Symbol: 

Function: the command aligns the selected visualization elements along a line through the upper edge of the element that is positioned highest.

Call: Menu “Visualization → Alignment”, context menu

Requirement: Several elements are selected.

**Command 'Align Right'**

Symbol: 

Function: the command aligns the selected visualization elements along a line through the right-hand edge of the element that is positioned furthest right.

Call: Menu “Visualization → Alignment”, context menu

Requirement: Several elements are selected.

**Command 'Align Bottom'**

Symbol: 

Function: the command aligns the selected visualization elements along a line through the lower edge of the element that is positioned lowest.

Call: Menu “Visualization → Alignment”, context menu

Requirement: Several elements are selected.

**Command 'Align Vertical Center'**

Symbol: 

Function: the command aligns the selected visualization elements to their common vertical center.

Call: Menu “Visualization → Alignment”, context menu

Requirement: Several elements are selected.

**Command 'Align Horizontal Center'**

Symbol: 

Function: The command aligns the selected visualization elements to their common horizontal center.

Call: Menu “Visualization → Alignment”, context menu

Requirement: Several elements are selected.

**Command 'Make Horizontal Spacing Equal'**

Symbol: 

Function: The command aligns the selected visualization elements so that the elements positioned furthest left and furthest right retain their position and the elements between them are positioned with the same horizontal spacing.

Call: Menu “Visualization → Alignment”, context menu
Requirement: 3 or more elements are selected. The first element is blue, while the other elements are displayed in grey.

**Command 'Increase Horizontal Spacing'**

Symbol: ⬆️

Function: The command aligns the selected visualization elements so that the blue element retains its position and the other elements are positioned with a larger horizontal spacing. The spacing increases by 1 pixel each time.

Call: Menu “Visualization ➔ Alignment”, context menu

Requirement: Several elements are selected.

**Command 'Decrease Horizontal Spacing'**

Symbol: ⬅️

Function: The command aligns the selected visualization elements so that the blue element retains its position and the other elements are positioned with a smaller horizontal spacing. The spacing decreases by 1 pixel each time.

Call: Menu “Visualization ➔ Alignment”, context menu

Requirement: Several elements are selected.

**Command 'Remove Horizontal Spacing'**

Symbol: ❌

Function: The command aligns the selected visualization elements so that the blue element retains its position and the other elements are positioned with no horizontal spacing between them.

Call: Menu “Visualization ➔ Alignment”, context menu

Requirement: Several elements are selected.

**Command 'Make Vertical Spacing Equal'**

Symbol: ⬇️

Function: The command aligns the selected visualization elements so that the uppermost and lowermost elements retain their position and the elements between them are positioned with the same vertical spacing.

Call: Menu “Visualization”, context menu

Requirement: 3 or more elements are selected. The first element is blue, while the other elements are displayed in grey.

**Command 'Increase Vertical Spacing'**

Symbol: ⬆️

Function: The command aligns the selected visualization elements so that the blue element retains its position and the other elements are positioned with a larger vertical spacing. The spacing increases by 1 pixel each time.

Call: Menu “Visualization ➔ Alignment”, context menu

Requirement: Several elements are selected.

**Command 'Decrease Vertical Spacing'**

Symbol: ⬅️

Function: The command aligns the selected visualization elements so that the blue element retains its position and the other elements are positioned with a smaller vertical spacing. The spacing decreases by 1 pixel each time.

Call: Menu “Visualization ➔ Alignment”, context menu

Requirement: Several elements are selected.
**Command 'Remove Vertical Spacing'**

Symbol: 

**Function:** The command aligns the selected visualization elements so that the blue element retains its position and the other elements are positioned with no horizontal spacing between them.

**Call:** Menu “Visualization ➔ Alignment”, context menu

**Requirement:** Several elements are selected.

---

**Command 'Make Same Width'**

Symbol: 

**Function:** The command makes the width of the selected visualization elements the same as the width of the blue selected element.

**Call:** Menu “Visualization ➔ Alignment”, context menu

**Requirement:** Several elements are selected. The first element is blue, while the other elements are displayed in grey.

```
The command does not work with lines or polygons.
```

---

**Command 'Make Same Height'**

Symbol: 

**Function:** The command makes the height of the selected visualization elements the same as the height of the blue selected element.

**Call:** Menu “Visualization ➔ Alignment”, context menu

**Requirement:** Several elements are selected. The first element is blue, while the other elements are displayed in grey.

```
The command does not work with lines or polygons.
```

---

**Command 'Make Same Size'**

Symbol: 

**Function:** The command makes the size of the selected visualization elements the same as the size of the blue selected element.

**Call:** Menu “Visualization ➔ Alignment”, context menu

**Requirement:** Several elements are selected. The first element is blue, while the other elements are displayed in grey.

```
The command does not work with lines or polygons.
```

---

**Command 'Size to Grid'**

Symbol: 

**Function:** The command aligns the size and position of the selected visualization elements to the grid.

**Call:** Menu “Visualization ➔ Alignment”, context menu

**Requirement:** Several elements are selected.
Command 'Group'

Symbol: 🎨

**Function:** The command groups the selected visualization elements and displays them as one.

**Call:** Menu “Visualization”, context menu

**Requirement:** At least 2 elements are selected.

To select more elements you can drag a window around the desired elements with the mouse. Alternatively you can click on the desired elements while keeping the [Shift] key pressed.

To select all elements you can open the context menu of the visualization editor and choose the “Select All” command.

You can also drag and drop elements to a group. For that, press the [Shift] key while dragging the element to the group. Meanwhile the cursor changes its appearance (display a small plus sign).

See also
- ✉ Chapter 1.3.5.3.2 “Positioning elements, adapting size and layer” on page 1113
- ✉ Chapter 1.3.5.19.2.8 “Command ‘Ungroup’” on page 1564
- ✉ Chapter 1.3.5.19.2.22 “Command ‘Select None’” on page 1581

Command 'Ungroup'

Symbol: 🎨

**Function:** The command ungroups elements again.

**Call:** Menu “Visualization”, context menu

**Requirement:** A grouping is selected.

See also
- ✉ Chapter 1.3.5.19.2.7 “Command ‘Group’” on page 1564

Command 'Frame Selection'

**Function:** The command opens the “Frame Configuration” dialog.

**Call:**
- Menu bar: “Visualization”
- Click the “Configure” button in the “References” property.

**Requirement:** A “Frame” element or “Tab control” element is selected in the editor. The “Element Properties” view is open.

Dialog 'Frame Configuration'

The dialog allows you to select one or more of all available visualizations. The selected visualizations are displayed at runtime in the window area of the frame element or tab control element.
**NOTICE!**
Visualizations can be nested at any depth by means of frame elements. In order to use the “Switch to any visualization” frame selection type without any problems, a frame must not contain more than 21 referenced visualizations.

For more information, see also the description for the “Input configuration” of an element: Action “Switch frame visualization”.

---

### Table 284: “Available Visualizations”

<table>
<thead>
<tr>
<th>“By Visualization Name”</th>
<th>The list of available visualizations of the project and libraries is sorted alphabetically.</th>
</tr>
</thead>
<tbody>
<tr>
<td>“By Type or Instance”</td>
<td>The list of available visualizations of the project and libraries is sorted by type or instance.</td>
</tr>
<tr>
<td>Input field for a filter</td>
<td>If a filter text is specified, then only those visualizations whose names contain the filter text are listed.</td>
</tr>
<tr>
<td>()</td>
<td>Project with project visualizations below it</td>
</tr>
<tr>
<td>()</td>
<td>Library with project visualizations below it</td>
</tr>
</tbody>
</table>

### Table 285: “Selected Visualizations”

<table>
<thead>
<tr>
<th>“Add”</th>
<th>Click the symbol to add a visualization to the list of selected visualizations.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Requirement: This is selected in “Available Visualizations”.</td>
</tr>
<tr>
<td></td>
<td>Hint: To add a visualization, double-click a visualization in “Available Visualizations”.</td>
</tr>
<tr>
<td>“Delete”</td>
<td>Click the symbol to delete a visualization from the list.</td>
</tr>
<tr>
<td></td>
<td>Requirement: This is selected in “Selected Visualizations”.</td>
</tr>
</tbody>
</table>

The visualizations are automatically numerically indexed via the order in the list. The top visualization has the index 0. The next visualization has the index 1 and so on.

Note: A frame and a tab control element use the variables specified in the index of the “Switch frame variable” property.

| “Move Up” | Click the symbol to move a visualization up in the list. |
|           | Requirement: This is selected in “Selected Visualizations”. |
| “Move Down” | Click the symbol to move a visualization down in the list. |
|           | Requirement: This is selected in “Selected Visualizations”. |

---

See also

- Chapter 1.3.5.18.1.6 “Visualization element ‘Frame’” on page 1287
- Chapter 1.3.5.18.1.10 “Visualization element ‘Tab control’” on page 1316
- “Element property ‘Switch frame variable’” on page 1298

---

**Command ‘Background’**

**Symbol:** 🎨

**Function:** The dialog “Background” opens. You can define here whether the background of the visualization is colored or displayed with an image.

**Call:** Menu “Visualization”, context menu
See also

- §Chapter 1.3.5.3.7 “Designing a background” on page 1124

**Dialog 'Background'**

**Table 286: “Color Settings”**

| “Use Color” | ☑: Background in color | Color defined as a style color or as a fixed value.

**Table 287: “Image Setting”**

| “Use Image” | ☑: Display of a background image | Reference to an image from an image pool in the project, formally specified as an instance path: <Name of the image pool>..<ID>

Example:
- ImagePool_A.Factory
- ImagePool_B.ID_B

**Command 'Multiply Visu Element’**

Symbol: 🠌

**Function:** The command opens the “Multiply Visu Element” dialog, which contains a configuration derived from the template element and the array declaration. You can rearrange the elements here, as well as their quantity and the index access to the array data. When you exit the dialog, a field of similar elements is created from the template element. In the properties of the new elements, array variables are now configured with precise array indexes. These new elements are those in which you have configured an array variable with index access placeholders in the template.

**Call:** Menu bar: “Visualization”; context menu

**Requirement:** The visualization is active and a configured template element is selected.

**Dialog 'Multiply Visu Element’**

**Table 288: Tab “Basic Settings”**

| “Total number of elements” | The total number is determined by the index range of the placeholders, including the setting on the “Advanced Settings” tab. The layout of the elements can be one-dimensional (as a column or row) or two-dimensional (as a table field).

| “Horizontal” | Number of elements per row |

Default: Number of array components (index range) of the placeholder $\text{FIRSTDIM}$

**Example for array:** axLampIsOn: ARRAY[0..4] OF BOOL; = 5
### “Vertical”

Number of rows required for the layout of all elements

Default

- When using index access placeholder \$FIRSTDIM\$:
  - If the index range of the placeholder is less than five, then the layout of elements is horizontal. If the index range is greater than five, then the layout the elements is quadratic whenever possible.
- When using index access placeholders \$FIRSTDIM\$ and \$SECONDDIM\$:
  - The number of horizontal elements is equal to the number of index ranges specified by the placeholder \$FIRSTDIM\$. The number of vertical elements is equal to the number of index ranges specified by the placeholder \$SECONDDIM\$.

### “Offset between elements”

Distance between the new elements; affects the positions of the new elements

- “0”: The frames of the elements overlap by one pixel.
- “1”: The elements touch.
- “<n>”: A distance of \(n-1\) pixel is visible between the elements.

### “Horizontal”

Distance between the elements within a row (in pixels)

Example: 2 for a distance of one pixel

### “Vertical”

Distance between the elements within the columns (in pixels)

Example for a distance of three pixels: 4

### “Arrangement of elements”

Origin from which the new elements are positioned and arranged

If “Vertical” or “Horizontal” <> 1

- “From top left”
- “From top right”
- “From bottom left”
- “From bottom right”

If “Horizontal” or “Vertical” = 1

- “From top”
- “From bottom”

### “Orientation”

Determines the layout of the elements in the field (row by row, or column by column)

- “Line by line”
- “Column by column”

### “Preview”

Displays the set layout and orientation of the elements as an arrow
Based on the template element, the precise index for accessing the array variable is calculated for each new element. The calculation is based on the array index limits as specified in the array declaration. The settings are also taken into account here.

**“1st dimension”** Calculation guideline for the index of the first dimension that replaces $FIRSTDIM$

The first new element gets the value specified below in “Start index” in the first dimension. The other elements each get an index incremented by “Increment” until an index is calculated for all elements.

Example
- “Start index”: 1
- “Increment”: 1

**“2nd dimension”** Calculation guideline for the index of the second dimension that replaces $SECONDDIM$

The first new element gets the value specified below in “Start index” in the second dimension. The other elements each get an index incremented by “Increment”.

Example
- “Start index”: 1
- “Increment”: 1

“OK” First, it is validated whether the calculated indices are in the index range of the array variable. If so, then the elements that match the template element are created and arranged as a field (row, column, or table). The placeholder indexes are replaced by the calculated indexes.
Example

Declaration of array variables

```plaintext
VAR
  asTexts_Example: ARRAY[1..2,1..2] OF STRING :=
    ['1A Text', '2A Text',
     '1B Text', '2B Text'];
  asToolTips_Example: ARRAY[1..2,1..2] OF STRING :=
    ['1A Tooltip', '2A Tooltip',
     '1B Tooltip', '2B Tooltip'];
  axUserInput_Example: ARRAY[1..2,1..2] OF BOOL;
END_VAR
```

Visualization with template element and its property configuration

Table 290: Dialog 'Multiply Visu Element'

<table>
<thead>
<tr>
<th>Tab</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Basic Settings&quot;</td>
<td></td>
</tr>
<tr>
<td>&quot;Total number of elements&quot;</td>
<td></td>
</tr>
<tr>
<td>&quot;Horizontal&quot;</td>
<td>2</td>
</tr>
<tr>
<td>&quot;Vertical&quot;</td>
<td>2</td>
</tr>
</tbody>
</table>
"Offset between elements"

"Horizontal"  2
"Vertical"  2

"Arrangement of elements"  "From top left"
"Orientation"  "Line by line"

Tab "Advanced Settings"

"Array access"

"1st dimension"

"Start index"  1
"Increment"  1

"2nd dimension"

"Start index"  1
"Increment"  1

Visualization at runtime:

See also

●  § Chapter 1.3.5.9.2 “Configuring and multiplying visualization elements as templates” on page 1156

Command 'Configure Display Settings of the Trend'

Symbol: 

**Function:** When you execute this command in “Visualization” or in the context menu, the “Edit Display Settings” dialog box opens.
Call:
- Menu bar: “Visualization”
- Context menu of a “Trend” element in the visualization editor
- Property “Display Settings”

**Requirement:** A trend is selected in the active visualization editor.

### Tab “X-Axis”

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Grid”</td>
<td>✓: Trend diagram with grid lines in the x-direction in the selected color</td>
</tr>
<tr>
<td>“Font”</td>
<td>Font for the axis label</td>
</tr>
</tbody>
</table>

### Tab “Y-Axis”

**Table 291: “Display mode”**

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Auto”</td>
<td>✓: The visualization scales automatically.</td>
</tr>
<tr>
<td>“Fixed”</td>
<td>❌: Fixed range from “Minimum” to “Maximum”</td>
</tr>
<tr>
<td>“Minimum”</td>
<td>Start value of the range</td>
</tr>
<tr>
<td></td>
<td>Requirement: “Fixed” is activated.</td>
</tr>
<tr>
<td>“Maximum”</td>
<td>End value of the range</td>
</tr>
<tr>
<td></td>
<td>Requirement: “Fixed” is activated.</td>
</tr>
<tr>
<td>“Grid”</td>
<td>✓: Trend diagram with grid lines in the y-direction in the selected color</td>
</tr>
<tr>
<td>“Description”</td>
<td>✓: Text for labeling the y-axis (for example, DC/mA)</td>
</tr>
</tbody>
</table>

**Table 292: “Tick marks”**

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Fixed spacing”</td>
<td>✓: Axis scale with tick marks for “Distance” and “Subdivisions”</td>
</tr>
<tr>
<td>“Distance”</td>
<td>Distance between tick marks (example: 2)</td>
</tr>
<tr>
<td>“Subdivisions”</td>
<td>Number of subdivisions between tick marks (example: 4)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Font”</td>
<td>Font for the axis label</td>
</tr>
</tbody>
</table>

**Table 293: “Background”**

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>“From visualization style”</td>
<td>Background color as defined in the visualization style</td>
</tr>
<tr>
<td>“Draw background”</td>
<td>Background color that is selected in the lower input field</td>
</tr>
<tr>
<td>“No background”</td>
<td>Trend diagram with transparent background</td>
</tr>
<tr>
<td>Background color of the trend diagram</td>
<td>Requirement: “Draw background” is activated.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Reset”</td>
<td>Resets the settings to the default settings</td>
</tr>
<tr>
<td>“Use as default”</td>
<td>Stores settings as default</td>
</tr>
</tbody>
</table>
### Add y-axis
Extends the trend diagram by one y-axis.

Result: The “Trend recording” editor contains an extended selection of y-axes in the “Additional axes” option of the “Variable Settings”.

### Delete y-axis
Deletes the y-axis of the visible tab.

See also
- Chapter 1.3.5.19.2.12 “Command ‘Configure Display Settings of the Trend’” on page 1570
- Editor ‘Trend recording’

### Command ‘Configure Trace’

**Symbol:**

**Function:** This command opens the “Trace Configuration” dialog box.

**Call:** Context menu of the visualization element; “Trace” property of the visualization element.

**Requirement:** An element of type “Trace” is open in the editor.

### Dialog box 'Trace Configuration'

The tree view shows the trace configuration and allows navigation.

The top entry contains the trace name. When this entry is selected, the “Record Settings” group appears in the adjacent view.

An entry is located here for each variable that data was recorded continuously. When a variable is selected, the “Variable Settings” group appears in the adjacent view.

### Add variable
Adds a new entry to the trace configuration.

Result: A blank configuration appears next to the new variable under “Variable Settings”. You configure the variable there.

### Delete variable
Removes the selected variable.

### 'Recording Settings'

A trigger can be configured in the trace only.

<table>
<thead>
<tr>
<th><strong>“Task”</strong></th>
<th>Task where data was recorded.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>“Record condition”</strong></td>
<td>Recording condition for which the application records data in runtime mode: Variable (BOOL)</td>
</tr>
<tr>
<td><strong>“Comment”</strong></td>
<td>Example: Acquiring data only when all conditions are true.</td>
</tr>
<tr>
<td><strong>“Resolution”</strong></td>
<td>Measure for the time stamp that is recorded per data set.</td>
</tr>
<tr>
<td></td>
<td>- “ms”: Time stamp (in milliseconds).</td>
</tr>
<tr>
<td></td>
<td>- “µs”: Time stamp (in microseconds) for a task cycle time of 1 ms or less</td>
</tr>
<tr>
<td><strong>“Automatic restart”</strong></td>
<td>[✓]: Recording starts automatically as soon as the trace has been started one time and then the controller was restarted. The trace configuration and the contents of the trace buffer are saved persistently to a file on the target system. Format: .trace.csv</td>
</tr>
<tr>
<td><strong>“Display”</strong></td>
<td>The “Edit Appearance” dialog box opens.</td>
</tr>
</tbody>
</table>
The “Advanced Trace Settings” dialog box opens.

The “Copy Settings from Trace Instance” dialog box opens. If you have already created an existing trace configuration from a trace object, then you can copy the configuration data to the visualization element. To do this, select the respective object.

See also
- Chapter 1.3.5.19.3.19 “Dialog ‘Display Settings’” on page 1606
- Chapter 1.3.5.19.3.18 “Dialog ‘Advanced Trace Settings’” on page 1606
- Chapter 1.3.5.18.1.34 “Visualization element ‘Trace’” on page 1466

<table>
<thead>
<tr>
<th>'Variable Settings'</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>&quot;Variable&quot;</strong></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>

| **"Parameters"** | Parameter whose value is acquired. |
|                  | ☑: Input assistance lists are valid parameters of the PLC. |

| **"Attached axis"** | Y-axis of the trace diagram for the "Variable". |
|                    | ☑: Selection of the standard y-axis and the additional configured y-axes |
| Note: The additional configured y-axes are configured in the "Edit Display Settings" dialog box. |

| **"Display variable name"** | ✅: The trace graphs are displayed in tooltip with their variable names. |
|                            | If a text is also specified in "Description", then the text is displayed first with the variable names in parentheses. |
| Example: Sensor A (PLC_PRG.iSensor_A) |
| If "Description" does not contain any text, then the "Display Variable Name" property is activated automatically. Then only the name is displayed (example: PLC_PRG.iSensor_A). |
| ☑: The trace graphs are displayed in tooltip without their variable names. Only the text in "Description" is displayed. |

| **"Description"** | Text for the tooltip. It is displayed when a visualization user moves the cursor in the trace diagram. |
|                  | Example: Sensor A |
|                  | The text is also entered into the “GlobalTextList” object and can be translated there. |

| **"Color"** | Color of the graph in the diagram. |

<p>| <strong>&quot;Line type&quot;</strong> | Representation of the graph as a line chart |
|                | ● &quot;Line&quot;: Values are linked to form a line. |
|                | ● &quot;Step&quot;: Values are linked in the form of steps. |
|                | ● &quot;None&quot;: Values are not linked. |</p>
<table>
<thead>
<tr>
<th><strong>“Line width”</strong></th>
<th>In pixels</th>
<th>Example: 1</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>“Line style”</strong></td>
<td>The display of the line is solid, dash, dot, dash-dot, or dash-dot-dot.</td>
<td></td>
</tr>
</tbody>
</table>
| **“Dot type”** | Representation of the graph as a scatter chart. This configuration entry with the “Line type” determines the appearance of the graph.  
  ● “Dot”: Each value as a dot.  
  ● “Cross”: Each value as a cross.  
  ● “None”  
  Note: For “Dot” or “Cross”, a paint buffer overflow can result from many recorded variables. |
| **“Warning at minimum”** | [ ] When below the lower limit, the visualization shows the trace graphs in the alert color. |
| **“Critical lower limit”** | Minimum Value | Example: 10. |
| **“Color”** | Warning color on falling below the limit |
| **“Warning at maximum”** | [ ] When above the upper limit, the visualization shows the trace graphs in the alert color. |
| **“Critical upper limit”** | Maximum value | Example: 90 |
| **“Color”** | Warning color on exceeding the limit |
| **“Dynamic appearance options”** | |
| **“Variable for visibility”** | Variable (BOOL) or as bit access. This controls the visibility of the variables in the trace diagram.  
  ● TRUE: Visible  
  ● FALSE: Invisible |

See also  
● © Chapter 1.3.5.19.2.13 “Command ‘Configure Trace’” on page 1572

**Command ‘Export Trace Configuration’**

**Function:** This command opens the “Export Trace Configuration” dialog box.

**Call:** context menu (right-click) the upper node in the tree view of the trace configuration.

**Requirement:** The dialog box “Trace Configuration” is active and the name of the trace configuration is selected in the tree view (example: Visu_Trace1).

**Dialog box 'Export Trace Configuration’**

This dialog is used for saving the trace configuration to a text file that can be read by the run-time system.

<table>
<thead>
<tr>
<th><strong>“File name”</strong></th>
<th>Name of text file to be created.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>“File type”</strong></td>
<td>“Trace file (*.trace)”: Format that the runtime system component CmpTraceMgr expects for reading.</td>
</tr>
</tbody>
</table>
Command 'Insert Elements for Controlling Trace'

Symbol:  

**Function:** The command opens the “Trace Wizard” dialog. In this dialog, you select predefined visualization elements for controlling the trace recording. These elements are then inserted as configured into the visualization editor.

**Call:** Menu bar: “Visualization”; context menu of the trace element.

**Requirement:** The view is active and a trace element is selected.

---

**Dialog 'Trace Wizard'**

<table>
<thead>
<tr>
<th>“Control variable”</th>
<th>Corresponds to the “Control variables” property that is available in the element properties of the trace element.</th>
</tr>
</thead>
<tbody>
<tr>
<td>✓</td>
<td>The control element for this trace control variable is created in the visualization editor.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>“Variable”</th>
<th>Project variables that are assigned to the control element below the “Input configuration” property. In addition, the project variables are declared as local variables in the visualization when needed (in the interface editor).</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>This list corresponds to the assignments that are defined in the element properties of the trace element. If nothing is configured in the properties of the trace element (no project variables assigned as control variables), then a pre-allocation is offered with default variable names.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>“Type of element to insert”</th>
<th>For a Boolean variable, this element can be inserted as a button or rectangle. For a string variable, a rectangle or a text field is provided.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>“OK”</th>
<th>At the closing of the dialog, the selected control elements are inserted into the visualization editor and (when needed) its control variables are created as local variables of the visualization. They are declared in the interface editor and they are used by the control element (property “Input configuration” ➤ Toggle ➤ Variable”) and by the trace element (“Control variables” property). The control element writes to the variable and the trace element reads the variable.</th>
</tr>
</thead>
</table>

---

**Example**

**Standard control variables:**

```plaintext
VAR
  bResetTrigger : BOOL;
  bStart : BOOL;
  bStop : BOOL;
  bStore : BOOL;
  sStoreFilename : STRING;
  bRestore : BOOL;
  sRestoreFilename : STRING;
END_VAR
```

---

See also

-  § Chapter 1.3.5.19.2.13 “Command ‘Configure Trace’” on page 1572
-  § Chapter 1.3.5.10.1 “Getting started with trace” on page 1163
Command 'Configure Display Settings of the Trend'

Symbol: ☐

Function: When you execute this command in "Visualization" or in the context menu, the “Edit Display Settings” dialog box opens.

Call:
- Menu bar: “Visualization”
- Context menu of a “Trend” element in the visualization editor
- Property “Display Settings”

Requirement: A trend is selected in the active visualization editor.

Tab “X-Axis”

| “Grid”         | Trend diagram with grid lines in the x-direction in the selected color |
| “Font”         | Font for the axis label |

Tab “Y-Axis”

Table 294: “Display mode”

| “Auto” | ☐: The visualization scales automatically. |
| “Fixed” | ☐: Fixed range from “Minimum” to “Maximum” |
| “Minimum” | Start value of the range |
| “Maximum” | End value of the range |
| “Grid” | ☐: Trend diagram with grid lines in the y-direction in the selected color |
| “Description” | ☐: Text for labeling the y-axis (for example, DC/mA) |

Table 295: “Tick marks”

| “Fixed spacing” | ☐: Axis scale with tick marks for “Distance” and “Subdivisions” |
| “Distance” | Distance between tick marks (example: 2) |
| “Subdivisions” | Number of subdivisions between tick marks (example: 4) |

| “Font” | Font for the axis label |

Table 296: “Background”

| “From visualization style” | Background color as defined in the visualization style |
| “Draw background” | Background color that is selected in the lower input field |
| “No background” | Trend diagram with transparent background |
| Background color of the trend diagram | Requirement: “Draw background” is activated. |

| “Reset” | Resets the settings to the default settings |
| “Use as default” | Stores settings as default |
“Add y-axis” | Extends the trend diagram by one y-axis
---|---
Result: The “Trend recording” editor contains an extended selection of y-axes in the “Additional axes” option of the “Variable Settings”.

“Delete y-axis” | Deletes the y-axis of the visible tab.

See also
- Chapter 1.3.5.19.2.12 “Command 'Configure Display Settings of the Trend’” on page 1570
- Editor ‘Trend recording’

Command 'Edit Trend Recording'

Symbol: 🔄

Function: This command opens the “Trend Recording” object.

Call:
- Menu bar: “Visualization”
- Context menu of a “Trend” element in the visualization editor
- Property “Trend recording”

Requirement: An element of type “Trend recording” is selected in the visualization editor.

See also
- Object Trend recording

Command 'Insert Elements for Controlling the Trend'

Symbol: 🔄

Function: When you execute this command in “Visualization”, the “Trend Wizard” dialog box opens.

Call: Menu bar: “Visualization”; context menu of a “Trend” element in the visualization editor.

Requirement: A trend is selected in the active visualization editor.

Dialog 'Trend wizard'

Each row of the table contains a control element that can assigned to the trend. The elements are placed in the visualization next to the trend. The control elements are saved in the “Assigned control elements” property and can be modified there.

<table>
<thead>
<tr>
<th>“Attached control element”</th>
<th>The associated element is available in the visualization and connected with the trend via the property “Assigned Visu element”. The element is inserted into the visualization.</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Position”</td>
<td>Position of the control element in relation to the trend.</td>
</tr>
<tr>
<td>“Type of element to insert”</td>
<td>Drop-down list with the installed types of the control element</td>
</tr>
<tr>
<td>“Instance name”</td>
<td>Instance name of the control element</td>
</tr>
</tbody>
</table>

See also
-
Command 'Visualization Element Repository'

Symbol: 

**Function:** This command opens the “Visualization Element Repository” dialog box opens for editing the storage location and visualization profile.

**Call:** Menu bar: “Tools”.

**Requirement:** No project is open.

---

The visualization element repository is used for creating a visualization profile or visualization extension. This is necessary when developing your own visualization elements with the CODESYS VisuElement Toolkit. The CODESYS VisuElement Toolkit is required for this with a valid license. End users who do not wish to create their own visualization elements can use this dialog to find out which elements are included in which visualization profile. A reconfiguration of the storage location for a repository is also important only for element developers.

---

**NOTICE!**

1. Only an empty directory can be selected as a new storage location for a repository.

2. The "System" repository cannot be modified. This is indicated by the entry in italics in the repository list.

---

Currently only a single version of an element can be installed.

---

**Dialog box 'Visualization Element Repository'**

**Table 297: Editing the repository**

<table>
<thead>
<tr>
<th>“Location”</th>
<th>Storage location for the repository in the file system. The drop-down list contains the configured repositories for visualization elements.</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Edit locations”</td>
<td>Opens the “Edit Repositories” dialog box for modifying the repository currently selected in “Location” or for creating a new repository.</td>
</tr>
</tbody>
</table>

**Table 298: “Profile or extension selection”**

A profile is a collection of visualization elements in a specific version. These elements originate from one or more libraries. They are available in the toolbox of the visualization editor when the profile is used in the project. You can use an extension to add a specific selection of elements to an existing profile.

Creating and editing a profile is possible only if the CODESYS VisuElement Toolkit is installed. In this case, the buttons on the right side of the dialog box can be used.

| “Create or update profile” | You can configure a new profile in the dialog or modify an existing one. Then, the “New”, “Copy”, and “Delete” buttons are operable, as well as the “Installed Elements” and “Available Elements” views. |
"Create or update extension" In the dialog, you can configure an extension for the selected profile. Another drop-down list “Extension” appears with all currently available extensions. To configure a new extension, use the “New” or “Copy” buttons (see below). In both cases, the “Specify Visualization Extension” dialog box opens for you to define a new extension. In this dialog box, the “Name”, “Company”, and “Version” of the extension are displayed. Version syntax: Sequence of numbers and points with a number at the end.

“Profile” Currently selected profile. The drop-down list provides all profiles available in the repository set above.

“Extension” The extension that is currently selected for the specified profile. The drop-down list provides all extensions available for the profile.

“New” Pressing the button opens the “Specify Name of Visualization Profile” dialog or the “Specify Visualization Extension” dialog. Specify a unique name for the new profile, or for an extension also the company name and the version. CODESYS automatically enters the previously used name, appended with “_0”.

The “Installed elements” list is empty.

“Copy” Pressing the button opens the “Specify Name of Visualization Profile” dialog or the “Specify Visualization Extension” dialog (see above: “New”).

The elements of the selected profile are accepted and they appear in the “Installed elements” view.

“Delete” The currently set profile or the extension is deleted, and then the drop-down list is removed.

Table 299: “Installed Elements”

<table>
<thead>
<tr>
<th>Name, Vendor, Library</th>
<th>Elements that are assigned to the selected profile.</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Uninstall”</td>
<td>All elements currently selected in the list are uninstalled and removed from the “Available Elements” list.</td>
</tr>
<tr>
<td>“Update code”</td>
<td>The list is refreshed with any changes in the implementation code of the library POU.</td>
</tr>
<tr>
<td>“Update all”</td>
<td>The list is refreshed with any changes in the implementation code and in the interfaces (declaration part) of the library POU.</td>
</tr>
</tbody>
</table>

Table 300: “Available Elements”

<table>
<thead>
<tr>
<th>Name</th>
<th>Elements that are available in the system and can be installed into the current profile or extension. The selection depends on the installed element libraries and element packages.</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Library”</td>
<td>The tree structure displays the libraries with the contained elements below them. Elements display in green are already installed for the specified profile or extension. “Profiles” shows the elements installed for the profiles.</td>
</tr>
<tr>
<td>“Vendor”</td>
<td></td>
</tr>
<tr>
<td>“Version”</td>
<td></td>
</tr>
<tr>
<td>“Repositories”</td>
<td></td>
</tr>
<tr>
<td>“Profiles”</td>
<td></td>
</tr>
<tr>
<td>“Install element”</td>
<td>The elements selected in the list are added to the “Installed Elements” view. Existing elements are overwritten.</td>
</tr>
<tr>
<td>“Install library”</td>
<td>The “Library Repository” dialog box opens where another library can be installed in order to accept its elements in the “Available Elements” view.</td>
</tr>
</tbody>
</table>
**Table 301**

| “Note current library versions only” | :yellow: When refreshing the list, only the most current version of the library is searched, not all libraries. |
| “Overwrite profiles without prompting” | :yellow: For actions that change the profile, the usual prompt does not appear for confirming the change. |

**Dialog box 'Edit Repositories'**

| “Storage location, Name” | For managing the visualization elements, one or more repositories can be used. All currently defined storage locations are listed here with file path and name. The order from top to bottom is also the search order for the visualization elements. File path and name of the storage location selected previously in the “Repository for Visualization Elements” dialog. Note: A storage location "System" is always defined automatically, which cannot be modified or deleted. |
| “Add” | Opens the “Storage Location for Repository” dialog for creating a new storage location or for editing the current storage location. Specify: “Storage location” (file path of an empty directory) and “Name”. The name is symbolic (example: “Elements category 1”). |
| “Edit” | | |
| “Remove” | Deletes the repository currently selected in the repository list. |
| “Move Up, Move Down” | Moves the entries within the list. Note: The repositories are searched from top to bottom. |

**Command 'Visualization Style Repository'**

Symbol: ![visor](image)

**Function:** This command opens the “Visualization Styles” dialog box. It makes it possible to edit visualization style repositories.

**Call:** Menu bar: “Tools”.

See also
- Chapter 1.3.5.17.2 “Managing visualization styles in repositories” on page 1221

**Dialog Box 'Visualization Styles'**

| “Storage location” | Name of the currently selected repository Preset: “System” ![visor](image): Lists the repositories installed in the development system. |
| “(...)” | Storage location of the repository Example: (C:\ProgramData\CODESYS\Visualization Styles) |
| “Edit locations” | The “Edit Repository Locations” dialog box opens. |
Table 302: “Installed Visualization Styles”

| “Company” | When a company name is specified here, the tree view is filtered and only the styles of the selected company are listed.\n| Preset: “(All companies)” It is not filtered.\n| ❖: Lists all companies that are specified in the styles. |

Windows with styles | Tree view of all versions of the installed visualization styles in the selected repository |

“Display localized names” | ☑: The style name is localized and displayed in the language that is set in CODESYS.\n| ☐: The style is display as the source name. |

“Install” | The “Select Visualization Style(s)” dialog box opens. |

“Uninstall” | The selected style version is removed from the repository. |

“Preview” | The windows closes. A preview is displayed of the selected style in the selected version. Specific elements are displayed in the style. |

Dialog box ‘Edit Repository Locations’

Table 303: “Repositories (elements are searched in that order)”

| “Location” | Storage location of the configured repository on the development system\n| Example: C:\ProgramData\CODESYS\Visualization Styles |

| “Name” | Preset: System |

| “Add” | The “Repository Locations” dialog box opens. It makes it possible to manage other repositories. |

| “Edit” | |

| “Remove” | |

| “Move Up” | The order in the list of repositories is adapted. It defines the processing order when searching for elements. |

| “Move Down” | |

Command ‘Add Visual Element’

Function: The command opens a menu containing all available visualization elements as menu items.

Requirement: You have configured the command in the dialog box “Customize” in a way that you have a call in a (any) menu.

When you select an element in the menu, the element is added in the visualization editor in the upper left corner.

See also
\n● Chapter 1.3.5.3.1 “Selecting an element” on page 1112
\n● Dialog box ‘Customize’

Command ‘Select None’

Function: The command cancels at once any selection in the current visualization editor.

Requirement: You have configured the command in the dialog box “Customize” in a way that you have a call in a (any) menu.
Command 'Add Elements for Alarm Acknowledgement'

Symbol: 🔴

Function: This command adds buttons automatically to the visualization for acknowledging alarms. It opens an assistant for inserting controls below the table.

Call: Menu bar: “Visualization”; context menu of visualization element "Alarm table"

Requirement: An "Alarm table" visualization element is selected.

Dialog box 'Alarm Table Wizard'

| "Type of element(s) to insert" | ● “Button”  
|                               | ● “Rectangle” |
| "Action"                      | 🔄 A button or a rectangle with the selected function is added to the visualization. |
| "Variable"                    | If you have already specified a variable for an action, then this is displayed here in the "Variable" column. If you have not defined a variable yet, then a local visualization variable is created automatically. |

See also
- % Chapter 1.3.5.18.1.22 “Visualization element ‘Alarm table’” on page 1395
Dialog 'Access Rights'

**Function:** This dialog defines the permissions of user groups for a visualization element.

**Call:** Click in the “Value” field of the “Access Rights” element property of a visualization element.

**Requirement:** A visualization element is selected in a visualization element and the “Properties” is open.

<table>
<thead>
<tr>
<th>“User Groups”</th>
<th>Groups that were configured in the “Visualization Manager” (tab “User Management ➔ Groups”).</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Operable”</td>
<td>☑: The visualization element is available with full functionality.</td>
</tr>
<tr>
<td>“Only Visible”</td>
<td>☑: The visualization element is visible only and does not provide any functionality.</td>
</tr>
<tr>
<td>“Invisible”</td>
<td>☑: The visualization element is not displayed.</td>
</tr>
<tr>
<td>“Group hierarchy is used”</td>
<td>Display whether the option “Use group hierarchy” is activated in the “Visualization Manager” (tab “User Management ➔ Settings”).</td>
</tr>
</tbody>
</table>

A group of a higher hierarchy cannot have fewer permissions for an element than an element of a lower hierarchy.

If no user is logged in, then the permissions apply for the visualization elements that are configured for the user group “None”. If the permissions for a visualization element is restricted, then the group “None” should be granted the lowest permissions.

See also

●

Dialog 'Add Visualization'

**Function:** This dialog is used for creating a new object of type “Visualization”.

**Call:** “Project ➔ Add Object ➔ Visualization”; context menu of an application

**Requirement:** An application is selected in the device tree.

| “Name” | Name of the visualization  
Example: Visu_A |
|--------|-------------------------------------------------------------------|

The following settings are displayed only when you add an object of type “Visualization” to the project for the first time.

<table>
<thead>
<tr>
<th>“Symbol library”</th>
<th>List of all installed symbol libraries</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Assigned”</td>
<td>☑: Symbol library is selected</td>
</tr>
</tbody>
</table>

Tip: CODESYS manages this setting in the project settings.

<table>
<thead>
<tr>
<th>“Add”</th>
<th>CODESYS creates a new visualization, assigns the selected symbol libraries to the project, and provides them in the “ToolBox” view.</th>
</tr>
</thead>
</table>
See also

- Chapter 1.3.5.3.1 “Selecting an element” on page 1112
- Dialog ‘Project Settings’ - ‘Visualization’
- Command ‘Add Object’

Dialog ‘Updating the Frame Parameters’

**Function:** The dialog requests you, after changing an interface in the visualization references concerned, to re-assign the variables for the parameter transfer.

**Call:** The dialog appears automatically.

**Requirement:** You have changed the interface of a visualization, for example by adding an additional variable. After that you have selected either the menu command “File ➔ Save Project” or “Build ➔ Build” or opened a visualization.

<table>
<thead>
<tr>
<th>“Parameter”</th>
<th>Hierarchical structure of the interface parameters as a tree view</th>
</tr>
</thead>
<tbody>
<tr>
<td>![Image]</td>
<td>Top node of the visualization hierarchy with the name of the visualization. This contains a frame or a tab control.</td>
</tr>
<tr>
<td>![Image]</td>
<td>Name of the frame or the tab control</td>
</tr>
<tr>
<td>![Image]</td>
<td>Name of the referenced visualization</td>
</tr>
<tr>
<td>![Image]</td>
<td>Interface of the referenced visualization with the new parameters You can edit the parameter transfer here.</td>
</tr>
<tr>
<td>![Image]</td>
<td>Interface of the referenced visualization with the previously valid parameters You cannot edit the parameter transfer, but you can use it as a template.</td>
</tr>
<tr>
<td>![Image]</td>
<td>Variable for the parameter transfer (VAR_INPUT scope)</td>
</tr>
<tr>
<td>![Image]</td>
<td>Variable for the parameter transfer (VAR_IN_OUT scope)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>“Type”</th>
<th>Data type of the variable</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>Example:</strong> INT</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>“Value”</th>
<th>Variable that is transferred as a parameter and with whose value the visualization is initialized during instantiation.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>Example:</strong> PLC_PRG.iVisNr</td>
</tr>
</tbody>
</table>

If the variable lies under the current interface, which is marked in the tree view with ![Image] “(Current)”, then you can edit the parameter transfer.

- Click inside an input field to open it
- Double-click inside the field to open the input assistant
- Adopt the settings by copying assignments in the column “Value” and inserting them in another cell. Use the “Copy” and “Paste” links to do this

<table>
<thead>
<tr>
<th>“Copy”</th>
<th>Link to copy an assignment from the “Value” column.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Requirement: An assignment is selected.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>“Paste”</th>
<th>Link to insert an assignment</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Requirement: You have copied an assignment.</td>
</tr>
</tbody>
</table>
"OK"

Click on the button to close the dialog and confirm the changes made under "(Recent)".

Result: The assignment is entered in the "References" property and on the "Interface Editor" tab.

See also

●  
  ●  "Chapter 1.3.5.18.1.6 "Visualization element 'Frame'" on page 1287
  ●  "Chapter 1.3.5.19.2.1 "Command 'Interface Editor'" on page 1556

Dialog 'Configure Tags and Items'

**Function:** The dialog is for the management of the identifiers in a tree view. The assigned elements are listed underneath an identifier. User-defined identifiers can be created and their assignment to the visualization elements edited. The name of the identifier appears in the "ToolBox" view as a label on the button for opening the element selection.

**Call:** Click on symbol in the view "Tools"

See also

●  "Chapter 1.3.5.3.1 “Selecting an element” on page 1112
  ●  "Chapter 1.3.5.19.4.1.2 “View 'Toolbox'” on page 1609

**Tree view**

<table>
<thead>
<tr>
<th>&quot;Tags&quot;</th>
<th>Tree view</th>
</tr>
</thead>
</table>
| ●  "<Name>": preset identifier  
  ●  "<Name>": user-defined identifier |
  Example: "Favorite"

| + | Lists the assigned visualization elements. A selected visualization element can be removed by clicking on [Del].  
  Hint: the assignment is created in the view “ToolBox” with the help of the context menu of a selected element. |

| "Active" | a button for the identifier is visible in the "ToolBox" view. |

**Toolbar**

| + | The dialog "Add Tag" opens. |
| − or [Del] | The tag selected in the tree view is removed. If you close the dialog with "OK", the button is also removed from the view "ToolBox". |

**Dialog 'Add Tag**

**Call:** Click on the + symbol in the dialog “Configure Tags and Items”

| "Name" | Name of the tag  
  Example: tagA |
| "Description" | Example: Tagged with A |
**Dialog 'Gradient Editor'**

**Function:** The dialog is for setting the color gradient of visualization elements. If you define two colors, the color graduates between them. If you only select 1 color, the color graduates within this color through its brightness. The detailed settings are for a special specification of the initial position and the angle of the color gradient.

**Call:** Click in the value field of the property "Gradient settings"

**Requirement:** You have selected a visualization element in the editor that has the property “Gradient settings”.

| "Gradient type" | ● “Linear”  
| ● “Radial”  
| ● “Axial”: The color gradient runs along an axis, with the colors extending perpendicular to the axis on both sides. |
| "Color 1” | First color of the gradient. |
| "Color 2” | Second color of the gradient. |
| “Transparency” | Transparency of the associated color. Permissible values: Integers in the value range from 255 to 0. 255: The color is opaque. 0: The color is fully transparent. |
| “Angle (degrees)” | Requirement: “Linear” or “Axial” color gradient. |
| “Center X (%):” | Requirement: “Radial” color gradient. X-position of the center point (0 – 100%) |
| “Center Y (%):” | Requirement: “Radial” color gradient. Y-position of the center point (0 – 100%) |
| “Use one color” | Color gradient between “Color 1” and the same color with a different brightness. |
| “Brightness” | Requirement: The option “Use one color” is selected. Setting from 0 (black) to 100 (white) |
| “Use two colors” | Color gradient between the two selected colors “Color 1” and “Color 2”. |

See also
- ⇨ Chapter 1.3.5.3.3 “Assigning a color” on page 1115
- ⇨ Chapter 1.3.5.8.3 “Animating a color display” on page 1152

**Dialog 'Input Configuration'**

**Symbol:** 📅

**Function:** The dialog is used to assign input actions to specific input events. It also includes specific settings for the selected input action.

**Call:** Click “Configure” in the “Input configuration” property.

**Requirement:** An element is selected in the editor.

See also
### Input action 'User Management'

| Dialogs and actions | Configures which one of the possible user management dialogs or which action follows the input event  
|---------------------|---|
| Note | The dialog used at runtime is configured in the "Dialog settings" tab of the Visualization Manager.  
| See also |  
| - "Login dialog" |  
| - "Change password dialog" |  
| - "Change configuration dialog" |  
| Default | Dialogs from the VisuUserManagement library  
| Login | The login prompt opens.  
| Default | VisuUserManagement.VUM_Login in "Login dialog"  
| Logout | The current user is logged out.  
| Change user password | The dialog for changing the password opens.  
| Default | VisuUserManagement.VUM_ChangePassword in "Change password dialog"  
| Open user configuration | The dialog opens for changing the configuration.  
| Default | VisuUserManagement.VUM_UserManagement in "Change configuration dialog"  

See also  
- ☞ Chapter 1.3.5.5 “Setting up user management” on page 1139  
-  

### Input action 'Close Dialog'

| Dialog | The visualization of type "Dialog" that will be closed.  
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Examples</td>
<td></td>
</tr>
</tbody>
</table>
| Default dialogs of the VisuDialogs library, which is usually integrated in the project.  
| - FileOpenSave |  
| - Keypad |  
| - Login |  
| - Numpad |  
| - NumpadExtended |  
| - TextInputWithLimits |  
| Note | The setting in the object property ("Visualization" tab) of a visualization determines whether or not a visualization can be used as a dialog.  
| Result | Return value for closing the dialog.  
| Note | If there are more input actions after closing, then they configured in the "Input configuration ➔ OnDialogClosed" property of the element.  
| None | ☑ No return value
“OK”  ⚫ The set return value is returned. The return value refers to the button in the dialog. The value OK is returned for the OK button. The value Cancel is returned for the cancel button.

“Cancel”

“Abort”

“Retry”

“Ignore”

“Yes”

“No”

See also

●  
- Chapter 1.3.5.19.3.15 “Dialog 'Properties' of Visualization Objects” on page 1603

Input action 'Open Dialog'

“Dialog”  Visualization (type “Dialog”). The dialog opens.

List box with all dialogs available in the project.

Note: The “VisuDialogs” library provides visualizations (type “Dialog”).

- VisuDialogs.FileOpenSave
- VisuDialogs.Login

Transfer parameters of the dialog

“Parameter”  Interface parameter as declared in the interface editor of the visualization

Example: filelistProvider

“Type”  Data type of the parameter as declared in the interface editor of the visualization.

Example: VISU_FBFILELISTPROVIDER

“Value”  Variable (data type corresponds to the data type of the parameter). The value of the variable is read when the dialog opens and passed to the parameter.

Example: PLC_PRG.fileListProvider // Instance of function block VisuDialogs.Visu_FbFileListProvider

�� The input assistance offers all variables available in the entire project.

Here the return value of the dialog is activated for which the Var_OUTPUT variable and VAR_IN_OUT variable are written. The dialog closes afterwards.

“Update”  “and”  “parameter in case of result”  Note: The parameters are updated before the dialog is closed. Until then, the values are stored temporarily. They are stored as a copy, not as a reference.

“None”  ⚫ No return value

“OK”  ⚫ Defines the return value for which the transfer parameter is written

“Cancel”

“Abort”

“Retry”

“Ignore”

“Yes”

“No”
“Open dialog modal”  
☑️ Only the dialog processes user inputs. The remaining visualizations are blocked to user input.

“Position to open”

“Centered”  
The dialog opens in the center of the visualization.

“Position”  
The dialog opens at the position defined by “X” and “Y”.

“X”  
Position (in pixels) or variable (integer data type)

“Y”  
Position (in pixels) or variable (integer data type)

See also

Input action 'Change the language'

“Language”  
Language to be switched

Example: en

ари: The input assistance offers all languages available in the project.

Input action 'Change shown visualization'

Table 304: “Zoom to visualization”

<table>
<thead>
<tr>
<th>Visualization that is shown at the user input</th>
<th>Visualization that is selected from all available visualizations in the project or libraries.</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Assign”</td>
<td>Example: visMain</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>“Assign expression”</th>
<th>Variable (STRING) that contains the name of the visualization</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example: PLC_PRG.strVisu</td>
<td>for the following application code: strVisu: STRING := 'visMain';</td>
</tr>
</tbody>
</table>

The order in which visualizations are displayed by user inputs is saved internally. The following options use this information.

“Previous shown visualization”  
☑️: Visualization that has already been shown before the current one

Requirement: A visualization switch has already occurred.

“Next shown visualization”  
☑️: Visualization that is next in the call order after the current one.

Requirement: A visualization switch has already occurred which was called by “Previous shown visualization”.

Input action ”Execute command”  
Commands are listed here with transfer parameters that the visualization processes when an input event occurs.
“Configure commands”

Command from the list box. Click + to add the command to the list.

- “Execute program on the plc”
- “Execute program on client”
- “Print”
- “Navigate to URL (Webvisu)”
- “Create recipe”
- “Read Recipe”
- “Write Recipe”
- “Save a recipe in a file”
- “Load a recipe from a file”
- “Delete recipe”

+ The command in “Configure commands” is added to the list.

− The command is removed.

Requirement: A command is selected.

The order in the list defines the order of execution.

▼ The selected command is moved down one position in the list.

▲ The selected command is moved up one position in the list.

Table 305: Command “Execute program on the plc”

<table>
<thead>
<tr>
<th>“Command”</th>
<th>“1st parameter”</th>
<th>“2nd parameter”</th>
</tr>
</thead>
<tbody>
<tr>
<td>ExecutePlcProgram</td>
<td>'C:\programs\notepad.exe'</td>
<td>'Notes_A.txt'</td>
</tr>
<tr>
<td></td>
<td>EXE file that is executed on the controller</td>
<td></td>
</tr>
<tr>
<td></td>
<td>The program is executed on the PLC and therefore it must not be interactive or have any user interfaces.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>It is possible, for example, for a program to copy a file.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Program name with directory as STRING in single straight quotation marks</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Arguments of the program as STRING in single straight quotation marks</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Example: Name of the file that the program opens</td>
<td></td>
</tr>
</tbody>
</table>

Table 306: Command “Execute program on client”

<table>
<thead>
<tr>
<th>“Command”</th>
<th>“1st parameter”</th>
<th>“2nd parameter”</th>
</tr>
</thead>
<tbody>
<tr>
<td>ExecuteClientProgram</td>
<td>'C:\programs\notepad.exe'</td>
<td>'Notes_A.txt'</td>
</tr>
<tr>
<td></td>
<td>EXE file that is executed on the display variant (exception: WebVisu)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>The program is executed within the context of the display variant. After this, the program may be interactive and have a user interface.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Program name with directory as STRING in single straight quotation marks</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Arguments of the program as STRING in single straight quotation marks</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Example: Name of the file that the program opens</td>
<td></td>
</tr>
</tbody>
</table>

NOTICE!

If the visualization is displayed as a CODESYS WebVisu, then no program (EXE file) can be started.
Table 307: Command “Navigate to URL (Webvisu) ”

<table>
<thead>
<tr>
<th>“Command”</th>
<th>“1st parameter”</th>
<th>“2nd parameter”</th>
</tr>
</thead>
<tbody>
<tr>
<td>NavigateURL</td>
<td>'<a href="http://en.wikipedia.org">http://en.wikipedia.org</a>'</td>
<td>'replace'</td>
</tr>
<tr>
<td></td>
<td>PLC_PRG.stURL</td>
<td></td>
</tr>
</tbody>
</table>

The visualization navigates to the web page of the URL.

Requirement: The visualization is executed as a CODESYS WebVisu.

URL
- As a literal in single straight quotation marks
- As a variable (STRING)

If a parameter is not specified, then the web page is displayed in a new window or a new tab.

If 'replace' is specified, then the CODESYS WebVisu is replaced by the web page.

Table 308: Command “Read Recipe”

<table>
<thead>
<tr>
<th>“Command”</th>
<th>“1st parameter”</th>
<th>“2nd parameter”</th>
</tr>
</thead>
<tbody>
<tr>
<td>ReadRecipe</td>
<td>'RecipeDefinitionForModules'</td>
<td>'RecipeModuleA'</td>
</tr>
<tr>
<td></td>
<td>Name of the recipe definition</td>
<td>Name of the recipe</td>
</tr>
<tr>
<td></td>
<td>- As a literal</td>
<td>- As a literal</td>
</tr>
<tr>
<td></td>
<td>- As a variable (STRING)</td>
<td>- As a variable (STRING)</td>
</tr>
</tbody>
</table>

At visualization runtime, the controller reads the actual values from the variables of the recipe definition and writes them to the specified recipe. The values are saved implicitly (to a file on the controller) and shown in the recipe definition in the Recipe Manager of CODESYS. In other words, the recipe that is managed in CODESYS is updated with values from the controller.

Table 309: Command “Write Recipe”

<table>
<thead>
<tr>
<th>“Command”</th>
<th>“1st parameter”</th>
<th>“2nd parameter”</th>
</tr>
</thead>
<tbody>
<tr>
<td>WriteRecipe</td>
<td>PLC_PRG.stRecipeDef</td>
<td>PLC_PRG.stRecipe</td>
</tr>
<tr>
<td></td>
<td>Name of the affected recipe definition</td>
<td>Name of the recipe (from the recipe definition)</td>
</tr>
<tr>
<td></td>
<td>- As a literal</td>
<td>- As a literal</td>
</tr>
<tr>
<td></td>
<td>- As a variable (STRING)</td>
<td>- As a variable (STRING)</td>
</tr>
</tbody>
</table>

At visualization runtime, the values of the recipe are written to the variables on the controller as they are in the Recipe Manager.

Table 310: Command “Save a recipe in a file”

<table>
<thead>
<tr>
<th>“Command”</th>
<th>“1st parameter”</th>
<th>“2nd parameter”</th>
</tr>
</thead>
<tbody>
<tr>
<td>SaveRecipeAs</td>
<td>PLC_PRG.stRecipeDef</td>
<td>PLC_PRG.stRecipe</td>
</tr>
</tbody>
</table>
At visualization runtime, the “Save Recipe as” dialog opens and prompts the user for a file name and a storage location on the controller. The file name must not be `<recipe>.<recipe definition>`. The file extension is `.txtrecipe`.

The user can then save the file that includes the actual values from the recipe variables. If a transfer parameter is not specified in the 2nd parameter, then the file is saved without changing an implicit recipe file. If a transfer parameter is given in the 2nd parameter, then the implicit recipe file is also updated.

Note: If the “Save recipe changes to recipe files automatically” option is selected in the “Recipe Manager - General” tab, then the recipe definition in CODESYS and the implicit recipe files are kept the same automatically.

Note: Implicit (automatically generated) recipe files exist on the controller with names in the following syntax: `<recipe>.<recipe definition>.txtrecipe`. These are typically used in the application as a buffer when reading and writing recipe variables.

---

Table 311: Command “Load a recipe from a file”

<table>
<thead>
<tr>
<th>“Command”</th>
<th>“1st parameter”</th>
<th>“2nd parameter”</th>
</tr>
</thead>
<tbody>
<tr>
<td>LoadRecipeFrom</td>
<td>PLC_PRG.stRecipeDef</td>
<td>PLC_PRG.stRecipe</td>
</tr>
<tr>
<td></td>
<td>Name of the affected recipe definition</td>
<td>Name of the affected recipe definition</td>
</tr>
<tr>
<td></td>
<td>● As a literal</td>
<td>● As a literal</td>
</tr>
<tr>
<td></td>
<td>● As a variable (STRING)</td>
<td>● As a variable (STRING)</td>
</tr>
</tbody>
</table>

At visualization runtime, the “Load Recipe” dialog opens. It provides the visualization user with a file list that is located in the file system of the controller and filters by the extension `.txtrecipe`. The selected file can be downloaded. Then the recipes from the file are written to the implicit files and read to the given recipe in the recipe definition of the Recipe Manager.

Requirement: The file was created with the `SaveRecipeAs` command.

---

Table 312: Command “Create recipe”

<table>
<thead>
<tr>
<th>“Command”</th>
<th>“1st parameter”</th>
<th>“2nd parameter”</th>
</tr>
</thead>
<tbody>
<tr>
<td>CreateRecipe</td>
<td>PLC_PRG.stRecipeDef</td>
<td>PLC_PRG.stRecipe_New</td>
</tr>
<tr>
<td></td>
<td>Name of the affected recipe definition</td>
<td>Name of the new recipe</td>
</tr>
<tr>
<td></td>
<td>● As a literal</td>
<td>● As a literal</td>
</tr>
<tr>
<td></td>
<td>● As a variable (STRING)</td>
<td>● As a variable (STRING)</td>
</tr>
</tbody>
</table>

At visualization runtime, a new recipe is created in the given recipe definition.
Table 313: Command “Delete recipe”

<table>
<thead>
<tr>
<th>“Command”</th>
<th>“1st parameter”</th>
<th>“2nd parameter”</th>
</tr>
</thead>
<tbody>
<tr>
<td>DeleteRecipe</td>
<td>PLC_PRG.stRecipeDef</td>
<td>PLC_PRG.stRecipe</td>
</tr>
<tr>
<td>Name of the affected recipe definition</td>
<td>• As a literal</td>
<td>Name of the recipe</td>
</tr>
<tr>
<td></td>
<td>• As a variable (STRING)</td>
<td>• As a literal</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• As a variable (STRING)</td>
</tr>
</tbody>
</table>

At visualization runtime, the specified recipe is deleted from the recipe definition.

Table 314: Command “Print”

<table>
<thead>
<tr>
<th>“Command”</th>
<th>“1st parameter”</th>
<th>“2nd parameter”</th>
</tr>
</thead>
<tbody>
<tr>
<td>Print</td>
<td>Optional: File name for the visualization screen to be printed (example: ‘Startup image’)</td>
<td>A second parameter cannot be specified for the Print command.</td>
</tr>
</tbody>
</table>

The default “Printer” dialog opens while the visualization is running. In the dialog, you select a printer and configure additional print settings. When you confirm the dialog, the currently displayed visualization screen is printed.

Note: The command cannot be executed in the WebVisu.

See also
- Changing Values with Recipes
- Object ‘Recipe Definition’

Input action 'Switch frame visualization'

When the input event occurs, the display switches to another visualization within one frame.

<table>
<thead>
<tr>
<th>“Frame selection type”</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Switch local visualization”</td>
</tr>
<tr>
<td>“Switch to any visualization”</td>
</tr>
</tbody>
</table>

Requirement: “Switch local visualization” is selected.

<table>
<thead>
<tr>
<th>“Frame selection”</th>
<th>List of all frames that contain the active visualization. The referenced visualizations are listed below each frame, as determined in the “References” property of the respective frame.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Example:</td>
</tr>
<tr>
<td></td>
<td><img src="image.png" alt="Diagram" /></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>“Assign selection”</th>
<th>The selection in the “Frame selection” input field is accepted. Then it appears in the “Selected frame” and “Selected visualization” settings.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Requirement: A visualization is selected in the “Frame selection” input field.</td>
</tr>
</tbody>
</table>
**Selected Frame**

Name of the frame to be switched to

Example: MainArea

Hint: Use the "Assign selection" command for changing the setting here.

**Selected Visu**

Name of the switched visualization.

Example: visMainArea

Hint: Use the "Assign selection" command for changing the setting here.

---

**Frame and visualization selection**

Contains the frame to be switched to

**Assign**

Frame to be switched to (with complete path). The index determines the visualization.

Example: visMain.frameA.visB.frameB

The path is specified in the following syntax:

```
<visualization name>.<frame name> { <visualization name>.<frame name> }
```

Caution: Visualizations can be nested at any depth by means of frame elements. In order to use the "Switch to any visualization" frame selection type without any problems, a frame must not contain more than 21 referenced visualizations.

**Assign expression**

Variable (STRING). Contains the path of the frame to be switched to

Example: strFrame: STRING := 'visMain.frameA.visB.frameB';

**Index to select**

Index that determines which of the referenced visualizations is displayed

- As an integer
- As a variable (integer data type)
  
  Example: PLC_PRG.iIndex

Note: The referenced visualizations of a frame are indexed automatically according to their order.

---

Requirement: The project contains visualizations that form a structure.

See also

- "Dialog 'Frame Configuration'" on page 1564

---

**Input action**

The configuration of the input action defines how a visualization user specifies a value and to which variable the value is written.

**Input type**

How the input is prompted.

**Standard**

An input field also opens, or if necessary a virtual keyboard (when the display variant does not have a physical keyboard).

Note: The default option for text input at runtime is set in the Visualization Manager: "Dialog settings" tab, "Settings for default text input".

**Text input**

An input field appears. You use the keyboard to specify a number or a text.

Requirement: The display variant has a keyboard as input device.
<table>
<thead>
<tr>
<th>“Text input with limits”</th>
<th>An input field appears. You use the keyboard to specify a number or a text. The field also shows the value range for the input. When a limit is passed, the input value is displayed in red. Requirement: The display variant has a keyboard as input device.</th>
</tr>
</thead>
<tbody>
<tr>
<td>“VisuDialogs.Keypad”</td>
<td>A virtual keyboard opens. You use it to specify a number or a text.</td>
</tr>
<tr>
<td>“VisuDialogs.Numpad”</td>
<td>A virtual keyboard opens. You use it to specify a number.</td>
</tr>
<tr>
<td>“VisuDialogs.NumpadExtended”</td>
<td>A virtual keyboard opens. You use it to specify a number. Hexadecimal and exponential notation are also permitted here.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>“Choose variable to edit”</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>“Use text output variable”</strong></td>
</tr>
<tr>
<td><strong>“Use another variable”</strong></td>
</tr>
<tr>
<td><strong>“Initial display format”</strong></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>“Min”</th>
<th>Minimum value of the input limit. If a user specifies a lesser value, then it is not accepted.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>● As a fixed value</td>
</tr>
<tr>
<td></td>
<td>● As a variable (data type corresponds to selected variable)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>“Max”</th>
<th>Maximum value of the input limit. If a user specifies a greater value, then it is not accepted.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>● As a fixed value</td>
</tr>
<tr>
<td></td>
<td>● As a variable (data type corresponds to selected variable)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>“Dialog title”</th>
<th>Text displayed in the title bar of the dialog. Optional.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>● As a fixed string</td>
</tr>
<tr>
<td></td>
<td>Example: Insert value</td>
</tr>
<tr>
<td></td>
<td>● As a variable (STRING)</td>
</tr>
<tr>
<td></td>
<td>Example: PLC_PRG.stTitle : STRING := 'Insert value';</td>
</tr>
</tbody>
</table>

| “Password field” | ☑️: Unseen text input. *** is shown instead of the input text. |

Table 315: “Position to open input dialog”

<table>
<thead>
<tr>
<th>“Use global setting (from Visualization manager)”</th>
<th>☑️: This option is applies only for use in a TargetVisu or WebVisu. The settings are used which are available in the “Dialog settings” tab of the Visualization Manager.</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Centered”</td>
<td>☑️: The dialog opens in the center of the visualization window.</td>
</tr>
<tr>
<td>“Position”</td>
<td>☑️: The dialog opens in the visualization at the position defined here.</td>
</tr>
<tr>
<td></td>
<td>“X”, “Y”: Variable or explicit number (in pixels) for the definition of the upper left corner of the dialog in the coordinate system of the visualization window.</td>
</tr>
<tr>
<td></td>
<td>You can use the placeholders ElementRectangle.ptTopLeft.iX and .iY ElementRectangle.ptBottomRight.iY. It is replaced at runtime by the coordinates of the calling element.</td>
</tr>
</tbody>
</table>
See also

- "Chapter 1.3.5.18.2 “Placeholders with format definition in the output text” on page 1546"
- "Chapter 1.3.5.19.3.15 “Dialog 'Properties' of Visualization Objects” on page 1603"
- "Chapter 1.3.5.18.2 “Placeholders with format definition in the output text” on page 1546"

Input action 'Execute ST-Code'

<table>
<thead>
<tr>
<th>Input field</th>
<th>Editor for code as structured text</th>
</tr>
</thead>
</table>

Input action 'Toggle a variable'

<table>
<thead>
<tr>
<th>&quot;Variable&quot;</th>
<th>Variable (BOOL). It toggles between TRUE and FALSE for an input event. Example: PLC_Prg.bSwitch</th>
</tr>
</thead>
</table>

Input action 'File transfer'

Table 316: “Transfer”

| “Direction” | “From PLC to Visualization”
|-------------|----------------------------------|
| | “From Visualization to PLC”

| “Type” | “File”
|--------|-----------------
| | “Streaming”

| “File name” | Requirement: The transfer type is “File”.
|-------------|--------------------------------------------------------------------------------------------------|
| | Name of the file in the file system of the controller. The file name can be specified as a variable of type STRING or it can be specified directly in single straight quotation marks.
| | An absolute or relative path with directory information can also be specified as the file name.
| | Examples:
| | 'E:\temp\test.txt' -> absolute path.
| | 'MyData/test.txt' -> relative path. The file is stored in the PlcLogic/MyData subdirectory. "PlcLogic" is the default resolution for the directory placeholder $PLCLOGIC$. This placeholder does not also have to be specified, but it can: '$$PLCLOGIC$$/MyData/test.txt').
| | '$$VISU$$/test.txt' -> relative path. The file is stored in the PlcLogic/visu subdirectory. "visu" is the default resolution for the placeholder $VISU$). It can also be specified explicitly 'visu/test.txt'.
| | '$$PLCLOGIC$$/test.txt' -> relative path. The file is stored in the PlcLogic subdirectory. Note: If the path in the visualization is edited by means of a "Text field" element, then the masking character $ does not have to be specified: $VISU$/dummy.txt
“Streaming instance name” | Instance name of the object that yields the data for the transfer in streaming mode. The object has to implement the interface IVisuStreamReader (PLC --> visualization) or IVisuStreamWriter (visualization --> PLC). Requirement: The transfer type is “Streaming”.

“Control flags” | Variable of type DWORD. These flags control the options for data transfer. This variable can be a combination of the following flags:
- VisuElems.VisuEnumFileTransferControlFlags.UseOriginalFileName: Defines whether or not the selected file name is also used in the controller.
- VisuEnumFileTransferControlFlags.ConfirmFileOverwriteInPlc: Defines whether or not the user has to confirm when overwriting an existing file.

Table 317: “Status Variables”

| “Transfer active” | Boolean variable (optional)  
| TRUE: The transfer is in progress. |
| “Transfer successful” | Boolean variable (optional)  
| TRUE: The transfer has completed successfully. |
| “Error code” |  
| 0: No errors  
| 1: Unspecified error  
| 2: Cancellation of file dialog  
| 3: Other file transfer in progress  
| 4: Error during file transfer  
| 5: Cancellation by timeout  
| 6: File read error – The file is not available or cannot be read.  
| 7: No device support for file transfer  
| Possible causes:  
| CODESYS WebVisu: By default, file transfer is not possible.  
| Communication with a controller of a version < 3.5.11: Functionality not implemented.  
| Communication with a controller of a version >= 3.5.11: File transfer not activated (device description).  
| Note: In this case, contact the CODESYS support team. |

Behavior in online mode (type “File”)
- Transfer from the controller to the visualization (operating panel): The “Save File” dialog opens. A file name and storage location can be defined on the operating panel for the file that is specified in “File name” in the input configuration. Then the file is stored there.
- Transfer from the visualization (operating panel) to the controller: The “Open file” dialog opens. Any file can be selected here which is stored as specified in “File name” on the controller.

Dialog 'Options' - 'Visualization Styles'

Symbol: 🎨

Function: This dialog is used for configuring the display of library visualizations and visualizations in the POU’s view in the visualization editor. In addition, it is used for configuring the tab “Visualization Manager” - “Settings” (group “Style Settings”).

Call: Menu bar: “Tools ➔ Options” (‘Visualization Styles’ category).

These settings are not applied at visualization runtime. In runtime mode, only the settings of the visualization manager are available in the “Settings” tab.
See also

- “Chapter 1.3.5.17 “Applying visualization styles” on page 1216

'Style Configuration for Libraries and Global Visualizations'

<table>
<thead>
<tr>
<th>Setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Use no visualization style&quot;</td>
<td>Display without using style properties. Elements are displayed as defined by presets.</td>
</tr>
<tr>
<td>&quot;Use the following visualization style&quot;</td>
<td>Style with style properties used for displaying visualizations.</td>
</tr>
<tr>
<td>&quot;Derive visualization style automatically&quot;</td>
<td>Display with the style that was selected in the application in the visualization manager (when possible). Therefore, the display is derived from this style. It is actually possible for this to cause an incorrect display. Then the fallback solution is used.</td>
</tr>
<tr>
<td>&quot;Fallback if no visualization style could be derived &quot;</td>
<td>Another style that is applied after the selected style. Then a style property is assigned from the style specified here. This is done for element properties that could not be assigned style properties. Requirement: The selected style causes a device-specific, deficient display on the display variant.</td>
</tr>
</tbody>
</table>

'Preference Selection'

The drop-down list of “Selected style” can be configured in the visualization manager ("Settings" tab, "Style settings" group).

- "Display all versions" | All other styles of the repository, including the selected style, are listed for selection, but only in the latest version. If newer versions are installed for the selected style, then these are also listed.                                                                                  |
| □: All installed styles in all installed versions are available for selection. |

'Style for New Visualization Managers'

- "Last used: <style, version, vendor>" | Style that is selected automatically when you add a new visualization application. Note: It is actually possible that a display variant is displayed another way depending on the device despite this setting. |
- "Preset: <style, version, vendor>" |
- "<style, version, vendor>" |

Dialog 'Options' - 'Visualization User Management'

Symbol: 🌐

Function: The options define the use of visualization user management for global visualizations in the "POUs" view and for visualizations that are linked from libraries.

Call: Menu bar: “Tools”.

Requirement: A visualization user management exists.
Table 318: “User Management Configuration for Libraries and Global Visualizations”

| “Do not use visualization user management” | The affected visualizations behave as when no user management is configured. |
| “Use the following visualization user group list” | • You can edit the list.  
• The list is created in the “Visualization manager” (“User management Groups”) by clicking “Export groups for global visualizations”. |
| “Derive visualization user management automatically” | The affected visualizations use the user management configuration of the visualization manager selected here.  
The drop-down list shows all visualization managers of the project.  
If this is not possible, then the user groups are used from the option “Use the following user group list for the visualization”. |

See also

Dialog Box 'Options' - 'Visualization'

Symbol: 📖

Function: The dialog serves for the configuration of the visualization editor and during runtime it serves the configuration of the Integrated Visualization.

Call: Main menu “Tools ➔ Options”, category “Visualization”

Tab 'General'

These settings will not be applied for the following visualization clients: CODESYS TargetVisu, CODESYS WebVisu.

Table 319: “Presentation options (visualization editor in the programming system)”

| “Fixed” | The visualization maintains its original size |
| “Isotropic” | The visualization maintains its proportions |
| “Anisotropic” | The visualization adapts to the size of the visualization window |
| “Antialiased Drawing” | ☑: The visualization is drawn with the help of antialiasing methods. This applies while you are editing and also when the visualization is running as Diagnosis Visualization. |

Table 320: “Editing options”

| “Link to toggle/tap variable when appropriate” | ☑: The placeholder “<toggle/tap variable>” in the visualization element properties is enabled.  
Effect: If you drag an element having the property “Color variable ➔ Toggle color” in the visualization editor, this property will be configured with the placeholder “<toggle/tap variable>”.  
See also

- Chapter 1.3.5.18.1.11 “Visualization element 'Button'” on page 1321
- Chapter 1.3.5.18.1.6 “Visualization element 'Frame'” on page 1287
- Chapter 1.3.5.18.1.5 “Visualization element 'Image'” on page 1272
- Chapter 1.3.5.18.1.2 “Visualization element 'Line'” on page 1236
- Chapter 1.3.5.18.1.14 “Visualization element 'Scrollbar'” on page 1356
- Chapter 1.3.5.18.1.4 “Visualization element 'Pie'” on page 1260
- Chapter 1.3.5.18.1.3 “Visualization element 'Polygon', 'Polyline', 'Bézier curve'” on page 1248
- Chapter 1.3.5.18.1.1 “Visualization element 'Rectangle', 'Round rectangle', 'Ellipse'” on page 1224
- Chapter 1.3.5.18.1.15 “Visualization element 'Text field'” on page 1345

**Tab 'Grid'**

| **Visible** | ☑: The visualization editor contains a grid. The spacing of the grid lines is defined by “Size”. |
| **Active** | ☑: The visualization elements get aligned to the grid, defined by “Size”, even if the grid lines are not visible. When you insert or move an element, its center will be positioned on the grid. When you modify the size of an element, you can move the position markers onto grid lines only. Elements already available in a visualization, will not be aligned automatically, until you change their position. |
| **Size** | Spacing of the grid lines in pixel. |

**Tab 'File options'**

| **Text list files for textual "IntelliSense"** | File name and path of a file of type .csv. The file contains texts in the format of a text list. The file entries will be available when using the function "List Components" as input assistance. Note: You can create this file as an export file of the global text list. For this purpose use the command “Import/Export Text Lists”. |
| **Visualization Directories** | |
| **Text list files** | Storage path for text lists. Note: This setting will be used in CODESYS only if no storage path for “Text list files” is defined in the “Project Settings”, category “Visualization”. |
| **Image files** | Storage path for image files. Multiple paths get separated by semicolons. CODESYS uses this path for example when exporting or importing image files. Note: This setting will be used in CODESYS only if no storage path for “Image files” is defined in the “Project Settings”, category “Visualization”. |

See also

- Provide export file for the input assistance
- Command 'Import/Export Text Lists'

See also

- Chapter 1.3.5.19.3.13 “Dialog 'Project Settings' - 'Visualization'” on page 1602
- Chapter 1.3.5.19.4.7 “Object 'TargetVisu'” on page 1623
Dialog 'Project Environment' - 'Visualization Profile'

**Function:** The dialog displays the current visualization profile of the project. The profile can be updated here.

**Call:** Main menu “Project ➔ Project Environment”, Tab “Visualization Profile”.

<table>
<thead>
<tr>
<th>“Current visualization profile in project”</th>
<th>The currently set visualization profile of the opened project.</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Recommended, newest profile”</td>
<td>The newest profile</td>
</tr>
<tr>
<td>“Action”</td>
<td></td>
</tr>
<tr>
<td>“Do not update”</td>
<td>The visualization profile of the project remains unchanged.</td>
</tr>
<tr>
<td>“Update to x.x.x.x”</td>
<td>CODESYS updates the project to the chosen visualization profile.</td>
</tr>
</tbody>
</table>
| “Check for updates when loading this project” | ☑: CODESYS checks for new profiles when the project is opened. If there are updates available an update dialog opens automatically.  
  ☐: No check of the profile when loading the project. The update dialogs do not open automatically any longer. |
| “Set All to Newest”                      | CODESYS updates the profile.                                  |

Dialog 'Project Environment' - 'Visualization Styles'

**Function:** The dialog displays the currently used visualization style of the project. The visualization style can be updated here.

**Call:** Main menu “Project ➔ Project Environment”, tab “Visualization Styles”

<table>
<thead>
<tr>
<th>“For the following visualization styles currently in use, newer versions are available:”</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>“Visualization Styles”</td>
<td>Version of the currently set visualization style of the opened project.</td>
</tr>
<tr>
<td>“Current”</td>
<td>Current version of the visualization style, for example 3.5.6.0</td>
</tr>
<tr>
<td>“Recommended”</td>
<td>Recommend version of the visualization style, for example 3.5.7.0</td>
</tr>
<tr>
<td>“Action”</td>
<td></td>
</tr>
<tr>
<td>“Do not update”</td>
<td>The visualization style of the project remains unchanged.</td>
</tr>
<tr>
<td>“Update to x.x.x.x”</td>
<td>CODESYS updates the project to the version of the chosen visualization style.</td>
</tr>
</tbody>
</table>
| “Check for updates when loading the project” | ☑: CODESYS checks for new versions when the project is opened. If there are updates available an update dialog opens automatically.  
  ☐: No check of the version. The update dialogs do not open automatically any longer. |
| “Set All to Newest”                         | CODESYS updates the version.                                  |

See also
- ☐ Chapter 1.3.5.17 “Applying visualization styles” on page 1216

Dialog 'Project environment' - 'Visualization Symbols'

**Function:** The dialog box lists installed symbol libraries and enables you to assign a symbol library to the project.

**Call:** Main menu “Project ➔ Project Environment”, tab “Visualization Symbols”.

---

2020/12/10

3ADR010583, 1, en_US

1601
**Requirement:** The opened project contains a visualization and has been saved with a compiler version of < 3.5.7.0. CODESYS knows symbol libraries from compiler version 3.5.7.0.

<table>
<thead>
<tr>
<th>“Symbol libraries”</th>
<th>List of all installed symbol libraries</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Active”</td>
<td>☑: Symbol library activated for the project. CODESYS makes its symbols available in the “ToolBox” view.</td>
</tr>
<tr>
<td></td>
<td>☐: Symbol library is installed in the library repository.</td>
</tr>
</tbody>
</table>

See also

- ☑ Chapter 1.3.5.3.1 “Selecting an element” on page 1112

**Dialog ‘Project Settings’ - ‘Visualization’**

**Symbol:** ☑

**Function:** This dialog box is for the configuration of the project-wide settings for objects of the type “Visualization”.

**Call:** Main menu “Project ➔ Project Settings”, category “Visualization”

**Requirement:** A project is open.

**Tab ‘General’**

**Table 321: “Visualization Directories”**

<table>
<thead>
<tr>
<th>“Text list files”</th>
<th>Directory containing text lists that are available in the project for configuring texts for different languages. CODESYS uses the directory, for example, when exporting or importing text lists.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>After clicking on ☑ the dialog box “Select Directory” appears, which enables the selection of a directory in the file system.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>“Image files”</th>
<th>Directory containing image files that are available in the project. Multiple folders are separated with a semicolon. CODESYS uses the directory, for example, when exporting or importing image files.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>After clicking on ☑ the dialog box “Select Directory” appears, which enables the selection of a directory in the file system.</td>
</tr>
</tbody>
</table>

**Table 322: “Advanced”**

<table>
<thead>
<tr>
<th>“Activate property handling in all element properties”</th>
<th>☑: You can configure a visualization element with a property ☑ instead of with an IEC variable in those of its properties in which you select a variable. CODESYS then creates additional code for the property handling when compiling a visualization.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Requirement: Your IEC code contains at least one object of the type “Interface property”, i.e. a property ☑.</td>
</tr>
</tbody>
</table>

See also

- ‘Property’ object
Tab ‘Symbol Libraries’

Table 323: “Visualization Symbol Libraries”

<table>
<thead>
<tr>
<th>“Symbol Libraries”</th>
<th>List of all installed symbol libraries, for example VisuSymbols</th>
</tr>
</thead>
</table>
| “Active”           | ☑: Symbol library is selected in the project and CODESYS makes it available in the “Tools” view of a visualization.  
                        ☐: Symbol library is installed in the library repository and CODESYS does not make it available in the “ToolBox” view of a visualization. |

See also

- Chapter 1.3.5.19.3.2 “Dialog ‘Add Visualization’” on page 1583

Dialog ‘Project Settings’ - ‘Visualization Profile’

Symbol: ☑

Function: The dialog box enables the setting of the visualization profile.

Call: Menu “Project ➔ Project Settings”, category “Visualization Profile”

Requirement: A project is open.

Table 324: “Visualization Profile”

| “Specific profile” | Profile that CODESYS uses in the project and that determines the visualization elements that are available in the project.  
                        The selection list contains all the profiles installed so far. |

Dialog ‘Properties’ of Visualization Objects

Function: This dialog is used for configuring object-dependent properties.

Call: Menu bar: “View”; context menu of the visualization object in the “Devices” view or “POUs” view.

Tab ‘General’

Table 325

<table>
<thead>
<tr>
<th>“Name”</th>
<th>Example: visMain</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Object type”</td>
<td>visualization</td>
</tr>
<tr>
<td>“Open with”</td>
<td>visualization</td>
</tr>
</tbody>
</table>

See also

- Dialog ‘Properties’ - Tab ‘General’

Tab ‘Access Control’

This tab is used for defining which user group can execute which actions on the object.

See also

- Dialog ‘Properties’ - Tab ‘Access Control’

Tab ‘Visualization’

This tab assigns a visualization type to a visualization.

In addition, it includes settings for window size that are used at runtime.
Table 326: “Use visualization as”

<table>
<thead>
<tr>
<th>“Visualization”</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Dialog”</td>
<td>Visualization type for a visualization that opens as a dialog in its own window for an input event. The input action for this is “Open dialog”. The “Close dialog” input action closes the window.</td>
</tr>
<tr>
<td></td>
<td>Tip: A dialog usually includes an “OK” button or “Cancel” button at the bottom edge for confirming or rejecting user input, and for closing the dialog. A simple dialog or a dialog prompt includes only a question or information and buttons for closing the dialog with either “Yes” or “No”. A dialog is part of a user interface. While a dialog is open, the rest of the user interface is usually disabled.</td>
</tr>
<tr>
<td>“Numpad / keypad / dialog for input configuration”</td>
<td>Visualization type for a visualization that displays a virtual numeric keypad or a virtual keyboard. It appears when the user is prompted to specify text. The input action for this is “Write variable”.</td>
</tr>
<tr>
<td></td>
<td>Note: The interface of this visualization must also conform with the interfaces for the standard visualizations for the numeric keypad or that keyboard that provides the VisuDialogs library: Numpad, Keypad, NumpadExtended, or TextinputwithLimits.</td>
</tr>
<tr>
<td></td>
<td>Tip: The VisuDialogs library contains templates for virtual keyboards or numeric keypads.</td>
</tr>
<tr>
<td>“Dialog is opaque”</td>
<td>☑: The screen area that is covered by the dialog is not refreshed. This has a positive effect on the character and input performance.</td>
</tr>
<tr>
<td></td>
<td>Use this option when your drawn dialog is rectangular and opaque, containing no transparent parts.</td>
</tr>
<tr>
<td>“Use automatic detected visualization size”</td>
<td>☐: The size is determined so that all visualization elements are enclosed.</td>
</tr>
<tr>
<td>“Include background image”</td>
<td>☑: All elements and the background image are completely visible.</td>
</tr>
<tr>
<td></td>
<td>☐: All elements are visible, but a larger background image is truncated.</td>
</tr>
<tr>
<td>“Use specified visualization size”</td>
<td>☑: The values “Height” and “Width” define the window size of the visualization (in pixels).</td>
</tr>
<tr>
<td>“Internal”</td>
<td>☑: The visualization is internal. It is used exclusively as an internal module of a complete visualization in a library.</td>
</tr>
<tr>
<td></td>
<td>When editing as a library project while the project is open in CODESYS, an internal visualization is handled like all visualizations. The internal visualization appears in drop-down lists. Or in the visualization manager (“Visualizations” tab).</td>
</tr>
<tr>
<td></td>
<td>The internal visualizations that include a linked library are not visible to you.</td>
</tr>
</tbody>
</table>

See also

- “Dialog ‘Frame Configuration’” on page 1564

Tab ‘Build’

This tab includes options for compiling the object.

See also

- Dialog ‘Properties’ - Tab ‘Access Control’

Dialog ‘Selected Alarm Class’

Function: In this dialog box, you define the alarm classes that are considered for the alarm table or alarm banner.
**Call:** Property "Alarm configuration" / "Alarm classes" of the alarm table or alarm banner visualization element.

**Requirement:** An alarm table visualization element or alarm banner visualization element is added to the visualization.

### Available Alarm Classes
Shows all alarm classes created in the project.

### Selected Alarm Classes
The alarm classes in this column are displayed in the alarm table.

### All
All alarm classes are listed in an alarm table.

- **Moves all available alarm classes to the "Selected Alarm Classes" column.**
- **Moves the selected alarm classes to the "Selected Alarm Classes" column.**
- **Removes the selected alarm classes from the "Selected Alarm Classes" column.**
- **Removes all selected alarm classes from the "Selected Alarm Classes" column.**

See also
- Chapter 1.3.5.18.1.22 “Visualization element ‘Alarm table’” on page 1395
- Chapter 1.3.5.18.1.22 “Visualization element ‘Alarm table’” on page 1395
- Chapter 1.3.5.7 “Visualizing alarm management” on page 1146

### Dialog 'Selected Alarm Group'
**Function:** In this dialog box, you define the alarm groups that are considered for the alarm table or alarm banner.

**Call:** Property "Alarm configuration" / "Alarm groups" of the alarm table or alarm banner visualization element.

**Requirement:** An alarm table visualization element or alarm banner visualization element is added to the visualization.

### Available Alarm Groups
Shows all alarm groups created in the project.

### Selected Alarm Groups
The alarm groups in this column are displayed in the alarm table.

### All
All alarm classes are listed in an alarm table.

- **Moves all available alarm groups to the "Selected Alarm Groups" column.**
- **Moves the selected alarm groups to the "Selected Alarm Groups" column.**
- **Removes the selected alarm groups from the "Selected Alarm Groups" column.**
- **Removes all selected alarm groups from the "Selected Alarm Groups" column.**

See also
- Chapter 1.3.5.18.1.22 “Visualization element ‘Alarm table’” on page 1395
- Chapter 1.3.5.18.1.22 “Visualization element ‘Alarm table’” on page 1395
- Chapter 1.3.5.7 “Visualizing alarm management” on page 1146
Dialog 'Advanced Trace Settings'

**Function:** The recording rate of the “Trace” visualization element is configured in this dialog box.

**Call**
- Properties: “Trace”, “Advanced”
- Context menu: “Configure trace”, “Advanced”

**Requirement:** A trace is selected in the active visualization editor.

<table>
<thead>
<tr>
<th>Table 327</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>“Measurement in every nth cycle”</strong></td>
</tr>
<tr>
<td><strong>“Buffer size (samples)”</strong></td>
</tr>
</tbody>
</table>

See also
- ☛ Chapter 1.3.5.19.2.13 “Command ’Configure Trace’” on page 1572

Dialog 'Display Settings'

**Function:** This dialog box includes the configuration for the display settings of the trace diagram (for both the x-axis and y-axis) and provides a preview in the trace diagram.

**Call:** “Display Settings” button in “Trace Configuration” dialog box.

<table>
<thead>
<tr>
<th>Table 328: “Tick Marks”</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>“Fixed spacing”</strong></td>
</tr>
<tr>
<td><strong>“Distance”</strong></td>
</tr>
<tr>
<td><strong>“Subdivisions”</strong></td>
</tr>
</tbody>
</table>

**Tab ‘X-Axis’**

**“Display Mode”**
- “Auto”: CODESYS Scales automatically.
- “Fixed length”: CODESYS displays a segment of constant “Length”.
- “Fixed”: CODESYS displays a segment from “Minimum” to “Maximum”.

**“Minimum”**
Initial value of the segment. Requirement: “Display mode” is “Fixed”.

**“Maximum”**
End value of the segment. Requirement: “Display mode” is “Fixed”.

**“Length”**
Constant length of the segment.

**“Grid”**
- Diagram with vertical grid lines. Select the line color from the dropdown list of colors.
"Font" Font for the x-axis

Tab 'Y-Axis'

"Display Mode" Scaling
- "Auto": CODESYS Scales automatically.
- "Fixed": CODESYS displays a segment from "Minimum" to "Maximum".

"Minimum" Initial value of the displayed segment. Requirement: "Display mode" is "Fixed".

"Maximum" End value of the displayed segment. Requirement: "Display mode" is "Fixed".

"Grid" Diagram with a grid line. Select the line color from the dropdown list of colors.

"Label" The description is displayed on the axis.

Table 329: “Tick Marks”

"Fixed spacing" CODESYS draws tick marks with “Distance” and “Subdivisions”.

"Distance" Distance between tick marks

"Subdivisions" Number of subdivisions between two tick marks

"Font" Font for the y-axis

Preview of the trace diagram

"Background color"  
- "No background": Transparent display without background color.
- "Draw background": Background color according to selection below.
- "From visualization style": Background color as defined in the visualization style.

"Reset" CODESYS resets all settings to the presets.

"Use as Default" CODESYS saves the settings as presets.
1.3.5.19.4 Objects

1.3.5.19.4.1 Object 'Visualization' and visualization editor............................. 1608
1.3.5.19.4.2 Object 'Visualization manager'................................................... 1613
1.3.5.19.4.3 Tab 'Visualization Manager' - 'Default Hotkeys'.......................... 1617
1.3.5.19.4.4 Tab 'Visualization manager' – 'Visualizations'............................. 1617
1.3.5.19.4.5 Tab 'Visualization manager' - 'User management'...................... 1618
1.3.5.19.4.6 Tab 'Visualization Manager' - 'Font'........................................... 1622
1.3.5.19.4.7 Object 'TargetVisu'...................................................................... 1623
1.3.5.19.4.8 Object 'WebVisu'........................................................................ 1624

Object 'Visualization' and visualization editor

Symbol: 🎨

The object represents a single visualization. You can insert a visualization under an application or, so that it is available project-wide, under the root node of the view “Devices” or directly in the view “POUs”. You can open the visualization editor for editing by double-clicking on the object entry in the device tree or in the view POUs.

See also
- Chapter 1.3.5.19.3.15 “Dialog ‘Properties’ of Visualization Objects” on page 1603
- Chapter 1.3.5.19.4.1.1 “Visualization editor” on page 1608

Visualization editor

The visualization editor opens after a double-click on a visualization object.

![Visualization editor interface]

- (1) Graphical editor: this is where you create a visualization from the visualization elements that are provided in the view Tools.
- (2) View “Tools”: available visualization elements
- (3) View “Properties”: configuration editor for the visualization element currently selected in the editor area
- (4) Menu “Visualization”: commands for working in the visualization editor
Amongst other things, the menu “Visualization” contains commands for opening additional editors:

- (5) “Interface Editor”: declaration of variables via which the references of the visualization can be parameterized.
- (6) “Hotkeys Configuration”: definition of shortcuts for inputs on the visualization in online mode.
- (7) “Elementlist”: List of all elements used in the visualization, possibility to change their position on the z-axis.

See also

- Chapter 1.3.5.4.4 “Configuring keyboard shortcuts” on page 1131
- Chapter 1.3.5.15.2 “Calling a visualization with an interface” on page 1188
- Chapter 1.3.5.19.4.1.2 “View ‘Toolbox’” on page 1609
- Chapter 1.3.5.19.4.1.3 “View ‘Properties’ of a visualization element” on page 1611
- Chapter 1.3.5.19.2.1 “Command ‘Interface Editor’” on page 1556
- Chapter 1.3.5.19.2.2 “Command ‘Keyboard Configuration’” on page 1558
- Chapter 1.3.5.19.2.3 “Command ‘Element List’” on page 1559

View ‘Toolbox’

Symbol: 🏗

**Function:** This view provides the elements that are usable in the editor. The individual elements are provided with certain identifiers in order to be able to group them. There is a button for each identifier. The elements of selected identifiers are displayed with thumbnails and can be dragged into the editor. Apart from the standard identifier, you can also define your own. You can change the size of the thumbnails with the sliders and a full-text search for element names is possible.

**Call:** Menu bar: “View ➤ Toolbox”

**Requirement:** A visualization is active.
● (1) “Toolbox” view
● (2) Toolbar with commands
● (3) Buttons with the name of the underlying identifier for selecting element categories
● (4) Selection of an individual visualization element
● (5) Control elements

See also

● Command ‘Toolbox’
Only one button can be selected.

Several buttons can be selected.

The “Manage Identifiers and Entries” dialog opens.

See also
- % Chapter 1.3.5.19.3.4 “Dialog ‘Configure Tags and Items’” on page 1585

A button labeled with the associated identifier is displayed for each defined element category. A selected button is shown in green.

<table>
<thead>
<tr>
<th>[Shift]+click a button</th>
<th>Changing the selection of the identifier and the selection type (single or multiple choice possible).</th>
</tr>
</thead>
<tbody>
<tr>
<td>Right-click on a button</td>
<td>Context menu appears.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>“Hide identifier”</th>
<th>Removes the button. The identifier is then invisible.</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Activate identifier”</td>
<td>Button turns green and the identifier is activated irrespective of the selection type.</td>
</tr>
<tr>
<td>“Deactivate identifier”</td>
<td>Button turns green and the identifier is deactivated irrespective of the selection type.</td>
</tr>
</tbody>
</table>

See also
- % Chapter 1.3.5.3.1 “Selecting an element” on page 1112
- % Chapter 1.3.5.19.3.4 “Dialog ‘Configure Tags and Items’” on page 1585
- Command ‘Toolbox’

The visualization elements are offered as thumbnails and labeled with name. The selection depends on the search query in or on the selection of the buttons.

<table>
<thead>
<tr>
<th>Slider</th>
<th>To change the size of the thumbnails.</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="Slider" /></td>
<td></td>
</tr>
<tr>
<td><img src="image" alt="Input Field" /></td>
<td>For a full-text search by element name among all available elements</td>
</tr>
<tr>
<td>“&lt;number&gt; Entries”</td>
<td>Number of entries of visualization elements that are currently offered by the selection of buttons or due to the search query in .</td>
</tr>
</tbody>
</table>

View ‘Properties’ of a visualization element

Symbol: ☐

**Function:** This view is used for configuring the element properties of the selected visualization element.

**Call:** Menu bar: “View ➤ Element Properties”
See also

Menu bar

Table 331: “Filters”

<table>
<thead>
<tr>
<th>Filter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>“All categories”</td>
<td>List of all element properties</td>
</tr>
<tr>
<td>“Default”</td>
<td>List of the most frequently used element properties</td>
</tr>
<tr>
<td>“Simple”</td>
<td>List of certain basic element properties, such as “Texts”, “Colors”, and “Input configuration”</td>
</tr>
<tr>
<td>“Animation”</td>
<td>List of element properties for animation with variables</td>
</tr>
<tr>
<td>“Colors”</td>
<td>List of element properties for designing with color</td>
</tr>
<tr>
<td>“Texts”</td>
<td>List of element properties for designing with text</td>
</tr>
<tr>
<td>“Input”</td>
<td>List of element properties for configuring user input</td>
</tr>
</tbody>
</table>

Table 332: “Sort”

<table>
<thead>
<tr>
<th>Sort Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Sort by type”</td>
<td>Element properties are sorted by the original order of categories.</td>
</tr>
<tr>
<td>“Sort by name”</td>
<td>Element properties are sorted in alphabetical order.</td>
</tr>
</tbody>
</table>

Table 333: “Order”

<table>
<thead>
<tr>
<th>Order Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Sort ascending”</td>
<td>The properties are sorted from A to Z.</td>
</tr>
<tr>
<td>“Sort descending”</td>
<td>The properties are sorted from Z to A.</td>
</tr>
</tbody>
</table>

“Expert”

- The table includes all properties. The menu command “Filter ➔ Show all categories” is enabled at the same time.

Element properties display in a table

<table>
<thead>
<tr>
<th>Column “Property”</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Element properties of the selected element</td>
</tr>
<tr>
<td>Column “Value”</td>
<td>Description</td>
</tr>
<tr>
<td></td>
<td>The assigned value is applied in the editor view.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Double-click in the “Value” column</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>A line editor, drop-down list, or dialog opens for editing the value.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Single-click for a selected field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>[Blank] for a selected field</td>
<td>A style selected in the visualization manager can include single, predefined element properties. As a result, these do not appear in this view because a fixed value is already assigned to them. They do not have to be configured anymore.</td>
</tr>
</tbody>
</table>

A style selected in the visualization manager can include single, predefined element properties. As a result, these do not appear in this view because a fixed value is already assigned to them. They do not have to be configured anymore.
Visualizations can be configured with device-specific restrictions that block the availability of element properties.

Device-specific restrictions:
- Elements with restricted availability
- Fonts with restricted availability
- Colors with restricted display
- Image formats with restricted display
- Maximum number of visualization elements
- Maximum number of visualizations below the device

Object 'Visualization manager'

Symbol: 📖

The visualization manager manages the configuration settings for all display variants of the visualizations of the current application.

The object is automatically inserted when a visualization object is inserted below the application. On a double-click the configuration dialog opens with several tabs.

If the device employed supports display variants of the visualization, the visualization manager automatically brings along the corresponding objects (CODESYS WebVisu, CODESYS TargetVisu).

If the device employed supports CODESYS TargetVisu, the visualization manager automatically brings along the corresponding object CODESYS TargetVisu.

See also
- § Chapter 1.3.5.19.4.7 “Object ‘TargetVisu’” on page 1623
- § Chapter 1.3.5.19.4.8 “Object ‘WebVisu’” on page 1624

Tab 'Visualization manager' – 'Settings'

Symbol: 📖

Function: the tab contains settings for all visualizations that are available application-wide.

'General settings'

| "Use unicode strings" | ✅: The visualization codes character strings as Unicode. |
| "Use CurrentVisu variable" | ✅: the application knows and uses the global variable VisuElems.CurrentVisu of the type STRING. It contains the name of the currently active visualization at the runtime of the application. The application can read from the variable in order to obtain the name of the currently active visualization. The application can cause a visualization change by a write access. Requirement: the application contains a visualization that calls further visualizations. Example
  - Variable assignment: VisuElems.CurrentVisu:=strVisuName;
  - Visualization name assignment: VisuElems.CurrentVisu:='visu1'; |
The “Visualization Style Editor” enables new styles to be generated, checked and installed in the visualization styles repository.

**“Selected style”**
- Style from the visualization styles repository that every visualization in the application uses, for example “Flat Style”.
- “<None>”: The visualization displays its elements without style or according to the internal default. A standard dialog appears instead of a selection list for selection in the element properties “Color” und “Font”.

Open a selection list with styles that are installed in the visualization styles repository.

**“Display all versions (for experts only)”**
- The selection list contains only the latest version of each selected style and all other styles. If a newer version of the selected style is installed it is also displayed.
- The selection list contains all versions of all installed styles.

**Button ➔**
Opens a selection list with commands for the use of the “Visualization Styles Editor”.

**“Open Style Editor”**
The “Visualization Styles Editor” opens.

**“Create and edit derived style”**
The “Visualization Styles Editor” opens with the dialog “Create a new visualization style”. The dialog contains the settings for the first configuration step.

Requirement: a style is selected in “Selected style”.

**“Copy and edit the selected style”**
The “Visualization Style Editor” opens with the dialog “Open existing style as a copy”. The dialog contains the settings for the first configuration step.

Requirement: a style is selected in “Selected style”.

**“Preview”**
The elements displayed represent the style specified in “Selected style”.

See also
- Chapter 1.3.5.17 “Applying visualization styles” on page 1216
- Chapter 1.3.5.17.2 “Managing visualization styles in repositories” on page 1221
- Chapter 1.3.5.20.1 “Dialog ‘Create a New Visualization Style’” on page 1946
- Chapter 1.3.5.20.2 “Dialog ‘Open Existing Style as a Copy’” on page 1947
- Chapter 1.3.5.19.3.7 “Dialog ‘Options’ - ‘Visualization Styles’” on page 1597

**'Language settings'**

**“Selected language”**
Language used by the display variants at the start of a visualization.

**'Settings for default text input'**
For an element with standard text input, a dialog that supports the input appears at runtime. You can specify which dialog appears.
<table>
<thead>
<tr>
<th>Dialog Name</th>
<th>Description</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Numpad”</td>
<td>Dialog that calls the visualization if a user activates the input field for a number at runtime. The dialog represents a numeric keypad. Default: “VisuDialogs.Numpad”</td>
<td></td>
</tr>
<tr>
<td>“Keypad”</td>
<td>Dialog that calls the visualization if a user activates the input field for a text at runtime. The dialog represents a keyboard. Preset: “VisuDialogs.Keypad”</td>
<td></td>
</tr>
</tbody>
</table>
| “Use text input with limits” | Requirement: CODESYS TargetVisu or CODESYS WebVisu are configured as display variants and the “standard text input” is “keyboard”. The visualization then supports input via keyboard at runtime. The input thus generally takes place via an input field.  
☑️ Instead of the input field you can call a dialog that displays the value range for inputs with a limited value range.  
Default: “VisuDialogs.TextinputWithLimits”. This dialog displays the value range and doesn't accept any value outside these limits. |                              |

See also
- ☰ Chapter 1.3.5.19.4.7 “Object 'TargetVisu’” on page 1623
- ☰ Chapter 1.3.5.19.4.8 “Object 'WebVisu’” on page 1624

'Settings for user management dialogs'

You can configure your visualization with a user management. To do this, configure an input to an element that causes a user management dialog to appear. The VisuUserManagement library contains ready-to-use dialog visualizations for this purpose. The library is located in the installation directory, for example in C:\Program Files (x86)\3S CODESYS\CODESYS\Projects\Visu\Dialogs\VisuUserMgmtDialogs.library.

You can also use other visualizations as user management dialogs. To do that you have to change the defaults here.

<table>
<thead>
<tr>
<th>Dialog Name</th>
<th>Description</th>
<th>Default</th>
</tr>
</thead>
</table>
| “Login dialog”           | User management dialog that enables logging in; typically a request to enter a user name and a password. It appears upon an input event on an element that executes as a consequential action “User management”, action “Login”.  
Preset: VisuUserManagement.VUM_Login |                              |
| “Change password dialog” | User management dialog that enables a password to be changed; typically a request to enter the current password and a new one. It appears upon an input event on an element that executes as a consequential action “User management”, action “Change user password”.  
Preset: VisuUserManagement.ChangePassword |                              |
| “Change configuration dialog” | User management dialog that enables a configuration change of the user management, i.e. typically a display of the current user configuration and a possibility to change it. It appears upon an input event on an element that executes as a consequential action “User management”, action “Open user configuration”.  
Preset: VisuUserManagement.VUM_UserManagement |                              |

See also
- ☰ Chapter 1.3.5.5 “Setting up user management” on page 1139
- ☰ Chapter 1.3.5.19.3.6 “Dialog ‘Input Configuration’” on page 1586
'Additional settings'

"Activate multitouch handling"  
☑️ At runtime the visualization expects user inputs via gestures and touch events.  
Elements concerned  
- Elements with input configuration  
- Element of the type “Frame”  
- Component of the type “Tab control”

"Activate semi-transparent drawing"  
☑️ The visualization draws the elements in a semi-transparent color.  
To do this you can additionally specify a graduation value for the transparency when defining a color. The transparency is defined in the “Transparency” property.  
The leading byte is evaluated in color variables.  
Preset: Activated. Requirement: you create a new visualization and the display variants can paint semi-transparently.

"Activate standard keyboard handling"  
☑️  
- [Tab]  
- [Shift] + [Tab]  
- [Input]  
- [Up arrow]  
- [Down arrow]  
- [Right arrow]  
- [Left arrow]

See also  
- ☞ Chapter 1.3.5.3.3 “Assigning a color” on page 1115  
- ☞ Chapter 1.3.5.19.1 “Keyboard Shortcuts for Default Keyboard Action” on page 1555

'Extended settings'

Table 334: “Memory settings”

<table>
<thead>
<tr>
<th>Setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Size of memory for visualization</td>
<td>Memory size in bytes allocated by the visualization at runtime. Preset: “400000”</td>
</tr>
<tr>
<td>Size of the paintbuffer (per client)</td>
<td>Memory size in bytes allocated by the visualization per display variant and used for painting actions. Preset: “50000”</td>
</tr>
</tbody>
</table>

Table 335: “File transfer mode”

<table>
<thead>
<tr>
<th>Setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transfer visualization files to the PLC</td>
<td>☑️: When downloading the application from the visualization directories to the controller, CODESYS copies image files and text list files that the visualization references. A CODESYS TargetVisu needs the files on the PLC and similarly the dialogs that support a numerical input or a keyboard input.</td>
</tr>
<tr>
<td>Use local visualization files</td>
<td>☑️: The visualization uses image files and text list files from local directories</td>
</tr>
<tr>
<td>Note:</td>
<td>In order to achieve that the visualization can access the files stored locally, it is necessary that the file paths are relative. The paths are given in dialog box “Project ➔ Project settings” in tab “Visualization”.</td>
</tr>
<tr>
<td>Note:</td>
<td>It is also necessary that the link type of a image is “Link to file”. The link type is specified in the image pool.</td>
</tr>
</tbody>
</table>

See also  
- ☞ Chapter 1.3.5.16.3 “Configure File Transfer Mode” on page 1215  
- ☞ Chapter 1.3.5.19.3.13 “Dialog ’Project Settings’ - ’Visualization’” on page 1602
Object 'Image pool'

Display variant denotes the type of visualization, for example WebVisu, TargetVisu. A visualization client is a currently connected display medium. Thus, several browsers (clients) can be connected in parallel for the display variant WebVisu.

Table 336: “Client settings”

<table>
<thead>
<tr>
<th>“Maximum number of visualization clients”</th>
<th>Limits the number of visualization clients that are executed at the same time. If you configure the elements so that they vary depending on the display variant, then you have to limit the number of display variants. A visualization is given an ID at runtime that identifies the display variant and then processes data accordingly. CODESYS can query the ID using the system variable CURRENTCLIENTID and thus obtains the information as to which of the running variants is concerned. Example: arr[CURRENTCLIENTID].dwColor</th>
</tr>
</thead>
<tbody>
<tr>
<td>Requirement: VisuGlobalClientManager library is integrated in the project. Tip: You can find in the CODESYS store. example &quot;Global Client Manager&quot;</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>“Transfer both svg images and converted images”</th>
<th>This option is visible only if both a WebVisu and a TargetVisu exist. It concerns images in svg format only. The option is available if the device description for the controller of the TargetVisu does not support the svg (full) format.</th>
<th>✓: The images are transmitted in the png or bmp formats (for TargetVisu) and additionally in svg format (for WebVisu).</th>
</tr>
</thead>
<tbody>
<tr>
<td>Not all settings are available with an integrated CODESYS visualization.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Tab 'Visualization Manager’ - ‘Default Hotkeys’

Symbol: □

The tab includes a list of configured keyboard shortcuts that are valid for all visualizations available throughout the application. Therefore, the tab is the central location for defining keyboard shortcuts for all visualizations that are below an application.

Keyboard shortcuts of the default keyboard action are not listed here.

The tab is similar to the “Keyboard Configuration” tab and provides the same editing options.

See also
- “Tab 'Keyboard configuration’” on page 1558

Tab 'Visualization manager’ – ‘Visualizations’

Symbol: 🎨
**Function:** The tab lists all visualizations that are available project-wide and enables an assignment of the visualizations for the loading behaviour, depending on the display variants.

**Tab 'Flags'**

<table>
<thead>
<tr>
<th>“Standard behaviour”</th>
<th>The visualizations of the application and the actually referenced visualizations are automatically loaded to the target system. The activated checkboxes show which one that is.</th>
</tr>
</thead>
</table>
|                      | ☑: The loading behaviour is explicitly defined for each visualization. Hint: use the explicit selection if you reference visualizations indirectly via IEC variables.                                                                 |}

<table>
<thead>
<tr>
<th>“Visualizations”</th>
<th>The list contains all created visualizations from the device tree and the POU view.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>“Dialogs”</th>
<th>The list contains all the referenceable visualizations that are available via the libraries of the library management.</th>
</tr>
</thead>
</table>

Only those visualizations selected here using checkboxes are loaded.

<table>
<thead>
<tr>
<th>“Remote target visualization, target visualization, web visualization,”</th>
<th>The column settings affect the loading behaviour for the display variants “remote target visualization”, “target visualization” and “web visualization”.</th>
</tr>
</thead>
</table>

**Tab 'Visualization manager' - 'User management’**

Symbol: 🌐

The “User management” tab is used for creating and configuring the user management for visualizations and their users and groups.

If a user management has not been configured yet, then the following buttons are available:

<table>
<thead>
<tr>
<th>“Create empty user management”</th>
<th>The user management opens. The “None” group is created.</th>
</tr>
</thead>
</table>

| “Create user management with default groups and users” | The user management opens. The following groups and users are created:  
- “Admin” group with “Admin” user  
- “Service” group with “Service” user  
- “Operator” group “Operator” user  
- “None” group |
|--------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

**Project with multiple visualization user managements**

**Table 337: “Choose between local and remote user management”**

<table>
<thead>
<tr>
<th>Requirement: The project includes several devices with a visualization user management.</th>
<th>The user management of this visualization manager is used for the visualization.</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Use local user management”</td>
<td>Drop-down list with all devices of the project that have their own visualization user management.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>“Use remote user management”</th>
<th>Drop-down list with all devices of the project that have their own visualization user management.</th>
</tr>
</thead>
</table>
Even if there are no display variants of the visualization in the application, it may be required that the visualization user management is located on the controller. This is the case, for example, when HMIs connect to the controller.

| Requirement: The visualization does not have any display variants. This means that the objects “Web visualization”, “Target visualization”, or “Remote target visualization” are inserted below the visualization manager. |

See also
● Chapter 1.3.5.5.2 “Configuring users and groups” on page 1140

**Tab 'Groups'**

| “Group name” | When you click the node, all users are listed that belong to the group. |
| “Automatic logout” | ☑: The “Logout time” input field is active and editable. |
| “Logout time” | Input field for integer value |
| | Drop-down list for time unit “Min”, “Sec”, or “Hr” |
| “Permission to change user data” | ☑: The group is granted permission to edit user data when the visualization is in online mode. |
| “Description” | The text is visible in the development system only. It is not downloaded to the controller. |
| “ID” | Unique ID for each group. Assigned automatically by the system. |

Add a new group
In the last row of the table, click in the “Group name” field and specify the name for the new group.

Delete a group
Select a group and press [Del]. The “None” group cannot be deleted.

**Table 338: Buttons**

| “Update visualizations / hotkeys” | Opens the “Update visualizations and hotkeys” dialog box. |
| | Update, if groups were changed at a time when visualizations or keyboard shortcuts already had restricted permissions. |
| “List usage of groups” | List of visualizations and keyboard shortcuts with restricted permissions. |
| | The list is displayed in the “Messages” view. |
| “Export groups for global visualizations” | The defined group names are transmitted to “Tools ➔ Options ➔ Visualization user management”. They are then listed in “Use the following user group list for the visualization”. The list can be changed there as well. |
| “Delete complete user management” | The user management is deleted and the start view is shown with the following buttons: |
| | “Create empty user management” and |
| | “Create user management with default groups and users”. |
| “Export user management” | The drop-down list opens. |
| | ● “Before V3.5 SP6” |
| | ● “V3.5 SP6 and later” |
| | A standard dialog opens for saving the user management as a CSV file with any name in any directory. |
“Import user management”  A standard dialog opens for importing a user management. The user management must be a CSV file.

↑, ↓  Moves the selected group one line up/down, thus changing the hierarchy of the group. A group of a higher hierarchy cannot have fewer permissions for an element than a group of a lower hierarchy.

Table 339: Dialog box “Update visualizations and hotkeys”

<table>
<thead>
<tr>
<th>Table 339: Dialog box “Update visualizations and hotkeys”</th>
</tr>
</thead>
<tbody>
<tr>
<td>This dialog updates only visualization elements and keyboard shortcuts with configured permissions.</td>
</tr>
<tr>
<td>“Add new group”  Drop-down list with all new created groups of this user management. Requirement: A new user group was created.</td>
</tr>
</tbody>
</table>
| “Setting for new group”  ● “new group in visualization / hotkey will get the right like group”: Drop-down list with all existing groups of this user management
  ● “new group should get the following right”
    – “for visualization elements”: Drop-down list with the permissions: “Operable”, “Only visible”, and “Invisible”.
    – “for hotkeys”: Drop-down list with the permissions: “Operable”, “Not operable”. |
| “Delete not existing groups”  If no affected visualization elements or keyboard shortcuts are found for updating, then this is displayed as a message in the “Messages” view (“Visualization” category). |
| “Rename groups”  Updates the permissions of the affected visualization elements and keyboard shortcuts |

Tab ‘User’

| “Login name”  Name for the user to log in to the visualization at runtime. This name is unique. |
| “Full name”  This name may exist more than one time in the user management. |
| “Password”  Encrypted by CODESYS. By default, the “Login name” is displayed here. If you click the “Password” field of a selected line, then the “Change password” dialog box opens. |
| “User group”  Group(s) that the user belongs to. Clicking the “User group” field of a selected user opens the dialog box “User groups the user belongs to”.
  ● “Groups”
  ● “Assigned”: The user is assigned to this group. |
| “Deactivate”  The user is deactivated. |
| “Description”  Descriptive text is available in the development system only and is not downloaded to the controller. |

Table 340: Buttons

| “Upload user from device”  The data of the user management is uploaded from the controller. If user data is already configured, then it is overwritten. |
| “Download user to device”  The data of the user management is downloaded to the controller. The existing user management on the controller is overwritten. |
“Export user management” The drop-down list opens.
- “Before V3.5 SP6”
- “V3.5 SP6 and later”
A standard dialog opens for saving the user management as a CSV file with any name in any directory.

“Import user management” A standard dialog opens for selecting the user management (in CSV format) from the file system.

Tab 'Settings'

“Settings for download of user data”

“Download user data on every login” The data of the user management is downloaded to the controller at login. Existing data is overwritten.

“Never download user data on login” The data of the user management is never downloaded to the controller, even if it changes.

“Allow decision on every download” A “Warning” dialog box opens for you to accept or refuse the download.

“Access rights for elements”

“Use group hierarchy” [✓]: The permissions can be granted to the group hierarchy of the “Groups” tab only.
The group in the first line of the “Group” list is the highest in the hierarchy.
A group of a higher hierarchy cannot have fewer permissions for an element than a group of a lower hierarchy.

“Logout behavior”

“Change to start visualization at logout” [✓]: Switches at logout to the visualization that is configured as the “Start visualization” in the respective display variant.

CSV file with the data for user management The data for user management is saved to a CSV file in the following format:
- **User groups**: ID;group name; automatic logoff TRUE/FALSE; logoff time; unit logoff time; permission to change user date TRUE/FALSE
- **Users**: login name; full name; password encrypt TRUE/FALSE; password; group ID; user deactivated TRUE/FALSE

Use this format when you want to edit data for user management by means of any tool. If you set password encrypt to FALSE, then an unencrypted password can be used. In the example, the unencrypted password Yellow was specified for the user Hugo. If you import the CSV file with the command "Import user management", then the password is encrypted automatically.
Example

V1.0.0.1
Usergroups:
1;Admin;TRUE;1;Minute;TRUE
3;Operator;FALSE;1;Minute;FALSE
7;Service;FALSE;1;Minute;FALSE
0;None;FALSE;1;Minute;FALSE
4;Early and late shift;FALSE;1;Minute;FALSE
2;Early shift;TRUE;1;Minute;FALSE
6;Late shift;FALSE;1;Minute;FALSE
User:
Service;Service;TRUE;C08298D42A35732CFFB7DF43771B7607;2;FALSE
Operator;Operator;TRUE;3D94AB9540B025B07773DE7037F19B37;3;FALSE
John;Blue;TRUE;62ED5DE29E5DD4164A01F3AF1B81EFA0;4;FALSE
Paul;White;TRUE;01E2CBD4AE5442D9EACE33669549A3CC;2;FALSE
Hugo;Green;FALSE;Yellow;6;FALSE

See also
● Chapter 1.3.5.5.2 “Configuring users and groups” on page 1140
● Chapter 1.3.5.5.4 “Configuring permissions for groups” on page 1142

Tab 'Visualization Manager' - 'Font'

Symbol: 📐

Function: This tab provides settings for adapting the font and font size in the visualization according to the language. The settings apply to all visualizations of the application, including the visualization manager.

Table 341: “Language Specific Font Settings”

<table>
<thead>
<tr>
<th>“Language”</th>
<th>Language used in the project. A column is created for each language. All text lists, including those from integrated libraries, are scanned for this.</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Font”</td>
<td>Font used by the visualization depending on the language.</td>
</tr>
</tbody>
</table>
| “Size factor” | The factor affects the type size of all texts in the visualization.  
    Preset: 1  
    If the factor is smaller than 1, this leads to a reduction of the type size. If the factor is 1, all texts are displayed unchanged as defined in “Properties”. |

Red highlighting of a cell
The highlighted language is no longer present in the text lists of the project or the libraries.  
This highlighting is not available in runtime mode.

Context menu of a selected table row

| “Delete Language” | The associated column is removed. This is advisable above all if settings in the column are highlighted in red. |
| “Copy Language Settings” | All settings in the column are copied to the clipboard. |
| “Paste Language Settings” | All settings in the column are overwritten with the values from the clipboard. |
Table 342: “General Font Settings”

| “Automatic decrease of font size” | ❑: If the text to be displayed does not fit in the text field in the set format, then the font size is decreased automatically until the text fits completely in the text field. 
Tip: This prevents a text from being truncated when changing to a language that needs more space. The requirement is that a font is available which has a sufficiently small font. |

See also
● ᵉ Chapter 1.3.5.6 “Setting up multiple languages” on page 1143

Object ‘TargetVisu’

Symbol: 📚

**Function:** The object is used for configuring CODESYS TargetVisu in order to display the visualization directly on the controller of an integrated or connected panel.

**Requirement:** The CODESYS control runtime environment is equipped with the CODESYS TargetVisu component. The object itself is inserted below the visualization manager.

CODESYS TargetVisu can be executed on different platforms, from embedded controllers to powerful PC-based systems on different operating systems. Therefore, it can be run on Windows, Windows Embedded CE, Linux, QNX, or VxWorks. A ready-made adaptation to the graphics interface of the systems is available on these operating systems. An adaptation is required for embedded controllers or other operating systems. In addition, there are device manufacturers that integrate visualizations into external applications by means of ActiveX controls.

See also
●

| “Start Visualization” | Name of the visualization where the start is displayed as CODESYS TargetVisu.
Hint: Use input assistance for selecting another visualization. |
| “Update rate (ms)” | Refresh rate (in milliseconds) in the visualization
Example: 200 |
| “Show used visualizations” | The link opens the “Visualizations” tab in the “Visualization manager” editor.
The tab provides information of the visualizations loaded on the display variants. |

See also
●

Table 343: “Scaling Options”

| “Fixed” | ❑: Fixed size of the visualization (original size). |
| “Isotropic” | ❑: The size of the visualization is adapted to the dimensions of the display device, retaining the proportions of the visualization. |
| “Anisotropic” | ❑: The size of the visualization is adjusted to the size of the display device, for example a screen. |
| “Use scaling options for dialogs” | ❑ The dialogs, also for keypad and numpad, are scaled like the visualization (drawn with the same scaling factor). This is an advantage when a dialog was created to match the visualization because then they are scaled together. |
“Use automatically detected client size”

- The visualization fills the screen of the display device completely.

“Use specified client size”

- The values in “Client height” and “Client width” are used for the size of the visualization. The visualization fills this screen area only.

“Client height”

- Height of the visualization (in pixels).

“Client width”

- Width of the visualization (in pixels).

<table>
<thead>
<tr>
<th>Table 344: “Presentation Options”</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Antialiased drawing”</td>
</tr>
<tr>
<td>- Antialiasing is used in the visualization editor for drawing a visualization as a TargetVisu and a TargetVisu variant.</td>
</tr>
<tr>
<td>Hint: If a horizontal or vertical line is drawn blurry on a specific visualization platform, then this can be corrected by an offset of 0.5px in the direction of the line thickness (see element property “Absolute movement”, option “Use REAL values”). Requirement: The platform in use supports using REAL coordinates.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Table 345: “Default Text Input”</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Input with”</td>
</tr>
<tr>
<td>“Touchscreen”</td>
</tr>
<tr>
<td>- Text input on the display variant with touchscreen. The keypad or numpad dialog opens.</td>
</tr>
<tr>
<td>“Keyboard”</td>
</tr>
<tr>
<td>- Text input on the display variant with an ordinary keyboard or a virtual keyboard (on Linux for example)</td>
</tr>
</tbody>
</table>

Effect:

When you configure a user input for default text input, select an input configuration for input action “Write variable”, and configure the “Input type” as “Default”, then the settings are used here.

See also

- “Input action ‘Write a Variable’” on page 1594

Object ‘WebVisu’

Symbol: 🌐

Function: This object is used for configuring the web-based display variant for remote display of the visualization of the controller in a web browser. This allows for remote access, remote monitoring, as well as service and diagnostics of an application over the Internet.

Requirement: The object is inserted below the visualization manager, and the target system has a web server with CODESYS WebVisu support. The web server allows for the communication between the target system and the web browser.

See also

-
“Start Visualization” | Name of the visualization where the start is displayed as CODESYS WebVisu. Hint: Use input assistance for selecting another visualization.

“Name of the .htm file” | Base URL of the web page. The URL is also specified as the address in the web browser.
Example: http://localhost:8080/webvisu.htm

Note: If you use a BeagleBone Black as a visualization device, then you must note that a BeagleBone Black uses port 9090 for its web server. A valid IP address is as follows: http://192.168.7.2:9090/webvisu.htm

“Use as default page” | ☑: The page specified in “Name of .htm file” is preset as the default page. Now this page will always open when a user specified in the web browser the IP address and port of the web server that is running on the controller: http://<IP address web server>:<port web server>.
Example: http://localhost:8080

Notice: Even if you have created multiple web visualizations, you can activate this option for exactly one web page only and therefore preset only one page as the default page.

“Update rate (ms)” | Refresh rate (in milliseconds) in the web browser

“Default communication buffer size” | Default size for communication buffer (in bytes). Defines the maximum available memory for data transfer between the web client and the web server.
Example: 50000

“Show used visualizations” | The link opens the “Visualizations” tab in the “Visualization Manager” editor.
The tab provides information of the visualizations loaded on the display variants.

See also
●
●

Table 346: “Scaling Options”

“Fixed” | ☐: Fixed size of the visualization. The values used are “Client height” and “Client width”.

“Isotropic” | ☐: The size of the visualization is adapted to the dimensions of the web browser, retaining the proportions of the visualization.

“Anisotropic” | ☐: The size of the visualization is adapted to the web browser.

“Use scaling options for dialogs” | ☑: The dialogs, also for keypad and numpad, are scaled like the visualization (drawn with the same scaling factor). This is an advantage when a dialog was created to match the visualization because then they are scaled together.

“Client height” | Height of the visualization (in pixels).

“Client width” | Width of the visualization (in pixels).

Table 347: “Presentation Options”

“Antialiased drawing” | ☑: Antialiasing is used when drawing the visualization in the web browser.

Table 348: “Default Text Input”

“Input with” |  
“Touchscreen” | Text input on the WebVisu with touchscreen. The keypad or numpad dialog opens.
<table>
<thead>
<tr>
<th>&quot;Keyboard&quot;</th>
<th>Text input on the WebVisu with an ordinary keyboard or a virtual keyboard (on Android OS for example)</th>
</tr>
</thead>
</table>

**Effect:**
When you configure a user input for default text input, select an input configuration for input action "Write variable", and configure the "Input type" as "Default", then the settings are used here.

**See also**
- "Input action 'Write a Variable'" on page 1594
### 1.3.5.19.5 Visualization Elements

<table>
<thead>
<tr>
<th>Code</th>
<th>Visualization Element</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.3.5.19.5.1</td>
<td>Visualization element 'Rectangle', 'Round rectangle', 'Ellipse'...</td>
</tr>
<tr>
<td>1.3.5.19.5.2</td>
<td>Visualization element 'Line'</td>
</tr>
<tr>
<td>1.3.5.19.5.3</td>
<td>Visualization element 'Polygon', 'Polyline', 'Bézier curve'...</td>
</tr>
</tbody>
</table>
| 1.3.5.19.5.4 | Visualization element 'Pie'
| 1.3.5.19.5.5 | Visualization element 'Image'
| 1.3.5.19.5.6 | Visualization element 'Frame'
| 1.3.5.19.5.7 | Visualization element 'Label'
| 1.3.5.19.5.8 | Visualization element 'Combo box array'
| 1.3.5.19.5.9 | Visualization element 'Combo box integer'
| 1.3.5.19.5.10 | Visualization element 'Tab control'
| 1.3.5.19.5.11 | Visualization element 'Button'
| 1.3.5.19.5.12 | Visualization element 'Group box'
| 1.3.5.19.5.13 | Visualization element 'Table'
| 1.3.5.19.5.14 | Visualization element 'Text field'
| 1.3.5.19.5.15 | Visualization element 'Scrollbar'
| 1.3.5.19.5.16 | Visualization element 'Slider'
| 1.3.5.19.5.17 | Visualization element 'SpinControl'
| 1.3.5.19.5.18 | Visualization element 'Invisible input'
| 1.3.5.19.5.19 | Visualization element 'Checkbox'
| 1.3.5.19.5.20 | Visualization element 'Progress bar'
| 1.3.5.19.5.21 | Visualization element 'Radio button'
| 1.3.5.19.5.22 | Visualization element 'Alarm table'
| 1.3.5.19.5.23 | Visualization element 'Alarm banner'
| 1.3.5.19.5.24 | Visualization element 'Bar display'
| 1.3.5.19.5.25 | Visualization element 'Meter 90°'
| 1.3.5.19.5.26 | Visualization element 'Meter 180°'
| 1.3.5.19.5.27 | Visualization element 'Meter'
| 1.3.5.19.5.28 | Visualization element 'Potentiometer'
| 1.3.5.19.5.29 | Visualization element 'Histogram'
| 1.3.5.19.5.30 | Visualization element 'Image switcher'
| 1.3.5.19.5.31 | Visualization element 'Lamp'
| 1.3.5.19.5.32 | Visualization element 'Dip switch', 'Power switch', 'Push switch', 'Push switch LED', 'Rocker switch'
| 1.3.5.19.5.33 | Visualization element 'Rotary switch'
| 1.3.5.19.5.34 | Visualization element 'Trace'
| 1.3.5.19.5.35 | Visualization element 'Trend'
| 1.3.5.19.5.36 | Visualization element 'Legend'
| 1.3.5.19.5.37 | Visualization element 'ActiveX'
| 1.3.5.19.5.38 | Visualization element 'Webbrowser'
| 1.3.5.19.5.39 | Visualization element 'Waiting symbol cube'
| 1.3.5.19.5.40 | Visualization element 'Waiting symbol flower'
| 1.3.5.19.5.41 | Visualization element 'Text editor'
| 1.3.5.19.5.42 | Visualization element 'Path3D'
| 1.3.5.19.5.43 | Visualization element 'Control Panel'
| 1.3.5.19.5.44 | Visualization element 'Date range picker'
| 1.3.5.19.5.45 | Visualization element 'Time range picker'
| 1.3.5.19.5.46 | Visualization element 'Date picker'
| 1.3.5.19.5.47 | Visualization Element 'Analog clock'
**Visualization element 'Rectangle', 'Round rectangle', 'Ellipse'**

Symbol:

![Rectangle Symbol](image)

Tag: "Basic"

The rectangle, round rectangle, and ellipse are the same element type. They can be converted into another element type by changing the "Element type" property.

**Element properties**

<table>
<thead>
<tr>
<th>&quot;Element name&quot;</th>
<th>Optional</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example: Werkstueck_3</td>
<td></td>
</tr>
</tbody>
</table>

Hint: Assign individual names for elements so that they are found faster in the element list.

<table>
<thead>
<tr>
<th>&quot;Type of element&quot;</th>
<th>&quot;Rectangle&quot;, &quot;Round rectangle&quot;, &quot;Ellipse&quot;</th>
</tr>
</thead>
</table>

**Element property 'Position'**

The position defines the location and size of the element in the visualization window. This is based on the Cartesian coordinate system. The origin is located at the upper left corner of the window. The positive horizontal x-axis runs to the right. The positive vertical y-axis runs downwards.

<table>
<thead>
<tr>
<th>&quot;X&quot;</th>
<th>The x-coordinate of the upper left corner of the element</th>
</tr>
</thead>
<tbody>
<tr>
<td>Specified in pixels</td>
<td></td>
</tr>
<tr>
<td>Example: 10</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>&quot;Y&quot;</th>
<th>The y-coordinate of the upper left corner of the element</th>
</tr>
</thead>
<tbody>
<tr>
<td>Specified in pixels</td>
<td></td>
</tr>
<tr>
<td>Example: 10</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>&quot;Width&quot;</th>
<th>Specified in pixels</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example: 150</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>&quot;Height&quot;</th>
<th>Specified in pixels</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example: 30</td>
<td></td>
</tr>
</tbody>
</table>

Tip: You can change the values in "X", "Y", "Width", and "Height" by dragging the corresponding symbols to another position in the editor.
**“Angle”**

Static angle of rotation (in degrees)

Example: 35

The element is displayed rotated in the editor. The point of rotation is the center of the element. A positive value rotates clockwise.

Tip: You can change the value in the editor by focusing the element to the handle. When the cursor is displayed as a rotating arrow, you can rotate the element about its center as a handle.

(1): Handle

Note: If a dynamic angle of rotation is also configured in the property “Absolute movement ➔ Internal rotation”, then the static and dynamic angles of rotation are added in runtime mode. The static angle of rotation acts as an offset.

See also

- § Chapter 1.3.5.3.2 “Positioning elements, adapting size and layer” on page 1113

**Radius setting**

Visible only when “Round rectangle” is selected in the “Type of element” property.

| “Radius” | How the corners are rounded.
| ——— | ——— |
| “From style” | &gt;&gt; |
| “Relative to the element size” | |
| “Explicit” | Allows for specifying a custom value in the “Value” setting. |

| “Value” | Radius of the rounded corners (in pixels)
| ——— | ——— |
| Example: 5 | |
| Requirement: “Explicit” is selected in the “Radius” setting. | |

**Element property ‘Center’**

The properties contain fixed values for the coordinates of the point of rotation. This point of rotation is shown as the symbol. The point is used as the center for rotating and scaling.

| “X” | X-coordinate of the point of rotation |
| “Y” | Y-coordinate of the point of rotation |

You can also change the values by dragging the symbols (4) to other positions in the editor.

**Element property ‘Colors’**
**“Normal state”**
The normal state is in effect if the variable in “Color variables ➔ Toggle color” is not defined or it has the value FALSE.

**“Frame color”**
Frame and fill color for the corresponding state of the variable.

**“Fill color”**

**“Transparency”**
Transparency value (0 to 255) for defining the transparency of the selected color. Example: 255: The color is opaque. 0: The color is completely transparent.

**“Alarm state”**
The alarm state is in effect if the variable in “Color variables ➔ Toggle color” has the value TRUE.

**“Use gradient color”**
☑: The element is displayed with a gradient of two colors.

**“Gradient setting”**
The “Gradient editor” dialog box opens.

See also
- ☀ Chapter 1.3.5.19.3.5 “Dialog ‘Gradient Editor’” on page 1586

**Element property ‘Appearance’**
The properties contain fixed values for setting the look of the element.

<table>
<thead>
<tr>
<th>“Line width”</th>
<th>Value in pixels</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example: 2</td>
<td></td>
</tr>
</tbody>
</table>

Note: The values 0 and 1 both result in a line weight of 1 pixel. If no line should be displayed, then the “Line style” property must be set to the option “Invisible”.

<table>
<thead>
<tr>
<th>“Fill attributes”</th>
<th>The way in which the element is filled.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>☐ “Filled”: The element is filled with the color from property “Colors ➔ Fill color”.</td>
</tr>
<tr>
<td></td>
<td>☐ “Invisible”: The fill color is invisible.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>“Line style”</th>
<th>Type of line representation</th>
</tr>
</thead>
<tbody>
<tr>
<td>☐ “Solid”</td>
<td></td>
</tr>
<tr>
<td>☐ “Dashes”</td>
<td></td>
</tr>
<tr>
<td>☐ “Dots”</td>
<td></td>
</tr>
<tr>
<td>☐ “Dash Dot”</td>
<td></td>
</tr>
<tr>
<td>☐ “Dash Dot Dot”</td>
<td></td>
</tr>
<tr>
<td>☐ “not visible”</td>
<td></td>
</tr>
</tbody>
</table>

You can assign variables in the “Appearance variables” property for controlling the appearance dynamically. The fixed values here are overwritten.

See also
- ☀ “Element property ‘Appearance variables’” on page 1285

**Element property ‘Texts’**
The properties contain character strings for labeling the element. The character string can also contain a placeholder with a format definition. In runtime mode, the placeholder is replaced by the current value in the specified format.

CODESYS accepts the specified texts automatically into the “GlobalTextList” text list. Therefore, these texts can be localized.
### “Text”

Character string (without single straight quotation marks) for the labeling the element. Add a line break by pressing the keyboard shortcut [Ctrl] + [Enter].

**Example:** `Accesses: %i`

The variable that contains the current value for the placeholder is specified in the property “Text variable ➔ Text”.

### “Tooltip”

Character string (without single straight quotation marks) that is displayed as the tooltip of an element.

**Example:** `Number of valid accesses`.

The variable that contains the current value for the placeholder is specified in the property “Text variable ➔ Tooltip”.

See also
- “Element property ‘Text variables’” on page 1229
- Chapter 1.3.5.18.2 “Placeholders with format definition in the output text” on page 1546

### Element property ‘Text properties’

The properties contain fixed values for the text properties.

<table>
<thead>
<tr>
<th><strong>“Horizontal alignment”</strong></th>
<th>Horizontal alignment of the text within the element.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>“Vertical alignment”</strong></td>
<td>Vertical alignment of the text within the element.</td>
</tr>
<tr>
<td><strong>“Text format”</strong></td>
<td>Definition for displaying texts that are too long</td>
</tr>
<tr>
<td></td>
<td>- “Default”: The long text is truncated.</td>
</tr>
<tr>
<td></td>
<td>- “Line break”: The text is split into parts.</td>
</tr>
<tr>
<td></td>
<td>- “Ellipsis”: The visible text ends with “...” indicating that it is not complete.</td>
</tr>
</tbody>
</table>

**“Font”**

**Example:** “Default”

➡️ The “Font” dialog box opens.

▶: Drop-down list with style fonts.

**“Font color”**

**Example:** “Black”

➡️ The “Color” dialog box opens.

▶: Drop-down list with style colors.

**“Transparency”**

Whole number (value range from 0 to 255). This determines the transparency of the respective color.

**Example:** 255: The color is opaque.

0: The color is completely transparent.

Please note: If the color is a style color and already has a transparency value, then this property is write-protected.

### Element property ‘Absolute movement’

The properties contain IEC variables for controlling the position of the element dynamically. The reference point is the upper left corner of the element. In runtime mode, the entire element is moved.

**“Movement”**

<table>
<thead>
<tr>
<th><strong>“X”</strong></th>
<th>Variable (numeric data type). Defines the X position (in pixels).</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Example:</strong></td>
<td>PLC_PRG.iPos_X.</td>
</tr>
<tr>
<td></td>
<td>Increasing this value in runtime mode moves the element to the right.</td>
</tr>
<tr>
<td>Property</td>
<td>Description</td>
</tr>
<tr>
<td>---------------</td>
<td>-------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>“Y”</td>
<td>Variable (numeric data type). Defines the Y position (in pixels).</td>
</tr>
<tr>
<td>“Rotation”</td>
<td>Variable (numeric data type). Defines the angle of rotation (in degrees).</td>
</tr>
<tr>
<td>“Scaling”</td>
<td>Variable (integer data type). Causes centric stretching.</td>
</tr>
<tr>
<td>“Interior rotation”</td>
<td>Variable (numeric data type). Defines the angle of rotation (in degrees).</td>
</tr>
<tr>
<td>“Use REAL values”</td>
<td>Note: Only available if the device supports the use of REAL coordinates. The properties of the absolute movement are interpreted as REAL values. The values are not rounded. The option allows for the individual fine-tuning of drawing the element, for example for the visualization of a smoother rotation. If a horizontal or vertical line is drawn blurry on a specific visualization platform, then this can be corrected by an offset of 0.5px in the direction of the line thickness.</td>
</tr>
</tbody>
</table>

You can link the variables to a unit conversion.
The properties “X”, “Y”, “Rotation”, and “Interior rotation” are supported by the "Client Animation" functionality.

See also
- Unit conversion

Element property 'Relative movement'

The properties contains variables for moving the element. The reference point is the position of the element ("Position" property). The shape of the element can change.

<table>
<thead>
<tr>
<th>“Movement top-left”</th>
</tr>
</thead>
<tbody>
<tr>
<td>“X” Variable (integer data type). It contains the number (in pixels) that the left edge is moved horizontally. Incrementing the value moves the element to the right. Example: PLC_PRG.iDeltaX</td>
</tr>
<tr>
<td>“Y” Variable (integer data type). It contains the number (in pixels) that the top edge is moved vertically. Incrementing the value moves the element to the down. Example: PLC_PRG.iDeltaY</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>“Movement bottom-right”</th>
</tr>
</thead>
<tbody>
<tr>
<td>“X” Variable (integer data type). It contains the number (in pixels) that the right edge is moved horizontally. Incrementing the value moves the element to the right. Example: PLC_PRG.iDeltaWidth</td>
</tr>
<tr>
<td>“Y” Variable (integer data type). It contains the number (in pixels) that the bottom edge is moved vertically. Incrementing the value moves the element to the down. Example: PLC_PRG.iDeltaHeight</td>
</tr>
</tbody>
</table>

See also
- “Element property ‘Absolute movement’” on page 1227

Element property 'Text variables'

These properties are variables with contents that replace a format definition.

<table>
<thead>
<tr>
<th>“Text variable”</th>
</tr>
</thead>
<tbody>
<tr>
<td>Variable (data type compliant with the format definition). It contains what is printed instead of the format definition. Example: PLC_PRG.iAccesses</td>
</tr>
<tr>
<td>Note: The format definition is part of the text in the property “Texts ➔ Text”. Note: If you specify a variable of type enumeration with text list support, then the name of the enumeration data type is added automatically in angle brackets after the variable name. Example: PLC_PRG.enVar &lt;enumeration name&gt;. Then the symbolic value of the enumeration component is printed instead of the numeric value when text is printed. Refer to the help page for the enumerations.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>“Tooltip variable”</th>
</tr>
</thead>
<tbody>
<tr>
<td>Variable (data type compliant with the format definition). It contains what is printed instead of the format definition. Example: PLC_PRG.iAccessesInTooltip</td>
</tr>
<tr>
<td>Note: The format definition is part of the text in the property “Texts ➔ Tooltip”.</td>
</tr>
</tbody>
</table>
See also

- % Chapter 1.3.5.18.2 “Placeholders with format definition in the output text” on page 1546
- % “Element property ‘Texts’” on page 1226
- Enumerations

**Element property ‘Dynamic texts’**

Dynamic texts are variably indexed texts of a text list. At runtime, the text is displayed that is currently indexed in the variable.

<table>
<thead>
<tr>
<th><strong>“Text list”</strong></th>
<th>Variable (string) or name of the text list as a fixed string in single straight quotation marks.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example: 'Errorlist'</td>
<td>▼: Drop-down list with the dialogs available in the text lists.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>“Text index”</strong></th>
<th>Text list ID. This refers to the desired output text.</th>
</tr>
</thead>
<tbody>
<tr>
<td>- As fixed string with the ID in single straight quotation marks.</td>
<td>Example: '1'</td>
</tr>
<tr>
<td>- As a variable (STRING) for dynamically controlling the text output.</td>
<td>Example: strTextID Sample assignment: PLC_PRG.strTextID := '1';</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>“Tooltip index”</strong></th>
<th>Text list ID. This refers to the desired output text.</th>
</tr>
</thead>
<tbody>
<tr>
<td>- As fixed string with the ID in single straight quotation marks.</td>
<td>Example: '2'</td>
</tr>
<tr>
<td>- As a variable (STRING) for dynamically controlling the text output.</td>
<td>Example: strToolTipID Sample assignment: PLC_PRG.strToolTipID := '2';</td>
</tr>
</tbody>
</table>

See also

- Text list

**Element property ‘Font variables’**

The variables allow for dynamic control of the text display.
| “Font name” | Variable (STRING). Includes the font of the text.  
Example: PLC_PRG.stFontVar := 'Arial';  
The selection of fonts corresponds to the default “Font” dialog. |
| “Size” | Variable (numeric data type). Contains the font size (in pixels or points). The applied unit is specified in brackets after the variable name.  
- <pt>: Points (default)  
  Example: PLC_PRG.iFontHeight <pt>  
  Code: iFontHeight : INT := 12;  
- <px>: Pixels  
  Example: PLC_PRG.iFontHeight <px>  
  Code: iFontHeight : INT := 19;  
  If you click in the value field, a drop-down list opens on the right for setting the unit.  
  Hint: The font size is specified in points (example: Arial 12). Use points when the variable font size should match a font, for example if a font is set in the property “Text property ➔ Font”. |
| “Flags” | Variable (DWORD). Contains the flags for displaying fonts.  
Flags:  
- 1: Italics  
- 2: Bold  
- 4: Underline  
- 8: Strikethrough  
  Note: You can combine the font displays by adding the coding of the flags. For example, a bold and underlined text: PLC_PRG.dwFontType := 6; |
| “Character set” | Variable (DWORD). Contains a character set number for the font.  
The selection of character set numbers corresponds to the “Script” setting of the standard “Font” dialog. |
| “Color” | Variable (DWORD). Includes the color of the text.  
Example: PLC_PRG.dwColorFont := 16#FF000000; |
| “Flags for text alignment” | Variable (integer data type). Contains the coding for text alignment.  
Example: PLC_PRG.dwTextAlignment.  
Coding:  
- 0: Top left  
- 1: Horizontal center  
- 2: Right  
- 4: Vertical center  
- 8: Bottom  
  Note: You can combine the text alignments by adding the coding of the flags. For example, a vertical and horizontal centered text: PLC_PRG.dwFontType := 5; |
Fixed values for displaying texts are set in “Text properties”.

See also
- "Element property 'Text properties'” on page 1227

---

**Element property 'Color variables'**

The Element property is used as an interface for project variables to dynamically control colors at runtime.

<table>
<thead>
<tr>
<th><strong>“Toggle color”</strong></th>
<th>The property controls the toggled color at runtime.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Value assignment:</strong></td>
<td><strong>FALSE:</strong> The element is displayed with the color specified in the “Color” property.</td>
</tr>
<tr>
<td></td>
<td><strong>TRUE:</strong> The element is displayed with the color specified in the “Alarm color” property.</td>
</tr>
<tr>
<td><strong>Assignment options:</strong></td>
<td><strong>Placeholder for the user input variable</strong></td>
</tr>
<tr>
<td></td>
<td>– “&lt;toggle/tap variable&gt;”</td>
</tr>
<tr>
<td></td>
<td>– “&lt;NOT toggle/tap variable&gt;”</td>
</tr>
<tr>
<td></td>
<td>The color change is not controlled by its own variable, but by a user input variable.</td>
</tr>
<tr>
<td></td>
<td>Note: Specify a variable for the mouse events “Tap” or “Toggle” in the input configuration of the element. Only then is the placeholder set. If you configure a variable in both “Toggle” and “Tap”, then the variable specified in “Tap” is used.</td>
</tr>
<tr>
<td></td>
<td>Hint: Click the symbol to insert the placeholder “&lt;toggle/tap variable&gt;”. When you activate the “Input configuration”, “Tap FALSE” property, then the “&lt;NOT toggle/tap variable&gt;” placeholder is displayed.</td>
</tr>
<tr>
<td></td>
<td><strong>Instance path of a project variable (BOOL)</strong></td>
</tr>
<tr>
<td></td>
<td>Example: PLC_PRG.xColorIsToggeled</td>
</tr>
<tr>
<td></td>
<td>Note: In the code, declare and implement the variable specified here. Its value assignment determines when the color changes.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>“Normal state”</strong></th>
<th>The properties listed below control the color depending on the state. The normal state is in effect if the variable in “Color variables”, “Toggle color” is not defined or it has the value FALSE. The alarm state is in effect if the variable in “Color variables”, “Toggle color” has the value TRUE.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>“Alarm state”</strong></td>
<td><strong>Frame color</strong> Assignment options:</td>
</tr>
<tr>
<td></td>
<td><strong>Variable (DWORD) for the frame color</strong></td>
</tr>
<tr>
<td></td>
<td>Example: PLC_PRG.dwBorderColor</td>
</tr>
<tr>
<td></td>
<td><strong>Color literal</strong></td>
</tr>
<tr>
<td></td>
<td>Example of green and opaque: 16#FF00FF00</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Filling color</strong></th>
<th>Assignment options:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>Variable (DWORD) for the fill color</strong></td>
</tr>
<tr>
<td></td>
<td>Example: PLC_PRG.dwFillColor</td>
</tr>
<tr>
<td></td>
<td><strong>Color literal</strong></td>
</tr>
<tr>
<td></td>
<td>Example of gray and opaque: 16#FF888888</td>
</tr>
</tbody>
</table>
The transparency part of the color value is evaluated only if the “Activate semi-transparent drawing” option of the visualization manager is selected.

Select the “Advanced” option in the toolbar of the properties view. Then all element properties are visible.

See also
- Chapter 1.3.5.8.3 “Animating a color display” on page 1152

Element property 'Appearance variables'

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Line width”</td>
<td>Variable (integer data type). Contains the line weight (in pixels).</td>
</tr>
<tr>
<td>“Fill attributes”</td>
<td>Variable (DWORD). Controls whether the fill color of the element is visible.</td>
</tr>
<tr>
<td></td>
<td>- Variable value = 0: Filled</td>
</tr>
<tr>
<td></td>
<td>- Variable value &gt; 0: Invisible; no fill color</td>
</tr>
<tr>
<td>“Line style”</td>
<td>Variable (DWORD). Controls the line style.</td>
</tr>
<tr>
<td></td>
<td>Coding:</td>
</tr>
<tr>
<td></td>
<td>- 0: Solid line</td>
</tr>
<tr>
<td></td>
<td>- 1: Dashed line</td>
</tr>
<tr>
<td></td>
<td>- 2: Dotted line</td>
</tr>
<tr>
<td></td>
<td>- 3: Line type &quot;Dash Dot&quot;</td>
</tr>
<tr>
<td></td>
<td>- 3: Line type &quot;Dash Dot Dot&quot;</td>
</tr>
<tr>
<td></td>
<td>- 8: Invisible; no line</td>
</tr>
</tbody>
</table>

Fixed values can be set in the “Appearance” property. These values can be overwritten by dynamic variables at runtime.

See also
- “Element property ‘Appearance’” on page 1238

Element property 'State variables'

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Invisible”</td>
<td>Variable (BOOL). Toggles the visibility of the element.</td>
</tr>
<tr>
<td></td>
<td>TRUE: The element is not visible at runtime.</td>
</tr>
<tr>
<td>“Deactivate inputs”</td>
<td>Variable (BOOL). Toggles the operability of the element.</td>
</tr>
<tr>
<td></td>
<td>TRUE: User inputs do not have any effect in runtime more. The element is shown as deactivated.</td>
</tr>
</tbody>
</table>
The “Invisible” property is supported by the “Client Animation” functionality.

See also

These properties are available only when you select the “Preview: Support client animations and overlay of native elements” option in the Visualization Manager.

You can animate a movement, a rotation, and the visibility of a visualization element.

<table>
<thead>
<tr>
<th>“Animation duration”</th>
<th>Time that the element executes an animation (in milliseconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Example: 500</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>“Move to the foreground”</th>
<th>Property value (BOOL)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>TRUE: At runtime, the element is displayed in the foreground.</td>
</tr>
<tr>
<td></td>
<td>FALSE: At runtime, the element is displayed in the layer where it was inserted in the visualization editor.</td>
</tr>
</tbody>
</table>

See also

- § Chapter 1.3.5.21.5 “Using client animation” on page 1962

### Element property 'Input configuration'

The properties contain the configurations for the user input when using the mouse or keyboard. A user input defines an event and one or more actions that are executed when an event occurs.

The “Configure” button opens the “Input configuration” dialog box for creating or modifying a user input.

Configured user inputs are briefly listed below the events. They each include the action that is triggered and the setting in short form.

Example: “Execute ST code”: \#PLC_PRG.i.x := 0;

<table>
<thead>
<tr>
<th>“OnDialogClosed”</th>
<th>Input event: The user closes the dialog box.</th>
</tr>
</thead>
<tbody>
<tr>
<td>“OnMouseClicked”</td>
<td>Input event: The user clicks the mouse button completely. The mouse button is clicked and released.</td>
</tr>
<tr>
<td>“OnMouseDown”</td>
<td>Input event: The user clicks down on the mouse button.</td>
</tr>
<tr>
<td>“OnMouseEnter”</td>
<td>Input event: The user drags the mouse pointer to the element.</td>
</tr>
<tr>
<td>“OnMouseLeave”</td>
<td>Input event: The user drags the mouse pointer away from the element.</td>
</tr>
<tr>
<td>“OnMouseMove”</td>
<td>Input event: The user moves the mouse pointer over the element area.</td>
</tr>
<tr>
<td>“OnMouseUp”</td>
<td>Input event: The user releases the mouse button over the element area.</td>
</tr>
<tr>
<td><strong>“Tap”</strong></td>
<td>When a mouse click event occurs, the variable defined in “Variable” is described in the application. The coding depends on the options “Tap FALSE” and “Tap on enter if captured”.</td>
</tr>
<tr>
<td>---------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>“Variable”</strong></td>
<td>Variable <code>(BOOL)</code> that is set on mouse click.</td>
</tr>
<tr>
<td>Example</td>
<td>PLC_PRG.bIsTapped</td>
</tr>
<tr>
<td>TRUE:</td>
<td>A mouse click event exists. It lasts while the user presses the mouse button over the element. It ends when the button is released.</td>
</tr>
<tr>
<td>FALSE:</td>
<td>A mouse click event does not exist.</td>
</tr>
<tr>
<td>Requirement:</td>
<td>The “Tap FALSE” option is not activated.</td>
</tr>
<tr>
<td><strong>“Tap FALSE”</strong></td>
<td>☑: The mouse click event leads to a complementary value in “Variable”.</td>
</tr>
<tr>
<td>TRUE:</td>
<td>A mouse click event does not exist.</td>
</tr>
<tr>
<td>FALSE:</td>
<td>While the mouse click event exists.</td>
</tr>
<tr>
<td><strong>“Tap on enter if captured”</strong></td>
<td>☑: During user input, it is also taken into consideration whether the mouse pointer is dragged within the element area or not while the mouse button is pressed.</td>
</tr>
<tr>
<td>TRUE:</td>
<td>While the mouse click event exists and the mouse pointer is moved over the element area.</td>
</tr>
<tr>
<td>FALSE:</td>
<td>A mouse click event does not exist. Or the user moves the mouse pointer outside of the element area while the mouse button is pressed.</td>
</tr>
<tr>
<td></td>
<td>The value is TRUE again as soon as the user moves the pointer back to the element area. The mouse is then captured.</td>
</tr>
<tr>
<td><strong>“Shift”</strong></td>
<td>With the onset of a mouse click event, the variable is set; when the mouse click event is completed, the variable is reset.</td>
</tr>
<tr>
<td><strong>“Variable”</strong></td>
<td>Variable <code>(BOOL)</code>. Its value toggled when the mouse click event is ended. This is when the user releases the mouse button while the mouse pointer is over the element area.</td>
</tr>
<tr>
<td></td>
<td>If the user releases the mouse button while the mouse pointer is outside of the element area, then the mouse click event is not ended and the value is not toggled.</td>
</tr>
<tr>
<td></td>
<td>Tip: The user can cancel a started toggle input by dragging the mouse pointer out of the element area.</td>
</tr>
<tr>
<td><strong>“Toggle on up if captured”</strong></td>
<td>☑: The value toggles regardless of where the mouse pointer is when the mouse button is released. The mouse is then captured.</td>
</tr>
<tr>
<td><strong>“Keyboard shortcuts”</strong></td>
<td>Keyboard shortcut on the element for triggering specific input actions.</td>
</tr>
<tr>
<td></td>
<td>When the keyboard shortcut event occurs, the input actions in the “Event(s)” property are triggered. In this way, it is not the input action itself that leads to this input action, but the mouse input action.</td>
</tr>
<tr>
<td><strong>“Key”</strong></td>
<td>Key pressed for input action.</td>
</tr>
<tr>
<td>Example</td>
<td><code>[T]</code></td>
</tr>
<tr>
<td>Note:</td>
<td>The following properties appear when a key is selected.</td>
</tr>
</tbody>
</table>
### Event(s)
- "None"
- "Mouse down": Pressing the key triggers the input actions that are configured in the "OnMouseDown" property.
- "Mouse up": Releasing the key triggers the input actions that are configured in the "OnMouseUp" property.
- "Mouse down/up": Pressing and releasing the key triggers the input actions that are configured in the "OnMouseDown" property and the "OnMouseUp" property.

### Shift
- Combination with the Shift key
- Example: [Shift]+[T].

### Control
- Combination with the Ctrl key
- Example: [Ctrl]+[T].

### Alt
- Combination with the Alt key
- Example: [Alt]+[T].

---

> All keyboard shortcuts and their actions that are configured in the visualization are listed in the “Keyboard configuration” tab.

---

**See also**
- § Chapter 1.3.5.19.2.2 “Command ‘Keyboard Configuration’” on page 1558
- § Chapter 1.3.5.19.3.6 “Dialog ‘Input Configuration’” on page 1586

### Element property 'Access rights'

<table>
<thead>
<tr>
<th>&quot;Access rights&quot;</th>
</tr>
</thead>
<tbody>
<tr>
<td>Opens the “Access rights” dialog. There you can edit the access privileges for the element.</td>
</tr>
<tr>
<td>Status messages:</td>
</tr>
<tr>
<td>- &quot;Not set. Full rights.&quot;: Access rights for all user groups: “operable”</td>
</tr>
<tr>
<td>- &quot;Rights are set: Limited rights&quot;: Access is restricted for at least one group.</td>
</tr>
</tbody>
</table>

**See also**
- § Chapter 1.3.5.19.3.1 “Dialog ‘Access Rights’” on page 1583

---

### Visualization element 'Line'

**Symbol:**

Tag: "Basic"

The element draws a single line.
Element properties

"Element name" | Optional.
Example: Separator_Header
Hint: Assign individual names for elements so that they are found faster in the element list.

"Type of element" | Line

Element property 'Position'
The following properties define the position and length of the element in the visualization window. These are based on the Cartesian coordinate system. The origin is located at the upper left corner of the window. The positive horizontal x-axis runs to the right. The positive vertical y-axis runs downwards.

"Dots" | "[0]": Coordinates of the starting point
"[1]": Coordinate of the end point
You can also change the values by dragging the box symbols (_drag_to_other_positions_) to other positions in the editor.

"Angle" | Static angle of rotation (in degrees).
Example: 35
The element is displayed rotated in the editor. The point of rotation is the center of the element. A positive value rotates clockwise.
Tip: You can change the value in the editor by focusing the element to the handle. When the cursor is displayed as a rotating arrow (rotate_handle_), you can rotate the element about its center as a handle.
Example:

(1): Handle
Note: If a dynamic angle of rotation is also configured in the property "Absolute movement ➔ Internal rotation", then the static and dynamic angles of rotation are added in runtime mode. The static angle of rotation acts as an offset.

See also

Element property 'Center'
The properties contain fixed values for the coordinates of the point of rotation. This point of rotation is shown as the symbol. The point is used as the center for rotating and scaling.

"X" | X-coordinate of the point of rotation
"Y" | Y-coordinate of the point of rotation
Element property 'Colors'

The properties contain fixed values for setting colors.

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Color”</td>
<td>Color of the line in normal state.</td>
</tr>
<tr>
<td></td>
<td>Please note that the normal state is in effect if the expression in the</td>
</tr>
<tr>
<td></td>
<td>“Color variables ➔ Toggle color” property is not defined or it has the value</td>
</tr>
<tr>
<td></td>
<td>FALSE.</td>
</tr>
<tr>
<td>“Alarm color”</td>
<td>Color of the line in alarm state.</td>
</tr>
<tr>
<td></td>
<td>Please note that the alarm state is in effect if the expression in the</td>
</tr>
<tr>
<td></td>
<td>“Color variables ➔ Toggle color” property has the value TRUE.</td>
</tr>
<tr>
<td>“Transparency”</td>
<td>Value (0 to 255) for defining the transparency of the selected color.</td>
</tr>
<tr>
<td></td>
<td>Example 255: The color is opaque. 0: The color is completely transparent.</td>
</tr>
</tbody>
</table>

See also
- "Element property 'Appearance variables'"

Element property 'Appearance'

The properties contain fixed values for setting the look of the element.

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Line width”</td>
<td>Value in pixels</td>
</tr>
<tr>
<td></td>
<td>Example: 2</td>
</tr>
<tr>
<td></td>
<td>Note: The values 0 and 1 both result in a line weight of one pixel. If no line</td>
</tr>
<tr>
<td></td>
<td>should be displayed, then the “Line style” property must be set to the option</td>
</tr>
<tr>
<td></td>
<td>“Invisible”.</td>
</tr>
<tr>
<td>“Line style”</td>
<td>Type of line representation</td>
</tr>
<tr>
<td></td>
<td>• &quot;Solid&quot;</td>
</tr>
<tr>
<td></td>
<td>• &quot;Dashes&quot;</td>
</tr>
<tr>
<td></td>
<td>• “Dots”</td>
</tr>
<tr>
<td></td>
<td>• “Dash Dot”</td>
</tr>
<tr>
<td></td>
<td>• “Dash Dot Dot”</td>
</tr>
<tr>
<td></td>
<td>• “not visible”</td>
</tr>
</tbody>
</table>

You can assign variables in the “Appearance variables” property for controlling the appearance dynamically. The fixed values are defined here.

See also
- "Element property 'Appearance variables'" on page 1285

Element property 'Texts'

The properties contain character strings for labeling the element. The character string can also contain a placeholder with a format definition. In runtime mode, the placeholder is replaced by the current value in the specified format.

CODESYS accepts the specified texts automatically into the “GlobalTextList” text list. Therefore, these texts can be localized.
### Text

Character string (without single straight quotation marks) for the labeling the element. Add a line break by pressing the keyboard shortcut [Ctrl] + [Enter].

**Example:** Accesses: %i

The variable that contains the current value for the placeholder is specified in the property “Text variable ➔ Text”.

### Tooltip

Character string (without single straight quotation marks) that is displayed as the tooltip of an element.

**Example:** Number of valid accesses.

The variable that contains the current value for the placeholder is specified in the property “Text variable ➔ Tooltip”.

### See also

- “Element property ’Text variables’” on page 1229
- Chapter 1.3.5.18.2 “Placeholders with format definition in the output text” on page 1546

### Horizontal alignment

Horizonal alignment of the text within the element.

### Vertical alignment

Vertical alignment of the text within the element.

### Text format

Definition for displaying texts that are too long

- **Default**: The long text is truncated.
- **Line break**: The text is split into parts.
- **Ellipsis**: The visible text ends with “...” indicating that it is not complete.

### Font

**Example:** “Default”

- The “Font” dialog box opens.
- : Drop-down list with style fonts.

### Font color

**Example:** “Black”

- The “Color” dialog box opens.
- : Drop-down list with style colors.

### Transparency

Whole number (value range from 0 to 255). This determines the transparency of the respective color.

**Example:** 255: The color is opaque.

0: The color is completely transparent.

Please note: If the color is a style color and already has a transparency value, then this property is write-protected.

### Movement

**X**

Variable (numeric data type). Defines the X position (in pixels).

**Example:** PLC_PRG.iPos_X.

Increasing this value in runtime mode moves the element to the right.
| **“Y”** | Variable (numeric data type). Defines the Y position (in pixels).  
Example: PLC_PRG.iPos_Y.  
Increasing this value in runtime mode moves the element downwards. |
| **“Rotation”** | Variable (numeric data type). Defines the angle of rotation (in degrees).  
Example: PLC_PRG.iAngle1.  
The midpoint of the element rotates at the “Center” point. This rotation point is shown as the symbol.  
In runtime mode, the alignment of the element remains the same with respect to the coordinate system of the visualization. Increasing the value rotates the element to the right. |
| **“Scaling”** | Variable (integer data type). Causes centric stretching.  
Example: PLC_PRG.iScaling.  
The reference point is the “Center” property.  
The value 1 shrinks the element by a factor of 0.001. The value 1000 returns the element to its original size. |
| **“Interior rotation”** | Variable (numeric data type). Defines the angle of rotation (in degrees).  
Example: PLC_PRG.iAngle2.  
In runtime mode, the element rotates about the point of rotation specified in “Center” according to the value of the variable. In addition, the alignment of the element rotates according to the coordinate system of the visualization. Increasing the value in the code rotates clockwise.  
The rotation point is shown as the symbol.  
Note: If a static angle of rotation is specified in the property “Position ➔ Angle”, then the static angle of rotation is added to the variable angle of rotation (offset) when the visualization is executed. |
| **“Use REAL values”** | Note: Only available if the device supports the use of REAL coordinates.  
The properties of the absolute movement are interpreted as REAL values. The values are not rounded.  
The option allows for the individual fine-tuning of drawing the element, for example for the visualization of a smoother rotation.  
Hint: If a horizontal or vertical line is drawn blurry on a specific visualization platform, then this can be corrected by an offset of 0.5px in the direction of the line thickness. |
The properties “X”, “Y”, “Rotation”, and “Interior rotation” are supported by the “Client Animation” functionality.

See also
- Unit conversion

Element property ‘Relative movement’
The properties contain variables for moving the element. The reference point is the position of the element (‘Position’ property). The shape of the element can change.

<table>
<thead>
<tr>
<th>“Movement point[0]”</th>
<th>Variable (numeric data type). It contains the number (in pixels) that the starting point of the line is moved.</th>
</tr>
</thead>
<tbody>
<tr>
<td>“X”</td>
<td>Incrementing the X value moves the element to the right.</td>
</tr>
<tr>
<td>“Y”</td>
<td>Incrementing the Y value moves the element to the down.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>“Movement point[1]”</th>
<th>Variable (numeric data type). It contains the number (in pixels) that the end point of the line is moved.</th>
</tr>
</thead>
<tbody>
<tr>
<td>“X”</td>
<td>Incrementing the X value moves the element to the right.</td>
</tr>
<tr>
<td>“Y”</td>
<td>Incrementing the Y value moves the element to the down.</td>
</tr>
</tbody>
</table>

See also
- “Element property ‘Absolute movement’” on page 1227

Element property ‘Text variables’
These properties are variables with contents that replace a format definition.

<table>
<thead>
<tr>
<th>“Text variable”</th>
<th>Variable (data type compliant with the format definition). It contains what is printed instead of the format definition.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example: PLC_PRG.iAccesses</td>
<td></td>
</tr>
<tr>
<td>Note: The format definition is part of the text in the property “Texts ➔ Text”.</td>
<td></td>
</tr>
<tr>
<td>Note: If you specify a variable of type enumeration with text list support, then the name of the enumeration data type is added automatically in angle brackets after the variable name. Example: PLC_PRG.enVar &lt;enumeration name&gt;. Then the symbolic value of the enumeration component is printed instead of the numeric value when text is printed. Refer to the help page for the enumerations.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>“Tooltip variable”</th>
<th>Variable (data type compliant with the format definition). It contains what is printed instead of the format definition.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example: PLC_PRG.iAccessesInTooltip</td>
<td></td>
</tr>
</tbody>
</table>
| Note: The format definition is part of the text in the property “Texts ➔ Tooltip”.

See also
- “Chapter 1.3.5.18.2 “Placeholders with format definition in the output text” on page 1546”
- “Element property ‘Texts’” on page 1226
- Enumerations

Element property ‘Dynamic texts’
Dynamic texts are variably indexed texts of a text list. At runtime, the text is displayed that is currently indexed in the variable.
| **“Text list”** | Variable (string) or name of the text list as a fixed string in single straight quotation marks.  
Example: 'Errorlist'  
▼: Drop-down list with the dialogs available in the text lists. |
| **“Text index”** | Text list ID. This refers to the desired output text.  
- As fixed string with the ID in single straight quotation marks.  
  Example: '1'  
- As a variable (STRING) for dynamically controlling the text output.  
  Example: strTextID  
  Sample assignment: PLC_PRG.strTextID := '1'; |
| **“Tooltip index”** | Text list ID. This refers to the desired output text.  
- As fixed string with the ID in single straight quotation marks.  
  Example: '2'  
- As a variable (STRING) for dynamically controlling the text output.  
  Example: strToolTipID  
  Sample assignment: PLC_PRG.strToolTipID := '2'; |

See also
- **Text list**

**Element property “Font variables”**

The variables allow for dynamic control of the text display.

| **“Font name”** | Variable (STRING). Includes the font of the text.  
Example: PLC_PRG.stFontVar := 'Arial';  
The selection of fonts corresponds to the default “Font” dialog. |
| **“Size”** | Variable (numeric data type). Contains the font size (in pixels or points). The applied unit is specified in brackets after the variable name.  
- <pt>: Points (default)  
  Example: PLC_PRG.iFontHeight <pt>  
  Code: iFontHeight : INT := 12;  
- <px>: Pixels  
  Example: PLC_PRG.iFontHeight <px>  
  Code: iFontHeight : INT := 19; |

If you click in the value field, a drop-down list opens on the right for setting the unit.

Hint: The font size is specified in points (example: Arial 12). Use points when the variable font size should match a font, for example if a font is set in the property “Text property ➔ Font”.

---

**PLC Automation with V3 CPUs**  
**Programming with IEC 61131-3 editor > CODESYS Visualization**  
2020/12/10  
3ADR010583, 1, en_US  
1646
### “Flags”

Variable (DWORD). Contains the flags for displaying fonts.

Flags:
- 1: Italics
- 2: Bold
- 4: Underline
- 8: Strikethrough

Note: You can combine the font displays by adding the coding of the flags. For example, a bold and underlined text: ```PLC_PRG.dwFontType := 6;``` 

### “Character set”

Variable (DWORD). Contains a character set number for the font.

The selection of character set numbers corresponds to the “Script” setting of the standard “Font” dialog.

### “Color”

Variable (DWORD). Includes the color of the text.

Example: ```PLC_PRG.dwColorFont := 16#FF000000;``` 

### “Flags for text alignment”

Variable (integer data type). Contains the coding for text alignment.

Example: ```PLC_PRG.dwTextAlignment := 5;``` 

- 0: Top left
- 1: Horizontal center
- 2: Right
- 4: Vertical center
- 8: Bottom

Note: You can combine the text alignments by adding the coding of the flags. For example, a vertical and horizontal centered text: ```PLC_PRG.dwFontType := 5;``` 

---

*Fixed values for displaying texts are set in “Text properties”.*

See also
- ₍ “Element property ‘Text properties’” on page 1227

---

**Element property ‘Colorvariables’**

The Element property is used as an interface for project variables to dynamically control colors at runtime.
"Toggle color"  The property controls the toggled color at runtime.

Value assignment:
- **FALSE:** The element is displayed with the color specified in the "Color" property.
- **TRUE:** The element is displayed with the color specified in the "Alarm color" property.

Assigning the property:
- Placeholder for the user input variable
  - "<toggle/tap variable>"
  - "<NOT toggle/tap variable>"

The color change is not controlled by its own variable, but by a user input variable.

Note: Specify a variable for the mouse events "Tap" or "Toggle" in the input configuration of the element. Only then is the placeholder set. If you configure a variable in both "Toggle" and "Tap", then the variable specified in "Tap" is used.

Hint: Click the symbol to insert the placeholder "<toggle/tap variable>". When you activate the "Inputconfiguration", "Tap FALSE" property, then the "<NOT toggle/tap variable>" placeholder is displayed.

- Instance path of a project variable (BOOL)
  Example: PLC_PRG.xColorIsToggeled

Note: In the code, declare and implement the variable specified here. Its value assignment determines when the color changes.

"Color"

- Variable (DWORD) for the color
  Example: PLC_PRG.dwColor
- Color literal
  Example of gray and opaque: 16#FF888888

Please note that the normal state is in effect if the expression in the "Colorvariables ➔ Toggle color" property is not defined or it has the value FALSE.

"Alarm color"

- Color variable in the alarm state
- Variable (DWORD) for the alarm color
  Example: PLC_PRG.dwAlarmColor
- Color literal
  Example of red and opaque: 16#FFFF0000

Please note that the alarm state is in effect if the expression in the "Colorvariables ➔ Toggle color" property has the value TRUE.

The transparency part of the color value is evaluated only if the “Activate semi-transparent drawing” option of the visualization manager is selected.

Select the “Advanced” option in the toolbar of the properties view. Then all element properties are visible.

See also:
- ¶ Chapter 1.3.5.8.3 “Animating a color display” on page 1152
The variables control the element behavior dynamically.

<table>
<thead>
<tr>
<th>“Invisible”</th>
<th>Variable (BOOL). Toggles the visibility of the element.</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRUE</td>
<td>The element is not visible at runtime.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>“Deactivate inputs”</th>
<th>Variable (BOOL). Toggles the operability of the element.</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRUE</td>
<td>User inputs do not have any effect in runtime more. The element is shown as deactivated.</td>
</tr>
</tbody>
</table>

The “Invisible” property is supported by the “Client Animation” functionality.

See also

- Addition

Element property 'Line width variable

Dynamic definition of the weight of a line element using a variable.

| “Integer value” | Variable (integer data type). Defines the line weight of the element (in pixels). This overwrites the fixed value that is defined in “Appearance → Line weight”. Note: The value 0 codes the same as 1 and sets the line weight to one pixel. |

Element property 'Line style variable

<table>
<thead>
<tr>
<th>“Integer value”</th>
<th>Variable (integer data type). Defines the appearance of the line at runtime.</th>
</tr>
</thead>
<tbody>
<tr>
<td>1: Solid</td>
<td></td>
</tr>
<tr>
<td>2: Dashes</td>
<td></td>
</tr>
<tr>
<td>3: Dots</td>
<td></td>
</tr>
<tr>
<td>4: Dash Dot</td>
<td></td>
</tr>
<tr>
<td>5: Dash Dot Dot</td>
<td></td>
</tr>
<tr>
<td>6: Invisible:</td>
<td>The line is not drawn.</td>
</tr>
</tbody>
</table>

These properties are available only when you select the “Preview: Support client animations and overlay of native elements” option in the Visualization Manager.

You can animate a movement, a rotation, and the visibility of a visualization element.

<table>
<thead>
<tr>
<th>“Animation duration”</th>
<th>Time that the element executes an animation (in milliseconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example: 500</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>“Move to the foreground”</th>
<th>Property value (BOOL)</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRUE</td>
<td>At runtime, the element is displayed in the foreground.</td>
</tr>
<tr>
<td>FALSE</td>
<td>At runtime, the element is displayed in the layer where it was inserted in the visualization editor.</td>
</tr>
</tbody>
</table>

See also

- Chapter 1.3.5.21.5 “Using client animation” on page 1962
The properties contain the configurations for the user input when using the mouse or keyboard. A user input defines an event and one or more actions that are executed when an event occurs.

The “Configure” button opens the “Input configuration” dialog box for creating or modifying a user input. Configured user inputs are briefly listed below the events. They each include the action that is triggered and the setting in short form.

Example: “Execute ST code”:  \$PLC_PRG.i_x := 0;

<table>
<thead>
<tr>
<th>Event</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>“OnDialogClosed”</td>
<td>Input event: The user closes the dialog box.</td>
</tr>
<tr>
<td>“OnMouseClick”</td>
<td>Input event: The user clicks the mouse button completely. The mouse button is clicked and released.</td>
</tr>
<tr>
<td>“OnMouseDown”</td>
<td>Input event: The user clicks down on the mouse button.</td>
</tr>
<tr>
<td>“OnMouseEnter”</td>
<td>Input event: The user drags the mouse pointer to the element.</td>
</tr>
<tr>
<td>“OnMouseLeave”</td>
<td>Input event: The user drags the mouse pointer away from the element.</td>
</tr>
<tr>
<td>“OnMouseMove”</td>
<td>Input event: The user moves the mouse pointer over the element area.</td>
</tr>
<tr>
<td>“OnMouseUp”</td>
<td>Input event: The user releases the mouse button over the element area.</td>
</tr>
</tbody>
</table>

“Tap”
When a mouse click event occurs, the variable defined in “Variable” is described in the application. The coding depends on the options “Tap FALSE” and “Tap on enter if captured”.

“Variable”
Variable (BOOL) that is set on mouse click.

Example: PLC_PRG.bIsTapped

TRUE: A mouse click event exists. It lasts while the user presses the mouse button over the element. It ends when the button is released.
FALSE: A mouse click event does not exist.

Requirement: The “Tap FALSE” option is not activated.

“Tap FALSE”

<table>
<thead>
<tr>
<th>boolean</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRUE</td>
<td>The mouse click event leads to a complementary value in “Variable”.</td>
</tr>
<tr>
<td>FALSE</td>
<td>A mouse click event does not exist.</td>
</tr>
<tr>
<td>FALSE</td>
<td>While the mouse click event exists.</td>
</tr>
</tbody>
</table>

“Tap on enter if captured”

<table>
<thead>
<tr>
<th>boolean</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRUE</td>
<td>During user input, it is also taken into consideration whether the mouse pointer is dragged within the element area or not while the mouse button is pressed.</td>
</tr>
<tr>
<td>TRUE</td>
<td>While the mouse click event exists and the mouse pointer is moved over the element area.</td>
</tr>
<tr>
<td>FALSE</td>
<td>A mouse click event does not exist. Or the user moves the mouse pointer outside of the element area while the mouse button is pressed. The value is TRUE again as soon as the user moves the pointer back to the element area. The mouse is then captured.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRUE</td>
<td>The mouse is then captured.</td>
</tr>
<tr>
<td>FALSE</td>
<td>The mouse is not captured.</td>
</tr>
</tbody>
</table>
**“Shift”**

With the onset of a mouse click event, the variable is set; when the mouse click event is completed, the variable is reset.

**“Variable”**

Variable (BOOL). Its value toggled when the mouse click event is ended. This is when the user releases the mouse button while the mouse pointer is over the element area.

If the user releases the mouse button while the mouse pointer is outside of the element area, then the mouse click event is not ended and the value is not toggled.

Tip: The user can cancel a started toggle input by dragging the mouse pointer out of the element area.

**“Toggle on up if captured”**

☑: The value toggles regardless of where the mouse pointer is when the mouse button is released. The mouse is then captured.

**“Keyboard shortcuts”**

Keyboard shortcut on the element for triggering specific input actions.

When the keyboard shortcut event occurs, the input actions in the “Event(s)” property are triggered. In this way, it is not the input action itself that leads to this input action, but the mouse input action.

**“Key”**

Key pressed for input action.

Example: [T]

Note: The following properties appear when a key is selected.

**“Event(s)”**

- “None”
- “Mouse down”: Pressing the key triggers the input actions that are configured in the “OnMouseDown” property.
- “Mouse up”: Releasing the key triggers the input actions that are configured in the “OnMouseUp” property.
- “Mouse down/up”: Pressing and releasing the key triggers the input actions that are configured in the “OnMouseDown” property and the “OnMouseUp” property.

**“Shift”**

☑: Combination with the Shift key

Example: [Shift]+[T].

**“Control”**

☑: Combination with the Ctrl key

Example: [Ctrl]+[T].

**“Alt”**

☑: Combination with the Alt key

Example: [Alt]+[T].

---

**All keyboard shortcuts and their actions that are configured in the visualization are listed in the “Keyboard configuration” tab.**

See also

- ☰ Chapter 1.3.5.19.2.2 “Command ‘Keyboard Configuration’” on page 1558
- ☰ Chapter 1.3.5.19.3.6 “Dialog ‘Input Configuration’” on page 1586

**Requirement:** User management is set up for the visualization.
“Access rights”

Opens the “Access rights” dialog. There you can edit the access privileges for the element.

Status messages:

- “Not set. Full rights.”: Access rights for all user groups: “operable”
- “Rights are set: Limited rights”: Access is restricted for at least one group.

See also

- ☀ Chapter 1.3.5.19.3.1 "Dialog ‘Access Rights’" on page 1583

See also

Visualization element 'Polygon', 'Polyline', 'Bézier curve'

Symbol:

Tag: “Basic”

The polygon, polyline, and Bézier curve are the same element type. They can be converted into another type by changing the “Element type” property.

Elements can be dragged and dropped to the editor. The element is then drawn with five points: [0] to [4].

Other positions are added as follows: Move the mouse pointer over a corner point; the mouse pointer changes shape. Now if you press and hold [Ctrl] and click the left mouse button, another point is created. You can delete a point by pressing and holding [Shift]+[Ctrl] and click the selected point.

As an alternative, you can select the element in the toolbox area and in the editor click multiple times. At the same time, a connecting line is drawn from one point to the other. End by double-clicking the element or right-clicking it one time.

Element properties

<table>
<thead>
<tr>
<th>&quot;Element name&quot;</th>
<th>Optional.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Hint: Assign individual names for elements so that they are found faster in the element list.</td>
</tr>
<tr>
<td>Example: Werkstück_1</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>&quot;Type of element&quot;</th>
<th>● “Polygon”</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>● “Polyline”</td>
</tr>
<tr>
<td></td>
<td>● “Bézier curve”</td>
</tr>
</tbody>
</table>

Element property 'Position'

The following properties define the position of the corner points in the visualization window. These are based on the Cartesian coordinate system. The origin is located at the upper left corner of the window. The positive horizontal x-axis runs to the right. The positive vertical y-axis runs downwards.
### "Dots"

[0...n]: Coordinates of the corner points

Specified in pixels

You can also change the values by dragging the box symbols (ogene) to other positions in the editor.

### "Angle"

Static angle of rotation (in degrees).

Example: 35

The element is displayed rotated in the editor. The point of rotation is the center of the element. A positive value rotates clockwise.

Tip: You can change the value in the editor by focusing the element to the handle. When the cursor is displayed as a rotating arrow (ogene), you can rotate the element about its center as a handle.

(1): Handle

Note: If a dynamic angle of rotation is also configured in the property "Absolute movement ➜ Internal rotation", then the static and dynamic angles of rotation are added in runtime mode. The static angle of rotation acts as an offset.

See also

- 

### Element property 'Center'

The properties contain fixed values for the coordinates of the point of rotation. This point of rotation is shown as the symbol. The point is used as the center for rotating and scaling.

<table>
<thead>
<tr>
<th>&quot;X&quot;</th>
<th>X-coordinate of the point of rotation</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Y&quot;</td>
<td>Y-coordinate of the point of rotation</td>
</tr>
</tbody>
</table>

You can also change the values by dragging the symbols (ogene) to other positions in the editor.

### Element property 'Colors'

| "Normal state" | The normal state is in effect if the variable in "Color variables ➜ Toggle color" is not defined or it has the value FALSE. |
| "Frame color"  | Frame and fill color for the corresponding state of the variable. |
| "Fill color"   | |
| "Transparency" | Transparency value (0 to 255) for defining the transparency of the selected color. Example: 255: The color is opaque. 0: The color is completely transparent. |
| "Alarm state"  | The alarm state is in effect if the variable in "Color variables ➜ Toggle color" has the value TRUE. |
| "Use gradient color" | : The element is displayed with a gradient of two colors. |
| "Gradient setting" | The "Gradient editor" dialog box opens. |
The properties contain fixed values for setting the look of the element.

**Element property 'Appearance'**

<table>
<thead>
<tr>
<th><strong>“Line width”</strong></th>
<th>Value in pixels</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example: 2</td>
<td></td>
</tr>
<tr>
<td>Note: The values 0 and 1 both result in a line weight of 1 pixel. If no line should be displayed, then the “Line style” property must be set to the option “Invisible”.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>“Fill attributes”</strong></th>
<th>The way in which the element is filled.</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Filled”: The element is filled with the color from property “Colors ➔ Fill color”.</td>
<td></td>
</tr>
<tr>
<td>“Invisible”: The fill color is invisible.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>“Line style”</strong></th>
<th>Type of line representation</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Solid”</td>
<td></td>
</tr>
<tr>
<td>“Dashes”</td>
<td></td>
</tr>
<tr>
<td>“Dots”</td>
<td></td>
</tr>
<tr>
<td>“Dash Dot”</td>
<td></td>
</tr>
<tr>
<td>“Dash Dot Dot”</td>
<td></td>
</tr>
<tr>
<td>“not visible”</td>
<td></td>
</tr>
</tbody>
</table>

You can assign variables in the “Appearance variables” property for controlling the appearance dynamically. The fixed values here are overwritten.

See also

- [Chapter 1.3.5.19.3.5 “Dialog ‘Gradient Editor’” on page 1586](#

**Element property 'Texts'**

The properties contains character strings for labeling the element. The character string can also contain a placeholder with a format definition. In runtime mode, the placeholder is replaced by the current value in the specified format.

CODESYS accepts the specified texts automatically into the “GlobalTextList” text list. Therefore, these texts can be localized.

<table>
<thead>
<tr>
<th><strong>“Text”</strong></th>
<th>Character string (without single straight quotation marks) for the labeling the element. Add a line break by pressing the keyboard shortcut [Ctrl] + [Enter].</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example:</td>
<td>Accesses: %i</td>
</tr>
<tr>
<td>The variable that contains the current value for the placeholder is specified in the property “Text variable ➔ Text”.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>“Tooltip”</strong></th>
<th>Character string (without single straight quotation marks) that is displayed as the tooltip of an element.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example:</td>
<td>Number of valid accesses.</td>
</tr>
<tr>
<td>The variable that contains the current value for the placeholder is specified in the property “Text variable ➔ Tooltip”.</td>
<td></td>
</tr>
</tbody>
</table>
See also

- ☰ “Element property ‘Text variables’” on page 1229
- ☰
- ☰ Chapter 1.3.5.18.2 “Placeholders with format definition in the output text” on page 1546

### Element property ‘Text properties’

The properties contain fixed values for the text properties.

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Horizontal alignment&quot;</td>
<td>Horizontal alignment of the text within the element.</td>
</tr>
<tr>
<td>&quot;Vertical alignment&quot;</td>
<td>Vertical alignment of the text within the element.</td>
</tr>
<tr>
<td>&quot;Text format&quot;</td>
<td>Definition for displaying texts that are too long</td>
</tr>
<tr>
<td>- &quot;Default&quot;: The long text is truncated.</td>
<td></td>
</tr>
<tr>
<td>- &quot;Line break&quot;: The text is split into parts.</td>
<td></td>
</tr>
<tr>
<td>- &quot;Ellipsis&quot;: The visible text ends with &quot;...&quot; indicating that it is not complete.</td>
<td></td>
</tr>
<tr>
<td>&quot;Font&quot;</td>
<td>Example: &quot;Default&quot;</td>
</tr>
<tr>
<td></td>
<td>☰: The “Font” dialog box opens.</td>
</tr>
<tr>
<td></td>
<td>☰: Drop-down list with style fonts.</td>
</tr>
<tr>
<td>&quot;Font color&quot;</td>
<td>Example: “Black”</td>
</tr>
<tr>
<td></td>
<td>☰: The “Color” dialog box opens.</td>
</tr>
<tr>
<td></td>
<td>☰: Drop-down list with style colors.</td>
</tr>
<tr>
<td>&quot;Transparency&quot;</td>
<td>Whole number (value range from 0 to 255). This determines the transparency of the respective color.</td>
</tr>
<tr>
<td></td>
<td>Example: 255: The color is opaque.</td>
</tr>
<tr>
<td></td>
<td>0: The color is completely transparent.</td>
</tr>
<tr>
<td></td>
<td>Please note: If the color is a style color and already has a transparency value, then this property is write-protected.</td>
</tr>
</tbody>
</table>

### Element property ‘Absolute movement’

The properties contain IEC variables for controlling the position of the element dynamically. The reference point is the upper left corner of the element. In runtime mode, the entire element is moved.

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Movement&quot;</td>
<td></td>
</tr>
<tr>
<td>&quot;X&quot;</td>
<td>Variable (numeric data type). Defines the X position (in pixels).</td>
</tr>
<tr>
<td></td>
<td>Example: PLC_PRG.iPos_X.</td>
</tr>
<tr>
<td></td>
<td>Increasing this value in runtime mode moves the element to the right.</td>
</tr>
<tr>
<td>&quot;Y&quot;</td>
<td>Variable (numeric data type). Defines the Y position (in pixels).</td>
</tr>
<tr>
<td></td>
<td>Example: PLC_PRG.iPos_Y.</td>
</tr>
<tr>
<td></td>
<td>Increasing this value in runtime mode moves the element downwards.</td>
</tr>
</tbody>
</table>
### "Rotation"

Variable (numeric data type). Defines the angle of rotation (in degrees).

**Example:** PLC_PRG.iAngle1.

The midpoint of the element rotates at the "Center" point. This rotation point is shown as the symbol.

In runtime mode, the alignment of the element remains the same with respect to the coordinate system of the visualization. Increasing the value rotates the element to the right.

### "Scaling"

Variable (integer data type). Causes centric stretching.

**Example:** PLC_PRG.iScaling.

The reference point is the "Center" property.

The value 1 shrinks the element by a factor of 0.001. The value 1000 returns the element to its original size.

### "Interior rotation"

Variable (numeric data type). Defines the angle of rotation (in degrees).

**Example:** PLC_PRG.iAngle2.

In runtime mode, the element rotates about the point of rotation specified in "Center" according to the value of the variable. In addition, the alignment of the element rotates according to the coordinate system of the visualization. Increasing the value in the code rotates clockwise.

The rotation point is shown as the symbol.

Note: If a static angle of rotation is specified in the property "Position → Angle", then the static angle of rotation is added to the variable angle of rotation (offset) when the visualization is executed.

### "Use REAL values"

Note: Only available if the device supports the use of REAL coordinates.

- The properties of the absolute movement are interpreted as REAL values. The values are not rounded.

The option allows for the individual fine-tuning of drawing the element, for example for the visualization of a smoother rotation.

**Hint:** If a horizontal or vertical line is drawn blurry on a specific visualization platform, then this can be corrected by an offset of 0.5px in the direction of the line thickness.

>You can link the variables to a unit conversion.
The properties “X”, “Y”, “Rotation”, and “Interior rotation” are supported by the “Client Animation” functionality.

See also
- Unit conversion

Element property 'Dynamic points'

| “Array of points” | Variable (POINTER TO). Points to an array of the structure VisuElems.VisuStructPoint. The elements iX and iY of VisuStructPoint contain the xy-coordinates of a point. The current number of array elements implicitly contains the variable in the property “Number of points”.

The variable that is assigned to the property “Number of points” contains the number of array elements and therefore the number of corner points.

Example:
```c
pPoints : POINTER TO ARRAY[0..100] OF VisuElems.VisuStructPoint;
```

| “Number of points” | Variable (integer data type): Contains the number of array elements and therefore the number of corner points for displaying the element.

Example:
```c
PLC_PRG.iNumberOfPoints := 24;
```

In the example, the element has 24 points. This definition is necessary because the individual points are defined by a pointer and this does not allow control over the number of points.

Note: In this way, it is possible to adapt the display of the element dynamically by updating the number of corner points.

Element property 'Text variables'

These properties are variables with contents that replace a format definition.

| “Text variable” | Variable (data type compliant with the format definition). It contains what is printed instead of the format definition.

Example:
```c
PLC_PRG.iAccesses
```

Note: The format definition is part of the text in the property “Texts ➔ Text”.

Note: If you specify a variable of type enumeration with text list support, then the name of the enumeration data type is added automatically in angle brackets after the variable name. Example: `PLC_PRG.enVar <enumeration name>`. Then the symbolic value of the enumeration component is printed instead of the numeric value when text is printed. Refer to the help page for the enumerations.

| “Tooltip variable” | Variable (data type compliant with the format definition). It contains what is printed instead of the format definition.

Example:
```c
PLC_PRG.iAccessesInTooltip
```

Note: The format definition is part of the text in the property “Texts ➔ Tooltip”.

See also
- § Chapter 1.3.5.1.8.2 “Placeholders with format definition in the output text” on page 1546
- § “Element property ‘Texts’” on page 1226
- Enumerations
Element property 'Dynamic texts'

Dynamic texts are variably indexed texts of a text list. At runtime, the text is displayed that is currently indexed in the variable.

| “Text list” | Variable (string) or name of the text list as a fixed string in single straight quotation marks.  
Example: 'Errorlist'  
▼: Drop-down list with the dialogs available in the text lists. |
|———|———|
| “Text index” | Text list ID. This refers to the desired output text.  
● As fixed string with the ID in single straight quotation marks.  
Example: '1'  
● As a variable (STRING) for dynamically controlling the text output.  
Example: strTextID  
Sample assignment: PLC_PRG.strTextID := '1'; |
|———|———|
| “Tooltip index” | Text list ID. This refers to the desired output text.  
● As fixed string with the ID in single straight quotation marks.  
Example: '2'  
● As a variable (STRING) for dynamically controlling the text output.  
Example: strToolTipID  
Sample assignment: PLC_PRG.strToolTipID := '2'; |

See also

● Text list

Element property 'Font variables'

The variables allow for dynamic control of the text display.

| “Font name” | Variable (STRING). Includes the font of the text.  
Example: PLC_PRG.stFontVar := 'Arial';  
The selection of fonts corresponds to the default “Font” dialog. |
|———|———|
| “Size” | Variable (numeric data type). Contains the font size (in pixels or points). The applied unit is specified in brackets after the variable name.  
● <pt>: Points (default)  
Example: PLC_PRG.iFontHeight <pt>  
Code: iFontHeight : INT := 12;  
● <px>: Pixels  
Example: PLC_PRG.iFontHeight <px>  
Code: iFontHeight : INT := 19; |

If you click in the value field, a drop-down list opens on the right for setting the unit.

Hint: The font size is specified in points (example: Arial 12). Use points when the variable font size should match a font, for example if a font is set in the property “Text property ➔ Font.”
**“Flags”**

Variable (DWORD). Contains the flags for displaying fonts.

Flags:
- 1: Italics
- 2: Bold
- 4: Underline
- 8: Strikethrough

Note: You can combine the font displays by adding the coding of the flags. For example, a bold and underlined text: PLC_PRG.dwFontType := 6;

---

**“Character set”**

Variable (DWORD). Contains a character set number for the font.

The selection of character set numbers corresponds to the “Script” setting of the standard “Font” dialog.

---

**“Color”**

Variable (DWORD). Includes the color of the text.

Example: PLC_PRG.dwColorFont := 16#FF000000;

---

**“Flags for text alignment”**

Variable (integer data type). Contains the coding for text alignment.

Example: PLC_PRG.dwTextAlignment

Coding:
- 0: Top left
- 1: Horizontal center
- 2: Right
- 4: Vertical center
- 8: Bottom

Note: You can combine the text alignments by adding the coding of the flags. For example, a vertical and horizontal centered text: PLC_PRG.dwFontType := 5;

---

**Fixed values for displaying texts are set in “Text properties”**

---

See also

- “Element property ‘Text properties’” on page 1227

---

**Element property ‘Colorvariables’**

The Element property is used as an interface for project variables to dynamically control colors at runtime.
"Toggle color"  The property controls the toggled color at runtime.

Value assignment:
- **FALSE**: The element is displayed with the color specified in the “Color” property.
- **TRUE**: The element is displayed with the color specified in the “Alarm color” property.

Assignment options:
- Placeholder for the user input variable
  - "<toggle/tap variable>"
  - "<NOT toggle/tap variable>"

  The color change is not controlled by its own variable, but by a user input variable.

  Note: Specify a variable for the mouse events “Tap” or “Toggle” in the input configuration of the element. Only then is the placeholder set. If you configure a variable in both “Toggle” and “Tap”, then the variable specified in “Tap” is used.

  Hint: Click the symbol to insert the placeholder “<toggle/tap variable>". When you activate the “Input configuration” “Tap FALSE” property, then the “<NOT toggle/tap variable>” placeholder is displayed.

- Instance path of a project variable (BOOL)
  - Example: PLC_PRG.xColorIsToggeled

  Note: In the code, declare and implement the variable specified here. Its value assignment determines when the color changes.

"Normal state"

"Alarm state"

The properties listed below control the color depending on the state. The normal state is in effect if the variable in “Color variables”, “Toggle color” is not defined or it has the value FALSE. The alarm state is in effect if the variable in “Colorvariables”, “Toggle color” has the value TRUE.

"Frame color"  Assignment options:
- Variable (DWORD) for the frame color
  - Example: PLC_PRG.dwBorderColor
- Color literal
  - Example of green and opaque: 16#FF00FF00

"Filling color"  Assignment options:
- Variable (DWORD) for the fill color
  - Example: PLC_PRG.dwFillColor
- Color literal
  - Example of gray and opaque: 16#FF888888

The transparency part of the color value is evaluated only if the “Activate semi-transparent drawing” option of the visualization manager is selected.

Select the “Advanced” option in the toolbar of the properties view. Then all element properties are visible.

See also
- Chapter 1.3.5.8.3 “Animating a color display” on page 1152
The properties contain IEC variables for controlling the appearance of the element dynamically.

### Element property 'Appearance variables'

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Line width&quot;</td>
<td>Variable (integer data type). Contains the line weight (in pixels).</td>
</tr>
</tbody>
</table>
| "Fill attributes" | Variable (DWORD). Controls whether the fill color of the element is visible.  
|                  | ● Variable value = 0: Filled                                                                                                                |
|                  | ● Variable value > 0: Invisible; no fill color                                                                                              |
| "Line style"      | Variable (DWORD). Controls the line style. Coding:  
|                  | ● 0: Solid line                                                                                                                             |
|                  | ● 1: Dashed line                                                                                                                            |
|                  | ● 2: Dotted line                                                                                                                             |
|                  | ● 3: Line type "Dash Dot"                                                                                                                   |
|                  | ● 3: Line type "Dash Dot Dot"                                                                                                                |
|                  | ● 8: Invisible; no line                                                                                                                     |

Fixed values can be set in the "Appearance" property. These values can be overwritten by dynamic variables at runtime.

See also
- ¶ "Element property 'Appearance'" on page 1238

### Element property 'State variables'

The variables control the element behavior dynamically.

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
</table>
| "Invisible"        | Variable (BOOL). Toggles the visibility of the element.  
|                    | TRUE: The element is not visible at runtime.                                                                                               |
| "Deactivate inputs"| Variable (BOOL). Toggles the operability of the element.  
|                    | TRUE: User inputs do not have any effect in runtime more. The element is shown as deactivated.                                              |

The "Invisible" property is supported by the "Client Animation" functionality.

See also
- ¶

These properties are available only when you select the "Preview: Support client animations and overlay of native elements" option in the Visualization Manager.

You can animate a movement, a rotation, and the visibility of a visualization element.
“Animation duration”
Time that the element executes an animation (in milliseconds)
Example: 500

“Move to the foreground”
Property value (BOOL)
TRUE: At runtime, the element is displayed in the foreground.
FALSE: At runtime, the element is displayed in the layer where it was inserted in the visualization editor.

See also
- Chapter 1.3.5.21.5 “Using client animation” on page 1962

Element property 'Input configuration'
The properties contain the configurations for the user input when using the mouse or keyboard.
A user input defines an event and one or more actions that are executed when an event occurs.

The “Configure” button opens the “Input configuration” dialog box for creating or modifying a user input.
Configured user inputs are briefly listed below the events. They each include the action that is triggered and the setting in short form.
Example: “Execute ST code”:

```
PLC_PRG.i_x := 0;
```

<table>
<thead>
<tr>
<th>Event</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OnDialogClosed</td>
<td>Input event: The user closes the dialog box.</td>
</tr>
<tr>
<td>OnMouseClick</td>
<td>Input event: The user clicks the mouse button completely. The mouse button is clicked and released.</td>
</tr>
<tr>
<td>OnMouseDown</td>
<td>Input event: The user clicks down on the mouse button.</td>
</tr>
<tr>
<td>OnMouseEnter</td>
<td>Input event: The user drags the mouse pointer to the element.</td>
</tr>
<tr>
<td>OnMouseLeave</td>
<td>Input event: The user drags the mouse pointer away from the element.</td>
</tr>
<tr>
<td>OnMouseMove</td>
<td>Input event: The user moves the mouse pointer over the element area.</td>
</tr>
<tr>
<td>OnMouseUp</td>
<td>Input event: The user releases the mouse button over the element area.</td>
</tr>
</tbody>
</table>

“Tap”
When a mouse click event occurs, the variable defined in “Variable” is described in the application. The coding depends on the options “Tap FALSE” and “Tap on enter if captured”.

“Variable”
Variable (BOOL) that is set on mouse click.
Example: PLC_PRG.bIsTapped
TRUE: A mouse click event exists. It lasts while the user presses the mouse button over the element. It ends when the button is released.
FALSE: A mouse click event does not exist.
Requirement: The “Tap FALSE” option is not activated.
| **“Tap FALSE”** | ![Checkmark] The mouse click event leads to a complementary value in “Variable”.  
**TRUE**: A mouse click event does not exist.  
**FALSE**: While the mouse click event exists. |

| **“Tap on enter if captured”** | ![Checkmark] During user input, it is also taken into consideration whether the mouse pointer is dragged within the element area or not while the mouse button is pressed.  
**TRUE**: While the mouse click event exists and the mouse pointer is moved over the element area.  
**FALSE**: A mouse click event does not exist. Or the user moves the mouse pointer outside of the element area while the mouse button is pressed.  
The value is **TRUE** again as soon as the user moves the pointer back to the element area. The mouse is then captured. |

| **“Shift”** | With the onset of a mouse click event, the variable is set; when the mouse click event is completed, the variable is reset. |

| **“Variable”** | Variable (BOOL). Its value toggled when the mouse click event is ended. This is when the user releases the mouse button while the mouse pointer is over the element area.  
If the user releases the mouse button while the mouse pointer is outside of the element area, then the mouse click event is not ended and the value is not toggled.  
Tip: The user can cancel a started toggle input by dragging the mouse pointer out of the element area. |

| **“Toggle on up if captured”** | ![Checkmark] The value toggles regardless of where the mouse pointer is when the mouse button is released. The mouse is then captured. |

| **“Keyboard shortcuts”** | Keyboard shortcut on the element for triggering specific input actions.  
When the keyboard shortcut event occurs, the input actions in the “Event(s)” property are triggered. In this way, it is not the input action itself that leads to this input action, but the mouse input action. |

| **“Key”** | Key pressed for input action.  
Example: [T]  
Note: The following properties appear when a key is selected. |

| **“Event(s)”** |  
- “None”  
- “Mouse down”: Pressing the key triggers the input actions that are configured in the “OnMouseDown” property.  
- “Mouse up”: Releasing the key triggers the input actions that are configured in the “OnMouseUp” property.  
- “Mouse down/up”: Pressing and releasing the key triggers the input actions that are configured in the “OnMouseDown” property and the “OnMouseUp” property. |

| **“Shift”** | ![Checkmark] Combination with the Shift key  
Example: [Shift]+[T]. |

| **“Control”** | ![Checkmark] Combination with the Ctrl key  
Example: [Ctrl]+[T]. |

| **“Alt”** | ![Checkmark] Combination with the Alt key  
Example: [Alt]+[T]. |
All keyboard shortcuts and their actions that are configured in the visualization are listed in the “Keyboard configuration” tab.

See also
- Chapter 1.3.5.19.2.2 “Command 'Keyboard Configuration'” on page 1558
- Chapter 1.3.5.19.3.6 “Dialog 'Input Configuration'” on page 1586

Requirement: User management is set up for the visualization.

<table>
<thead>
<tr>
<th>“Access rights”</th>
<th>Opens the “Access rights” dialog. There you can edit the access privileges for the element.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Status messages:</td>
<td>- “Not set. Full rights.”: Access rights for all user groups: “operable”</td>
</tr>
<tr>
<td></td>
<td>- “Rights are set: Limited rights”: Access is restricted for at least one group.</td>
</tr>
</tbody>
</table>

See also
- Chapter 1.3.5.19.3.1 “Dialog ‘Access Rights’” on page 1583

Visualization element 'Pie'

Symbol:

Tag: “Basic”
The element draws a pie of any angle.

Element properties

<table>
<thead>
<tr>
<th>“Element name”</th>
<th>Example: Error_rate_part_1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Optional</td>
<td>Hint: Assign individual names for elements so that they are found faster in the element list.</td>
</tr>
</tbody>
</table>

| “Type of element” | Pie |

Element property 'Position'
The position defines the location and size of the element in the visualization window. This is based on the Cartesian coordinate system. The origin is located at the upper left corner of the window. The positive horizontal x-axis runs to the right. The positive vertical y-axis runs downwards.
“X”
The x-coordinate of the upper left corner of the element
Specified in pixels
Example: 10

“Y”
The y-coordinate of the upper left corner of the element
Specified in pixels
Example: 10

“Width”
Specified in pixels
Example: 150

“Height”
Specified in pixels
Example: 30

Tip: You can change the values in “X”, “Y”, “Width”, and “Height” by dragging the corresponding symbols to another position in the editor.

“Angle”
Static angle of rotation (in degrees).
Example: 35

The element is displayed rotated in the editor. The point of rotation is the center of the element. A positive value rotates clockwise.

Tip: You can change the value in the editor by focusing the element to the handle. When the cursor is displayed as a rotating arrow, you can rotate the element about its center as a handle.

(1): Handle

Note: If a dynamic angle of rotation is also configured in the property “Absolute movement ➔ Internal rotation”, then the static and dynamic angles of rotation are added in runtime mode. The static angle of rotation acts as an offset.

See also

- ☰ Chapter 1.3.5.3.2 “Positioning elements, adapting size and layer” on page 1113
**“Begin”**
Start angle of the pie. If you also define a variable for the start, then the start angle is calculated from the sum of the values for “Begin” and “Variable for begin”.

**Example:**
- “Begin”: 330
- “End”: 90

**“End”**
End angle of the pie. If you also define a variable for the end, then the end angle is calculated from the sum of the values for “End” and “Variable for end”.

The pie is drawn clockwise from the start angle to the end angle.

**“Variable for begin”**
The start of the sector is defined dynamically by a variable.

**“Variable for end”**
The end of the sector is defined dynamically by a variable.

**“Only show circle line”**
☑: The pie is drawn without the radius line or filling color.

---

**Element property ‘Center’**

<table>
<thead>
<tr>
<th><strong>“X”</strong></th>
<th>Display of the center coordinates. You cannot modify these values here in the properties.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>“Y”</strong></td>
<td>If the pie is selected in the editor, then the center of the pie (as well as the center of the enveloping box) is visualized with the symbol  dàng. Moreover, the element is decorated with a position, begin, and end boxes that you can move.</td>
</tr>
</tbody>
</table>

The center coordinates change when you move the center symbol  dàng in the editor. This also changes the size of the pie so that the position box retains its position and the center remains in the middle of the element.

---

**Element property ‘Colors’**

| **“Normal state”** | The normal state is in effect if the variable in “Color variables ➔ Toggle color” is not defined or it has the value FALSE. |
| **“Frame color”** | Frame and fill color for the corresponding state of the variable. |
| **“Fill color”** | Transparency value (0 to 255) for defining the transparency of the selected color. Example: 255: The color is opaque. 0: The color is completely transparent. |
The alarm state is in effect if the variable in “Color variables ➔ Toggle color” has the value TRUE.

“Use gradient color”  
☑: The element is displayed with a gradient of two colors.

“Gradient setting”  
The “Gradient editor” dialog box opens.

See also
●  Chapter 1.3.5.19.3.5 “Dialog ‘Gradient Editor’” on page 1586

Element property ‘Appearance’  
The properties contain fixed values for setting the look of the element.

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Line width”</td>
<td>Value in pixels&lt;br&gt;Note: The values 0 and 1 both result in a line weight of 1 pixel. If no line should be displayed, then the “Line style” property must be set to the option “Invisible”.</td>
</tr>
<tr>
<td>“Fill attributes”</td>
<td>The way in which the element is filled.&lt;br&gt;• “Filled”: The element is filled with the color from property “Colors ➔ Fill color”.&lt;br&gt;• “Invisible”: The fill color is invisible.</td>
</tr>
<tr>
<td>“Line style”</td>
<td>Type of line representation&lt;br&gt;• “Solid”&lt;br&gt;• “Dashes”&lt;br&gt;• “Dots”&lt;br&gt;• “Dash Dot”&lt;br&gt;• “Dash Dot Dot”&lt;br&gt;• “not visible”</td>
</tr>
</tbody>
</table>

You can assign variables in the “Appearance variables” property for controlling the appearance dynamically. The fixed values here are overwritten.

See also
●  “Element property ‘Appearance variables’” on page 1285

Element property ‘Texts’  
The properties contains character strings for labeling the element. The character string can also contain a placeholder with a format definition. In runtime mode, the placeholder is replaced by the current value in the specified format.

CODESYS accepts the specified texts automatically into the “GlobalTextList” text list. Therefore, these texts can be localized.
**“Text”**
Character string (without single straight quotation marks) for the labeling the element. Add a line break by pressing the keyboard shortcut [Ctrl] + [Enter].

**Example:** Accesses: %i
The variable that contains the current value for the placeholder is specified in the property “Text variable ➔ Text”.

**“Tooltip”**
Character string (without single straight quotation marks) that is displayed as the tooltip of an element.

**Example:** Number of valid accesses.
The variable that contains the current value for the placeholder is specified in the property “Text variable ➔ Tooltip”.

See also
- § “Element property ’Text variables’” on page 1229
- § Chapter 1.3.5.18.2 “Placeholders with format definition in the output text” on page 1546

---

**Element property ’Text properties’**
The properties contain fixed values for the text properties.

<table>
<thead>
<tr>
<th>“Horizontal alignment”</th>
<th>Horizontal alignment of the text within the element.</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Vertical alignment”</td>
<td>Vertical alignment of the text within the element.</td>
</tr>
<tr>
<td><strong>“Text format”</strong></td>
<td>Definition for displaying texts that are too long</td>
</tr>
<tr>
<td></td>
<td>● “Default”: The long text is truncated.</td>
</tr>
<tr>
<td></td>
<td>● “Line break”: The text is split into parts.</td>
</tr>
<tr>
<td></td>
<td>● “Ellipsis”: The visible text ends with “...” indicating that it is not complete.</td>
</tr>
</tbody>
</table>

| **“Font”**            | Example: “Default”                                   |
|                       | ![The “Font” dialog box opens.]                      |
|                       | ![▼: Drop-down list with style fonts.]              |

| **“Font color”**      | Example: “Black”                                     |
|                       | ![The “Color” dialog box opens.]                    |
|                       | ![▼: Drop-down list with style colors.]             |

| **“Transparency”**    | Whole number (value range from 0 to 255). This determines the transparency of the respective color. |
|                       | Example: 255: The color is opaque.                   |
|                       | 0: The color is completely transparent.              |
|                       | Please note: If the color is a style color and already has a transparency value, then this property is write-protected. |

---

**Element property ’Absolute movement’**
The properties contain IEC variables for controlling the position of the element dynamically. The reference point is the upper left corner of the element. In runtime mode, the entire element is moved.

| **“Movement”**        | Variable (integer data type). Defines the X position (in pixels). |
|                      | Example: PLC_PRG.iPos_X.                                   |
|                      | Increasing this value in runtime mode moves the element to the right. |
### “Y”

Variable (integer data type). Defines the Y position (in pixels).

**Example:** PLC_PRG.iPos_Y.

Increasing this value in runtime mode moves the element downwards.

### “Scaling”

Variable (integer data type). Causes centric stretching.

**Example:** PLC_PRG.iScaling.

The reference point is the “Center” property.

The value 1 shrinks the element by a factor of 0.001. The value 1000 returns the element to its original size.

### “Interior rotation”

Variable (integer data type). Defines the angle of rotation (in degrees).

**Example:** PLC_PRG.iAngle2.

In runtime mode, the element rotates about the point of rotation specified in “Center” according to the value of the variable. In addition, the alignment of the element rotates according to the coordinate system of the visualization. Increasing the value in the code rotates clockwise.

The rotation point is shown as the symbol.

If a static angle of rotation is specified in “Position ➔ Angle”, then the static angle of rotation and the angle of rotation are added.

---

**You can link the variables to a unit conversion.**

**The “X”, “Y”, and “Interior rotation” properties are supported by the "Client Animation" functionality.**

See also

- **Unit conversion**
- **"Client Animation" functionality.**

---

**Element property 'Text variables'**

These properties are variables with contents that replace a format definition.
**“Text variable”**

Variable (data type compliant with the format definition). It contains what is printed instead of the format definition.

**Example:** PLC_PRG.iAccesses

Note: The format definition is part of the text in the property “Texts ➔ Text”.

Note: If you specify a variable of type enumeration with text list support, then the name of the enumeration data type is added automatically in angle brackets after the variable name. Example: PLC_PRG.enVar <enumeration name>. Then the symbolic value of the enumeration component is printed instead of the numeric value when text is printed. Refer to the help page for the enumerations.

**“Tooltip variable”**

Variable (data type compliant with the format definition). It contains what is printed instead of the format definition.

**Example:** PLC_PRG.iAccessesInTooltip

Note: The format definition is part of the text in the property “Texts ➔ Tooltip”.

See also

- § Chapter 1.3.5.18.2 “Placeholders with format definition in the output text” on page 1546
- § “Element property ‘Texts’” on page 1226
- Enumerations

**Element property 'Dynamic texts'**

Dynamic texts are variably indexed texts of a text list. At runtime, the text is displayed that is currently indexed in the variable.

**“Text list”**

Variable (string) or name of the text list as a fixed string in single straight quotation marks.

**Example:** 'Errorlist'

확: Drop-down list with the dialogs available in the text lists.

**“Text index”**

Text list ID. This refers to the desired output text.

- As fixed string with the ID in single straight quotation marks.
  
  **Example:** '1'

- As a variable (STRING) for dynamically controlling the text output.
  
  **Example:** strTextID
  
  Sample assignment: PLC_PRG.strTextID := '1';

**“Tooltip index”**

Text list ID. This refers to the desired output text.

- As fixed string with the ID in single straight quotation marks.
  
  **Example:** '2'

- As a variable (STRING) for dynamically controlling the text output.
  
  **Example:** strToolTipID
  
  Sample assignment: PLC_PRG.strToolTipID := '2';

See also

- Text list

**Element property 'Font variables'**

The variables allow for dynamic control of the text display.
### “Font name”

Variable (STRING). Includes the font of the text.

Example: `PLC_PRG.stFontVar := 'Arial';`

The selection of fonts corresponds to the default “Font” dialog.

### “Size”

Variable (numeric data type). Contains the font size (in pixels or points). The applied unit is specified in brackets after the variable name.

- `<pt>`: Points (default)
  
  Example: `PLC_PRG.iFontHeight <pt>`
  
  Code: `iFontHeight : INT := 12;`

- `<px>`: Pixels
  
  Example: `PLC_PRG.iFontHeight <px>`
  
  Code: `iFontHeight : INT := 19;`

If you click in the value field, a drop-down list opens on the right for setting the unit.

Hint: The font size is specified in points (example: Arial 12). Use points when the variable font size should match a font, for example if a font is set in the property “Text property → Font”.

### “Flags”

Variable (DWORD). Contains the flags for displaying fonts.

Flags:

- 1: Italics
- 2: Bold
- 4: Underline
- 8: Strikethrough

Note: You can combine the font displays by adding the coding of the flags. For example, a bold and underlined text: `PLC_PRG.dwFontType := 6;`

### “Character set”

Variable (DWORD). Contains a character set number for the font.

The selection of character set numbers corresponds to the “Script” setting of the standard “Font” dialog.

### “Color”

Variable (DWORD). Includes the color of the text.

Example: `PLC_PRG.dwColorFont := 16#FF000000;`

### “Flags for text alignment”

Variable (integer data type). Contains the coding for text alignment.

Example: `PLC_PRG.dwTextAlignment`

Coding:

- 0: Top left
- 1: Horizontal center
- 2: Right
- 4: Vertical center
- 8: Bottom

Note: You can combine the text alignments by adding the coding of the flags. For example, a vertical and horizontal centered text: `PLC_PRG.dwFontType := 5;`
Fixed values for displaying texts are set in “Text properties”.

See also
- “Element property ‘Text properties’” on page 1227

Element property ‘Colorvariables’
The Element property is used as an interface for project variables to dynamically control colors at runtime.

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Toggle color”</td>
<td>The property controls the toggled color at runtime.</td>
</tr>
<tr>
<td>Value assignment:</td>
<td></td>
</tr>
<tr>
<td>- FALSE: The element is displayed with the color specified in the “Color” property.</td>
<td></td>
</tr>
<tr>
<td>- TRUE: The element is displayed with the color specified in the “Alarm color” property.</td>
<td></td>
</tr>
<tr>
<td>Assignment options:</td>
<td></td>
</tr>
<tr>
<td>- Placeholder for the user input variable</td>
<td></td>
</tr>
<tr>
<td>- “&lt;toggle/tap variable&gt;”</td>
<td></td>
</tr>
<tr>
<td>- “&lt;NOT toggle/tap variable&gt;”</td>
<td></td>
</tr>
<tr>
<td>The color change is not controlled by its own variable, but by a user input variable.</td>
<td></td>
</tr>
<tr>
<td>Note: Specify a variable for the mouse events “Tap” or “Toggle” in the input configuration of the element. Only then is the placeholder set. If you configure a variable in both “Toggle” and “Tap”, then the variable specified in “Tap” is used.</td>
<td></td>
</tr>
<tr>
<td>Hint: Click the symbol 🔄 to insert the placeholder “&lt;toggle/tap variable&gt;”. When you activate the “Inputconfiguration”, “Tap FALSE” property, then the “&lt;NOT toggle/tap variable&gt;” placeholder is displayed.</td>
<td></td>
</tr>
<tr>
<td>- Instance path of a project variable (BOOL)</td>
<td></td>
</tr>
<tr>
<td>Example: PLC_PRG.xColorIsToggled</td>
<td></td>
</tr>
<tr>
<td>Note: In the code, declare and implement the variable specified here. Its value assignment determines when the color changes.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Normal state”</td>
<td></td>
</tr>
<tr>
<td>“Alarm state”</td>
<td></td>
</tr>
<tr>
<td>The properties listed below control the color depending on the state. The normal state is in effect if the variable in “Color variables”, “Toggle color” is not defined or it has the value FALSE. The alarm state is in effect if the variable in “Colorvariables”, “Toggle color” has the value TRUE.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Frame color”</td>
<td>Assignment options:</td>
</tr>
<tr>
<td></td>
<td>- Variable (DWORD) for the frame color</td>
</tr>
<tr>
<td></td>
<td>Example: PLC_PRG.dwBorderColor</td>
</tr>
<tr>
<td></td>
<td>- Color literal</td>
</tr>
<tr>
<td></td>
<td>Example of green and opaque: 16#FF00FF00</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Filling color”</td>
<td>Assignment options:</td>
</tr>
<tr>
<td></td>
<td>- Variable (DWORD) for the fill color</td>
</tr>
<tr>
<td></td>
<td>Example: PLC_PRG.dwFillColor</td>
</tr>
<tr>
<td></td>
<td>- Color literal</td>
</tr>
<tr>
<td></td>
<td>Example of gray and opaque: 16#FF888888</td>
</tr>
</tbody>
</table>
The transparency part of the color value is evaluated only if the "Activate semi-transparent drawing" option of the visualization manager is selected.

Select the "Advanced" option in the toolbar of the properties view. Then all element properties are visible.

See also
- Chapter 1.3.5.8.3 “Animating a color display” on page 1152

The properties contain IEC variables for controlling the appearance of the element dynamically.

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Line width&quot;</td>
<td>Variable (integer data type). Contains the line weight (in pixels).</td>
</tr>
</tbody>
</table>
| "Fill attributes"| Variable (DWORD). Controls whether the fill color of the element is visible.  
  ● Variable value = 0: Filled  
  ● Variable value > 0: Invisible; no fill color |
| "Line style"     | Variable (DWORD). Controls the line style.  
  Coding:  
  ● 0: Solid line  
  ● 1: Dashed line  
  ● 2: Dotted line  
  ● 3: Line type "Dash Dot"  
  ● 3: Line type "Dash Dot Dot"  
  ● 8: Invisible; no line |

Fixed values can be set in the "Appearance" property. These values can be overwritten by dynamic variables at runtime.

See also
- “Element property ‘Appearance’” on page 1238

The variables control the element behavior dynamically.

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
</table>
| "Invisible"       | Variable (BOOL). Toggles the visibility of the element.  
  TRUE: The element is not visible at runtime. |
| "Deactivate inputs" | Variable (BOOL). Toggles the operability of the element.  
  TRUE: User inputs do not have any effect in runtime more. The element is shown as deactivated. |
The “Invisible” property is supported by the “Client Animation” functionality.

See also

These properties are available only when you select the “Preview: Support client animations and overlay of native elements” option in the Visualization Manager.

You can animate a movement, a rotation, and the visibility of a visualization element.

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Animation duration&quot;</td>
<td>Time that the element executes an animation (in milliseconds)</td>
</tr>
<tr>
<td></td>
<td>Example: 500</td>
</tr>
<tr>
<td>&quot;Move to the foreground&quot;</td>
<td>Property value (BOOL)</td>
</tr>
<tr>
<td></td>
<td>TRUE: At runtime, the element is displayed in the foreground.</td>
</tr>
<tr>
<td></td>
<td>FALSE: At runtime, the element is displayed in the layer where it was inserted in the visualization editor.</td>
</tr>
</tbody>
</table>

See also

- Chapter 1.3.5.21.5 “Using client animation” on page 1962

**Element property 'Input configuration'**

The properties contain the configurations for the user input when using the mouse or keyboard. A user input defines an event and one or more actions that are executed when an event occurs.

The "Configure" button opens the “Input configuration” dialog box for creating or modifying a user input.

Configured user inputs are briefly listed below the events. They each include the action that is triggered and the setting in short form.

**Example:**

```
Execute ST code": PLC_PRG.i_x := 0;
```

<table>
<thead>
<tr>
<th>Event</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;OnDialogClosed&quot;</td>
<td>Input event: The user closes the dialog box.</td>
</tr>
<tr>
<td>&quot;OnMouseClick&quot;</td>
<td>Input event: The user clicks the mouse button completely. The mouse button is clicked and released.</td>
</tr>
<tr>
<td>&quot;OnMouseDown&quot;</td>
<td>Input event: The user clicks down on the mouse button.</td>
</tr>
<tr>
<td>&quot;OnMouseEnter&quot;</td>
<td>Input event: The user drags the mouse pointer to the element.</td>
</tr>
<tr>
<td>&quot;OnMouseLeave&quot;</td>
<td>Input event: The user drags the mouse pointer away from the element.</td>
</tr>
<tr>
<td>&quot;OnMouseMove&quot;</td>
<td>Input event: The user moves the mouse pointer over the element area.</td>
</tr>
<tr>
<td>&quot;OnMouseUp&quot;</td>
<td>Input event: The user releases the mouse button over the element area.</td>
</tr>
</tbody>
</table>
### “Tap”
When a mouse click event occurs, the variable defined in “Variable” is described in the application. The coding depends on the options “Tap FALSE” and “Tap on enter if captured”.

### “Variable”
Variable (BOOL) that is set on mouse click.

- **TRUE**: A mouse click event exists. It lasts while the user presses the mouse button over the element. It ends when the button is released.
- **FALSE**: A mouse click event does not exist.

#### Example:
```
PLC_PRG.bIsTapped
```

**Requirement**: The “Tap FALSE” option is not activated.

### “Tap FALSE”
- **TRUE**: The mouse click event leads to a complementary value in “Variable”.
- **FALSE**: A mouse click event does not exist.

### “Tap on enter if captured”
- **TRUE**: During user input, it is also taken into consideration whether the mouse pointer is dragged within the element area or not while the mouse button is pressed.
- **FALSE**: While the mouse click event exists and the mouse pointer is moved over the element area.
- **FALSE**: A mouse click event does not exist. Or the user moves the mouse pointer outside of the element area while the mouse button is pressed.

The value is **TRUE** again as soon as the user moves the pointer back to the element area. The mouse is then captured.

### “Shift”
With the onset of a mouse click event, the variable is set; when the mouse click event is completed, the variable is reset.

### “Variable”
Variable (BOOL). Its value toggled when the mouse click event is ended. This is when the user releases the mouse button while the mouse pointer is over the element area.

- If the user releases the mouse button while the mouse pointer is outside of the element area, then the mouse click event is not ended and the value is not toggled.
- **Tip**: The user can cancel a started toggle input by dragging the mouse pointer out of the element area.

### “Toggle on up if captured”
- **TRUE**: The value toggles regardless of where the mouse pointer is when the mouse button is released. The mouse is then captured.

### “Keyboard shortcuts”
Keyboard shortcut on the element for triggering specific input actions.

When the keyboard shortcut event occurs, the input actions in the “Event(s)” property are triggered. In this way, it is not the input action itself that leads to this input action, but the mouse input action.

### “Key”
Key pressed for input action.

#### Example: 
```
[T]
```

**Note**: The following properties appear when a key is selected.
**“Event(s)”**

- **“None”**
- **“Mouse down”**: Pressing the key triggers the input actions that are configured in the “OnMouseDown” property.
- **“Mouse up”**: Releasing the key triggers the input actions that are configured in the “OnMouseUp” property.
- **“Mouse down/up”**: Pressing and releasing the key triggers the input actions that are configured in the “OnMouseDown” property and the “OnMouseUp” property.

**“Shift”**

☑️ Combination with the Shift key
Example: [Shift]+[T].

**“Control”**

☑️ Combination with the Ctrl key
Example: [Ctrl]+[T].

**“Alt”**

☑️ Combination with the Alt key
Example: [Alt]+[T].

---

All keyboard shortcuts and their actions that are configured in the visualization are listed in the “Keyboard configuration” tab.

See also
- ☮ Chapter 1.3.5.19.2.2 “Command ‘Keyboard Configuration’” on page 1558
- ☮ Chapter 1.3.5.19.3.6 “Dialog ‘Input Configuration’” on page 1586

---

**Element property 'Access rights'**

Requirement: User management is set up for the visualization.

**“Access rights”**

Opens the “Access rights” dialog. There you can edit the access privileges for the element.
Status messages:
- **“Not set. Full rights.”**: Access rights for all user groups: “operable”
- **“Rights are set: Limited rights”**: Access is restricted for at least one group.

See also
- ☮ Chapter 1.3.5.19.3.1 “Dialog ‘Access Rights’” on page 1583

---

**Visualization element 'Image'**

Symbol:

Tag: “Basic”

The element adds an image to the visualization. The displayed image is managed in the image pool and referenced in the visualization element by means of a static ID. In addition, you can change the displayed image dynamically by using a variable instead of the static ID.
With the “Background” command, you can define a background for the entire visualization.

Directories that contain the images for use in visualizations can be defined in the project settings (category “Visualization”).

# Element properties

<table>
<thead>
<tr>
<th>“Element name”</th>
<th>Example: Status bar</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Optional</td>
</tr>
<tr>
<td></td>
<td>Hint: Assign individual names for elements so that they are found faster in the element list.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>“Type of element”</th>
<th>“Image”</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>“Static ID”</th>
<th>Identifier for static assignment</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID of the image file on, as it is defined in the corresponding image pool. If the image is not included in the global image pool in the POU view, then the instance path must be specified. Then the name of the image pool is preceded to make the entry unique. Example: <code>imagepool2.button_image</code>.</td>
<td></td>
</tr>
</tbody>
</table>

When entering a new ID, a file selection opens. The selected file is saved to the “GlobalImagePool”.

See also: Help for the “Image pool” object.

<table>
<thead>
<tr>
<th>“Show frame”</th>
<th>☑ The image file is displayed with a frame.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>“Clipping”</th>
<th>Requirement: “Scaling type” is “Fixed”.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>☑ Only part of the visualization is displayed that fits in the element frame.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>“Transparent”</th>
<th>☑ The image pixels that have the “Transparent color” are displayed as transparent.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>“Transparent color”</th>
<th>Effective only if the “Transparent” option is activated.</th>
</tr>
</thead>
<tbody>
<tr>
<td>The button opens the color selection dialog. This is where you select the transparent color.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>“Scaling type”</th>
<th>Definition how an image fits in the element frame.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>“Isotropic”: The entire image is shown in the element frame, either larger or smaller. As a result, the proportion of height and width are retained. If the alignment of the elements to each other should also be retained within a scaled frame element, then note the following. Unwanted horizontal or vertical offsets can be prevented by setting the properties “Horizontal alignment” and “Vertical alignment” to “Centered”. The alignment of the elements is retained and there are no resulting horizontal or vertical offsets. Example: A lamp is centered above a switch. The lamp should remain in the horizontally centered position, even if the frame is resized.</td>
</tr>
<tr>
<td></td>
<td>“Anisotropic”: The image resizes automatically to the dimensions of the element frame, filling the entire element frame. As a result, the proportions are not retained.</td>
</tr>
<tr>
<td></td>
<td>“Fixed”: The image retains its original size, even if the element frame is resized. Please note also that the “Clipping” option is selected. For each reassignment of an image ID, the element size is adapted automatically to the image size.</td>
</tr>
</tbody>
</table>
**“Horizontal alignment”**

Horizontal alignment of the element within the element frame:
- “Left”
- “Centered”
- “Right”

Requirement: The scaling type of the image is “Isotropic” or “Fixed”.

Note: If the visualization is referenced, then the horizontal alignment takes effect within the frame position.

- The property “Variable” is shown below.

<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Enumeration:</td>
</tr>
<tr>
<td></td>
<td>TYPE VisuElemBase.VisuEnumHorizontalAlignment</td>
</tr>
<tr>
<td></td>
<td>LEFT                                                                  HCENTER                                                                  RIGHT</td>
</tr>
<tr>
<td></td>
<td>END_TYPE</td>
</tr>
<tr>
<td>Example:</td>
<td>PLC_PRG.eHorizontalAlignment</td>
</tr>
</tbody>
</table>

Declaration:

```plaintext
PROGRAM PLC_PRG
VAR
  eHorizontalAlignment :
  VisuElemBase.VisuEnumHorizontalAlignment :=
  VisuElemBase.VisuEnumHorizontalAlignment.HCENTER;
END_VAR
```
**Vertical alignment**

Vertical alignment of the element within the element frame:

- **“Top”**
- **“Centered”**
- **“Bottom”**

Requirement: The scaling type of the image is “Isotropic” or “Fixed”.

Note: If the visualization is referenced, then the horizontal alignment takes effect within the frame position.

**: The “Variable” property is shown below.

**Variable**

Enumeration variable (ENUM VisuElemBase.VisuEnumVerticalAlignment). Contains the vertical alignment.

Enumeration:

```
TYPE VisuElemBase.VisuEnumVerticalAlignment
  DOWN
  VCENTER
  BOTTOM
END_TYPE
```

**Example:** PLC_PRG.eHorizontalAlignment

**Declaration:**

```
PROGRAM PLC_PRG
VAR
  eVerticalAlignment :
    VisuElemBase.VisuEnumVerticalAlignment :=
    VisuElemBase.VisuEnumVerticalAlignment.VCENTER;
END_VAR
```

See also

- **Object ‘Image Pool’**

**Element property ‘Position’**

The position defines the location and size of the element in the visualization window. This is based on the Cartesian coordinate system. The origin is located at the upper left corner of the window. The positive horizontal x-axis runs to the right. The positive vertical y-axis runs downwards.

<table>
<thead>
<tr>
<th><strong>X</strong></th>
<th>The x-coordinate of the upper left corner of the element</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Specified in pixels</td>
</tr>
<tr>
<td></td>
<td>Example: 10</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Y</strong></th>
<th>The y-coordinate of the upper left corner of the element</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Specified in pixels</td>
</tr>
<tr>
<td></td>
<td>Example: 10</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Width</strong></th>
<th>Specified in pixels</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Example: 150</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Height</strong></th>
<th>Specified in pixels</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Example: 30</td>
</tr>
</tbody>
</table>

Tip: You can change the values in “X”, “Y”, “Width”, and “Height” by dragging the corresponding symbols to another position in the editor.
"Angle"

Static angle of rotation (in degrees).

Example: 35

The element is displayed rotated in the editor. The point of rotation is the center of the element. A positive value rotates clockwise.

Tip: You can change the value in the editor by focusing the element to the handle. When the cursor is displayed as a rotating arrow, you can rotate the element about its center as a handle.

(1): Handle

Note: If a dynamic angle of rotation is also configured in the property "Absolute movement ➔ Internal rotation", then the static and dynamic angles of rotation are added in runtime mode. The static angle of rotation acts as an offset.

See also

- § Chapter 1.3.5.3.2 “Positioning elements, adapting size and layer” on page 1113

Element property 'Center'

The properties contain fixed values for the coordinates of the point of rotation. This point of rotation is shown as the symbol. The point is used as the center for rotating and scaling.

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>“X”</td>
<td>X-coordinate of the point of rotation</td>
</tr>
<tr>
<td>“Y”</td>
<td>Y-coordinate of the point of rotation</td>
</tr>
</tbody>
</table>

You can also change the values by dragging the symbols (⊕) to other positions in the editor.

Element property 'Colors'

The properties contain fixed values for setting colors.

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Color”</td>
<td>Color for the frame</td>
</tr>
<tr>
<td>Requirement: “Show frame” property is activated.</td>
<td></td>
</tr>
<tr>
<td>Please note that the normal state is in effect if the expression in the “Color variables ➔ Toggle color” property is not defined or it has the value FALSE.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Alarm color”</td>
<td>Color for the frame in alarm state</td>
</tr>
<tr>
<td>Requirement: “Show frame” property is activated.</td>
<td></td>
</tr>
<tr>
<td>Please note that the alarm state is in effect if the expression in the “Color variables ➔ Toggle color” property has the value TRUE.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Transparency”</td>
<td>Value (0 to 255) for defining the transparency of the selected color.</td>
</tr>
<tr>
<td>Example 255: The color is opaque. 0: The color is completely transparent.</td>
<td></td>
</tr>
</tbody>
</table>
See also

- The properties contain fixed values for setting the look of the element.

### Element property 'Appearance'

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Line width&quot;</td>
<td>Value in pixels. Example: 2. Note: The values 0 and 1 both result in a line weight of one pixel. If no line should be displayed, then the &quot;Line style&quot; property must be set to the option &quot;Invisible&quot;.</td>
</tr>
<tr>
<td>&quot;Line style&quot;</td>
<td>Type of line representation. Type of line representation.</td>
</tr>
<tr>
<td></td>
<td>● &quot;Solid&quot;</td>
</tr>
<tr>
<td></td>
<td>● &quot;Dashes&quot;</td>
</tr>
<tr>
<td></td>
<td>● &quot;Dots&quot;</td>
</tr>
<tr>
<td></td>
<td>● &quot;Dash Dot&quot;</td>
</tr>
<tr>
<td></td>
<td>● &quot;Dash Dot Dot&quot;</td>
</tr>
<tr>
<td></td>
<td>● &quot;not visible&quot;</td>
</tr>
</tbody>
</table>

You can assign variables in the "Appearance variables" property for controlling the appearance dynamically. The fixed values are defined here.

See also

- “Element property ‘Appearance variables’” on page 1285

### Element property 'Texts'

The properties contain character strings for labeling the element. The character string can also contain a placeholder with a format definition. In runtime mode, the placeholder is replaced by the current value in the specified format.

CODESYS accepts the specified texts automatically into the "GlobalTextList" text list. Therefore, these texts can be localized.

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Text&quot;</td>
<td>Character string (without single straight quotation marks) for the labeling of the element. Add a line break by pressing the keyboard shortcut [Ctrl] + [Enter]. Example: Accesses: %i The variable that contains the current value for the placeholder is specified in the property &quot;Text variable ➔ Text&quot;.</td>
</tr>
<tr>
<td>&quot;Tooltip&quot;</td>
<td>Character string (without single straight quotation marks) that is displayed as the tooltip of an element. Example: Number of valid accesses. The variable that contains the current value for the placeholder is specified in the property &quot;Text variable ➔ Tooltip&quot;.</td>
</tr>
</tbody>
</table>

See also

- “Element property ‘Text variables’” on page 1229
- Chapter 1.3.5.18.2 “Placeholders with format definition in the output text” on page 1546
The properties contain fixed values for the text properties.

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Horizontal alignment</td>
<td>Horizontal alignment of the text within the element.</td>
</tr>
<tr>
<td>Vertical alignment</td>
<td>Vertical alignment of the text within the element.</td>
</tr>
<tr>
<td>Text format</td>
<td>Definition for displaying texts that are too long</td>
</tr>
<tr>
<td></td>
<td>- &quot;Default&quot;: The long text is truncated.</td>
</tr>
<tr>
<td></td>
<td>- &quot;Line break&quot;: The text is split into parts.</td>
</tr>
<tr>
<td></td>
<td>- &quot;Ellipsis&quot;: The visible text ends with &quot;...&quot; indicating that it is not complete.</td>
</tr>
<tr>
<td>Font</td>
<td>Example: &quot;Default&quot;</td>
</tr>
<tr>
<td></td>
<td>📦: The &quot;Font&quot; dialog box opens.</td>
</tr>
<tr>
<td></td>
<td>◀️: Drop-down list with style fonts.</td>
</tr>
<tr>
<td>Font color</td>
<td>Example: &quot;Black&quot;</td>
</tr>
<tr>
<td></td>
<td>📦: The &quot;Color&quot; dialog box opens.</td>
</tr>
<tr>
<td></td>
<td>◀️: Drop-down list with style colors.</td>
</tr>
<tr>
<td>Transparency</td>
<td>Whole number (value range from 0 to 255). This determines the transparency of the respective color.</td>
</tr>
<tr>
<td></td>
<td>Example: 255: The color is opaque.</td>
</tr>
<tr>
<td></td>
<td>0: The color is completely transparent.</td>
</tr>
<tr>
<td></td>
<td>Please note: If the color is a style color and already has a transparency value, then this property is write-protected.</td>
</tr>
</tbody>
</table>

**Element property 'Image ID variable'**

| Image ID               | Variable (STRING). Contains the image ID. The contents of the string corresponds to the description of the "Static ID" property. Example: PLC_PRG.stImageID := 'ImagePool_A.Image3'; |

See also

- ☞ Chapter 1.3.5.18.1.5 "Visualization element 'Image'" on page 1272
- Object 'Image Pool'

You can use this element property for animating a series of image files.

**Element property 'Dynamic image'**

| Bitmap version         | Variable (integer data type). Contains the version of the image. If the variable changes, then the visualization re-reads the image referenced in the "Image ID" property and displays it. The visualization displays animations when the image file on the controller is updated continuously, thus incrementing the version variable. The application must be programmed for this. Possible applications |
|                       | - Displaying graphics that are generated by the application                                      |
|                       | - Displaying images that are refreshed by a camera                                               |
The properties contain IEC variables for controlling the position of the element dynamically. The reference point is the upper left corner of the element. In runtime mode, the entire element is moved.

**“Movement”**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>“X”</td>
<td>Variable (numeric data type). Defines the X position (in pixels). Increasing this value in runtime mode moves the element to the right.</td>
<td>PLC_PRG.iPos_X</td>
</tr>
<tr>
<td>“Y”</td>
<td>Variable (numeric data type). Defines the Y position (in pixels). Increasing this value in runtime mode moves the element downwards.</td>
<td>PLC_PRG.iPos_Y</td>
</tr>
</tbody>
</table>

**“Rotation”**

Variable (numeric data type). Defines the angle of rotation (in degrees).

Example: PLC_PRG.iAngle1.

The midpoint of the element rotates at the “Center” point. This rotation point is shown as the symbol.

In runtime mode, the alignment of the element remains the same with respect to the coordinate system of the visualization. Increasing the value rotates the element to the right.

**“Scaling”**

Variable (integer data type). Causes centric stretching.

Example: PLC_PRG.iScaling.

The reference point is the “Center” property.

The value 1 shrinks the element by a factor of 0.001. The value 1000 returns the element to its original size.

**“Interior rotation”**

Variable (numeric data type). Defines the angle of rotation (in degrees).

Example: PLC_PRG.iAngle2.

In runtime mode, the element rotates about the point of rotation specified in “Center” according to the value of the variable. In addition, the alignment of the element rotates according to the coordinate system of the visualization. Increasing the value in the code rotates clockwise.

The rotation point is shown as the symbol.

Note: If a static angle of rotation is specified in the property “Position ➔ Angle”, then the static angle of rotation is added to the variable angle of rotation (offset) when the visualization is executed.

---

You can link the variables to a unit conversion.
The properties “X”, “Y”, “Rotation”, and “Interior rotation” are supported by the “Client Animation” functionality.

See also
- Unit conversion

Element property 'Relative movement'
The properties contains variables for moving the element. The reference point is the position of the element ("Position" property). The shape of the element can change.

<table>
<thead>
<tr>
<th>“Movement top-left”</th>
</tr>
</thead>
<tbody>
<tr>
<td>“X” Variable (integer data type). It contains the number (in pixels) that the left edge is moved horizontally. Incrementing the value moves the element to the right. Example: PLC_PRG.iDeltaX</td>
</tr>
<tr>
<td>“Y” Variable (integer data type). It contains the number (in pixels) that the top edge is moved vertically. Incrementing the value moves the element to the down. Example: PLC_PRG.iDeltaY</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>“Movement bottom-right”</th>
</tr>
</thead>
<tbody>
<tr>
<td>“X” Variable (integer data type). It contains the number (in pixels) that the right edge is moved horizontally. Incrementing the value moves the element to the right. Example: PLC_PRG.iDeltaWidth</td>
</tr>
<tr>
<td>“Y” Variable (integer data type). It contains the number (in pixels) that the bottom edge is moved vertically. Incrementing the value moves the element to the down. Example: PLC_PRG.iDeltaHeight</td>
</tr>
</tbody>
</table>

See also
- “Element property ‘Absolute movement’” on page 1227

Element property 'Text variables'
These properties are variables with contents that replace a format definition.

<table>
<thead>
<tr>
<th>“Text variable”</th>
</tr>
</thead>
<tbody>
<tr>
<td>Variable (data type compliant with the format definition). It contains what is printed instead of the format definition. Example: PLC_PRG.iAccesses</td>
</tr>
<tr>
<td>Note: The format definition is part of the text in the property “Texts ➔ Text”. Note: If you specify a variable of type enumeration with text list support, then the name of the enumeration data type is added automatically in angle brackets after the variable name. Example: PLC_PRG.enVar &lt;enumeration name&gt;. Then the symbolic value of the enumeration component is printed instead of the numeric value when text is printed. Refer to the help page for the enumerations.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>“Tooltip variable”</th>
</tr>
</thead>
<tbody>
<tr>
<td>Variable (data type compliant with the format definition). It contains what is printed instead of the format definition. Example: PLC_PRG.iAccessesInTooltip</td>
</tr>
<tr>
<td>Note: The format definition is part of the text in the property “Texts ➔ Tooltip”.</td>
</tr>
</tbody>
</table>
Dynamic texts are variably indexed texts of a text list. At runtime, the text is displayed that is currently indexed in the variable.

<table>
<thead>
<tr>
<th><strong>Text list</strong></th>
<th>Variable (string) or name of the text list as a fixed string in single straight quotation marks.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example: 'Errorlist'</td>
<td></td>
</tr>
<tr>
<td>▼: Drop-down list with the dialogs available in the text lists.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Text index</strong></th>
<th>Text list ID. This refers to the desired output text.</th>
</tr>
</thead>
<tbody>
<tr>
<td>▪ As fixed string with the ID in single straight quotation marks.</td>
<td></td>
</tr>
<tr>
<td>Example: '1'</td>
<td></td>
</tr>
<tr>
<td>▪ As a variable (STRING) for dynamically controlling the text output.</td>
<td></td>
</tr>
<tr>
<td>Example: strTextID</td>
<td></td>
</tr>
<tr>
<td>Sample assignment: PLC_PRG.strTextID := '1';</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Tooltip index</strong></th>
<th>Text list ID. This refers to the desired output text.</th>
</tr>
</thead>
<tbody>
<tr>
<td>▪ As fixed string with the ID in single straight quotation marks.</td>
<td></td>
</tr>
<tr>
<td>Example: '2'</td>
<td></td>
</tr>
<tr>
<td>▪ As a variable (STRING) for dynamically controlling the text output.</td>
<td></td>
</tr>
<tr>
<td>Example: strToolTipID</td>
<td></td>
</tr>
<tr>
<td>Sample assignment: PLC_PRG.strToolTipID := '2';</td>
<td></td>
</tr>
</tbody>
</table>

The variables allow for dynamic control of the text display.

See also

- [Chapter 1.3.5.18.2 “Placeholders with format definition in the output text” on page 1546](#)
- [“Element property ‘Texts’” on page 1226](#)
- [Enumerations](#)
### “Font name”

Variable (STRING). Includes the font of the text.

Example: `PLC_PRG.stFontVar := 'Arial';`

The selection of fonts corresponds to the default “Font” dialog.

### “Size”

Variable (numeric data type). Contains the font size (in pixels or points). The applied unit is specified in brackets after the variable name.

- `<pt>`: Points (default)
  
  Example: `PLC_PRG.iFontHeight <pt>`
  
  Code:
  ```
  iFontHeight : INT := 12;
  ```

- `<px>`: Pixels
  
  Example: `PLC_PRG.iFontHeight <px>`
  
  Code:
  ```
  iFontHeight : INT := 19;
  ```

If you click in the value field, a drop-down list opens on the right for setting the unit.

Hint: The font size is specified in points (example: Arial 12). Use points when the variable font size should match a font, for example if a font is set in the property “Text property ➔ Font”.

### “Flags”

Variable (DWORD). Contains the flags for displaying fonts.

Flags:

- **1**: Italics
- **2**: Bold
- **4**: Underline
- **8**: Strikethrough

Note: You can combine the font displays by adding the coding of the flags. For example, a bold and underlined text: `PLC_PRG.dwFontType := 6;`

### “Character set”

Variable (DWORD). Contains a character set number for the font.

The selection of character set numbers corresponds to the “Script” setting of the standard “Font” dialog.

### “Color”

Variable (DWORD). Includes the color of the text.

Example: `PLC_PRG.dwColorFont := 16#FF000000;`

### “Flags for text alignment”

Variable (integer data type). Contains the coding for text alignment.

Example: `PLC_PRG.dwTextAlignment;`

Coding:

- **0**: Top left
- **1**: Horizontal center
- **2**: Right
- **4**: Vertical center
- **8**: Bottom

Note: You can combine the text alignments by adding the coding of the flags. For example, a vertical and horizontal centered text: `PLC_PRG.dwFontType := 5;`
Fixed values for displaying texts are set in “Text properties”.

See also

- “Element property 'Text properties’” on page 1227

**Element property 'Colorvariables'**

The Element property is used as an interface for project variables to dynamically control colors at runtime.
| **Toggle color** | The property controls the toggled color at runtime. Value assignment:  
  ● **FALSE**: The element is displayed with the color specified in the “Color” property.  
  ● **TRUE**: The element is displayed with the color specified in the “Alarm color” property.  
  Assigning the property:  
  ● Placeholder for the user input variable  
    – “<toggle/tap variable>”  
    – “<NOT toggle/tap variable>”  
  The color change is not controlled by its own variable, but by a user input variable.  
  Note: Specify a variable for the mouse events “Tap” or “Toggle” in the input configuration of the element. Only then is the placeholder set. If you configure a variable in both “Toggle” and “Tap”, then the variable specified in “Tap” is used.  
  Hint: Click the symbol to insert the placeholder “<toggle/tap variable>”. When you activate the “Inputconfiguration”, “Tap FALSE” property, then the “<NOT toggle/tap variable>” placeholder is displayed.  
  ● Instance path of a project variable (BOOL)  
    Example: PLC_PRG.xColorIsToggeled  
    Note: In the code, declare and implement the variable specified here. Its value assignment determines when the color changes. |
| **Color** | Color variable for the frame  
  ● Variable (DWORD) for the color  
    Example: PLC_PRG.dwColor  
  ● Color literal  
    Example of gray and opaque: 16#FF888888  
  Requirement: “Show frame” property is activated.  
  Please note that the normal state is in effect if the expression in the “Colorvariables” → “Toggle color” property is not defined or it has the value FALSE. |
| **Alarm color** | Color variable for the frame in alarm state  
  ● Variable (DWORD) for the alarm color  
    Example: PLC_PRG.dwAlarmColor  
  ● Color literal  
    Example of red and opaque: 16#FFFF0000  
  Please note that the alarm state is in effect if the expression in the “Colorvariables” → “Toggle color” property has the value TRUE. |

The transparency part of the color value is evaluated only if the “Activate semi-transparent drawing” option of the visualization manager is selected.

Select the “Advanced” option in the toolbar of the properties view. Then all element properties are visible.
Element property 'Appearance variables'

The properties contain variables for controlling the appearance of the element dynamically.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Line width&quot;</td>
<td>Variable (integer data type). Contains the line weight (in pixels). Note: The values 0 and 1 both result in a line weight of one pixel. If no line should be displayed, then the &quot;Line style&quot; property must be set to the option &quot;Invisible&quot;.</td>
</tr>
</tbody>
</table>

Fixed values can be set in the "Appearance" property. These values can be overwritten by dynamic variables at runtime.

See also
- § "Element property 'Appearance'" on page 1238

Element property 'State variables'

The variables control the element behavior dynamically.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Invisible&quot;</td>
<td>Variable (BOOL). Toggles the visibility of the element. TRUE: The element is not visible at runtime.</td>
</tr>
<tr>
<td>&quot;Deactivate inputs&quot;</td>
<td>Variable (BOOL). Toggles the operability of the element. TRUE: User inputs do not have any effect in runtime more. The element is shown as deactivated.</td>
</tr>
</tbody>
</table>

The "Invisible" property is supported by the "Client Animation" functionality.

See also
- §

These properties are available only when you select the "Preview: Support client animations and overlay of native elements" option in the Visualization Manager.

You can animate a movement, a rotation, and the visibility of a visualization element.
### Animation duration
Time that the element executes an animation (in milliseconds)

**Example:** 500

### Move to the foreground
Property value (BOOL)

- **TRUE:** At runtime, the element is displayed in the foreground.
- **FALSE:** At runtime, the element is displayed in the layer where it was inserted in the visualization editor.

See also
- Chapter 1.3.5.21.5 “Using client animation” on page 1962

### Element property 'Input configuration'
The properties contain the configurations for the user input when using the mouse or keyboard.

User input is a user event from the perspective of the element.

The “Configure” button opens the “Input configuration” dialog box for creating or modifying a user input configuration.

A configuration contains one or more input actions for the respective input event. Existing input actions are displayed below it.

**Example:** “Execute ST code”:

```
PLC_PRG.i_x := 0;
```

<table>
<thead>
<tr>
<th>Event(s)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>OnDialogClosed</strong></td>
<td>Input event: The user closes the dialog box.</td>
</tr>
<tr>
<td><strong>OnMouseClick</strong></td>
<td>Input event: A user clicks the element completely. The mouse button is clicked and released.</td>
</tr>
<tr>
<td><strong>OnMouseDown</strong></td>
<td>Input event: A user clicks down on the element only.</td>
</tr>
<tr>
<td><strong>OnMouseEnter</strong></td>
<td>Input event: A user drags the mouse pointer to the element.</td>
</tr>
<tr>
<td><strong>OnMouseLeave</strong></td>
<td>Input event: A user drags the mouse pointer away from the element.</td>
</tr>
<tr>
<td><strong>OnMouseMove</strong></td>
<td>Input event: A user moves the mouse pointer over the element area.</td>
</tr>
<tr>
<td><strong>OnMouseUp</strong></td>
<td>Input event: The user releases the mouse button over the element area.</td>
</tr>
</tbody>
</table>

See also
- Chapter 1.3.5.19.3.6 “Dialog ‘Input Configuration’” on page 1586

<table>
<thead>
<tr>
<th>Hotkeys</th>
<th>Keyboard shortcut on the element for triggering specific input actions.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Key</strong></td>
<td>Key pressed for input action.</td>
</tr>
<tr>
<td><strong>Event(s)</strong></td>
<td>When the keyboard shortcut event occurs, the input actions in the “Event(s)” property are triggered.</td>
</tr>
</tbody>
</table>

- **“None”**
- **“Mouse down”**: Pressing the key triggers the input actions that are configured in the “OnMouseDown” property.
- **“Mouse up”**: Releasing the key triggers the input actions that are configured in the “OnMouseUp” property.
- **“Mouse down/up”**: Pressing and releasing the key triggers the input actions that are configured in the “OnMouseDown” property and the “OnMouseUp” property.

<table>
<thead>
<tr>
<th>Shift</th>
<th>Combination with the Shift key</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Example:</strong> [Shift]+[T].</td>
<td></td>
</tr>
</tbody>
</table>
“Control”  ✔ Combination with the Ctrl key
Example: [Ctrl]+[T].

“Alt”  ✔ Combination with the Alt key
Example: [Alt]+[T].

All keyboard shortcuts and their actions that are configured in the visualization are listed in the “Keyboard configuration” tab.

See also
- § Chapter 1.3.5.19.2.2 “Command 'Keyboard Configuration’” on page 1558

Element property 'Access rights'

Requirement: User management is set up for the visualization.

“Access rights”  Opens the “Access rights” dialog. There you can edit the access privileges for the element.
Status messages:
- “Not set. Full rights.”: Access rights for all user groups : “operable”
- “Rights are set: Limited rights”: Access is restricted for at least one group.

See also
- § Chapter 1.3.5.19.3.1 “Dialog 'Access Rights’” on page 1583

See also
-  Project Settings - Visualization
- § Chapter 1.3.5.19.2.10 “Command 'Background’” on page 1565

Visualization element 'Frame'

Symbol:

Tag: “Basic”
The element serves as a frame in which to display one or more already existing visualizations. You get a structured user interface. The size of the frame can be fixed or scaled. The display area of the referenced visualization then adapts itself to the frame size.

Element properties

<table>
<thead>
<tr>
<th>“Element name”</th>
<th>Example: refVisUserInfo</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Type of element”</td>
<td>“Frame”</td>
</tr>
</tbody>
</table>
**“Clipping”**

- **Fixed size.** Only that part of the referenced visualization that fits inside the frame is displayed.

  Requirement: “Scaling type” property is “Fixed”.

**“Show frame”**

- Displays the frame
  - **“No frame”**: The displayed area of the frame does not have borders.
  - **“Frame”**: The displayed area of the frame has borders.
  - **“No frame with offset”**: The displayed area of the frame does not have a border and the displayed area of the referenced visualization is reduced inwards by one pixel as compared to the frame area. The gap prevents the referenced visualization from touching any adjacent elements.

**“Scaling type”**

- The method with which the height and width of the referenced visualization are scaled.
  - **“Isotropic”**: The visualization is scaled to the size of the element. The visualization retains its proportions with a fixed height/width ratio.
  - **“Anisotropic”**: The visualization is scaled to the size of the element. The height and width are adapted to the element independently of each other.
  - **“Fixed”**: the visualization is displayed in its original size without taking into account the size of the element.
  - **“Fixed and scrollable”**: The visualization is displayed fixed in the element. If it is larger than the element, the element will be provided with scrollbars.

  Please note: assign variables to the properties “Scroll position variable horizontal” or “Scroll position variable vertical”. You can then edit the data of the scrollbar position in the application.

**Element properties ‘Scrollbar settings’**

- The properties contain variables for the position of the scrollboxes in the scrollbars. You can then edit the data of the scrollbar position in the application.

  Requirement: the property “Scaling type” is “fixed and scrollable”.

**“Scroll position variable horizontal”**

- Variable (integer data type, also as array). Contains the position of the horizontal or vertical scrollbox. The array contains the position for every display variant. If the visualization runs on several display variants, then the position changes are decoupled from each other.

  Example:

  ```
  PLC_PRG.iScrollHor[CURRENTCLIENTID]
  PLC_PRG.iScrollVer[CURRENTCLIENTID]
  ```

  The variable is declared as an array in the example.

  ```
  iScrollHor: ARRAY[0..20] OF INT;
  iScrollVer: ARRAY[0..20] OF INT;
  CURRENTCLIENTID indexes the current display variant.
  ```

  You can combine the variables with a unit conversion.

**See also**

- **Unit conversion**
"Deactivation of the background character": The background is drawn. The non-animated element of the referenced visualization is drawn as a background bitmap in order to optimize the performance of the visualization.

Consequence: Elements can be displayed in an unexpected order at runtime. For example, an animated element can push itself behind the frame at runtime.

☑️ Background character is deactivated in order to avoid the behaviour described above.

Element property 'References'

Contains the currently configured visualization references as a subnode

"References": Clicking "Configure" opens the "Frame Configuration" dialog. This is used to manage the referenced visualizations.

Caution: Visualizations can be nested at any depth by means of frame elements. In order to use the "Switch to any visualization" frame selection type without any problems, a frame must not contain more than 21 referenced visualizations. For more information, see also the description for the "Input configuration" of an element: Action "Switch frame visualization".

List of the currently referenced visualizations

Visualizations that have a button also have this displayed as a subnode. Each interface variable is listed with the currently assigned transfer parameters.

Example:

```plaintext
vis_FormA
  ● iDataToDisplay_1: PLC_PRG.iVar1
  ● iDataToDisplay_2: PLC_PRG.iVar2
```

Hint: You can change the assignment of the variables to an interface variable here and edit the value field. Or click the "Configure" button instead.

See also

• ☸ Chapter 1.3.5.19.2.1 “Command ‘Interface Editor’” on page 1556

• ☸ Chapter 1.3.5.19.3.6 “Dialog ‘Input Configuration’” on page 1586

Element property 'Position'

The position defines the location and size of the element in the visualization window. These are based on the Cartesian coordinate system. The origin is located at the upper left corner of the window. The positive horizontal x-axis runs to the right. The positive vertical y-axis runs downwards.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;X&quot;</td>
<td>X coordinate of the upper left corner of the element. Specified in pixels. Example: 10.</td>
</tr>
<tr>
<td>&quot;Y&quot;</td>
<td>Y coordinate of the upper left corner of the element. Specified in pixels. Example: 10.</td>
</tr>
<tr>
<td>&quot;Width&quot;</td>
<td>Specified in pixels. Example: 150</td>
</tr>
<tr>
<td>&quot;Height&quot;</td>
<td>Specified in pixels. Example: 30</td>
</tr>
</tbody>
</table>
You can also change the values by dragging the box symbols (⊃) to other positions in the editor.

See also
- ☰ Chapter 1.3.5.2 “Positioning elements, adapting size and layer” on page 1113

**Element property 'Center'**

The properties contain fixed values for the coordinates of the point of rotation. This point of rotation is shown as the △ symbol. The point is used as the center for rotating and scaling.

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>“X”</td>
<td>X-coordinate of the point of rotation</td>
</tr>
<tr>
<td>“Y”</td>
<td>Y-coordinate of the point of rotation</td>
</tr>
</tbody>
</table>

You can also change the values by dragging the symbols (⊃) to other positions in the editor.

**Element property 'Colors'**

The properties contain fixed values for the colors.

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>“Color”</td>
<td>Color of the frame</td>
</tr>
<tr>
<td></td>
<td>▼: Selection list with style colors appears</td>
</tr>
<tr>
<td></td>
<td>□: Standard dialog &quot;Color&quot; opens for selecting a color.</td>
</tr>
<tr>
<td></td>
<td>Please note: the normal state is when the boolean variable in the property “Color variables ▶ Toggle color” is not defined or its value is FALSE.</td>
</tr>
<tr>
<td>“Alarm color”</td>
<td>Color with which the element is filled during the alarm state.</td>
</tr>
<tr>
<td></td>
<td>Please note: Alarm state is when the value of the boolean variable in the property “Color variables ▶ Toggle color” is FALSE.</td>
</tr>
<tr>
<td>“Transparency”</td>
<td>Integer number (value range from 255 to 0). Specifies the transparency of the associated color.</td>
</tr>
<tr>
<td></td>
<td>255: The color is opaque.</td>
</tr>
<tr>
<td></td>
<td>0: The color is fully transparent.</td>
</tr>
<tr>
<td></td>
<td>Please note: If the color is a style color and already contains a transparency value, then this property is write-protected.</td>
</tr>
</tbody>
</table>

See also
- ☰ Chapter 1.3.5.3.3 “Assigning a color” on page 1115

**Element property 'Appearance'**

The properties contain fixed values for setting the look of the element.
### “Line width”
Value in pixels

Example: 2

Note: The values 0 and 1 both result in a line weight of one pixel. If no line should be displayed, then the “Line style” property must be set to the option “Invisible”.

### “Line style”
Type of line representation

- “Solid”
- “Dashes”
- “Dots”
- “Dash Dot”
- “Dash Dot Dot”
- “not visible”

You can assign variables in the “Appearance variables” property for controlling the appearance dynamically. The fixed values are defined here.

See also
- “Element property ‘Appearance variables’” on page 1285

---

**Element property ‘Texts’**

The properties contain character strings for labeling the element. The character string can also contain a placeholder with a format definition. In runtime mode, the placeholder is replaced by the current value in the specified format.

CODESYS accepts the specified texts automatically into the “GlobalTextList” text list. Therefore, these texts can be localized.

### “Text”
Character string (without single straight quotation marks) for the labeling the element. Add a line break by pressing the keyboard shortcut [Ctrl] + [Enter].

Example: Accesses: %i

The variable that contains the current value for the placeholder is specified in the property “Text variable Text”.

### “Tooltip”
Character string (without single straight quotation marks) that is displayed as the tooltip of an element.

Example: Number of valid accesses.

The variable that contains the current value for the placeholder is specified in the property “Text variable Tooltip”.

See also
- “Element property ‘Text variables’” on page 1229
- “Chapter 1.3.5.18.2 “Placeholders with format definition in the output text” on page 1546

---

**Element property ‘Text properties’**

The properties contain fixed values for the text properties.

<table>
<thead>
<tr>
<th>“Horizontal alignment”</th>
<th>Horizontal alignment of the text within the element.</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Vertical alignment”</td>
<td>Vertical alignment of the text within the element.</td>
</tr>
</tbody>
</table>
### "Text format"

Definition for displaying texts that are too long

- **"Default"**: The long text is truncated.
- **"Line break"**: The text is split into parts.
- **"Ellipsis"**: The visible text ends with "..." indicating that it is not complete.

### "Font"

Example: "Default"

- The "Font" dialog box opens.
- ➤: Drop-down list with style fonts.

### "Font color"

Example: "Black"

- The "Color" dialog box opens.
- ➤: Drop-down list with style colors.

### "Transparency"

Whole number (value range from 0 to 255). This determines the transparency of the respective color.

- Example: 255: The color is opaque.
- 0: The color is completely transparent.

Please note: If the color is a style color and already has a transparency value, then this property is write-protected.

### Element property 'Absolute movement'

The properties contain IEC variables for controlling the position of the element dynamically. The reference point is the upper left corner of the element. In runtime mode, the entire element is moved.

### "Movement"

#### "X"

Variable (numeric data type). Defines the X position (in pixels).

- Example: PLC_PRG.iPos_X.
- Increasing this value in runtime mode moves the element to the right.

#### "Y"

Variable (numeric data type). Defines the Y position (in pixels).

- Example: PLC_PRG.iPos_Y.
- Increasing this value in runtime mode moves the element downwards.

#### "Rotation"

Variable (numeric data type). Defines the angle of rotation (in degrees).

- Example: PLC_PRG.iAngle1.
- The midpoint of the element rotates at the "Center" point. This rotation point is shown as the symbol.
- In runtime mode, the alignment of the element remains the same with respect to the coordinate system of the visualization. Increasing the value rotates the element to the right.

#### "Scaling"

Variable (integer data type). Causes centric stretching.

- Example: PLC_PRG.iScaling.
- The reference point is the "Center" property.
- The value 1 shrinks the element by a factor of 0.001. The value 1000 returns the element to its original size.
"Interior rotation" Variable (numeric data type). Defines the angle of rotation (in degrees).

Example: PLC_PRG.iAngle2.

In runtime mode, the element rotates about the point of rotation specified in "Center" according to the value of the variable. In addition, the alignment of the element rotates according to the coordinate system of the visualization. Increasing the value in the code rotates clockwise.

The rotation point is shown as the \( \mathbb{R} \) symbol.

Note: If a static angle of rotation is specified in the property "Position \( \rightarrow \) Angle", then the static angle of rotation is added to the variable angle of rotation (offset) when the visualization is executed.

You can link the variables to a unit conversion.

The properties "X", "Y", "Rotation", and "Interior rotation" are supported by the "Client Animation" functionality.

See also
- Unit conversion

Element property 'Relative movement'

The properties contain variables for moving the element. The reference point is the position of the element ("Position" property). The shape of the element can change.

"Movement top-left"

| "X" | Variable (integer data type). It contains the number (in pixels) that the left edge is moved horizontally. Incrementing the value moves the element to the right. Example: PLC_PRG.iDeltaX |
| "Y" | Variable (integer data type). It contains the number (in pixels) that the top edge is moved vertically. Incrementing the value moves the element to the down. Example: PLC_PRG.iDeltaY |

"Movement bottom-right"

| "X" | Variable (integer data type). It contains the number (in pixels) that the right edge is moved horizontally. Incrementing the value moves the element to the right. Example: PLC_PRG.iDeltaWidth |
| "Y" | Variable (integer data type). It contains the number (in pixels) that the bottom edge is moved vertically. Incrementing the value moves the element to the down. Example: PLC_PRG.iDeltaHeight |

See also
- “Element property ‘Absolute movement’” on page 1227
Element property 'Text variables'

These properties are variables with contents that replace a format definition.

<table>
<thead>
<tr>
<th>&quot;Text variable&quot;</th>
<th>Variable (data type compliant with the format definition). It contains what is printed instead of the format definition. Example: PLC_PRG.iAccesses</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Note: The format definition is part of the text in the property “Texts ➔ Text”. Note: If you specify a variable of type enumeration with text list support, then the name of the enumeration data type is added automatically in angle brackets after the variable name. Example: PLC_PRG.enVar &lt;enumeration name&gt;. Then the symbolic value of the enumeration component is printed instead of the numeric value when text is printed. Refer to the help page for the enumerations.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>&quot;Tooltip variable&quot;</th>
<th>Variable (data type compliant with the format definition). It contains what is printed instead of the format definition. Example: PLC_PRG.iAccessesInTooltip</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Note: The format definition is part of the text in the property “Texts ➔ Tooltip”.</td>
</tr>
</tbody>
</table>

See also
- § Chapter 1.3.5.18.2 “Placeholders with format definition in the output text” on page 1546
- § “Element property 'Texts’” on page 1226
- Enumerations

Element property 'Dynamic texts'

Dynamic texts are variably indexed texts of a text list. At runtime, the text is displayed that is currently indexed in the variable.

<table>
<thead>
<tr>
<th>&quot;Text list&quot;</th>
<th>Variable (string) or name of the text list as a fixed string in single straight quotation marks. Example: 'Errorlist'</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>➔: Drop-down list with the dialogs available in the text lists.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>&quot;Text index&quot;</th>
<th>Text list ID. This refers to the desired output text. As fixed string with the ID in single straight quotation marks. Example: '1' As a variable (STRING) for dynamically controlling the text output. Example: strTextID Sample assignment: PLC_PRG.strTextID := '1';</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>&quot;Tooltip index&quot;</th>
<th>Text list ID. This refers to the desired output text. As fixed string with the ID in single straight quotation marks. Example: '2' As a variable (STRING) for dynamically controlling the text output. Example: strToolTipID Sample assignment: PLC_PRG.strToolTipID := '2';</th>
</tr>
</thead>
</table>

See also
- Text list

Element property 'Font variables'

The variables allow for dynamic control of the text display.
### “Font name”

Variable *(STRING)*. Includes the font of the text.

Example: `PLC_PRG.stFontVar := 'Arial';`

The selection of fonts corresponds to the default “Font” dialog.

### “Size”

Variable (numeric data type). Contains the font size (in pixels or points). The applied unit is specified in brackets after the variable name.

- **<pt>: Points (default)**
  
  Example: `PLC_PRG.iFontHeight <pt>`
  
  Code: `iFontHeight : INT := 12;`

- **<px>: Pixels**
  
  Example: `PLC_PRG.iFontHeight <px>`
  
  Code: `iFontHeight : INT := 19;`

If you click in the value field, a drop-down list opens on the right for setting the unit.

Hint: The font size is specified in points (example: Arial 12). Use points when the variable font size should match a font, for example if a font is set in the property “Text property → Font”.

### “Flags”

Variable *(DWORD)*. Contains the flags for displaying fonts.

Flags:

- 1: Italics
- 2: Bold
- 4: Underline
- 8: Strikethrough

Note: You can combine the font displays by adding the coding of the flags. For example, a bold and underlined text: `PLC_PRG.dwFontType := 6;`

### “Character set”

Variable *(DWORD)*. Contains a character set number for the font.

The selection of character set numbers corresponds to the “Script” setting of the standard “Font” dialog.

### “Color”

Variable *(DWORD)*. Includes the color of the text.

Example: `PLC_PRG.dwColorFont := 16#FF000000;`

### “Flags for text alignment”

Variable (integer data type). Contains the coding for text alignment.

Example: `PLC_PRG.dwTextAlignment.`

Coding:

- 0: Top left
- 1: Horizontal center
- 2: Right
- 4: Vertical center
- 8: Bottom

Note: You can combine the text alignments by adding the coding of the flags. For example, a vertical and horizontal centered text: `PLC_PRG.dwFontType := 5;`
Fixed values for displaying texts are set in “Text properties”.

See also

- “Element property ‘Text properties’” on page 1227

**Element property ‘Colorvariables’**

The Element property is used as an interface for project variables to dynamically control colors at runtime.
### "Toggle color"

The property controls the toggled color at runtime.

**Value assignment:**
- **FALSE:** The element is displayed with the color specified in the "Color" property.
- **TRUE:** The element is displayed with the color specified in the "Alarm color" property.

**Assigning the property:**
- Placeholder for the user input variable
  - "<toggle/tap variable>"
  - "<NOT toggle/tap variable>"

The color change is not controlled by its own variable, but by a user input variable.

Note: Specify a variable for the mouse events "Tap" or "Toggle" in the input configuration of the element. Only then is the placeholder set. If you configure a variable in both "Toggle" and "Tap", then the variable specified in "Tap" is used.

Hint: Click the symbol to insert the placeholder "<toggle/tap variable>". When you activate the "Inputconfiguration", "Tap FALSE" property, then the "<NOT toggle/tap variable>" placeholder is displayed.

- Instance path of a project variable (BOOL)
  Example: PLC_PRG.xColorIsToggeled

Note: In the code, declare and implement the variable specified here. Its value assignment determines when the color changes.

### "Color"

Color variable for the frame

- Variable (DWORD) for the color
  Example: PLC_PRG.dwColor
- Color literal
  Example of gray and opaque: `16#FF888888`

Requirement: "Show frame" property is activated.

Please note that the normal state is in effect if the expression in the "Colorvariables ➔ Toggle color" property is not defined or it has the value FALSE.

### "Alarm color"

Color variable for the frame in alarm state

- Variable (DWORD) for the alarm color
  Example: PLC_PRG.dwAlarmColor
- Color literal
  Example of red and opaque: `16#FFFF0000`

Please note that the alarm state is in effect if the expression in the "Colorvariables ➔ Toggle color" property has the value TRUE.

---

The transparency part of the color value is evaluated only if the "Activate semi-transparent drawing" option of the visualization manager is selected.

Select the "Advanced" option in the toolbar of the properties view. Then all element properties are visible.
Element property 'Appearance variables'

The properties contain variables for controlling the appearance of the element dynamically.

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>&quot;Line width&quot;</strong></td>
<td>Variable (integer data type). Contains the line weight (in pixels). Note: The values 0 and 1 both result in a line weight of one pixel. If no line should be displayed, then the &quot;Line style&quot; property must be set to the option &quot;Invisible&quot;.</td>
</tr>
<tr>
<td><strong>&quot;Line style&quot;</strong></td>
<td>Variable (DWORD). Controls the line style. Coding:</td>
</tr>
<tr>
<td></td>
<td>● 0: Solid line</td>
</tr>
<tr>
<td></td>
<td>● 1: Dashed line</td>
</tr>
<tr>
<td></td>
<td>● 2: Dotted line</td>
</tr>
<tr>
<td></td>
<td>● 3: Line type &quot;Dash Dot&quot;</td>
</tr>
<tr>
<td></td>
<td>● 3: Line type &quot;Dash Dot Dot&quot;</td>
</tr>
<tr>
<td></td>
<td>● 8: Invisible: The line is not drawn.</td>
</tr>
</tbody>
</table>

Fixed values can be set in the "Appearance" property. These values can be overwritten by dynamic variables at runtime.

See also
- “Element property ‘Appearance” on page 1238

Element property 'Switch frame variable'

The variable controls the switching of the referenced visualizations. This variable indexes one of the referenced frame visualizations and this is displayed in the frame. When the value of the variable changes, it switches to the recently indexed visualization.

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>&quot;Variable&quot;</strong></td>
<td>Variable (integer data type) that contains the index of the active visualization Example: PLC_PRG.uiIndexVisu</td>
</tr>
<tr>
<td></td>
<td>Hint: The “Frame Configuration” dialog includes a list of referenced visualizations. The visualizations are automatically numerically indexed via the order in the list. Note: This variant of switching usually affects all connected display variants.</td>
</tr>
<tr>
<td></td>
<td>● Array element (integer data type) for index access via CURRENTCLIENTID Example: PLC_PRG.aIndexVisu[CURRENTCLIENTID]</td>
</tr>
<tr>
<td></td>
<td>Note: This variant of switching applies to the current client only, and therefore only on one display variant. That is the display variant where the value change was triggered (for example, by means of user input).</td>
</tr>
</tbody>
</table>

See also
- “Chapter 1.3.5.19.2.9 “Command ‘Frame Selection” on page 1564

Element property 'State variables'

The variables control the element behavior dynamically.
"Invisible" Variable (BOOL). Toggles the visibility of the element.
TRUE: The element is not visible at runtime.

"Deactivate inputs" Variable (BOOL). Toggles the operability of the element.
TRUE: User inputs do not have any effect in runtime more. The element is shown as deactivated.

The "Invisible" property is supported by the "Client Animation" functionality.

See also

These properties are available only when you select the "Preview: Support client animations and overlay of native elements" option in the Visualization Manager.

You can animate a movement, a rotation, and the visibility of a visualization element.

"Animation duration" Time that the element executes an animation (in milliseconds)
Example: 500

"Move to the foreground" Property value (BOOL)
TRUE: At runtime, the element is displayed in the foreground.
FALSE: At runtime, the element is displayed in the layer where it was inserted in the visualization editor.

See also

- Chapter 1.3.5.21.5 “Using client animation” on page 1962

Element property ’Input configuration’ The properties contain the configurations for the user input when using the mouse or keyboard. A user input defines an event and one or more actions that are executed when an event occurs.

Configured user inputs are briefly listed below the events. They each include the action that is triggered and the setting in short form.

Example: "Execute ST code": PLC_PRG.i_x := 0;

"OnDialogClosed" Input event: The user closes the dialog box.

"OnMouseClick" Input event: The user clicks the mouse button completely. The mouse button is clicked and released.

"OnMouseDown" Input event: The user clicks down on the mouse button.

"OnMouseEnter" Input event: The user drags the mouse pointer to the element.

"OnMouseLeave" Input event: The user drags the mouse pointer away from the element.

"OnMouseMove" Input event: The user moves the mouse pointer over the element area.

"OnMouseUp" Input event: The user releases the mouse button over the element area.
### “Tap”
When a mouse click event occurs, the variable defined in “Variable” is described in the application. The coding depends on the options “Tap FALSE” and “Tap on enter if captured”.

### “Variable”
Variable (BOOL) that is set on mouse click.

**Example:** PLC_PRG.bIsTapped
- **TRUE:** A mouse click event exists. It lasts while the user presses the mouse button over the element. It ends when the button is released.
- **FALSE:** A mouse click event does not exist.

**Requirement:** The “Tap FALSE” option is not activated.

### “Tap FALSE”
- **☑:** The mouse click event leads to a complementary value in “Variable”.
- **TRUE:** A mouse click event does not exist.
- **FALSE:** While the mouse click event exists.

### “Tap on enter if captured”
- **☑:** During user input, it is also taken into consideration whether the mouse pointer is dragged within the element area or not while the mouse button is pressed.
- **TRUE:** While the mouse click event exists and the mouse pointer is moved over the element area.
- **FALSE:** A mouse click event does not exist. Or the user moves the mouse pointer outside of the element area while the mouse button is pressed.

The value is **TRUE** again as soon as the user moves the pointer back to the element area. The mouse is then captured.

### “Shift”
With the onset of a mouse click event, the variable is set; when the mouse click event is completed, the variable is reset.

### “Variable”
Variable (BOOL). Its value toggled when the mouse click event is ended. This is when the user releases the mouse button while the mouse pointer is over the element area.

If the user releases the mouse button while the mouse pointer is outside of the element area, then the mouse click event is not ended and the value is not toggled.

**Tip:** The user can cancel a started toggle input by dragging the mouse pointer out of the element area.

### “Toggle on up if captured”
- **☑:** The value toggles regardless of where the mouse pointer is when the mouse button is released. The mouse is then captured.

### “Keyboard shortcuts”
Keyboard shortcut on the element for triggering specific input actions.

When the keyboard shortcut event occurs, the input actions in the “Event(s)” property are triggered. In this way, it is not the input action itself that leads to this input action, but the mouse input action.

### “Key”
Key pressed for input action.

**Example:** [T]

**Note:** The following properties appear when a key is selected.
"Event(s)"

- "None"
- "Mouse down": Pressing the key triggers the input actions that are configured in the "OnMouseDown" property.
- "Mouse up": Releasing the key triggers the input actions that are configured in the "OnMouseUp" property.
- "Mouse down/up": Pressing and releasing the key triggers the input actions that are configured in the "OnMouseDown" property and the "OnMouseUp" property.

"Shift"

☑: Combination with the Shift key
Example: [Shift]+[T].

"Control"

☑: Combination with the Ctrl key
Example: [Ctrl]+[T].

"Alt"

☑: Combination with the Alt key
Example: [Alt]+[T].

All keyboard shortcuts and their actions that are configured in the visualization are listed in the "Keyboard configuration" tab.

See also
- § Chapter 1.3.5.19.2.2 “Command 'Keyboard Configuration’” on page 1558
- § Chapter 1.3.5.19.3.6 “Dialog 'Input Configuration’” on page 1586

Requirement: User management is set up for the visualization.

Element property 'Access rights'

"Access rights"

Opens the "Access rights" dialog. There you can edit the access privileges for the element.
Status messages:
- "Not set. Full rights.": Access rights for all user groups: "operable"
- "Rights are set: Limited rights": Access is restricted for at least one group.

See also
- § Chapter 1.3.5.19.3.1 “Dialog 'Access Rights’” on page 1583

Visualization element 'Label'

Symbol:

Tag: "Common controls"
The element is used to label visualizations.
### Element properties

<table>
<thead>
<tr>
<th>“Element name”</th>
<th>Optional</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hint: Assign individual names for elements so that they are found faster in the element list.</td>
<td></td>
</tr>
<tr>
<td>Example: Header_Parameter</td>
<td></td>
</tr>
</tbody>
</table>

| “Type of element” | “Label” |

### Element property 'Texts'

The property requires a string that can contain a placeholder with a format definition. At runtime, the placeholder is replaced by the current value. Specify the actual value in the “Text variables” property.

This text is entered automatically into the `GlobalTextList` text list and can be localized there.

<table>
<thead>
<tr>
<th>“Text”</th>
<th>Character string (without single straight quotation marks)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example: Main page %s</td>
<td></td>
</tr>
</tbody>
</table>

See also
- Chapter 1.3.5.18.2 “Placeholders with format definition in the output text” on page 1546

### Element property 'Position'

The position defines the location and size of the element in the visualization window. These are based on the Cartesian coordinate system. The origin is located at the upper left corner of the window. The positive horizontal x-axis runs to the right. The positive vertical y-axis runs downwards.

<table>
<thead>
<tr>
<th>“X”</th>
<th>X coordinate of the upper left corner of the element</th>
</tr>
</thead>
<tbody>
<tr>
<td>Specified in pixels.</td>
<td></td>
</tr>
<tr>
<td>Example: 10.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>“Y”</th>
<th>Y coordinate of the upper left corner of the element</th>
</tr>
</thead>
<tbody>
<tr>
<td>Specified in pixels.</td>
<td></td>
</tr>
<tr>
<td>Example: 10.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>“Width”</th>
<th>Specified in pixels.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example: 150</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>“Height”</th>
<th>Specified in pixels.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example: 30</td>
<td></td>
</tr>
</tbody>
</table>

You can also change the values by dragging the box symbols (_draging_box_symbol_ ) to other positions in the editor.

See also
- Chapter 1.3.5.3.2 “Positioning elements, adapting size and layer” on page 1113

### Element property 'Center'

The properties contain fixed values for the coordinates of the point of rotation. This point of rotation is shown as the __rotation_center_symbol_ symbol. The point is used as the center for rotating and scaling.
<table>
<thead>
<tr>
<th><strong>X</strong></th>
<th>X-coordinate of the point of rotation</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Y</strong></td>
<td>Y-coordinate of the point of rotation</td>
</tr>
</tbody>
</table>

You can also change the values by dragging the symbols (_drag_ ) to other positions in the editor.

**Element property 'Text properties'**

<table>
<thead>
<tr>
<th><strong>Horizontal alignment</strong></th>
<th>Horizontal alignment of the text within the element.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Vertical alignment</strong></td>
<td>Vertical alignment of the text within the element.</td>
</tr>
<tr>
<td><strong>Text format</strong></td>
<td>Definition for displaying texts that are too long</td>
</tr>
<tr>
<td></td>
<td>● &quot;Default&quot;: The long text is truncated.</td>
</tr>
<tr>
<td></td>
<td>● &quot;Line break&quot;: The text is split into parts.</td>
</tr>
<tr>
<td></td>
<td>● &quot;Ellipsis&quot;: The visible text ends with &quot;...&quot; indicating that it is not complete.</td>
</tr>
</tbody>
</table>

**Font**

Example: "Default"

_drag_. The "Font" dialog box opens.  

▼: Drop-down list with style fonts.

**Font color**

Example: "Black"

_drag_. The "Color" dialog box opens.  

▼: Drop-down list with style colors.

**Transparency**

Whole number (value range from 0 to 255). This determines the transparency of the respective color.

Example: 255: The color is opaque.  
0: The color is completely transparent.  

Please note: If the color is a style color and already has a transparency value, then this property is write-protected.

**Element property 'Absolute movement'**

The properties contain IEC variables for controlling the position of the element dynamically. The reference point is the upper left corner of the element. In runtime mode, the entire element is moved.

<table>
<thead>
<tr>
<th><strong>Movement</strong></th>
<th></th>
</tr>
</thead>
</table>
| **X**        | Variable (numeric data type). Defines the X position (in pixels).  
|              | Example: PLC_PRG.iPos_X.  
|              | Increasing this value in runtime mode moves the element to the right. |
| **Y**        | Variable (numeric data type). Defines the Y position (in pixels).  
|              | Example: PLC_PRG.iPos_Y.  
|              | Increasing this value in runtime mode moves the element downwards. |
### "Rotation"

Variable (numeric data type). Defines the angle of rotation (in degrees).

**Example:** PLC_PRG.iAngle1.

The midpoint of the element rotates at the "Center" point. This rotation point is shown as the symbol.

In runtime mode, the alignment of the element remains the same with respect to the coordinate system of the visualization. Increasing the value rotates the element to the right.

### "Interior rotation"

Variable (numeric data type). Defines the angle of rotation (in degrees).

**Example:** PLC_PRG.iAngle2.

In runtime mode, the element rotates about the point of rotation specified in "Center" according to the value of the variable. In addition, the alignment of the element rotates according to the coordinate system of the visualization. Increasing the value in the code rotates clockwise.

The rotation point is shown as the symbol.

Note: If a static angle of rotation is specified in the "Position Angle" property, then the static angle of rotation is added to the variable angle of rotation (offset) when the visualization is executed.

---

**You can link the variables to a unit conversion.**

**The "X", "Y", "Rotation", and "Interior rotation" properties are supported by the "Client Animation" functionality.**

---

**See also**

- **Unit conversion**

---

**Element property 'State variables'**

The variables control the element behavior dynamically.

<table>
<thead>
<tr>
<th>&quot;Invisible&quot;</th>
<th>Variable (BOOL). Toggles the visibility of the element.</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRUE: The element is not visible at runtime.</td>
<td></td>
</tr>
</tbody>
</table>

**The "Invisible" property is supported by the "Client Animation" functionality.**
These properties are available only when you select the “Preview: Support client animations and overlay of native elements” option in the Visualization Manager.

You can animate a movement, a rotation, and the visibility of a visualization element.

<table>
<thead>
<tr>
<th>“Animation duration”</th>
<th>Time that the element executes an animation (in milliseconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example: 500</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>“Move to the foreground”</th>
<th>Property value (BOOL)</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRUE:</td>
<td>At runtime, the element is displayed in the foreground.</td>
</tr>
<tr>
<td>FALSE:</td>
<td>At runtime, the element is displayed in the layer where it was inserted in the visualization editor.</td>
</tr>
</tbody>
</table>

See also
● Chapter 1.3.5.21.5 “Using client animation” on page 1962

Requirement: User management is set up for the visualization.

<table>
<thead>
<tr>
<th>“Access rights”</th>
<th>Opens the “Access rights” dialog. There you can edit the access privileges for the element.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Status messages:</td>
<td>● “Not set. Full rights.”: Access rights for all user groups: “operable”</td>
</tr>
<tr>
<td></td>
<td>● “Rights are set: Limited rights”: Access is restricted for at least one group.</td>
</tr>
</tbody>
</table>

See also
● Chapter 1.3.5.19.3.1 “Dialog ‘Access Rights’” on page 1583

See also

Visualization element ‘Combo box array’

Symbol:

Tag: “Common controls”

The element shows values of an array as a drop-down list. When the visualization user clicks an entry, the array index of the entry is written to an integer variable.
**Element name**

Optional

Hint: Assign individual names for elements so that they are found faster in the element list.

Example: `List_Product_Number`

**Type of element**

“Combo box array”

---

**Element property ’Position’**

The position defines the location and size of the element in the visualization window. These are based on the Cartesian coordinate system. The origin is located at the upper left corner of the window. The positive horizontal x-axis runs to the right. The positive vertical y-axis runs downwards.

| **“X”** | X coordinate of the upper left corner of the element
| Specified in pixels.
| Example: 10. |
| **“Y”** | Y coordinate of the upper left corner of the element
| Specified in pixels.
| Example: 10. |
| **“Width”** | Specified in pixels.
| Example: 150 |
| **“Height”** | Specified in pixels.
| Example: 30 |

You can also change the values by dragging the box symbols (redicate) to other positions in the editor.

See also

- [Chapter 1.3.5.3.2 “Positioning elements, adapting size and layer” on page 1113](#)

---

**Element property ’Center’**

The properties contain fixed values for the coordinates of the point of rotation. This point of rotation is shown as the (predicate) symbol. The point is used as the center for rotating and scaling.

| **“X”** | X-coordinate of the point of rotation |
| **“Y”** | Y-coordinate of the point of rotation |

You can also change the values by dragging the symbols (predicate) to other positions in the editor.
**“Variable”**

At runtime, saves the array index of the list entry that the user clicks.

Property value
- Variable (integer data type)
  
  **Example:** `PLC_PRG.iIndexComboboxEntry`

- Enumeration variable with text list support
  
  **Example:** `PLC_PRG.eMyCombobox<COMBO>`

  **Note:** Value range of the enumeration value that lies within the DWORD or DINT value range.

**“Data array”**

Displayed as combo box. Every array component becomes a combo box entry.

Property value
- Array variable (ARRAY[... ] OF)
  
  **Example:** `PLC_PRG.astrCombobox`

  **Declaration:**
  ```plaintext```
  ```plaintext```

See also
- Enumerations
- Chapter 1.3.5.6 “Setting up multiple languages” on page 1143

**Element property 'Columns'**

The “Combo box – Array” element visualizes an array variable or structure variable in a tabular view. The index of array elements or structure members is shown in a column or row. Two-dimensional arrays or structure arrays are shown in several columns. You specify the visualized variable in the “Data array” property. If a variable is assigned there, then you can specify the display of the table columns where the array elements are shown. You can customize each column that is assigned to an index [<n>].

**“Columns”**

- [<n>]

  **Due to the structure of the variable that is defined in “Data array”, CODESYS determines the number of columns and defines them with the index <n>.**

  **Example:**
  ```plaintext```
  StringTable : ARRAY [0..2, 0..4] OF STRING :=
  ['BMW','Audi','Mercedes','VW','Fiat',
   '150','150','150','150','100','blue','gray','silver','blue',
   'red'];: three columns are formed [0],[1] and [2].
  ```plaintext```

**“Max. array index”**

Optional. Variable (integer data type) or value. Defines up to which array index the data is displayed.

**“Row height”**

Height of the rows (in pixels).

**“Number visible rows”**

Optional. If the array is larger than the number of visible rows, then a scrollbar is included.

**“Scrollbar size”**

Width of the vertical scrollbar (in pixels).

**Table 349: “Element property 'Columns: Column [<n>]’”**

<table>
<thead>
<tr>
<th>“Width”</th>
<th>Column width (in pixels).</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Image column”</td>
<td>Images can be displayed in the column. Images are used from the global image pool or user-defined image pools. The image IDs are shown in the cells of the table as defined in the image pool.</td>
</tr>
<tr>
<td>“Image configuration”</td>
<td></td>
</tr>
</tbody>
</table>
### "Fill mode"
- **"Fill cell"**
  The image resizes to the dimensions of the cell without fixing the height/width ratio.
- **"Centered"**
  The image is centered in the cell and retains its proportions (height-width ratio).

### "Transparency"
- : The color that is specified in "Transparent color" is displayed as transparent.

### "Transparent color"
When the "Transparent" property is enabled, the color specified here is not displayed. Pixels with this color are transparent.

### "Text alignment in column"
- **"Left"**
- **"Centered"**
- **"Right"**

### Element property 'Texts'  

<table>
<thead>
<tr>
<th>&quot;Tooltip&quot;</th>
<th>Character string (without single straight quotation marks) that is displayed as the tooltip of an element in runtime mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example:</td>
<td>Products of customer A</td>
</tr>
<tr>
<td>Hint:</td>
<td>The text is accepted automatically into the &quot;GlobalTextList&quot; text list and can be localized there.</td>
</tr>
</tbody>
</table>

See also
- 

### Element property 'Text properties'  

The properties contain fixed values for the text properties.

| "Usage of" |  
|-------------|------------------------------------------------------------------------------------------------------------------|
| - **"Default style values"**: The values of the visualization style are used.  
- **"Individual settings"**: The "Individual text properties" property group is shown |
| The values can be customized here. |

<table>
<thead>
<tr>
<th>&quot;Individual text properties&quot;</th>
</tr>
</thead>
<tbody>
<tr>
<td>Requirement: The &quot;Individual settings&quot; text property is defined.</td>
</tr>
</tbody>
</table>

| "Font" | 
|--------|--------------------------------------------------|
| Example: "Default" |
| : The "Font" dialog box opens. |
| : Drop-down list with style fonts. |

| "Font color" | 
|-------------|--------------------------------------------------|
| Example: "Black" |
| : The "Color" dialog box opens. |
| : Drop-down list with style colors. |

| "Transparency" | 
|----------------|--------------------------------------------------|
| Whole number (value range from 0 to 255). This determines the transparency of the respective color. |
| Example: 255: The color is opaque. |
| 0: The color is completely transparent. |
| Please note: If the color is a style color and already has a transparency value, then this property is write-protected. |
The properties contain IEC variables for controlling the position of the element dynamically. The reference point is the upper left corner of the element. In runtime mode, the entire element is moved.

### "Movement"

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>X</strong></td>
<td>Variable (numeric data type). Defines the X position (in pixels). Increasing this value in runtime mode moves the element to the right.</td>
<td>PLC_PRG.iPos_X</td>
</tr>
<tr>
<td><strong>Y</strong></td>
<td>Variable (numeric data type). Defines the Y position (in pixels). Increasing this value in runtime mode moves the element downwards.</td>
<td>PLC_PRG.iPos_Y</td>
</tr>
</tbody>
</table>

### "Rotation" Variable (numeric data type). Defines the angle of rotation (in degrees). The midpoint of the element rotates at the "Center" point. This rotation point is shown as the symbol. In runtime mode, the alignment of the element remains the same with respect to the coordinate system of the visualization. Increasing the value rotates the element to the right.

### "Interior rotation" Variable (numeric data type). Defines the angle of rotation (in degrees). In runtime mode, the element rotates about the point of rotation specified in "Center" according to the value of the variable. In addition, the alignment of the element rotates according to the coordinate system of the visualization. Increasing the value in the code rotates clockwise. The rotation point is shown as the symbol.

Note: If a static angle of rotation is specified in the "Position ➔ Angle" property, then the static angle of rotation is added to the variable angle of rotation (offset) when the visualization is executed.

You can link the variables to a unit conversion.

The "X", "Y", "Rotation", and "Interior rotation" properties are supported by the "Client Animation" functionality.
The variables control the element behavior dynamically.

<table>
<thead>
<tr>
<th><strong>“Invisible”</strong></th>
<th>Variable (BOOL). Toggles the visibility of the element. TRUE: The element is not visible at runtime.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>“Deactivate inputs”</strong></td>
<td>Variable (BOOL). Toggles the operability of the element. TRUE: User inputs do not have any effect in runtime more. The element is shown as deactivated.</td>
</tr>
</tbody>
</table>

The “Invisible” property is supported by the “Client Animation” functionality.

These properties are available only when you select the “Preview: Support client animations and overlay of native elements” option in the Visualization Manager.

You can animate a movement, a rotation, and the visibility of a visualization element.

<table>
<thead>
<tr>
<th><strong>“Animation duration”</strong></th>
<th>Time that the element executes an animation (in milliseconds) Example: 500</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>“Move to the foreground”</strong></td>
<td>Property value (BOOL) TRUE: At runtime, the element is displayed in the foreground. FALSE: At runtime, the element is displayed in the layer where it was inserted in the visualization editor.</td>
</tr>
</tbody>
</table>

Visualization element ‘Combo box integer’

Symbol:

Tag: “Common controls”
The element shows values as a drop-down list. When the user clicks an entry, the ID of the entry is written to an integer variable. The entries in the drop-down list can be from a list and contain images from an image pool.

**Element properties**

<table>
<thead>
<tr>
<th>“Element name”</th>
<th>Example: List of product numbers</th>
</tr>
</thead>
<tbody>
<tr>
<td>Optional</td>
<td>Hint: Assign individual names for elements so that they are found faster in the element list.</td>
</tr>
</tbody>
</table>

| “Type of element” | “Combo box integer” |

**Element property 'Position'**

The position defines the location and size of the element in the visualization window. These are based on the Cartesian coordinate system. The origin is located at the upper left corner of the window. The positive horizontal x-axis runs to the right. The positive vertical y-axis runs downwards.

<table>
<thead>
<tr>
<th>“X”</th>
<th>X coordinate of the upper left corner of the element</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Specified in pixels.</td>
</tr>
<tr>
<td></td>
<td>Example: 10.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>“Y”</th>
<th>Y coordinate of the upper left corner of the element</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Specified in pixels.</td>
</tr>
<tr>
<td></td>
<td>Example: 10.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>“Width”</th>
<th>Specified in pixels.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Example: 150</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>“Height”</th>
<th>Specified in pixels.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Example: 30</td>
</tr>
</tbody>
</table>

You can also change the values by dragging the box symbols ((strcmp)) to other positions in the editor.

See also

- Chapter 1.3.5.3.2 “Positioning elements, adapting size and layer” on page 1113

**Element property 'Center’**

The properties contain fixed values for the coordinates of the point of rotation. This point of rotation is shown as the symbol. The point is used as the center for rotating and scaling.

<table>
<thead>
<tr>
<th>“X”</th>
<th>X-coordinate of the point of rotation</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>“Y”</th>
<th>Y-coordinate of the point of rotation</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

You can also change the values by dragging the symbols (strcmp) to other positions in the editor.
At runtime, saves the text list ID of the list entry that the user clicks if only one image pool is displayed, then the image ID is saved.

**Property value**
- Variable (integer data type)
  - Example: `PLC_PRG.iIDComboboxEntry`
- Enumeration variable with text list support
  - Example: `PLC_PRG.eMyCombobox<COMBO>`

**Text list**
Displayed as combo box. Every text list entry becomes a combo box entry.

**Transfer value**
- Text list identifier as string
  - Example: 'TextList_A'
  - Note: The IDs of the text list must be lie within the value range of DWORD or DINT.
- Empty
  - When an enumeration variable with text list support is specified in the "Variable" property
  - When only one image pool is visualized

**Image pool**
Displayed as combo box. Every image in the image pool becomes a combo box entry.

Example: 'ImagePool_A'

See also
- Enumerations
- Chapter 1.3.5.6 “Setting up multiple languages” on page 1143

**Element property 'Settings of the list'**
Displayed list that expands when a visualization user clicks into the element.

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Number of rows setting&quot;</td>
<td>• &quot;From style&quot;:&lt;br&gt;• &quot;Explicit&quot;: Then the &quot;Number of visible rows&quot; property appears below it.</td>
</tr>
<tr>
<td>&quot;Number of visible rows&quot;</td>
<td>Number of visible lines of the combo box drop-down list defined here</td>
</tr>
<tr>
<td></td>
<td>• Integer literal&lt;br&gt;Example: 5</td>
</tr>
<tr>
<td></td>
<td>• Variable (integer data type)&lt;br&gt;Example: <code>PLC_PRG.iNumberOfVisibleRows</code></td>
</tr>
<tr>
<td></td>
<td>Note: The property is available when the &quot;Number of rows setting&quot; property is set to &quot;Explicit&quot;.</td>
</tr>
<tr>
<td>&quot;Row height&quot;</td>
<td>• &quot;From style&quot;:&lt;br&gt;• Literal&lt;br&gt;Example: 20</td>
</tr>
<tr>
<td>&quot;Height of image&quot;</td>
<td>Image height (in pixels) of the image displayed in the drop-down list entry</td>
</tr>
<tr>
<td></td>
<td>• &quot;From style&quot;:&lt;br&gt;• Integer literal&lt;br&gt;Example: 30</td>
</tr>
<tr>
<td></td>
<td>Note: Images are displayed only when a value is specified in the &quot;Image pool&quot; property.</td>
</tr>
</tbody>
</table>
### "Width of image"

Image width (in pixels) of the image displayed in the drop-down list entry

- **From style**:  
  - Literal  
    - Example: 30

Note: Images are displayed only when a value is specified in the "Image pool" property.

### "Offset of image"

Makes the images in the selection list appear offset (in pixels) from the left margin. An offset of 0 means that the images are displayed directly on the margin.

- **From style**:  
  - Literal  
    - Example: 4

Note: Images are displayed only when a value is specified in the "Image pool" property.

### "Scrollbar size"

Size of the scrollbar (in pixels). The scrollbar is displayed when more entries are specified in the drop-down list than in "Number of visible rows".

Default: 20

### Element property 'Texts'

#### "Tooltip"

Character string (without single straight quotation marks) that is displayed as the tooltip of an element in runtime mode

**Example**: Products of customer A

**Hint**: The text is accepted automatically into the "GlobalTextList" text list and can be localized there.

### Element property 'Value range'

#### "Limit valuerange"

Limits the text list to one subrange. This subrange is displayed by the combo box.

Requirement: A value is specified in the "Text list" property.

☑: Only the subrange that is defined by the "Minimum value" and "Maximum value" properties is displayed as a drop-down list.

#### "Minimum value"

ID of the text list entry from which a combo box entry is displayed

- **Literal** (ANY_NUM)  
  - Example: 5
- **Variable** (integer data type)  
  - Example: PLC_PRG.iFirstEntry
<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Maximum value</strong></td>
<td>ID of the text list entry up to which combo box entries are displayed</td>
</tr>
<tr>
<td></td>
<td>• <strong>Literal (ANY_NUM)</strong>&lt;br&gt;Example: 10&lt;br&gt;• <strong>Variable (integer data type)</strong>&lt;br&gt;Example: PLC_PRG.iLastEntry</td>
</tr>
<tr>
<td><strong>Filter missing text entries</strong></td>
<td>Text list is refreshed and any unused texts (IDs) are removed.&lt;br&gt;Requirement: A value is specified in the &quot;Text list&quot; property.</td>
</tr>
</tbody>
</table>

**Element properties 'Text properties'**<br>The properties contain fixed values for the text properties.

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Usage of</strong></td>
<td>• <strong>Default style values</strong>: The values of the visualization style are used.&lt;br&gt;• <strong>Individual settings</strong>: The &quot;Individual text properties&quot; property group is shown&lt;br&gt;The values can be customized here.</td>
</tr>
<tr>
<td><strong>Individual text properties</strong></td>
<td>Requirement: The &quot;Individual settings&quot; text property is defined.</td>
</tr>
<tr>
<td><strong>Horizontal alignment</strong></td>
<td>Horizontal alignment of the text within the element.</td>
</tr>
<tr>
<td><strong>Vertical alignment</strong></td>
<td>Vertical alignment of the text within the element.</td>
</tr>
<tr>
<td><strong>Font</strong></td>
<td>Example: &quot;Default&quot;&lt;br&gt;➡️: The &quot;Font&quot; dialog box opens.&lt;br&gt;▼: Drop-down list with style fonts.</td>
</tr>
<tr>
<td><strong>Font color</strong></td>
<td>Example: &quot;Black&quot;&lt;br&gt;➡️: The &quot;Color&quot; dialog box opens.&lt;br&gt;▼: Drop-down list with style colors.</td>
</tr>
<tr>
<td><strong>Transparency</strong></td>
<td>Whole number (value range from 0 to 255). This determines the transparency of the respective color.&lt;br&gt;Example: 255: The color is opaque.&lt;br&gt;0: The color is completely transparent.&lt;br&gt;Please note: If the color is a style color and already has a transparency value, then this property is write-protected.</td>
</tr>
</tbody>
</table>

**Element property 'Absolute movement'**<br>The properties contain IEC variables for controlling the position of the element dynamically. The reference point is the upper left corner of the element. In runtime mode, the entire element is moved.

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Movement</strong></td>
<td></td>
</tr>
<tr>
<td><strong>X</strong></td>
<td>Variable (numeric data type). Defines the X position (in pixels).&lt;br&gt;Example: PLC_PRG.iPos_X.&lt;br&gt;Increasing this value in runtime mode moves the element to the right.</td>
</tr>
<tr>
<td><strong>Y</strong></td>
<td>Variable (numeric data type). Defines the Y position (in pixels).&lt;br&gt;Example: PLC_PRG.iPos_Y.&lt;br&gt;Increasing this value in runtime mode moves the element downwards.</td>
</tr>
</tbody>
</table>
### “Rotation”

Variable (numeric data type). Defines the angle of rotation (in degrees).

**Example:** PLC_PRG.iAngle1.

The midpoint of the element rotates at the “Center” point. This rotation point is shown as the symbol.

In runtime mode, the alignment of the element remains the same with respect to the coordinate system of the visualization. Increasing the value rotates the element to the right.

### “Interior rotation”

Variable (numeric data type). Defines the angle of rotation (in degrees).

**Example:** PLC_PRG.iAngle2.

In runtime mode, the element rotates about the point of rotation specified in “Center” according to the value of the variable. In addition, the alignment of the element rotates according to the coordinate system of the visualization. Increasing the value in the code rotates clockwise.

The rotation point is shown as the symbol.

Note: If a static angle of rotation is specified in the “Position → Angle” property, then the static angle of rotation is added to the variable angle of rotation (offset) when the visualization is executed.

---

**You can link the variables to a unit conversion.**

**The “X”, “Y”, “Rotation”, and “Interior rotation” properties are supported by the “Client Animation” functionality.**

---

### “Invisible”

Variable (BOOL). Toggles the visibility of the element.

**TRUE:** The element is not visible at runtime.

### “Deactivate inputs”

Variable (BOOL). Toggles the operability of the element.

**TRUE:** User inputs do not have any effect in runtime more. The element is shown as deactivated.

---

Element property ‘State variables’

The variables control the element behavior dynamically.
The “Invisible” property is supported by the “Client Animation” functionality.

See also

- These properties are available only when you select the “Preview: Support client animations and overlay of native elements” option in the Visualization Manager.

You can animate a movement, a rotation, and the visibility of a visualization element.

<table>
<thead>
<tr>
<th>“Animation duration”</th>
<th>Time that the element executes an animation (in milliseconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example: 500</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>“Move to the foreground”</th>
<th>Property value (BOOL)</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRUE: At runtime, the element is displayed in the foreground.</td>
<td></td>
</tr>
<tr>
<td>FALSE: At runtime, the element is displayed in the layer where it was inserted in the visualization editor.</td>
<td></td>
</tr>
</tbody>
</table>

See also

- ☞ Chapter 1.3.5.21.5 “Using client animation” on page 1962

**Element property ‘Access rights’**

Requirement: User management is set up for the visualization.

<table>
<thead>
<tr>
<th>“Access rights”</th>
<th>Opens the “Access rights” dialog. There you can edit the access privileges for the element.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Status messages:</td>
<td>- “Not set. Full rights.”: Access rights for all user groups: “operable”</td>
</tr>
<tr>
<td></td>
<td>- “Rights are set: Limited rights”: Access is restricted for at least one group.</td>
</tr>
</tbody>
</table>

See also

- ☞ Chapter 1.3.5.19.3.1 “Dialog ‘Access Rights’” on page 1583

See also

- ☞ Visualization element ‘Tab control’

Symbol:

Tag: “Common controls”

The element displays selected visualizations in tabs. The tabs can be used by means of the tab header without having to configure an input configuration. A visualization user switches between visualizations by clicking the tab header.
Element properties

<table>
<thead>
<tr>
<th>“Element name”</th>
<th>Example: Assembly A</th>
</tr>
</thead>
<tbody>
<tr>
<td>Optional</td>
<td>Hint: Assign individual names for elements so that they are found faster in the element list.</td>
</tr>
</tbody>
</table>

| “Type of element” | “Tab control” |

| “Tab width” | Width of the tab (in pixels). If there is not space for all tab headers, then a scrollbar is added. |
| Example: 30  |

<table>
<thead>
<tr>
<th>“Tab height”</th>
<th>Height of the tab (in pixels)</th>
</tr>
</thead>
<tbody>
<tr>
<td>● Integer literal</td>
<td></td>
</tr>
<tr>
<td>Example: 15</td>
<td></td>
</tr>
<tr>
<td>● “From style”</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>“Scaling type”</th>
<th>The method with which the height and width of the referenced visualization are scaled.</th>
</tr>
</thead>
<tbody>
<tr>
<td>● “Isotropic”: The visualization is scaled to the size of the element. The visualization retains its proportions with a fixed height/width ratio.</td>
<td></td>
</tr>
<tr>
<td>● “Anisotropic”: The visualization is scaled to the size of the element. The height and width are adapted to the element independently of each other.</td>
<td></td>
</tr>
<tr>
<td>● “Fixed”: the visualization is displayed in its original size without taking into account the size of the element.</td>
<td></td>
</tr>
<tr>
<td>● “Fixed and scrollable”: The visualization is displayed fixed in the element. If it is larger than the element, the element will be provided with scrollbars. Please note: assign variables to the properties “Scroll position variable horizontal” or “Scroll position variable vertical”. You can then edit the data of the scrollbar position in the application.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>“Deactivate background drawing”</th>
<th>☐: The non-animated elements of the referenced visualization are displayed as background images in order to optimize the performance of the visualization.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Result: At runtime, the elements can be displayed in any order, for example when an element moves behind the frame at runtime.</td>
</tr>
<tr>
<td></td>
<td>☑: Deactivates the background display in order to prevent the behavior described above</td>
</tr>
<tr>
<td></td>
<td>The property is not available for the following settings:</td>
</tr>
<tr>
<td></td>
<td>● The “ Scaling type” property is set to “Fixed and scrollable”</td>
</tr>
<tr>
<td></td>
<td>● The client animation functionality is enabled.</td>
</tr>
</tbody>
</table>

See also

●

Element property 'Scrollbar settings'

The properties include variables for the position of the scroll boxes in the scrollbars. You can process the data for the scroll box position in the application.
Requirement: The “Scaling type” property is “Fixed and scrollable”.

| “Scroll position variable horizontal” | Variable (integer data type, also array). Includes the position of the horizontal or vertical scroll box. The array contains the position for each display variant. If the visualization is running on multiple display variants, then the position changes are disconnected from each other. Example: 
PLC_PRG.iScrollHor[CURRENTCLIENTID] 
PLC_PRG.iScrollVer[CURRENTCLIENTID] 
In this example, the variable is declared as an array: 
iScrollHor: ARRAY[0..20] OF INT; 
iScrollVer: ARRAY[0..20] OF INT; 
CURRENTCLIENTID indicates the current display variant. |
| “Scroll position variable, vertical” |

See also
● Unit conversion

Element property 'References’

| “References” | Clicking “Configure” opens the “Frame Configuration” dialog. You can select an existing visualization there. Selected visualization references are shown in the properties. Selected visualization references are listed here as subordinate properties. |
| Name pf the visualization reference (example: PLC_PRG.S1) |
| “Heading” | Tab caption (example: Panel) |
| “Image ID” | Image ID in the theme <image pool name>.<ID> Example: Imagepool_A.1 for the image with ID 1 in Imagepool_A |

Interface parameter of the visualization reference Example: iX If the visualization has an interface, then their parameters are displayed here as subordinate properties. Variable (data type conforms to data type of the interface parameter). Includes the initialization value for the instantiation of the visualization.

See also
●
● © Chapter 1.3.5.19.2.1 “Command 'Interface Editor’” on page 1556
● © Chapter 1.3.5.19.2.9 “Command 'Frame Selection’” on page 1564

Element property 'Position’ The position defines the location and size of the element in the visualization window. These are based on the Cartesian coordinate system. The origin is located at the upper left corner of the window. The positive horizontal x-axis runs to the right. The positive vertical y-axis runs downwards.
**"X"** | X coordinate of the upper left corner of the element  
Specified in pixels.  
Example: 10.

| **"Y"** | Y coordinate of the upper left corner of the element  
Specified in pixels.  
Example: 10.

| **"Width"** | Specified in pixels.  
Example: 150

| **"Height"** | Specified in pixels.  
Example: 30

---

General Information: You can also change the values by dragging the box symbols (_drag_ ) to other positions in the editor.

See also
- Chapter 1.3.5.3.2 “Positioning elements, adapting size and layer” on page 1113

**Element property 'Center'**

The properties contain fixed values for the coordinates of the point of rotation. This point of rotation is shown as the _rotation_ symbol. The point is used as the center for rotating and scaling.

| **"X"** | X-coordinate of the point of rotation

| **"Y"** | Y-coordinate of the point of rotation

---

General Information: You can also change the values by dragging the symbols (_rotation_ ) to other positions in the editor.

**Element property 'Switch frame variable'**

| **"Variable"** | Variable (integer data type). Specifies the index of the active visualization.  
Example: PLC_PRG.uiActiveVisuID.  
Tip: The “Frame Configuration” dialog box includes a list of selected visualizations. The visualizations are ordered automatically in numeric order in the list.

See also
- Chapter 1.3.5.19.2.9 “Command ‘Frame Selection’” on page 1564

**Element property 'Absolute movement'**

The properties contain IEC variables for controlling the position of the element dynamically. The reference point is the upper left corner of the element. In runtime mode, the entire element is moved.
**Movement**

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
<th>Example</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>“X”</td>
<td>Variable (numeric data type). Defines the X position (in pixels).</td>
<td>PLC_PRG.iPos_X.</td>
<td>Increasing this value in runtime mode moves the element to the right.</td>
</tr>
<tr>
<td>“Y”</td>
<td>Variable (numeric data type). Defines the Y position (in pixels).</td>
<td>PLC_PRG.iPos_Y.</td>
<td>Increasing this value in runtime mode moves the element downwards.</td>
</tr>
<tr>
<td>“Rotation”</td>
<td>Variable (numeric data type). Defines the angle of rotation (in degrees).</td>
<td>PLC_PRG.iAngle1.</td>
<td>The midpoint of the element rotates at the “Center” point. This rotation point is shown as the symbol. In runtime mode, the alignment of the element remains the same with respect to the coordinate system of the visualization. Increasing the value rotates the element to the right.</td>
</tr>
<tr>
<td>“Interior rotation”</td>
<td>Variable (numeric data type). Defines the angle of rotation (in degrees).</td>
<td>PLC_PRG.iAngle2.</td>
<td>In runtime mode, the element rotates about the point of rotation specified in “Center” according to the value of the variable. In addition, the alignment of the element rotates according to the coordinate system of the visualization. Increasing the value in the code rotates clockwise. The rotation point is shown as the symbol. Note: If a static angle of rotation is specified in the “Position ➤ Angle” property, then the static angle of rotation is added to the variable angle of rotation (offset) when the visualization is executed.</td>
</tr>
</tbody>
</table>

You can link the variables to a unit conversion.

The “X”, “Y”, “Rotation”, and “Interior rotation” properties are supported by the “Client Animation” functionality.

See also
- Unit conversion
**Element property 'State variables'**

The variables control the element behavior dynamically.

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Invisible&quot;</td>
<td>Variable (BOOL). Toggles the visibility of the element. &lt;br&gt;TRUE: The element is not visible at runtime.</td>
</tr>
<tr>
<td>&quot;Deactivate inputs&quot;</td>
<td>Variable (BOOL). Toggles the operability of the element. &lt;br&gt;TRUE: User inputs do not have any effect in runtime more. The element is shown as deactivated.</td>
</tr>
</tbody>
</table>

*The “Invisible” property is supported by the "Client Animation” functionality.*

See also

- 

These properties are available only when you select the “Preview: Support client animations and overlay of native elements” option in the Visualization Manager.

You can animate a movement, a rotation, and the visibility of a visualization element.

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Animation duration&quot;</td>
<td>Time that the element executes an animation (in milliseconds) &lt;br&gt;Example: 500</td>
</tr>
<tr>
<td>&quot;Move to the foreground&quot;</td>
<td>Property value (BOOL) &lt;br&gt;TRUE: At runtime, the element is displayed in the foreground. &lt;br&gt;FALSE: At runtime, the element is displayed in the layer where it was inserted in the visualization editor.</td>
</tr>
</tbody>
</table>

See also

-  Chapter 1.3.5.21.5 “Using client animation” on page 1962

Visualization element 'Button'

- Symbol:

  ![Symbol](image)

- Tag: “Common controls”

This element triggers an action, such as setting a variable.

**Element properties**
Element name

Optional

Hint: Assign individual names for elements so that they are found faster in the element list.

Example: Voltage_on

Type of element

“Button”

The position defines the location and size of the element in the visualization window. These are based on the Cartesian coordinate system. The origin is located at the upper left corner of the window. The positive horizontal x-axis runs to the right. The positive vertical y-axis runs downwards.

<table>
<thead>
<tr>
<th>“X”</th>
<th>X coordinate of the upper left corner of the element</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Specified in pixels.</td>
</tr>
<tr>
<td></td>
<td>Example: 10.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>“Y”</th>
<th>Y coordinate of the upper left corner of the element</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Specified in pixels.</td>
</tr>
<tr>
<td></td>
<td>Example: 10.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>“Width”</th>
<th>Specified in pixels.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Example: 150</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>“Height”</th>
<th>Specified in pixels.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Example: 30</td>
</tr>
</tbody>
</table>

You can also change the values by dragging the box symbols ((Rectangle) to other positions in the editor.

See also

- Chapter 1.3.5.3.2 “Positioning elements, adapting size and layer” on page 1113

Element property ‘Center’

The properties contain fixed values for the coordinates of the point of rotation. This point of rotation is shown as the (Blue) symbol. The point is used as the center for rotating and scaling.

<table>
<thead>
<tr>
<th>“X”</th>
<th>X-coordinate of the point of rotation</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>“Y”</th>
<th>Y-coordinate of the point of rotation</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

You can also change the values by dragging the symbols (Blue) to other positions in the editor.

Element property ‘Colors’

The properties contain fixed values for setting colors.
<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Color”</td>
<td>Color for the element in its normal state. Please note that the normal state is in effect if the expression in the “Color variables ➔ Toggle color” property is not defined or it has the value FALSE.</td>
</tr>
<tr>
<td>“Alarm color”</td>
<td>Color for the element in alarm state. Please note that the alarm state is in effect if the expression in the “Color variables ➔ Toggle color” property has the value TRUE.</td>
</tr>
<tr>
<td>“Transparency”</td>
<td>Value (0 to 255) for defining the transparency of the selected color. Example 255: The color is opaque. 0: The color is completely transparent.</td>
</tr>
<tr>
<td>“Use gradient color”</td>
<td>![ ]: The element is displayed with a color gradient.</td>
</tr>
<tr>
<td>“Gradient setting”</td>
<td>The “Color gradient editor” dialog box opens.</td>
</tr>
</tbody>
</table>

See also
- ![ ] Chapter 1.3.5.19.3.5 “Dialog ‘Gradient Editor’” on page 1586

**Element property 'Image’**

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Static ID”</td>
<td>Reference to an image in an image pool of the format &lt;name of image pool&gt; &lt;image ID&gt; (example: image_pool.GreenButton). If the image is from the “GlobalImagePool”, then you can omit the name of the image pool because CODESYS always searches this pool first. ![ ]: The “Input Assistant” dialog box opens and lists all available image pools and images in the entire project.</td>
</tr>
<tr>
<td>“Scale type”</td>
<td>Behavior of the image when resizing the button.</td>
</tr>
</tbody>
</table>
|             | - “Isotropic”: The image retains its proportions. The ratio of height to width is retained, even if you change the height or width of the button separately.  
|             | - “Anisotropic”: The image resizes to the dimensions of the button.                                                                        
|             | - “Fixed”: The image retains its original size, even if you change the size of the button.                                                |
| “Transparency”| The visualization displays the image with the transparency color that is selected in “Transparency color”.                                           |
| “Transparency color”| Color that is transparent in the image (example: “White”). If the image background that is reference by “Static ID” is white, then this background is displayed transparent. Clicking ![ ] opens a color selection dialog. Requirement: The “Transparency” option is activated. |
| “Horizontal alignment”| Horizontal alignment of the image                                                                                                        |
|             | - “Left”  
|             | - “Centered”  
|             | - “Right” |
| “Vertical alignment”| Vertical alignment of the image                                                                                                           |
|             | - “Top”  
|             | - “Centered”  
|             | - “Bottom” |

**Element property 'Texts’**

The properties contains character strings for labeling the element. The character string can also contain a placeholder with a format definition. In runtime mode, the placeholder is replaced by the current value in the specified format.
CODESYS accepts the specified texts automatically into the “GlobalTextList” text list. Therefore, these texts can be localized.

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Text”</td>
<td>Character string (without single straight quotation marks) for the labeling the element. Add a line break by pressing the keyboard shortcut [Ctrl] + [Enter]. Example: Accesses: %i The variable that contains the current value for the placeholder is specified in the property “Text variable ➤ Text”.</td>
</tr>
<tr>
<td>“Tooltip”</td>
<td>Character string (without single straight quotation marks) that is displayed as the tooltip of an element. Example: Number of valid accesses. The variable that contains the current value for the placeholder is specified in the property “Text variable ➤ Tooltip”.</td>
</tr>
</tbody>
</table>

See also
- “Element property Text variables” on page 1229
- Chapter 1.3.5.18.2 “Placeholders with format definition in the output text” on page 1546

The properties contain fixed values for the text properties.

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Horizontal alignment”</td>
<td>Horizontal alignment of the text within the element.</td>
</tr>
<tr>
<td>“Vertical alignment”</td>
<td>Vertical alignment of the text within the element.</td>
</tr>
</tbody>
</table>
| “Text format” | Definition for displaying texts that are too long
  - “Default”: The long text is truncated.
  - “Line break”: The text is split into parts.
  - “Ellipsis”: The visible text ends with “...” indicating that it is not complete. |
| “Font” | Example: “Default”
  - The “Font” dialog box opens.
  - : Drop-down list with style fonts. |
| “Font color” | Example: “Black”
  - The “Color” dialog box opens.
  - : Drop-down list with style colors. |
| “Transparency” | Whole number (value range from 0 to 255). This determines the transparency of the respective color.
  - Example: 255: The color is opaque.
  - 0: The color is completely transparent.
  - Please note: If the color is a style color and already has a transparency value, then this property is write-protected. |

The properties contain IEC variables for controlling the position of the element dynamically. The reference point is the upper left corner of the element. In runtime mode, the entire element is moved.
“Movement”

<table>
<thead>
<tr>
<th>“X”</th>
<th>Variable (numeric data type). Defines the X position (in pixels).</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example:</td>
<td>PLC_PRG.iPos_X.</td>
</tr>
<tr>
<td></td>
<td>Increasing this value in runtime mode moves the element to the right.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>“Y”</th>
<th>Variable (numeric data type). Defines the Y position (in pixels).</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example:</td>
<td>PLC_PRG.iPos_Y.</td>
</tr>
<tr>
<td></td>
<td>Increasing this value in runtime mode moves the element downwards.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>“Rotation”</th>
<th>Variable (numeric data type). Defines the angle of rotation (in degrees).</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example:</td>
<td>PLC_PRG.iAngle1.</td>
</tr>
<tr>
<td></td>
<td>The midpoint of the element rotates at the “Center” point. This rotation point is shown as the + symbol.</td>
</tr>
<tr>
<td></td>
<td>In runtime mode, the alignment of the element remains the same with respect to the coordinate system of the visualization. Increasing the value rotates the element to the right.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>“Scaling”</th>
<th>Variable (integer data type). Causes centric stretching.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example:</td>
<td>PLC_PRG.iScaling.</td>
</tr>
<tr>
<td></td>
<td>The reference point is the “Center” property.</td>
</tr>
<tr>
<td></td>
<td>The value 1 shrinks the element by a factor of 0.001. The value 1000 returns the element to its original size.</td>
</tr>
</tbody>
</table>

You can link the variables to a unit conversion.

The properties “X”, “Y”, “Rotation”, and “Interior rotation” are supported by the “Client Animation” functionality.

See also
- Unit conversion

Element property ‘Relative movement’

The properties contains variables for moving the element. The reference point is the position of the element (“Position” property). The shape of the element can change.

<table>
<thead>
<tr>
<th>“Movement top-left”</th>
<th>“X”</th>
<th>Variable (integer data type). It contains the number (in pixels) that the left edge is moved horizontally. Incrementing the value moves the element to the right.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example:</td>
<td>PLC_PRG.iDeltaX</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Y”</td>
<td>Variable (integer data type). It contains the number (in pixels) that the top edge is moved vertically. Incrementing the value moves the element to the down.</td>
<td>PLC_PRG.iDeltaY</td>
</tr>
<tr>
<td>“Movement bottom-right”</td>
<td></td>
<td></td>
</tr>
<tr>
<td>“X”</td>
<td>Variable (integer data type). It contains the number (in pixels) that the right edge is moved horizontally. Incrementing the value moves the element to the right.</td>
<td>PLC_PRG.iDeltaWidth</td>
</tr>
<tr>
<td>“Y”</td>
<td>Variable (integer data type). It contains the number (in pixels) that the bottom edge is moved vertically. Incrementing the value moves the element to the down.</td>
<td>PLC_PRG.iDeltaHeight</td>
</tr>
</tbody>
</table>

See also
- “Element property ‘Absolute movement’” on page 1227

**Element property ‘Text variables’**

These properties are variables with contents that replace a format definition.

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Text variable”</td>
<td>Variable (data type compliant with the format definition). It contains what is printed instead of the format definition.</td>
<td>PLC_PRG.iAccesses</td>
</tr>
<tr>
<td></td>
<td>Note: The format definition is part of the text in the property “Texts ➔ Text”.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Note: If you specify a variable of type enumeration with text list support, then the name of the enumeration data type is added automatically in angle brackets after the variable name. Example: PLC_PRG.enVar &lt;enumeration name&gt;. Then the symbolic value of the enumeration component is printed instead of the numeric value when text is printed. Refer to the help page for the enumerations.</td>
<td></td>
</tr>
<tr>
<td>“Tooltip variable”</td>
<td>Variable (data type compliant with the format definition). It contains what is printed instead of the format definition.</td>
<td>PLC_PRG.iAccessesInTooltip</td>
</tr>
<tr>
<td></td>
<td>Note: The format definition is part of the text in the property “Texts ➔ Tooltip”.</td>
<td></td>
</tr>
</tbody>
</table>

See also
- Chapter 1.3.5.18.2 “Placeholders with format definition in the output text” on page 1546
- “Element property ‘Texts’” on page 1226
- Enumerations

**Element property ‘Dynamic texts’**

Dynamic texts are variably indexed texts of a text list. At runtime, the text is displayed that is currently indexed in the variable.
| **“Text list”** | Variable (string) or name of the text list as a fixed string in single straight quotation marks.  
Example: 'Errorlist'  
▼: Drop-down list with the dialogs available in the text lists. |
| **“Text index”** | Text list ID. This refers to the desired output text.  
- As fixed string with the ID in single straight quotation marks.  
  Example: '1'  
- As a variable (STRING) for dynamically controlling the text output.  
  Example: strTextID  
  Sample assignment: PLC_PRG.strTextID := '1'; |
| **“Tooltip index”** | Text list ID. This refers to the desired output text.  
- As fixed string with the ID in single straight quotation marks.  
  Example: '2'  
- As a variable (STRING) for dynamically controlling the text output.  
  Example: strToolTipID  
  Sample assignment: PLC_PRG.strToolTipID := '2'; |

See also  
- *Text list*

**Element property "Font variables"**  
The variables allow for dynamic control of the text display.

| **“Font name”** | Variable (STRING). Includes the font of the text.  
Example: PLC_PRG.stFontVar := 'Arial';  
The selection of fonts corresponds to the default “Font” dialog. |
| **“Size”** | Variable (numeric data type). Contains the font size (in pixels or points). The applied unit is specified in brackets after the variable name.  
- <pt>: Points (default)  
  Example: PLC_PRG.iFontHeight <pt>  
  Code: iFontHeight : INT := 12;  
- <px>: Pixels  
  Example: PLC_PRG.iFontHeight <px>  
  Code: iFontHeight : INT := 19; |

If you click in the value field, a drop-down list opens on the right for setting the unit.  

Hint: The font size is specified in points (example: Arial 12). Use points when the variable font size should match a font, for example if a font is set in the property “Text property ➔ Font”.

---

2020/12/10

3ADR010583, 1, en_US
### Flags
Variable (DWORD). Contains the flags for displaying fonts.

Flags:
- 1: Italics
- 2: Bold
- 4: Underline
- 8: Strikethrough

Note: You can combine the font displays by adding the coding of the flags. For example, a bold and underlined text: `PLC_PRG.dwFontType := 6;`  

### Character set
Variable (DWORD). Contains a character set number for the font.

The selection of character set numbers corresponds to the “Script” setting of the standard “Font” dialog.

### Color
Variable (DWORD). Includes the color of the text.

Example: `PLC_PRG.dwColorFont := 16#FF000000;`

### Flags for text alignment
Variable (integer data type). Contains the coding for text alignment.

Example: `PLC_PRG.dwTextAlignment.`  

Coding:
- 0: Top left
- 1: Horizontal center
- 2: Right
- 4: Vertical center
- 8: Bottom

Note: You can combine the text alignments by adding the coding of the flags. For example, a vertical and horizontal centered text: `PLC_PRG.dwFontType := 5;`  

---

Fixed values for displaying texts are set in “Text properties”.

See also
- ☰ “Element property ‘Text properties’” on page 1227

**Element property ‘Colorvariables’**
The Element property is used as an interface for project variables to dynamically control colors at runtime.
### “Toggle color”

The property controls the toggled color at runtime.

#### Value assignment:
- **FALSE**: The element is displayed with the color specified in the “Color” property.
- **TRUE**: The element is displayed with the color specified in the “Alarm color” property.

#### Assigning the property:
- Placeholder for the user input variable
  - “<toggle/tap variable>”
  - “<NOT toggle/tap variable>”

  The color change is not controlled by its own variable, but by a user input variable.
  
  Note: Specify a variable for the mouse events “Tap” or “Toggle” in the input configuration of the element. Only then is the placeholder set. If you configure a variable in both “Toggle” and “Tap”, then the variable specified in “Tap” is used.

  Hint: Click the symbol to insert the placeholder “<toggle/tap variable>”. When you activate the “Inputconfiguration”, “Tap FALSE” property, then the “<NOT toggle/tap variable>” placeholder is displayed.

- Instance path of a project variable (BOOL)
  
  Example: PLC_PRG.xColorIsToggeled
  
  Note: In the code, declare and implement the variable specified here. Its value assignment determines when the color changes.

### “Color”

Color variable for the frame

- **Variable (DWORD)** for the color
  
  Example: PLC_PRG.dwColor

- Color literal
  
  Example of gray and opaque: 16#FF888888

**Requirement:** “Show frame” property is activated.

Please note that the normal state is in effect if the expression in the “Colorvariables ➔ Toggle color” property is not defined or it has the value FALSE.

### “Alarm color”

Color variable for the frame in alarm state

- **Variable (DWORD)** for the alarm color
  
  Example: PLC_PRG.dwAlarmColor

- Color literal
  
  Example of red and opaque: 16#FFFF0000

Please note that the alarm state is in effect if the expression in the “Colorvariables ➔ Toggle color” property has the value TRUE.

---

**The transparency part of the color value is evaluated only if the “Activate semi-transparent drawing” option of the visualization manager is selected.**

**Select the “Advanced” option in the toolbar of the properties view. Then all element properties are visible.**
**Element property 'State variables'**

The variables control the element behavior dynamically.

<table>
<thead>
<tr>
<th>“Invisible”</th>
<th>Variable (BOOL). Toggles the visibility of the element.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>TRUE: The element is not visible at runtime.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>“Deactivate inputs”</th>
<th>Variable (BOOL). Toggles the operability of the element.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>TRUE: User inputs do not have any effect in runtime more. The element is shown as deactivated.</td>
</tr>
</tbody>
</table>

The “Invisible” property is supported by the "Client Animation" functionality.

**Element property 'Button state variable'**

At runtime, the property controls whether the button is displayed as pressed or not.

**Values:**

- FALSE: The button is displayed as not pressed.
- TRUE: The button is displayed as pressed.

**Argument passed to the property:**

- Placeholder for the user input variable to couple the representation of the button with the input variable.
  - "<toggle/tap variable>"
  - "<NOT toggle/tap variable>"

  Note: Specify a variable for the mouse events “Tap” or “Toggle” in the input configuration of the button. Only then is the placeholder set. If you configure a variable in both “Toggle” and “Tap”, then the variable specified in “Tap” is used.

  Hint: Click the symbol ![insert symbol] to insert the placeholder “<toggle/tap variable>”. When you activate the “Inputconfiguration”, “Tap FALSE” property, then the “<NOT toggle/tap variable>” placeholder is displayed.

- Instance path of a project variable (BOOL) Example: prgA.xButtonState

  Note: Implement a value assignment in the code for the variable specified here.
Element property 'Image ID variable'

<table>
<thead>
<tr>
<th>&quot;Image ID&quot;</th>
<th>Variable (STRING). Contains the image ID. The contents of the string correspond to the description of the &quot;Static ID&quot; property.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Example: PLC_PRG.stImageID := 'ImagePool_A.Image3';</td>
</tr>
</tbody>
</table>

See also

- § Chapter 1.3.5.18.1.5 “Visualization element 'Image'” on page 1272
- § Object 'Image Pool'

These properties are available only when you select the “Preview: Support client animations and overlay of native elements” option in the Visualization Manager.

You can animate a movement, a rotation, and the visibility of a visualization element.

<table>
<thead>
<tr>
<th>&quot;Animation duration&quot;</th>
<th>Time that the element executes an animation (in milliseconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Example: 500</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>&quot;Move to the foreground&quot;</th>
<th>Property value (BOOL)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>TRUE: At runtime, the element is displayed in the foreground.</td>
</tr>
<tr>
<td></td>
<td>FALSE: At runtime, the element is displayed in the layer where it was inserted in the visualization editor.</td>
</tr>
</tbody>
</table>

See also

- § Chapter 1.3.5.21.5 “Using client animation” on page 1962

Element property 'Input configuration'

The properties contain the configurations for the user input when using the mouse or keyboard. A user input defines an event and one or more actions that are executed when an event occurs.

The “Configure” button opens the “Input configuration” dialog box for creating or modifying a user input.

Configured user inputs are briefly listed below the events. They each include the action that is triggered and the setting in short form.

Example: "Execute ST code": PLC_PRG.i_x := 0;

| "OnDialogClosed" | Input event: The user closes the dialog box. |
| "OnMouseClick"   | Input event: The user clicks the mouse button completely. The mouse button is clicked and released. |
| "OnMouseDown"    | Input event: The user clicks down on the mouse button. |
| "OnMouseEnter"   | Input event: The user drags the mouse pointer to the element. |
| "OnMouseLeave"   | Input event: The user drags the mouse pointer away from the element. |
| "OnMouseMove"    | Input event: The user moves the mouse pointer over the element area. |
| "OnMouseUp"      | Input event: The user releases the mouse button over the element area. |
**“Tap”**

When a mouse click event occurs, the variable defined in “Variable” is described in the application. The coding depends on the options “Tap FALSE” and “Tap on enter if captured”.

**“Variable”**

Variable (BOOL) that is set on mouse click.

Example: PLC_PRG.bIsTapped

TRUE: A mouse click event exists. It lasts while the user presses the mouse button over the element. It ends when the button is released.

FALSE: A mouse click event does not exist.

Requirement: The “Tap FALSE” option is not activated.

**“Tap FALSE”**

☑️: The mouse click event leads to a complementary value in “Variable”.

TRUE: A mouse click event does not exist.

FALSE: While the mouse click event exists.

**“Tap on enter if captured”**

☑️: During user input, it is also taken into consideration whether the mouse pointer is dragged within the element area or not while the mouse button is pressed.

TRUE: While the mouse click event exists and the mouse pointer is moved over the element area.

FALSE: A mouse click event does not exist. Or the user moves the mouse pointer outside of the element area while the mouse button is pressed.

The value is TRUE again as soon as the user moves the pointer back to the element area. The mouse is then captured.

**“Shift”**

With the onset of a mouse click event, the variable is set; when the mouse click event is completed, the variable is reset.

**“Variable”**

Variable (BOOL). Its value toggled when the mouse click event is ended. This is when the user releases the mouse button while the mouse pointer is over the element area.

If the user releases the mouse button while the mouse pointer is outside of the element area, then the mouse click event is not ended and the value is not toggled.

Tip: The user can cancel a started toggle input by dragging the mouse pointer out of the element area.

**“Toggle on up if captured”**

☑️: The value toggles regardless of where the mouse pointer is when the mouse button is released. The mouse is then captured.

**“Keyboard shortcuts”**

Keyboard shortcut on the element for triggering specific input actions.

When the keyboard shortcut event occurs, the input actions in the “Event(s)” property are triggered. In this way, it is not the input action itself that leads to this input action, but the mouse input action.

**“Key”**

Key pressed for input action.

Example: [T]

Note: The following properties appear when a key is selected.
### Event(s)
- "None"
- "Mouse down": Pressing the key triggers the input actions that are configured in the "OnMouseDown" property.
- "Mouse up": Releasing the key triggers the input actions that are configured in the "OnMouseUp" property.
- "Mouse down/up": Pressing and releasing the key triggers the input actions that are configured in the "OnMouseDown" property and the "OnMouseUp" property.

### Shift
- Combination with the Shift key
- Example: [Shift]+[T].

### Control
- Combination with the Ctrl key
- Example: [Ctrl]+[T].

### Alt
- Combination with the Alt key
- Example: [Alt]+[T].

All keyboard shortcuts and their actions that are configured in the visualization are listed in the "Keyboard configuration" tab.

### Access rights
Requirement: User management is set up for the visualization.

- Opens the "Access rights" dialog. There you can edit the access privileges for the element.

<table>
<thead>
<tr>
<th>Access rights</th>
</tr>
</thead>
<tbody>
<tr>
<td>Status messages:</td>
</tr>
<tr>
<td>- &quot;Not set. Full rights.&quot;: Access rights for all user groups: &quot;operable&quot;</td>
</tr>
<tr>
<td>- &quot;Rights are set: Limited rights&quot;: Access is restricted for at least one group.</td>
</tr>
</tbody>
</table>

See also
- Chapter 1.3.5.19.2.2 “Command 'Keyboard Configuration’” on page 1558
- Chapter 1.3.5.19.3.6 “Dialog ‘Input Configuration’” on page 1586

### Visualization element ‘Group box’
Symbol:

Tag: “Common controls”
The element provides visual grouping of visualization elements. The group box can be nested many times.
You can also use drag & drop to add elements to a group box. To do this, simply drag the element to the group box. The appearance of the cursor changes (a small plus sign is shown). When you click the [Shift] key at the same time, the element is not added.

You can remove elements from the group box by dragging them out of the group box.

Element properties

<table>
<thead>
<tr>
<th>“Element name”</th>
<th>Example: Parameter axis 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Optional</td>
<td></td>
</tr>
<tr>
<td>Hint: Assign individual names for elements so that they are found faster in the element list.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>“Type of element”</th>
<th>“Group box”</th>
</tr>
</thead>
</table>

| “Clipping” | Elements are clipped that protrude beyond the size of the group box. |

Element property 'Position'

The position defines the location and size of the element in the visualization window. These are based on the Cartesian coordinate system. The origin is located at the upper left corner of the window. The positive horizontal x-axis runs to the right. The positive vertical y-axis runs downwards.

<table>
<thead>
<tr>
<th>“X”</th>
<th>X coordinate of the upper left corner of the element</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Specified in pixels.</td>
</tr>
<tr>
<td></td>
<td>Example: 10.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>“Y”</th>
<th>Y coordinate of the upper left corner of the element</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Specified in pixels.</td>
</tr>
<tr>
<td></td>
<td>Example: 10.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>“Width”</th>
<th>Specified in pixels.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Example: 150</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>“Height”</th>
<th>Specified in pixels.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Example: 30</td>
</tr>
</tbody>
</table>

You can also change the values by dragging the box symbols (_drag_ ) to other positions in the editor.

See also

- Chapter 1.3.5.3.2 “Positioning elements, adapting size and layer” on page 1113

Element property 'Center'

The properties contain fixed values for the coordinates of the point of rotation. This point of rotation is shown as the _center_ symbol. The point is used as the center for rotating and scaling.
The properties contain character strings for labeling the element.

CODESYS accepts the specified texts automatically into the "GlobalTextList" text list. Therefore, these texts can be localized.

<table>
<thead>
<tr>
<th>Text</th>
<th>Character string (without single straight quotation marks) for the labeling the element. Example: Axis 1.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tooltip</td>
<td>Character string (without single straight quotation marks) that is displayed as the tooltip of an element. Example: Parameters of Axis 1.</td>
</tr>
</tbody>
</table>

See also

The properties contain fixed values for the text properties.

<table>
<thead>
<tr>
<th>Font</th>
<th>Example: &quot;Default&quot;</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>☐: The &quot;Font&quot; dialog box opens.</td>
</tr>
<tr>
<td></td>
<td>▼: Drop-down list with style fonts.</td>
</tr>
<tr>
<td>Font color</td>
<td>Example: &quot;Black&quot;</td>
</tr>
<tr>
<td></td>
<td>☐: The &quot;Color&quot; dialog box opens.</td>
</tr>
<tr>
<td></td>
<td>▼: Drop-down list with style colors.</td>
</tr>
<tr>
<td>Transparency</td>
<td>Whole number (value range from 0 to 255). This determines the transparency of the respective color. Example: 255: The color is opaque. 0: The color is completely transparent. Please note: If the color is a style color and already has a transparency value, then this property is write-protected.</td>
</tr>
</tbody>
</table>

The properties contain IEC variables for controlling the position of the element dynamically. The reference point is the upper left corner of the element. In runtime mode, the entire element is moved.
### Movement

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
<th>Example</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>X</strong></td>
<td>Variable (numeric data type). Defines the X position (in pixels).</td>
<td>PLC_PRG.iPos_X</td>
<td>Increasing this value in runtime mode moves the element to the right.</td>
</tr>
<tr>
<td><strong>Y</strong></td>
<td>Variable (numeric data type). Defines the Y position (in pixels).</td>
<td>PLC_PRG.iPos_Y</td>
<td>Increasing this value in runtime mode moves the element downwards.</td>
</tr>
</tbody>
</table>

### Rotation

Variable (numeric data type). Defines the angle of rotation (in degrees).

Example: PLC_PRG.iAngle1.

The midpoint of the element rotates at the “Center” point. This rotation point is shown as the symbol.

In runtime mode, the alignment of the element remains the same with respect to the coordinate system of the visualization. Increasing the value rotates the element to the right.

### Scaling

Variable (integer data type). Causes centric stretching.

Example: PLC_PRG.iScaling.

The reference point is the “Center” property.

The value 1 shrinks the element by a factor of 0.001. The value 1000 returns the element to its original size.

### Interior rotation

Variable (numeric data type). Defines the angle of rotation (in degrees).

Example: PLC_PRG.iAngle2.

In runtime mode, the element rotates about the point of rotation specified in “Center” according to the value of the variable. In addition, the alignment of the element rotates according to the coordinate system of the visualization. Increasing the value in the code rotates clockwise.

The rotation point is shown as the symbol.

Note: If a static angle of rotation is specified in the property “Position ➔ Angle”, then the static angle of rotation is added to the variable angle of rotation (offset) when the visualization is executed.

You can link the variables to a unit conversion.

The properties “X”, “Y”, “Rotation”, and “Interior rotation” are supported by the “Client Animation” functionality.
### Element property 'State variables'

The variables control the element behavior dynamically.

<table>
<thead>
<tr>
<th>Variable Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Invisible”</td>
<td>Variable (BOOL). Toggles the visibility of the element. TRUE: The element is not visible at runtime.</td>
</tr>
<tr>
<td>“Deactivate inputs”</td>
<td>Variable (BOOL). Toggles the operability of the element. TRUE: User inputs do not have any effect in runtime more. The element is shown as deactivated.</td>
</tr>
</tbody>
</table>

The “Invisible” property is supported by the "Client Animation" functionality.

These properties are available only when you select the “Preview: Support client animations and overlay of native elements” option in the Visualization Manager.

You can animate a movement, a rotation, and the visibility of a visualization element.

<table>
<thead>
<tr>
<th>Property Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Animation duration”</td>
<td>Time that the element executes an animation (in milliseconds) Example: 500</td>
</tr>
<tr>
<td>“Move to the foreground”</td>
<td>Property value (BOOL) TRUE: At runtime, the element is displayed in the foreground. FALSE: At runtime, the element is displayed in the layer where it was inserted in the visualization editor.</td>
</tr>
</tbody>
</table>

See also
- Chapter 1.3.5.21.5 “Using client animation” on page 1962

### Visualization element 'Table'

Symbol:

Tag: “Common controls”
The element displays data that can be represented as an array in a table. Therefore, the data type of the visualizing variable can be 1) a one-dimensional array, 2) a maximum two-dimensional array, 3) an array of an array, 4) an array of structures, or 5) an array of a function block.

**Element properties**

<table>
<thead>
<tr>
<th>“Element name”</th>
<th>Example: Data set component 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Optional</td>
<td>Hint: Assign individual names for elements so that they are found faster in the element list.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>“Type of element”</th>
<th>Table</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>“Data array”</th>
<th>Array whose data is visualized as a table</th>
</tr>
</thead>
<tbody>
<tr>
<td>Variable (ARRAY) whose data type determines the number of columns and rows in the table</td>
<td></td>
</tr>
<tr>
<td>Array types</td>
<td></td>
</tr>
<tr>
<td>● One-dimensional array: The table has one column.</td>
<td></td>
</tr>
<tr>
<td>● Two-dimensional array: The second dimension determines the number of columns.</td>
<td></td>
</tr>
<tr>
<td>● Array of an array: The number of array elements of the back array determines the number of columns.</td>
<td></td>
</tr>
<tr>
<td>● Array of a structure: The number of structure members determines the number of columns.</td>
<td></td>
</tr>
<tr>
<td>● Array of a function block: The number of local variables determines the number of columns.</td>
<td></td>
</tr>
<tr>
<td>Example: PLC_PRG.aiTable</td>
<td></td>
</tr>
<tr>
<td>Declaration: aiTable : ARRAY[0..3, 0..4] OF INT := [4(1, 2, 3, 4, 5)];</td>
<td></td>
</tr>
<tr>
<td>Hint: If the declaration of the array changes, then the table can be refreshed by placing the cursor in the data array value field and pressing the [Enter] key.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>“Max. array index”</th>
<th>Top index limit for the displayed table. Limits the number of displayed rows. The index begins at 0.</th>
</tr>
</thead>
<tbody>
<tr>
<td>● Variable (integer data type)</td>
<td></td>
</tr>
<tr>
<td>Example: PLC_PRG.iUpperIndexBoundToDisplay</td>
<td></td>
</tr>
<tr>
<td>● Integer literal</td>
<td></td>
</tr>
<tr>
<td>Example: 4 is displayed as 5 in the row of the table.</td>
<td></td>
</tr>
</tbody>
</table>

See also

- Data Type ‘ARRAY’

**Element property 'Columns'**
The table element shows the values of a variable in a tabular view. The array elements of structure members are shown in a column or in a row. Two-dimensional arrays or arrays of a structure are shown in multiple columns. The visualized variable is defined in the “Data array” property. When a variable is assigned there, you can specify the display of the table columns where the array elements are shown. An individual configuration is possible for each column that is assigned to an index [<n>].

<table>
<thead>
<tr>
<th>“Show row header”</th>
<th>☑: The row header is visible.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example: For an array, the index of the array element is displayed in the header.</td>
<td></td>
</tr>
</tbody>
</table>

| “Show column header” | ☑: The column label is visible. |
Table 350: “Element property 'Columns: Column [<n>]’”

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Column header”</td>
<td>By default, the name of the array or structure is applied as the heading with the index or structure member for the column. If an array of a function block has been selected for “Data array”, then the name of the array is applied to the column header with the local variables of the function block that belong to the column. The column label can be changed here by specifying a new title.</td>
</tr>
<tr>
<td>“Width”</td>
<td>Column width (in pixels)</td>
</tr>
<tr>
<td>“Image column”</td>
<td>Images can be displayed in the column. Images are used from the global image pool or custom image pools. The image IDs are shown in the cells of the table as they are defined in the image pool.</td>
</tr>
<tr>
<td>“Image configuration”</td>
<td></td>
</tr>
</tbody>
</table>
| “Fill mode”                    |  ● Fill cell: The image resizes to the dimensions of the cell without fixing the height/width ratio.  
  ● Centered: The image is centered in the cell and retains its proportions (height/width ratio).                                      |
| “Transparency”                 |  [✓]: The color which is specified in “Transparent color” is displayed as transparent.                                                           |
| “Transparent color”            | This color is displayed as transparent. Requirement: The “Transparency” property is activated.                                                |
| “Text alignment of header”     | Alignment of the column header:  
  ● Left  
  ● Centered  
  ● Right                                                                                                                                  |
| “Use template”                 |  [✓]: Another visualization element (type “Rectangle”, “Round rectangle”, or “Ellipse”) is inserted into each line of this table column. The properties list is extended automatically with the properties of this element in “Template”. |
| “Text alignment of the headline from the template” | Requirement: The “Use template” property is activated.  
  [✓]: When activated, the settings for font (size) and alignment in the inserted template are also applied to the column header. |
| “Template”                     | Requirement: The “Use template” property is activated.  
  The properties of all elements assigned to the column are listed in “Template”. They can be modified there as described in "Rectangle, Round rectangle and Ellipse” |

See also

- § Chapter 1.3.5.18.1.1 “Visualization element 'Rectangle', 'Round rectangle', 'Ellipse’” on page 1224

**Element property 'Position'**

The position defines the location and size of the element in the visualization window. These are based on the Cartesian coordinate system. The origin is located at the upper left corner of the window. The positive horizontal x-axis runs to the right. The positive vertical y-axis runs downwards.
### Element property 'Center'

The properties contain fixed values for the coordinates of the point of rotation. This point of rotation is shown as the symbol. The point is used as the center for rotating and scaling.

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>X</strong></td>
<td>X-coordinate of the point of rotation</td>
</tr>
<tr>
<td><strong>Y</strong></td>
<td>Y-coordinate of the point of rotation</td>
</tr>
</tbody>
</table>

You can also change the values by dragging the symbols (-symbol) to other positions in the editor.

### Element property 'Text properties'

The properties contain fixed values for the text properties.

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Horizontal alignment</strong></td>
<td>Horizontal alignment of the text within the element.</td>
</tr>
<tr>
<td><strong>Vertical alignment</strong></td>
<td>Vertical alignment of the text within the element.</td>
</tr>
<tr>
<td><strong>Font</strong></td>
<td>Example: “Default”&lt;br&gt; elevate: The “Font” dialog box opens. &lt;br&gt; ▼: Drop-down list with style fonts.</td>
</tr>
</tbody>
</table>
### Font color

Example: “Black”

- The “Color” dialog box opens.
- Drop-down list with style colors.

### Transparency

Whole number (value range from 0 to 255). This determines the transparency of the respective color.

Example: 255: The color is opaque.
0: The color is completely transparent.

Please note: If the color is a style color and already has a transparency value, then this property is write-protected.

---

**Element property 'Dynamic texts'**

Dynamic texts are variably indexed texts of a text list. At runtime, the text is displayed that is currently indexed in the variable.

<table>
<thead>
<tr>
<th><strong>Text list</strong></th>
<th>Variable (string) or name of the text list as a fixed string in single straight quotation marks.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example: 'Errorlist'</td>
<td>➣: Drop-down list with the dialogs available in the text lists.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Text index</strong></th>
<th>Text list ID. This refers to the desired output text.</th>
</tr>
</thead>
<tbody>
<tr>
<td>● As fixed string with the ID in single straight quotation marks.</td>
<td>Example: '1'</td>
</tr>
<tr>
<td>● As a variable (STRING) for dynamically controlling the text output.</td>
<td>Example: strTextID</td>
</tr>
<tr>
<td>Sample assignment: PLC_PRG.strTextID := '1';</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Tooltip index</strong></th>
<th>Text list ID. This refers to the desired output text.</th>
</tr>
</thead>
<tbody>
<tr>
<td>● As fixed string with the ID in single straight quotation marks.</td>
<td>Example: '2'</td>
</tr>
<tr>
<td>● As a variable (STRING) for dynamically controlling the text output.</td>
<td>Example: strToolTipID</td>
</tr>
<tr>
<td>Sample assignment: PLC_PRG.strToolTipID := '2';</td>
<td></td>
</tr>
</tbody>
</table>

See also

- **Text list**

---

**Element property 'Font variables'**

The variables enable dynamic control of the text display.

<table>
<thead>
<tr>
<th><strong>Font name</strong></th>
<th>Variable (STRING). Includes the font of the text.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example: PLC_PRG.stFontVar := 'Arial';</td>
<td>The selection of fonts corresponds to the default “Font” dialog box.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Size</strong></th>
<th>Variable (integer data type). Contains the font size (in pixels).</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example: PLC_PRG.iFontHeight := 16;</td>
<td>The selection of font sizes corresponds to the default “Font” dialog box.</td>
</tr>
</tbody>
</table>
**"Flags"**
Variable (DWORD). Contains the flags for displaying fonts.

Flags:
- 1: Italic
- 2: Bold
- 4: Underline
- 8: Strikethrough

Note: You can combine the font displays by adding the coding of the flags. For example, a bold and underlined text:
```
PLC_PRG.dwFontType := 6;
```

**"Charset"**
Variable (DWORD). Contains a character set number for the font.
The selection of character set numbers corresponds to the "Script" setting of the standard "Font" dialog box.

**"Color"**
Variable (DWORD). Includes the color of the text.
Example: 
```
PLC_PRG.dwColorFont := 16#FF000000;
```

**"Flags for text alignment"**
Variable (integer data type). Contains the coding for text alignment.

Coding:
- 0: Top left
- 1: Horizontal center
- 2: Right
- 4: Vertical center
- 8: Bottom

Note: You can combine the text alignments by adding the coding of the flags. For example, a vertical and horizontal centered text:
```
PLC_PRG.dwFontType := 5;
```

---

**Fixed values for displaying texts are set in "Text properties".**

See also
- "Element property 'Text properties'" on page 1227

---

**Element property 'Absolute movement'**
The properties contain IEC variables for controlling the position of the element dynamically. The reference point is the upper left corner of the element. In runtime mode, the entire element is moved.

**"Movement"**

**"X"**
Variable (numeric data type). Defines the X position (in pixels).
Example: 
```
PLC_PRG.iPos_X.
```
Increasing this value in runtime mode moves the element to the right.

**"Y"**
Variable (numeric data type). Defines the Y position (in pixels).
Example: 
```
PLC_PRG.iPos_Y.
```
Increasing this value in runtime mode moves the element downwards.
**“Rotation”**
Variable (numeric data type). Defines the angle of rotation (in degrees).

Example: PLC_PRG.iAngle1.

The midpoint of the element rotates at the “Center” point. This rotation point is shown as the symbol.

In runtime mode, the alignment of the element remains the same with respect to the coordinate system of the visualization. Increasing the value rotates the element to the right.

**“Interior rotation”**
Variable (numeric data type). Defines the angle of rotation (in degrees).

Example: PLC_PRG.iAngle2.

In runtime mode, the element rotates about the point of rotation specified in “Center” according to the value of the variable. In addition, the alignment of the element rotates according to the coordinate system of the visualization. Increasing the value in the code rotates clockwise.

The rotation point is shown as the symbol.

Note: If a static angle of rotation is specified in the “Position Angle” property, then the static angle of rotation is added to the variable angle of rotation (offset) when the visualization is executed.

---

You can link the variables to a unit conversion.

The “X”, “Y”, “Rotation”, and “Interior rotation” properties are supported by the “Client Animation” functionality.

See also
- Unit conversion
- 

---

**Element property ‘State variables’**
The variables control the element behavior dynamically.

<table>
<thead>
<tr>
<th>“Invisible”</th>
<th>Variable (BOOL). Toggles the visibility of the element. TRUE: The element is not visible at runtime.</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Deactivate inputs”</td>
<td>Variable (BOOL). Toggles the operability of the element. TRUE: User inputs do not have any effect in runtime more. The element is shown as deactivated.</td>
</tr>
</tbody>
</table>
The "Invisible" property is supported by the "Client Animation" functionality.

See also

- "Background color on selection" Fill color of the selected row.
- "Selection font color" Font color of the selected row.
- "Selection type" Selection when clicking the table row.
  - No selection: No selection
  - Cell selection: The clicked cell only.
  - Row selection: Row of the clicked cell.
  - Column selection: Column of the clicked cell.
  - Row and column selection: Row and column of the clicked cell.
- "Frame around selected cells" A frame is drawn around the selected cells.
- "Variable for selected column" Variable (INT). Contains the array index of the "Column" of the selected cell. If the data array points to a structure, then the structure components are indexed, starting at 0.
  Warning: This index represents the correct position in the array only if no columns have been removed from the table in the display.
- "Variable for selected row" Variable (INT). Contains the array index of the "Row" of the selected cell.
- "Variable for valid column selection" Variable (BOOL).
  TRUE: The "Variable for selected column" variable contains a valid value.
- "Variable for valid row selection" Variable (BOOL).
  TRUE: The "Variable for selected row" variable contains a valid value.

These properties are available only when you select the "Preview: Support client animations and overlay of native elements" option in the Visualization Manager.

You can animate a movement, a rotation, and the visibility of a visualization element.

- "Animation duration" Time that the element executes an animation (in milliseconds)
  Example: 500
- "Move to the foreground" Property value (BOOL)
  TRUE: At runtime, the element is displayed in the foreground.
  FALSE: At runtime, the element is displayed in the layer where it was inserted in the visualization editor.

See also

- "Chapter 1.3.5.21.5 "Using client animation" on page 1962"
Requirement: User management is set up for the visualization.

| **"Access rights"** | Opens the "Access rights" dialog. There you can edit the access privileges for the element. Status messages: ● "Not set. Full rights.": Access rights for all user groups: "operable" ● "Rights are set: Limited rights": Access is restricted for at least one group. |

See also
● "Chapter 1.3.5.19.3.1 "Dialog 'Access Rights''" on page 1583

See also
●
● "Chapter 1.3.5.9.1 "Displaying structured variable values in tables" on page 1155
● *Data Type 'ARRAY'*

Visualization element 'Text field'

Symbol:

Tag: "Common controls"
The element is used for the following:
● Static output of text. The contents of a variable can be part of the text.
● Showing a tooltip. The text is managed as static text and can also be defined so that the contents of a variable are also displayed.
● Dynamic output of text. Texts of a text list are displayed dynamically.
● Input of text. For example, a user can input a number or a text literal.

See also
●

Element properties

| **"Element name"** | Optional
| Example: FileName_A
| Hint: Assign individual names for elements so that they are found faster in the element list. |
| **"Type of element"** | "Text field"

The position defines the location and size of the element in the visualization window. These are based on the Cartesian coordinate system. The origin is located at the upper left corner of the window. The positive horizontal x-axis runs to the right. The positive vertical y-axis runs downwards.
<table>
<thead>
<tr>
<th><strong>X</strong></th>
<th>X coordinate of the upper left corner of the element</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Specified in pixels.</td>
</tr>
<tr>
<td></td>
<td>Example: 10.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Y</strong></th>
<th>Y coordinate of the upper left corner of the element</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Specified in pixels.</td>
</tr>
<tr>
<td></td>
<td>Example: 10.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Width</strong></th>
<th>Specified in pixels.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Example: 150</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Height</strong></th>
<th>Specified in pixels.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Example: 30</td>
</tr>
</tbody>
</table>

You can also change the values by dragging the box symbols (_drag_ ) to other positions in the editor.

See also

- Chapter 1.3.5.3.2 “Positioning elements, adapting size and layer” on page 1113

**Element property 'Colors'**

<table>
<thead>
<tr>
<th><strong>Normal state</strong></th>
<th>The normal state is in effect if the variable in “Color variables ➔ Toggle color” is not defined or it has the value FALSE.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Frame color</strong></td>
<td>Frame and fill color for the corresponding state of the variable.</td>
</tr>
<tr>
<td><strong>Fill color</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Transparency</strong></td>
<td>Transparency value (0 to 255) for defining the transparency of the selected color. Example: 255: The color is opaque. 0: The color is completely transparent.</td>
</tr>
<tr>
<td><strong>Alarm state</strong></td>
<td>The alarm state is in effect if the variable in “Color variables ➔ Toggle color” has the value TRUE.</td>
</tr>
</tbody>
</table>

See also

- Chapter 1.3.5.19.3.5 “Dialog ‘Gradient Editor’” on page 1586

**Element property 'Appearance'**

The properties contain fixed values for setting the look of the element.
**“Line width”**  
Value in pixels  
Example: 2  
Note: The values 0 and 1 both result in a line weight of 1 pixel. If no line should be displayed, then the “Line style” property must be set to the option “Invisible”.

**“Fill attributes”**  
The way in which the element is filled.  
- “Filled”: The element is filled with the color from property “Colors ➔ Fill color”.  
- “Invisible”: The fill color is invisible.

**“Line style”**  
The type of line representation  
- “Solid”  
- “Dashes”  
- “Dots”  
- “Dash Dot”  
- “Dash Dot Dot”  
- “not visible”

You can assign variables in the “Appearance variables” property for controlling the appearance dynamically. The fixed values here are overwritten.

See also  
- “Element property ‘Appearance variables’” on page 1285

**Element property 'Texts’**  
The properties contain character strings for labeling the element. The character string can also contain a placeholder with a format definition. In runtime mode, the placeholder is replaced by the current value in the specified format.  
CODESYS accepts the specified texts automatically into the “GlobalTextList” text list. Therefore, these texts can be localized.

**“Text”**  
Character string (without single straight quotation marks) for the labeling the element. Add a line break by pressing the keyboard shortcut [Ctrl] + [Enter].  
Example: Accesses: %i  
The variable that contains the current value for the placeholder is specified in the property “Text variable ➔ Text”.

**“Tooltip”**  
Character string (without single straight quotation marks) that is displayed as the tooltip of an element.  
Example: Number of valid accesses.  
The variable that contains the current value for the placeholder is specified in the property “Text variable ➔ Tooltip”.

See also  
- “Element property ‘Text variables’” on page 1229
- Chapter 1.3.5.18.2 “Placeholders with format definition in the output text” on page 1546

**Element property 'Text properties’**  
The properties contain fixed values for the text properties.
### Element property 'Text variables'

These properties are variables with contents that replace a format definition.

<table>
<thead>
<tr>
<th><strong>Text variable</strong></th>
<th>Variable (data type compliant with the format definition). It contains what is printed instead of the format definition.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example: PLC_PRG.iAccesses</td>
<td>Note: The format definition is part of the text in the property “Texts ➔ Text”. Note: If you specify a variable of type enumeration with text list support, then the name of the enumeration data type is added automatically in angle brackets after the variable name. Example: PLC_PRG.enVar &lt;enumeration name&gt;. Then the symbolic value of the enumeration component is printed instead of the numeric value when text is printed. Refer to the help page for the enumerations.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Tooltip variable</strong></th>
<th>Variable (data type compliant with the format definition). It contains what is printed instead of the format definition.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example: PLC_PRG.iAccessesTooltip</td>
<td>Note: The format definition is part of the text in the property “Texts ➔ Tooltip”.</td>
</tr>
</tbody>
</table>

See also

- % Chapter 1.3.5.18.2 “Placeholders with format definition in the output text” on page 1546
- % “Element property ‘Texts’” on page 1226
- Enumerations

### Element property 'Dynamic texts'

Dynamic texts are variably indexed texts of a text list. At runtime, the text is displayed that is currently indexed in the variable.
### "Text list"

Variable (string) or name of the text list as a fixed string in single straight quotation marks.

Example: 'Errorlist'

- Drop-down list with the dialogs available in the text lists.

### "Text index"

Text list ID. This refers to the desired output text.

- As fixed string with the ID in single straight quotation marks.
  - Example: '1'
- As a variable (STRING) for dynamically controlling the text output.
  - Example: \(\text{strTextID}\)
  - Sample assignment: \(\text{PLC_PRG.strTextID} := '1';\)

### "Tooltip index"

Text list ID. This refers to the desired output text.

- As fixed string with the ID in single straight quotation marks.
  - Example: '2'
- As a variable (STRING) for dynamically controlling the text output.
  - Example: \(\text{strToolTipID}\)
  - Sample assignment: \(\text{PLC_PRG.strToolTipID} := '2';\)

#### See also
- "Text list"

---

### Element property 'Font variables'

The variables allow for dynamic control of the text display.

#### "Font name"

Variable (STRING). Includes the font of the text.

- Example: \(\text{PLC_PRG.stFontVar} := 'Arial';\)

The selection of fonts corresponds to the default "Font" dialog.

#### "Size"

Variable (numeric data type). Contains the font size (in pixels or points). The applied unit is specified in brackets after the variable name.

- \(<\text{pt}>\) Points (default)
  - Example: \(\text{PLC_PRG.iFontHeight} <\text{pt}>\)
  - Code: \(\text{iFontHeight} : \text{INT} := 12;\)
- \(<\text{px}>\) Pixels
  - Example: \(\text{PLC_PRG.iFontHeight} <\text{px}>\)
  - Code: \(\text{iFontHeight} : \text{INT} := 19;\)

If you click in the value field, a drop-down list opens on the right for setting the unit.

Hint: The font size is specified in points (example: Arial 12). Use points when the variable font size should match a font, for example if a font is set in the property "Text property ➔ Font".
### "Flags"

Variable (DWORD). Contains the flags for displaying fonts.

Flags:
- 1: Italics
- 2: Bold
- 4: Underline
- 8: Strikethrough

Note: You can combine the font displays by adding the coding of the flags. For example, a bold and underlined text: PLC_PRG.dwFontType := 6;

### "Character set"

Variable (DWORD). Contains a character set number for the font.

The selection of character set numbers corresponds to the “Script” setting of the standard “Font” dialog.

### "Color"

Variable (DWORD). Includes the color of the text.

Example: PLC_PRG.dwColorFont := 16#FF000000;

### "Flags for text alignment"

Variable (integer data type). Contains the coding for text alignment.

Example: PLC_PRG.dwTextAlignment.

Coding:
- 0: Top left
- 1: Horizontal center
- 2: Right
- 4: Vertical center
- 8: Bottom

Note: You can combine the text alignments by adding the coding of the flags. For example, a vertical and horizontal centered text: PLC_PRG.dwFontType := 5;

---

**Fixed values for displaying texts are set in "Text properties".**

See also
- "Element property 'Text properties'" on page 1227

**Element property 'Colorvariables'**

The Element property is used as an interface for project variables to dynamically control colors at runtime.
"Toggle color"  The property controls the toggled color at runtime.  
**Value assignment:**  
- **FALSE**: The element is displayed with the color specified in the “Color” property.  
- **TRUE**: The element is displayed with the color specified in the “Alarm color” property.  

**Assignment options:**  
- Placeholder for the user input variable  
  – “<toggle/tap variable>”  
  – “<NOT toggle/tap variable>”  

The color change is not controlled by its own variable, but by a user input variable.  

Note: Specify a variable for the mouse events “Tap” or “Toggle” in the input configuration of the element. Only then is the placeholder set. If you configure a variable in both “Toggle” and “Tap”, then the variable specified in “Tap” is used.  

Hint: Click the symbol to insert the placeholder “<toggle/tap variable>”. When you activate the “Input configuration”, “Tap FALSE” property, then the “<NOT toggle/tap variable>” placeholder is displayed.  

- Instance path of a project variable (BOOL)  
  Example: PLC_PRG.xColorIsToggled  

Note: In the code, declare and implement the variable specified here. Its value assignment determines when the color changes.

"Normal state"  "Alarm state"  The properties listed below control the color depending on the state. The normal state is in effect if the variable in "Color variables", "Toggle color" is not defined or it has the value FALSE. The alarm state is in effect if the variable in "Color variables", "Toggle color" has the value TRUE.

"Frame color"  **Assignment options:**  
- Variable (DWORD) for the frame color  
  Example: PLC_PRG.dwBorderColor  
- Color literal  
  Example of green and opaque: 16#FF00FF00

"Filling color"  **Assignment options:**  
- Variable (DWORD) for the fill color  
  Example: PLC_PRG.dwFillColor  
- Color literal  
  Example of gray and opaque: 16#FF888888

The transparency part of the color value is evaluated only if the “Activate semi-transparent drawing” option of the visualization manager is selected.

Select the “Advanced” option in the toolbar of the properties view. Then all element properties are visible.

See also  
- Chapter 1.3.5.8.3 “Animating a color display” on page 1152
**Element property 'Absolute movement'**

The properties contain IEC variables for controlling the position of the element dynamically. The reference point is the upper left corner of the element. In runtime mode, the entire element is moved.

### "Movement"

| **"X"** | Variable (numeric data type). Defines the X position (in pixels).  
Example: PLC_PRG.iPos_X.  
Increasing this value in runtime mode moves the element to the right. |
| **"Y"** | Variable (numeric data type). Defines the Y position (in pixels).  
Example: PLC_PRG.iPos_Y.  
Increasing this value in runtime mode moves the element downwards. |

| **"Rotation"** | Variable (numeric data type). Defines the angle of rotation (in degrees).  
Example: PLC_PRG.iAngle1.  
The midpoint of the element rotates at the “Center” point. This rotation point is shown as the symbol.  
In runtime mode, the alignment of the element remains the same with respect to the coordinate system of the visualization. Increasing the value rotates the element to the right. |
| **"Interior rotation"** | Variable (numeric data type). Defines the angle of rotation (in degrees).  
Example: PLC_PRG.iAngle2.  
In runtime mode, the element rotates about the point of rotation specified in “Center” according to the value of the variable. In addition, the alignment of the element rotates according to the coordinate system of the visualization. Increasing the value in the code rotates clockwise.  
The rotation point is shown as the symbol.  
Note: If a static angle of rotation is specified in the "Position ➔ Angle" property, then the static angle of rotation is added to the variable angle of rotation (offset) when the visualization is executed. |

---

*You can link the variables to a unit conversion.*

*The "X", "Y", "Rotation", and "Interior rotation" properties are supported by the "Client Animation" functionality.*
The variables control the element behavior dynamically.

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Invisible”</td>
<td>Variable (BOOL). Toggles the visibility of the element. TRUE: The element is not visible at runtime.</td>
</tr>
<tr>
<td>“Deactivate inputs”</td>
<td>Variable (BOOL). Toggles the operability of the element. TRUE: User inputs do not have any effect in runtime more. The element is shown as deactivated.</td>
</tr>
</tbody>
</table>

The “Invisible” property is supported by the “Client Animation” functionality.

The variables control the element behavior dynamically.

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Caret position”</td>
<td>Variable (integer data type). Contains the position of the caret.</td>
</tr>
<tr>
<td>“Selection start”</td>
<td>Variable (integer data type). Contains the position of the first selected character. Example: PLC_PRG.iSelStart</td>
</tr>
<tr>
<td>“Selection end”</td>
<td>Variable (integer data type). Contains the position of the last selected character. Example: PLC_PRG.iSelEnd</td>
</tr>
<tr>
<td>“All selected”</td>
<td>Variable (BOOL). Toggles the selection of the entered text. TRUE: The text in the text field is selected. FALSE: The selection starts with the value in “Selection start” and ends with “Selection end”.</td>
</tr>
</tbody>
</table>

These properties are available only when you select the “Preview: Support client animations and overlay of native elements” option in the Visualization Manager.

You can animate a movement, a rotation, and the visibility of a visualization element.

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Animation duration”</td>
<td>Time that the element executes an animation (in milliseconds) Example: 500</td>
</tr>
<tr>
<td>“Move to the foreground”</td>
<td>Property value (BOOL) TRUE: At runtime, the element is displayed in the foreground. FALSE: At runtime, the element is displayed in the layer where it was inserted in the visualization editor.</td>
</tr>
</tbody>
</table>
The properties contain the configurations for the user input when using the mouse or keyboard. A user input defines an event and one or more actions that are executed when an event occurs.

Example: "Execute ST code": \$ PLC_PRG.i_x := 0;

<table>
<thead>
<tr>
<th>Input event</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;OnDialogClosed&quot;</td>
<td>Input event: The user closes the dialog box.</td>
</tr>
<tr>
<td>&quot;OnMouseClick&quot;</td>
<td>Input event: The user clicks the mouse button completely. The mouse button is clicked and released.</td>
</tr>
<tr>
<td>&quot;OnMouseDown&quot;</td>
<td>Input event: The user clicks down on the mouse button.</td>
</tr>
<tr>
<td>&quot;OnMouseEnter&quot;</td>
<td>Input event: The user drags the mouse pointer to the element.</td>
</tr>
<tr>
<td>&quot;OnMouseLeave&quot;</td>
<td>Input event: The user drags the mouse pointer away from the element.</td>
</tr>
<tr>
<td>&quot;OnMouseMove&quot;</td>
<td>Input event: The user moves the mouse pointer over the element area.</td>
</tr>
<tr>
<td>&quot;OnMouseUp&quot;</td>
<td>Input event: The user releases the mouse button over the element area.</td>
</tr>
</tbody>
</table>

"Tap" When a mouse click event occurs, the variable defined in "Variable" is described in the application. The coding depends on the options "Tap FALSE" and "Tap on enter if captured".

"Variable" Variable (BOOL) that is set on mouse click.
Example: PLC_PRG.bIsTapped
TRUE: A mouse click event exists. It lasts while the user presses the mouse button over the element. It ends when the button is released.
FALSE: A mouse click event does not exist.

Requirement: The "Tap FALSE" option is not activated.

"Tap FALSE" ☑: The mouse click event leads to a complementary value in "Variable".
TRUE: A mouse click event does not exist.
FALSE: While the mouse click event exists.

"Tap on enter if captured" ☑: During user input, it is also taken into consideration whether the mouse pointer is dragged within the element area or not while the mouse button is pressed.
TRUE: While the mouse click event exists and the mouse pointer is moved over the element area.
FALSE: A mouse click event does not exist. Or the user moves the mouse pointer outside of the element area while the mouse button is pressed.
The value is TRUE again as soon as the user moves the pointer back to the element area. The mouse is then captured.
### “Shift”
With the onset of a mouse click event, the variable is set; when the mouse click event is completed, the variable is reset.

### “Variable”
Variable (BOOL). Its value toggled when the mouse click event is ended. This is when the user releases the mouse button while the mouse pointer is over the element area.

If the user releases the mouse button while the mouse pointer is outside of the element area, then the mouse click event is not ended and the value is not toggled.

Tip: The user can cancel a started toggle input by dragging the mouse pointer out of the element area.

### “Toggle on up if captured”
☑️: The value toggles regardless of where the mouse pointer is when the mouse button is released. The mouse is then captured.

### “Keyboard shortcuts”
Keyboard shortcut on the element for triggering specific input actions.

When the keyboard shortcut event occurs, the input actions in the “Event(s)” property are triggered. In this way, it is not the input action itself that leads to this input action, but the mouse input action.

### “Key”
Key pressed for input action.

Example: [T]

Note: The following properties appear when a key is selected.

### “Event(s)”
- “None”
- “Mouse down”: Pressing the key triggers the input actions that are configured in the “OnMouseDown” property.
- “Mouse up”: Releasing the key triggers the input actions that are configured in the “OnMouseUp” property.
- “Mouse down/up”: Pressing and releasing the key triggers the input actions that are configured in the “OnMouseDown” property and the “OnMouseUp” property.

### “Shift”
☑️: Combination with the Shift key

Example: [Shift]+[T].

### “Control”
☑️: Combination with the Ctrl key

Example: [Ctrl]+[T].

### “Alt”
☑️: Combination with the Alt key

Example: [Alt]+[T].

---

All keyboard shortcuts and their actions that are configured in the visualization are listed in the “Keyboard configuration” tab.

See also
- Chapter 1.3.5.19.2.2 “Command ‘Keyboard Configuration’” on page 1558
- Chapter 1.3.5.19.3.6 “Dialog ‘Input Configuration’” on page 1586

---

Element property 'Access rights'

Requirement: User management is set up for the visualization.
“Access rights” Opens the “Access rights” dialog. There you can edit the access privileges for the element.

Status messages:
- “Not set. Full rights.”: Access rights for all user groups: “operable”
- “Rights are set: Limited rights”: Access is restricted for at least one group.

See also
- Chapter 1.3.5.19.3.1 “Dialog ‘Access Rights’” on page 1583

Visualization element ‘Scrollbar’

Symbol:

Tag: “Common controls”
The element sets the value of a variable, depending on the position of the scrollbar.

Element properties

<table>
<thead>
<tr>
<th>“Element name”</th>
<th>Example: Speed Conveyor Belt 1</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Optional</td>
</tr>
<tr>
<td></td>
<td>Hint: Assign individual names for elements so that they are found faster in the element list.</td>
</tr>
</tbody>
</table>

| “Type of element” | “Scrollbar” |

Element property ‘Center’
The properties contain fixed values for the coordinates of the point of rotation. This point of rotation is shown as the symbol. The point is used as the center for rotating and scaling.

<table>
<thead>
<tr>
<th>“X”</th>
<th>X-coordinate of the point of rotation</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Y”</td>
<td>Y-coordinate of the point of rotation</td>
</tr>
</tbody>
</table>

You can also change the values by dragging the symbols (⊕) to other positions in the editor.

<table>
<thead>
<tr>
<th>“Value”</th>
<th>Variable as type integer that includes the position of the scrollbar.</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Minimum value”</td>
<td>Smallest value of the scrollbar (fixed value or variable).</td>
</tr>
<tr>
<td>“Maximum value”</td>
<td>Largest value of the scrollbar (fixed value or variable).</td>
</tr>
<tr>
<td><strong>“Page size”</strong></td>
<td>Page size</td>
</tr>
<tr>
<td>----------------</td>
<td>-----------</td>
</tr>
<tr>
<td>● As a fixed value, for example 10</td>
<td></td>
</tr>
<tr>
<td>● As a variable of data type integer</td>
<td></td>
</tr>
<tr>
<td>Requirement: Visible when the “Move to click” property is <strong>not</strong> selected.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>“Move to click”</strong></th>
<th>Behavior of the scrollbar at visualization runtime when it is clicked:</th>
</tr>
</thead>
<tbody>
<tr>
<td>✓: The scrollbar moves to the clicked position.</td>
<td></td>
</tr>
<tr>
<td>□: The scrollbar moves to one “Page size” in the direction of the click.</td>
<td></td>
</tr>
</tbody>
</table>

**Element property ‘Position’**

The position defines the location and size of the element in the visualization window. These are based on the Cartesian coordinate system. The origin is located at the upper left corner of the window. The positive horizontal x-axis runs to the right. The positive vertical y-axis runs downwards.

<table>
<thead>
<tr>
<th><strong>“X”</strong></th>
<th>X coordinate of the upper left corner of the element</th>
</tr>
</thead>
<tbody>
<tr>
<td>Specified in pixels.</td>
<td></td>
</tr>
<tr>
<td>Example: 10.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>“Y”</strong></th>
<th>Y coordinate of the upper left corner of the element</th>
</tr>
</thead>
<tbody>
<tr>
<td>Specified in pixels.</td>
<td></td>
</tr>
<tr>
<td>Example: 10.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>“Width”</strong></th>
<th>Specified in pixels.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example: 150</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>“Height”</strong></th>
<th>Specified in pixels.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example: 30</td>
<td></td>
</tr>
</tbody>
</table>

You can also change the values by dragging the box symbols (▲) to other positions in the editor.

See also

- Chapter 1.3.5.3.2 “Positioning elements, adapting size and layer” on page 1113

**Element property ‘Bar’**

The property defines the representation of scaling and direction of travel.
| **Orientation** | Alignment of the slider; defined by the ratio of width to height.  
  ● “Horizontal”  
  ● “Vertical”  
  You can modify the alignment in the visualization editor by using the pointing device to adjust the width and height of the scrollbar. |
| **Running direction** | The drop-down list varies depending on the alignment of the slider.  
  Horizontal  
  ● “Left to right”: Scale starts at the left.  
  ● “Right to left”: Scale starts at the right.  
  Vertical  
  ● “Bottom to top”: Scale starts at the bottom.  
  ● “Top to bottom”: Scale starts at the top. |

**Element property 'Colors'**  
The properties contain fixed values for setting colors.

| **Color** | Color for the element in its normal state.  
  Please note that the normal state is in effect if the expression in the “Color variables ➔ Toggle color” property is not defined or it has the value FALSE. |
| **Alarm color** | Color for the element in alarm state.  
  Please note that the alarm state is in effect if the expression in the “Color variables ➔ Toggle color” property has the value TRUE. |
| **Transparency** | Value (0 to 255) for defining the transparency of the selected color.  
  Example 255: The color is opaque. 0: The color is completely transparent. |

**Element property 'Texts'**  
The properties contains character strings for labeling the element. The character string can also contain a placeholder with a format definition. In runtime mode, the placeholder is replaced by the current value in the specified format.

CODESYS accepts the specified texts automatically into the “GlobalTextList” text list. Therefore, these texts can be localized.

| **Text** | Character string (without single straight quotation marks) for the labeling the element. Add a line break by pressing the keyboard shortcut [Ctrl] + [Enter].  
  Example: Accesses: %i  
  The variable that contains the current value for the placeholder is specified in the property “Text variable ➔ Text”. |
| **Tooltip** | Character string (without single straight quotation marks) that is displayed as the tooltip of an element.  
  Example: Number of valid accesses.  
  The variable that contains the current value for the placeholder is specified in the property “Text variable ➔ Tooltip” |
The properties contain fixed values for the text properties.

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>“Horizontal alignment”</strong></td>
<td>Horizontal alignment of the text within the element.</td>
</tr>
<tr>
<td><strong>“Vertical alignment”</strong></td>
<td>Vertical alignment of the text within the element.</td>
</tr>
<tr>
<td><strong>“Font”</strong></td>
<td>Example: “Default”&lt;br&gt;The “Font” dialog box opens.&lt;br&gt;▼: Drop-down list with style fonts.</td>
</tr>
<tr>
<td><strong>“Font color”</strong></td>
<td>Example: “Black”&lt;br&gt;The “Color” dialog box opens.&lt;br&gt;▼: Drop-down list with style colors.</td>
</tr>
<tr>
<td><strong>“Transparency”</strong></td>
<td>Whole number (value range from 0 to 255). This determines the transparency of the respective color. Example: 255: The color is opaque. 0: The color is completely transparent. Please note: If the color is a style color and already has a transparency value, then this property is write-protected.</td>
</tr>
</tbody>
</table>

These properties are variables with contents that replace a format definition.

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>“Text variable”</strong></td>
<td>Variable (data type compliant with the format definition). It contains what is printed instead of the format definition. Example: PLC_PRG.iAccesses</td>
</tr>
<tr>
<td></td>
<td>Note: The format definition is part of the text in the property “Texts ➔ Text”. Note: If you specify a variable of type enumeration with text list support, then the name of the enumeration data type is added automatically in angle brackets after the variable name. Example: PLC_PRG.enVar &lt;enumeration name&gt;. Then the symbolic value of the enumeration component is printed instead of the numeric value when text is printed. Refer to the help page for the enumerations.</td>
</tr>
<tr>
<td><strong>“Tooltip variable”</strong></td>
<td>Variable (data type compliant with the format definition). It contains what is printed instead of the format definition. Example: PLC_PRG.iAccessesInTooltip</td>
</tr>
<tr>
<td></td>
<td>Note: The format definition is part of the text in the property “Texts ➔ Tooltip”.</td>
</tr>
</tbody>
</table>

See also
- "Element property ‘Text variables’" on page 1229
- Chapter 1.3.5.18.2 “Placeholders with format definition in the output text” on page 1546
- "Element property ‘Texts’" on page 1226
- Enumerations
Dynamic texts are variably indexed texts of a text list. At runtime, the text is displayed that is currently indexed in the variable.

**Text list**
Variable (string) or name of the text list as a fixed string in single straight quotation marks.
Example: 'Errorlist'
↓: Drop-down list with the dialogs available in the text lists.

**Text index**
Text list ID. This refers to the desired output text.
- As fixed string with the ID in single straight quotation marks.
  Example: '1'
- As a variable (STRING) for dynamically controlling the text output.
  Example: strTextID
  Sample assignment: PLC_PRG.strTextID := '1';

**Tooltip index**
Text list ID. This refers to the desired output text.
- As fixed string with the ID in single straight quotation marks.
  Example: '2'
- As a variable (STRING) for dynamically controlling the text output.
  Example: strToolTipID
  Sample assignment: PLC_PRG.strToolTipID := '2';

See also
- Text list

The variables allow for dynamic control of the text display.

**Font name**
Variable (STRING). Includes the font of the text.
Example: PLC_PRG.stFontVar := 'Arial';
The selection of fonts corresponds to the default “Font” dialog.

**Size**
Variable (numeric data type). Contains the font size (in pixels or points). The applied unit is specified in brackets after the variable name.
- <pt>: Points (default)
  Example: PLC_PRG.iFontHeight <pt>
  Code: iFontHeight : INT := 12;
- <px>: Pixels
  Example: PLC_PRG.iFontHeight <px>
  Code: iFontHeight : INT := 19;

If you click in the value field, a drop-down list opens on the right for setting the unit.

Hint: The font size is specified in points (example: Arial 12). Use points when the variable font size should match a font, for example if a font is set in the property “Text property ➔ Font”.

Element property 'Dynamic texts'

Element property 'Font variables'
"Flags" Variable (DWORD). Contains the flags for displaying fonts.

Flags:
- 1: Italics
- 2: Bold
- 4: Underline
- 8: Strikethrough

Note: You can combine the font displays by adding the coding of the flags. For example, a bold and underlined text: PLC_PRG.dwFontType := 6;

"Character set" Variable (DWORD). Contains a character set number for the font.

The selection of character set numbers corresponds to the "Script" setting of the standard "Font" dialog.

"Color" Variable (DWORD). Includes the color of the text.

Example: PLC_PRG.dwColorFont := 16#FF000000;

"Flags for text alignment" Variable (integer data type). Contains the coding for text alignment.

Example: PLC_PRG.dwTextAlignment.

Coding:
- 0: Top left
- 1: Horizontal center
- 2: Right
- 4: Vertical center
- 8: Bottom

Note: You can combine the text alignments by adding the coding of the flags. For example, a vertical and horizontal centered text: PLC_PRG.dwFontType := 5;

Fixed values for displaying texts are set in "Text properties".

See also
- § "Element property 'Text properties'" on page 1227

Element property 'Color variables'

The Element property is used as an interface for project variables to dynamically control colors at runtime.
**“Toggle color”**

The property controls the toggled color at runtime.

**Value assignment:**
- **FALSE**: The element is displayed with the color specified in the “Color” property.
- **TRUE**: The element is displayed with the color specified in the “Alarm color” property.

**Assignment options:**
- Placeholder for the user input variable
  - `<toggle/tap variable>`
  - `<NOT toggle/tap variable>`

The color change is not controlled by its own variable, but by a user input variable.

Note: Specify a variable for the mouse events “Tap” or “Toggle” in the input configuration of the element. Only then is the placeholder set. If you configure a variable in both “Toggle” and “Tap”, then the variable specified in “Tap” is used.

Hint: Click the symbol to insert the placeholder “<toggle/tap variable>”. When you activate the “Input configuration”, “Tap FALSE” property, then the “<NOT toggle/tap variable>” placeholder is displayed.

- Instance path of a project variable (BOOL)
  Example: PLC_PRG.xColorIsToggeled

Note: In the code, declare and implement the variable specified here. Its value assignment determines when the color changes.

---

**“Normal state”**

**“Alarm state”**

The properties listed below control the color depending on the state. The normal state is in effect if the variable in “Color variables”, “Toggle color” is not defined or it has the value FALSE. The alarm state is in effect if the variable in “Color variables”, “Toggle color” has the value TRUE.

---

**“Frame color”**

**Assignment options:**
- Variable (DWORD) for the frame color
  Example: PLC_PRG.dwBorderColor
- Color literal
  Example of green and opaque: 16#FF00FF00

---

**“Filling color”**

**Assignment options:**
- Variable (DWORD) for the fill color
  Example: PLC_PRG.dwFillColor
- Color literal
  Example of gray and opaque: 16#FF888888

---

The transparency part of the color value is evaluated only if the “Activate semi-transparent drawing” option of the visualization manager is selected.

Select the “Advanced” option in the toolbar of the properties view. Then all element properties are visible.

See also
- § Chapter 1.3.5.8.3 “Animating a color display” on page 1152
### Element property 'Absolute movement'

The properties contain IEC variables for controlling the position of the element dynamically. The reference point is the upper left corner of the element. In runtime mode, the entire element is moved.

#### "Movement"

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>X</strong></td>
<td>Variable (numeric data type). Defines the X position (in pixels).</td>
<td>PLC_PRG.iPos_X.</td>
</tr>
<tr>
<td></td>
<td>Increasing this value in runtime mode moves the element to the right.</td>
<td></td>
</tr>
<tr>
<td><strong>Y</strong></td>
<td>Variable (numeric data type). Defines the Y position (in pixels).</td>
<td>PLC_PRG.iPos_Y.</td>
</tr>
<tr>
<td></td>
<td>Increasing this value in runtime mode moves the element downwards.</td>
<td></td>
</tr>
</tbody>
</table>

#### "Rotation"

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Rotation</strong></td>
<td>Variable (numeric data type). Defines the angle of rotation (in degrees).</td>
<td>PLC_PRG.iAngle1.</td>
</tr>
<tr>
<td></td>
<td>The midpoint of the element rotates at the &quot;Center&quot; point. This rotation point is shown as the symbol.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>In runtime mode, the alignment of the element remains the same with respect to the coordinate system of the visualization. Increasing the value rotates the element to the right.</td>
<td><img src="image1" alt="Diagram" /></td>
</tr>
</tbody>
</table>

#### "Interior rotation"

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Interior rotation</strong></td>
<td>Variable (numeric data type). Defines the angle of rotation (in degrees).</td>
<td>PLC_PRG.iAngle2.</td>
</tr>
<tr>
<td></td>
<td>In runtime mode, the element rotates about the point of rotation specified in &quot;Center&quot; according to the value of the variable. In addition, the alignment of the element rotates according to the coordinate system of the visualization. Increasing the value in the code rotates clockwise.</td>
<td><img src="image2" alt="Diagram" /></td>
</tr>
<tr>
<td></td>
<td>The rotation point is shown as the symbol.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Note: If a static angle of rotation is specified in the &quot;Position ➔ Angle&quot; property, then the static angle of rotation is added to the variable angle of rotation (offset) when the visualization is executed.</td>
<td></td>
</tr>
</tbody>
</table>

You can link the variables to a unit conversion.

The "X", "Y", "Rotation", and "Interior rotation" properties are supported by the "Client Animation" functionality.
The variables control the element behavior dynamically.

### Element property 'State variables'

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Invisible&quot;</td>
<td>Variable (BOOL). Toggles the visibility of the element. TRUE: The element is not visible at runtime.</td>
</tr>
<tr>
<td>&quot;Deactivate inputs&quot;</td>
<td>Variable (BOOL). Toggles the operability of the element. TRUE: User inputs do not have any effect in runtime more. The element is shown as deactivated.</td>
</tr>
</tbody>
</table>

The "Invisible" property is supported by the "Client Animation" functionality.

See also

- Unit conversion

---

These properties are available only when you select the "Preview: Support client animations and overlay of native elements" option in the Visualization Manager.

You can animate a movement, a rotation, and the visibility of a visualization element.

### Element property 'Access rights'

**Requirement:** User management is set up for the visualization.

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
</table>
| "Access rights"          | Opens the "Access rights" dialog. There you can edit the access privileges for the element. Status messages:  
  - "Not set. Full rights.": Access rights for all user groups : "operable"  
  - "Rights are set. Limited rights": Access is restricted for at least one group. |

See also

- ☞ Chapter 1.3.5.21.5 “Using client animation” on page 1962
- ☞ Chapter 1.3.5.19.3.1 “Dialog ‘Access Rights’” on page 1583
Visualization element 'Slider'
Symbol:

Tag: “Common controls”
The element adjusts the value of a variable, depending on the position of the slider within the slider bar. You define the value range of the slider bar by means of the scale start and scale end.

Element properties

<table>
<thead>
<tr>
<th>“Element name”</th>
<th>Example: Speed controller conveyor belt 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Optional</td>
<td>Hint: Assign individual names for elements so that they are found faster in the element list.</td>
</tr>
<tr>
<td>“Type of element”</td>
<td>“Slider”</td>
</tr>
</tbody>
</table>

Element property 'Position'
The position defines the location and size of the element in the visualization window. These are based on the Cartesian coordinate system. The origin is located at the upper left corner of the window. The positive horizontal x-axis runs to the right. The positive vertical y-axis runs downwards.

<table>
<thead>
<tr>
<th>“X”</th>
<th>X coordinate of the upper left corner of the element Specified in pixels. Example: 10.</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Y”</td>
<td>Y coordinate of the upper left corner of the element Specified in pixels. Example: 10.</td>
</tr>
<tr>
<td>“Width”</td>
<td>Specified in pixels. Example: 150</td>
</tr>
<tr>
<td>“Height”</td>
<td>Specified in pixels. Example: 30</td>
</tr>
</tbody>
</table>

You can also change the values by dragging the box symbols (_drag_box_1_) to other positions in the editor.

See also

- Chapter 1.3.5.3.2 “Positioning elements, adapting size and layer” on page 1113
**Element property 'Center'**

The properties contain fixed values for the coordinates of the point of rotation. This point of rotation is shown as the symbol. The point is used as the center for rotating and scaling.

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>“X”</td>
<td>X-coordinate of the point of rotation</td>
</tr>
<tr>
<td>“Y”</td>
<td>Y-coordinate of the point of rotation</td>
</tr>
</tbody>
</table>

You can also change the values by dragging the symbols (.isDirectory) to other positions in the editor.

**"Variable"**

Variable (numeric data type)

Example: PLC_PRG.rSlider

When executed, the variable assigns a value that corresponds to the position of the slider in the bar.

**"Page size"**

Page size

- As a fixed value, for example 10
- As an IEC variable of data type integer

Requirement: The "Move to click" element property is not activated.

**"Move to click"**

Function of the slider at visualization runtime when it is clicked:

- The slider moves to the clicked position.
- The slider moves to the value (defined in the "Page size" element property) in the direction of the click.

**Element property 'Scale'**

**"Show scale"**

☑: The element has a visible scale.

Note: This option is available for the slider only.

**"Scale start"**

Least value of the scale and the lower limit of the value range for the element.

Example: 0

☑: The property "Variable" is shown below.

**"Variable"**

Variable (integer data type). Contains the scale start.

Example: PLC_PRG.iScaleStart

Declaration:

```plaintext```
PROGRAM PLC_PRG
VAR
    iScaleStart : INT := 0;
END_VAR
```plaintext```

**"Scale end"**

Greatest value of the scale and the upper limit of the value range for the element.

Example: 100

☑: The property "Variable" is shown below.
| **“Variable”** | Variable (integer data type). Contains the scale end.  
**Example:** PLC_PRG.iScaleEnd  
**Declaration:** |
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
PROGRAM PLC_PRG  
VAR  
iScaleEnd : INT := 120;  
END_VAR  
| |
| **“Main scale”** | Distance between two tick marks on the rough scale.  
**Example:** 10  
★: The property “Variable” is shown below. |
| **“Variable”** | Variable (integer data type). Contains the distance.  
**Example:** PLC_PRG.iMainScale  
**Declaration:** |
|  
PROGRAM PLC_PRG  
VAR  
iMainScale : INT := 20;  
END_VAR  
| |
| **“Subscale”** | Distance between two dashes on the fine scale. You can hide the fine scale by setting the value to 0.  
**Example:** 2  
★: The property “Variable” is shown below. |
| **“Variable”** | Variable (integer data type). Contains the distance.  
**Example:** PLC_PRG.iSubScale  
**Declaration:** |
|  
PROGRAM PLC_PRG  
VAR  
iMainScale : INT := 5;  
END_VAR  
| |
| **“Scale format (C Syntax)”** | Formatting of the scale label (example: %d %s)  
Note: This property is available for the slider only. |
| **“Scale proportion”** | Size of the scale (in %) of the total size |

**Element property ‘Bar’**  
The property defines the representation of scaling and direction of travel.
### Diagram type
The drop-down list varies depending on the alignment of the diagram.

**Horizontal**
- **"Top":** Scale is above the slider.
- **"Bottom":** Scale is below the slider.
- **"Top and bottom":** Two scales frame the slider above and below.

**Vertical**
- **Left:** Scale is left of the slider.
- **Right:** Scale is right of the slider.
- **Left and right:** Two scales frame the slider on the left and the right.

### Orientation
Alignment of the slider; defined by the ratio of width to height.
- **"Horizontal"**
- **"Vertical"**

You can modify the alignment in the visualization editor by using the pointing device to adjust the width and height of the scrollbar.

### Running direction
The drop-down list varies depending on the alignment of the slider.

**Horizontal**
- **"Left to right":** Scale starts at the left.
- **"Right to left":** Scale starts at the right.

**Vertical**
- **"Bottom to top":** Scale starts at the bottom.
- **"Top to bottom":** Scale starts at the top.

### Movement
The properties contain IEC variables for controlling the position of the element dynamically. The reference point is the upper left corner of the element. In runtime mode, the entire element is moved.

**X**
Variable (numeric data type). Defines the X position (in pixels).

Example: `PLC_PRG.iPos_X`.

Increasing this value in runtime mode moves the element to the right.

**Y**
Variable (numeric data type). Defines the Y position (in pixels).

Example: `PLC_PRG.iPos_Y`.

Increasing this value in runtime mode moves the element downwards.

**Rotation**
Variable (numeric data type). Defines the angle of rotation (in degrees).

Example: `PLC_PRG.iAngle1`.

The midpoint of the element rotates at the "Center" point. This rotation point is shown as the 📡 symbol.

In runtime mode, the alignment of the element remains the same with respect to the coordinate system of the visualization. Increasing the value rotates the element to the right.
**"Interior rotation"**

Variable (numeric data type). Defines the angle of rotation (in degrees).

**Example:** PLC_PRG.iAngle2.

In runtime mode, the element rotates about the point of rotation specified in "Center" according to the value of the variable. In addition, the alignment of the element rotates according to the coordinate system of the visualization. Increasing the value in the code rotates clockwise.

The rotation point is shown as the ✡ symbol.

Note: If a static angle of rotation is specified in the "Position ➔ Angle" property, then the static angle of rotation is added to the variable angle of rotation (offset) when the visualization is executed.

---

**You can link the variables to a unit conversion.**

---

**The “X”, “Y”, “Rotation”, and “Interior rotation” properties are supported by the ”Client Animation“ functionality.**

**See also**

- Unit conversion

---

**Element property ‘State variables’**

The variables control the element behavior dynamically.

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>&quot;Invisible&quot;</strong></td>
<td>Variable (BOOL). Toggles the visibility of the element. TRUE: The element is not visible at runtime.</td>
</tr>
<tr>
<td><strong>&quot;Deactivate inputs&quot;</strong></td>
<td>Variable (BOOL). Toggles the operability of the element. TRUE: User inputs do not have any effect in runtime more. The element is shown as deactivated.</td>
</tr>
</tbody>
</table>

**The "Invisible" property is supported by the "Client Animation" functionality.**

**See also**

- 

These properties are available only when you select the "Preview: Support client animations and overlay of native elements" option in the Visualization Manager.

You can animate a movement, a rotation, and the visibility of a visualization element.
**“Animation duration”**
Time that the element executes an animation (in milliseconds)
Example: 500

**“Move to the foreground”**
Property value (BOOL)
TRUE: At runtime, the element is displayed in the foreground.
FALSE: At runtime, the element is displayed in the layer where it was inserted in the visualization editor.

See also
- *Chapter 1.3.5.21.5 “Using client animation” on page 1962*

**Element property 'Access rights'**
Requirement: User management is set up for the visualization.

**“Access rights”**
Opens the “Access rights” dialog. There you can edit the access privileges for the element.
Status messages:
- “Not set. Full rights.”: Access rights for all user groups: “operable”
- “Rights are set: Limited rights”: Access is restricted for at least one group.

See also
- *Chapter 1.3.5.19.3.1 “Dialog ‘Access Rights’” on page 1583*

**Visualization element 'SpinControl'**
Symbol:

Tag: “Common controls”
The element increments or decrements the value of a variable in defined intervals.

**Element properties**

<table>
<thead>
<tr>
<th><strong>“Element name”</strong></th>
<th>Example: Speed controller conveyor belt</th>
</tr>
</thead>
<tbody>
<tr>
<td>Optional</td>
<td>Hint: Assign individual names for elements so that they are found faster in the element list.</td>
</tr>
</tbody>
</table>

| **“Type of element”** | “SpinControl” |

**Element property ‘Position’**
The position defines the location and size of the element in the visualization window. These are based on the Cartesian coordinate system. The origin is located at the upper left corner of the window. The positive horizontal x-axis runs to the right. The positive vertical y-axis runs downwards.
<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
</table>
| **“X”** | X coordinate of the upper left corner of the element | Specified in pixels.  
Example: 10. |
| **“Y”** | Y coordinate of the upper left corner of the element | Specified in pixels.  
Example: 10. |
| **“Width”** | Specified in pixels. | Example: 150 |
| **“Height”** | Specified in pixels. | Example: 30 |

You can also change the values by dragging the box symbols (_drag-) to other positions in the editor.

See also

- Chapter 1.3.5.3.2 “Positioning elements, adapting size and layer” on page 1113

**Element property ‘Center’**

The properties contain fixed values for the coordinates of the point of rotation. This point of rotation is shown as the symbol. The point is used as the center for rotating and scaling.

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>“X”</strong></td>
<td>X-coordinate of the point of rotation</td>
</tr>
<tr>
<td><strong>“Y”</strong></td>
<td>Y-coordinate of the point of rotation</td>
</tr>
</tbody>
</table>

You can also change the values by dragging the symbols (drag) to other positions in the editor.

**“Variable”**

Variable (numeric data type)  
Beispiel PLC_PRG.iTemp

**“Number format”**

Format of the value in printf syntax  
Example: %d, %5.2f

**“Interval”**

Interval used for modification of the value

**Element property ‘Value range’**
### Minimum value
Lower limit of the output value
- fixed value
- Variable (INT)

### Maximum value
Upper limit of the output value
- fixed value
- Variable (INT)

**Element property 'Text properties'**

<table>
<thead>
<tr>
<th>Usage of</th>
<th>The properties contain fixed values for the text properties.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Usage of</strong></td>
<td></td>
</tr>
<tr>
<td>“Default style values”</td>
<td>The values of the visualization style are used.</td>
</tr>
<tr>
<td>“Individual settings”</td>
<td>The &quot;Individual settings&quot; property group is shown</td>
</tr>
<tr>
<td></td>
<td>The values can be customized here.</td>
</tr>
</tbody>
</table>

**Requirement:**

The “Individual settings” text property is defined.

<table>
<thead>
<tr>
<th>Horizontal alignment</th>
<th>Horizontal alignment of the text within the element.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Horizontal alignment</strong></td>
<td></td>
</tr>
<tr>
<td>Vertical alignment</td>
<td>Vertical alignment of the text within the element.</td>
</tr>
<tr>
<td><strong>Vertical alignment</strong></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Font</th>
<th>Example: “Default”</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>The “Font” dialog box opens.</td>
</tr>
<tr>
<td></td>
<td>▼: Drop-down list with style fonts.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Font color</th>
<th>Example: “Black”</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>The “Color” dialog box opens.</td>
</tr>
<tr>
<td></td>
<td>▼: Drop-down list with style colors.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Transparency</th>
<th>Whole number (value range from 0 to 255). This determines the transparency of the respective color.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Example: 255: The color is opaque.</td>
</tr>
<tr>
<td></td>
<td>0: The color is completely transparent.</td>
</tr>
<tr>
<td></td>
<td>Please note: If the color is a style color and already has a transparency value, then this property is write-protected.</td>
</tr>
</tbody>
</table>

**Element property 'Colorvariables'**

The Element property is used as an interface for project variables to dynamically control colors at runtime.
"Toggle color"
The property controls the toggled color at runtime.
Value assignment:
- **FALSE**: The element is displayed with the color specified in the "Color" property.
- **TRUE**: The element is displayed with the color specified in the "Alarm color" property.
Assigning the property:
- Placeholder for the user input variable
  - "<toggle/tap variable>"
  - "<NOT toggle/tap variable>"
  
  The color change is not controlled by its own variable, but by a user input variable.
Note: Specify a variable for the mouse events "Tap" or "Toggle" in the input configuration of the element. Only then is the placeholder set. If you configure a variable in both "Toggle" and "Tap", then the variable specified in "Tap" is used.
Hint: Click the symbol to insert the placeholder "<toggle/tap variable>". When you activate the "Inputconfiguration", "Tap FALSE" property, then the "<NOT toggle/tap variable>" placeholder is displayed.
- Instance path of a project variable (BOOL)
  
  Example: PLC_PRG.xColorIsToggeled
  
  Note: In the code, declare and implement the variable specified here. Its value assignment determines when the color changes.

The transparency part of the color value is evaluated only if the "Activate semi-transparent drawing" option of the visualization manager is selected.

Select the "Advanced" option in the toolbar of the properties view. Then all element properties are visible.

See also
- § Chapter 1.3.5.8.3 “Animating a color display” on page 1152

Element property 'Absolute movement'
The properties contain IEC variables for controlling the position of the element dynamically. The reference point is the upper left corner of the element. In runtime mode, the entire element is moved.

"Movement"

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;X&quot;</td>
<td>Variable (numeric data type). Defines the X position (in pixels). Example: PLC_PRG.iPos_X. Increasing this value in runtime mode moves the element to the right.</td>
</tr>
<tr>
<td>&quot;Y&quot;</td>
<td>Variable (numeric data type). Defines the Y position (in pixels). Example: PLC_PRG.iPos_Y. Increasing this value in runtime mode moves the element downwards.</td>
</tr>
</tbody>
</table>
**“Rotation”**

Variable (numeric data type). Defines the angle of rotation (in degrees).

**Example:** PLC_PRG.iAngle1.

The midpoint of the element rotates at the “Center” point. This rotation point is shown as the symbol.

In runtime mode, the alignment of the element remains the same with respect to the coordinate system of the visualization. Increasing the value rotates the element to the right.

**“Interior rotation”**

Variable (numeric data type). Defines the angle of rotation (in degrees).

**Example:** PLC_PRG.iAngle2.

In runtime mode, the element rotates about the point of rotation specified in “Center” according to the value of the variable. In addition, the alignment of the element rotates according to the coordinate system of the visualization. Increasing the value in the code rotates clockwise.

The rotation point is shown as the symbol.

Note: If a static angle of rotation is specified in the “Position Angle” property, then the static angle of rotation is added to the variable angle of rotation (offset) when the visualization is executed.

---

You can link the variables to a unit conversion.

---

The “X”, “Y”, “Rotation”, and “Interior rotation” properties are supported by the “Client Animation” functionality.

See also

- Unit conversion

---

### Element property ‘State variables’

The variables control the element behavior dynamically.

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>“Invisible”</strong></td>
<td>Variable (BOOL). Toggles the visibility of the element. TRUE: The element is not visible at runtime.</td>
</tr>
<tr>
<td><strong>“Deactivate inputs”</strong></td>
<td>Variable (BOOL). Toggles the operability of the element. TRUE: User inputs do not have any effect in runtime more. The element is shown as deactivated.</td>
</tr>
</tbody>
</table>
The “Invisible” property is supported by the “Client Animation” functionality.

See also

These properties are available only when you select the “Preview: Support client animations and overlay of native elements” option in the Visualization Manager.

You can animate a movement, a rotation, and the visibility of a visualization element.

<table>
<thead>
<tr>
<th><strong>&quot;Animation duration&quot;</strong></th>
<th>Time that the element executes an animation (in milliseconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example: 500</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>&quot;Move to the foreground&quot;</strong></th>
<th>Property value (BOOL)</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRUE: At runtime, the element is displayed in the foreground.</td>
<td></td>
</tr>
<tr>
<td>FALSE: At runtime, the element is displayed in the layer where it was inserted in the visualization editor.</td>
<td></td>
</tr>
</tbody>
</table>

See also

- Chapter 1.3.5.21.5 “Using client animation” on page 1962

Element property 'Input configuration'
The properties contain the configurations for the user input when using the mouse or keyboard. User input is a user event from the perspective of the element.

The input configuration refers to the text area of the element only, not the two buttons.

The “Configure” button opens the “Input configuration” dialog box for creating or modifying a user input configuration.

A configuration contains one or more input actions for the respective input event. Existing input actions are displayed below it.

Example: “Execute ST code”: # PLC_PRG.i_x := 0;

<table>
<thead>
<tr>
<th><strong>&quot;OnDialogClosed&quot;</strong></th>
<th>Input event: The user closes the dialog box.</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;OnMouseClick&quot;</td>
<td>Input event: The user clicks the element completely. The mouse button is clicked and released.</td>
</tr>
<tr>
<td>&quot;OnMouseDown&quot;</td>
<td>Input event: The user clicks down on the element only.</td>
</tr>
<tr>
<td>&quot;OnMouseEnter&quot;</td>
<td>Input event: The user drags the mouse pointer to the element.</td>
</tr>
<tr>
<td>&quot;OnMouseLeave&quot;</td>
<td>Input event: The user drags the mouse pointer away from the element.</td>
</tr>
<tr>
<td>&quot;OnMouseMove&quot;</td>
<td>Input event: The user moves the mouse pointer over the element area.</td>
</tr>
<tr>
<td>&quot;OnMouseUp&quot;</td>
<td>Input event: The user releases the mouse button over the element area.</td>
</tr>
</tbody>
</table>

See also

- Chapter 1.3.5.19.3.6 “Dialog ‘Input Configuration’” on page 1586
### “Tap”
When a mouse click event occurs, the variable defined in “Variable” is described in the application. The coding depends on the options “Tap FALSE” and “Tap on enter if captured”.

### “Variable”
Variable (BOOL). Contains the information whether a mouse click event exists.
- **Example:** PLC_PRG.bIsTapped
  - **TRUE:** A mouse click event exists. It lasts while the user presses the mouse button over the element. It ends when the button is released.
  - **FALSE:** A mouse click event does not exist.
- **Requirement:** The “Tap FALSE” option is not activated.

### “Tap FALSE”
- **✔️:** The mouse click event leads to a complementary value in “Variable”.
- **TRUE:** A mouse click event does not exist.
- **FALSE:** While the mouse click event exists.

### “Tap on enter if captured”
- **✔️:** During user input, it is also taken into consideration whether the mouse pointer is dragged within the element area or not while the mouse button is pressed.
  - **TRUE:** While the mouse click event exists and the mouse pointer is moved over the element area.
  - **FALSE:** A mouse click event does not exist. Or the user moves the mouse pointer outside of the element area while the mouse button is pressed.
  - The value is **TRUE** again as soon as the user moves the pointer back to the element area. The mouse is then captured.

### “Shift”
When a mouse click event occurs, the variable here is described in the application. When the mouse click event ends, its value is toggled with the “Toggle on up if captured” option.

### “Variable”
Variable (BOOL). Its value toggled when the mouse click event is ended. This is when the user releases the mouse button while the mouse pointer is over the element area.
- If the user releases the mouse button while the mouse pointer is outside of the element area, then the mouse click event is not ended and the value is not toggled.
- **Tip:** The user can cancel a started toggle input by dragging the mouse pointer out of the element area.

### “Toggle on up if captured”
- **✔️:** The value toggles regardless of where the mouse pointer is when the mouse button is released. The mouse is then captured.

### “Hotkeys”
Keyboard shortcut on the element for triggering specific input actions.
- When the keyboard shortcut event occurs, the input actions in the “Event(s)” property are triggered. In this way, it is not the input action itself that leads to this input action, but the mouse input action.

### “Key”
Key pressed for input action.
- **Example:** [T]
- **Note:** The following properties appear when a key is selected.
“Event(s)”
- “None”
- “Mouse down”: Pressing the key triggers the input actions that are configured in the “OnMouseDown” property.
- “Mouse up”: Releasing the key triggers the input actions that are configured in the “OnMouseUp” property.
- “Mouse down/up”: Pressing and releasing the key triggers the input actions that are configured in the “OnMouseDown” property and the “OnMouseUp” property.

“Shift”
☑: Combination with the Shift key
Example: [Shift]+[T].

“Control”
☑: Combination with the Ctrl key
Example: [Ctrl]+[T].

“Alt”
☑: Combination with the Alt key
Example: [Alt]+[T].

All keyboard shortcuts and their actions that are configured in the visualization are listed in the “Keyboard configuration” tab.

See also
- § Chapter 1.3.5.19.2.2 “Command ‘Keyboard Configuration’” on page 1558
- § Chapter 1.3.5.19.3.6 “Dialog ‘Input Configuration’” on page 1586

Requirement: User management is set up for the visualization.

“Access rights”
Opens the “Access rights” dialog. There you can edit the access privileges for the element.
Status messages:
- “Not set. Full rights.”: Access rights for all user groups: “operable”
- “Rights are set: Limited rights”: Access is restricted for at least one group.

See also
- § Chapter 1.3.5.19.3.1 “Dialog ‘Access Rights’” on page 1583

See also
- 

Visualization element ‘Invisible input’
Symbol:

Tag: “Common controls”
This element is displayed in the editor with a dashed line and is not visible in online mode. You define the behavior of the el in the input configuration.
Element properties

<table>
<thead>
<tr>
<th>&quot;Element name&quot;</th>
<th>Optional</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hint: Assign individual names for elements so that they are found faster in the element list.</td>
<td></td>
</tr>
<tr>
<td>Example: invisible_input_1</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>&quot;Type of element&quot;</th>
<th>&quot;Invisible input&quot;</th>
</tr>
</thead>
</table>

The position defines the location and size of the element in the visualization window. These are based on the Cartesian coordinate system. The origin is located at the upper left corner of the window. The positive horizontal x-axis runs to the right. The positive vertical y-axis runs downwards.

<table>
<thead>
<tr>
<th>&quot;X&quot;</th>
<th>X coordinate of the upper left corner of the element</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Specified in pixels.</td>
</tr>
<tr>
<td></td>
<td>Example: 10.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>&quot;Y&quot;</th>
<th>Y coordinate of the upper left corner of the element</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Specified in pixels.</td>
</tr>
<tr>
<td></td>
<td>Example: 10.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>&quot;Width&quot;</th>
<th>Specified in pixels.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Example: 150</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>&quot;Height&quot;</th>
<th>Specified in pixels.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Example: 30</td>
</tr>
</tbody>
</table>

You can also change the values by dragging the box symbols (■) to other positions in the editor.

See also

- Chapter 1.3.5.3.2 "Positioning elements, adapting size and layer" on page 1113

Element property 'Center'

The properties contain fixed values for the coordinates of the point of rotation. This point of rotation is shown as the symbol. The point is used as the center for rotating and scaling.

<table>
<thead>
<tr>
<th>&quot;X&quot;</th>
<th>X-coordinate of the point of rotation</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>&quot;Y&quot;</th>
<th>Y-coordinate of the point of rotation</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

You can also change the values by dragging the symbols (♣) to other positions in the editor.

Element property 'Absolute movement'

The properties contain IEC variables for controlling the position of the element dynamically. The reference point is the upper left corner of the element. In runtime mode, the entire element is moved.
### Movement

**“X”**  
Variable (numeric data type). Defines the X position (in pixels).  
Example: PLC_PRG.iPos_X.  
Increasing this value in runtime mode moves the element to the right.

**“Y”**  
Variable (numeric data type). Defines the Y position (in pixels).  
Example: PLC_PRG.iPos_Y.  
Increasing this value in runtime mode moves the element downwards.

**“Rotation”**  
Variable (numeric data type). Defines the angle of rotation (in degrees).  
Example: PLC_PRG.iAngle1.  
The midpoint of the element rotates at the “Center” point. This rotation point is shown as the symbol.  
In runtime mode, the alignment of the element remains the same with respect to the coordinate system of the visualization. Increasing the value rotates the element to the right.

**“Interior rotation”**  
Variable (numeric data type). Defines the angle of rotation (in degrees).  
Example: PLC_PRG.iAngle2.  
In runtime mode, the element rotates about the point of rotation specified in “Center” according to the value of the variable. In addition, the alignment of the element rotates according to the coordinate system of the visualization. Increasing the value in the code rotates clockwise.  
The rotation point is shown as the symbol.  
Note: If a static angle of rotation is specified in the “Position ➔ Angle” property, then the static angle of rotation is added to the variable angle of rotation (offset) when the visualization is executed.

---

You can link the variables to a unit conversion.

---

The “X”, “Y”, “Rotation”, and “Interior rotation” properties are supported by the “Client Animation” functionality.

See also
- Unit conversion
The variables control the element behavior dynamically.

| “Deactivate inputs” | Variable (BOOL). Toggles the operability of the element.  
| TRUE: User inputs do not have any effect in runtime more. The element is shown as deactivated. |

These properties are available only when you select the “Preview: Support client animations and overlay of native elements” option in the Visualization Manager.

You can animate a movement, a rotation, and the visibility of a visualization element.

| “Animation duration” | Time that the element executes an animation (in milliseconds)  
| Example: 500 |
| “Move to the foreground” | Property value (BOOL)  
| TRUE: At runtime, the element is displayed in the foreground.  
| FALSE: At runtime, the element is displayed in the layer where it was inserted in the visualization editor. |

See also
- © Chapter 1.3.5.21.5 “Using client animation” on page 1962

The properties contain the configurations for the user input when using the mouse or keyboard. A user input defines an event and one or more actions that are executed when an event occurs.

The “Configure” button opens the “Input configuration” dialog box for creating or modifying a user input.

Configured user inputs are briefly listed below the events. They each include the action that is triggered and the setting in short form.

Example: “Execute ST code”: # PLC_PRG.i_x := 0;

| “OnDialogClosed” | Input event: The user closes the dialog box. |
| “OnMouseClick” | Input event: The user clicks the mouse button completely. The mouse button is clicked and released. |
| “OnMouseDown” | Input event: The user clicks down on the mouse button. |
| “OnMouseEnter” | Input event: The user drags the mouse pointer to the element. |
| “OnMouseLeave” | Input event: The user drags the mouse pointer away from the element. |
| “OnMouseMove” | Input event: The user moves the mouse pointer over the element area. |
| “OnMouseUp” | Input event: The user releases the mouse button over the element area. |
### "Tap"

When a mouse click event occurs, the variable defined in "Variable" is described in the application. The coding depends on the options "Tap FALSE" and "Tap on enter if captured".

#### "Variable"
Variable (BOOL) that is set on mouse click.

**Example:** PLC_PRG.bIsTapped
- **TRUE**: A mouse click event exists. It lasts while the user presses the mouse button over the element. It ends when the button is released.
- **FALSE**: A mouse click event does not exist.

**Requirement:** The "Tap FALSE" option is not activated.

#### "Tap FALSE"
- **☑**: The mouse click event leads to a complementary value in "Variable".
- **TRUE**: A mouse click event does not exist.
- **FALSE**: While the mouse click event exists.

#### "Tap on enter if captured"
- **☑**: During user input, it is also taken into consideration whether the mouse pointer is dragged within the element area or not while the mouse button is pressed.
- **TRUE**: While the mouse click event exists and the mouse pointer is moved over the element area.
- **FALSE**: A mouse click event does not exist. Or the user moves the mouse pointer outside of the element area while the mouse button is pressed.

The value is **TRUE** again as soon as the user moves the pointer back to the element area. The mouse is then captured.

### "Shift"

With the onset of a mouse click event, the variable is set; when the mouse click event is completed, the variable is reset.

#### "Variable"
Variable (BOOL). Its value toggled when the mouse click event is ended. This is when the user releases the mouse button while the mouse pointer is over the element area.

If the user releases the mouse button while the mouse pointer is outside of the element area, the mouse click event is not ended and the value is not toggled.

**Tip:** The user can cancel a started toggle input by dragging the mouse pointer out of the element area.

#### "Toggle on up if captured"
- **☑**: The value toggles regardless of where the mouse pointer is when the mouse button is released. The mouse is then captured.

### "Keyboard shortcuts"
Keyboard shortcut on the element for triggering specific input actions.

When the keyboard shortcut event occurs, the input actions in the "Event(s)" property are triggered. In this way, it is not the input action itself that leads to this input action, but the mouse input action.

#### "Key"
Key pressed for input action.

**Example:** [T]

**Note:** The following properties appear when a key is selected.
**“Event(s)”**

- **“None”**
- **“Mouse down”**: Pressing the key triggers the input actions that are configured in the “OnMouseDown” property.
- **“Mouse up”**: Releasing the key triggers the input actions that are configured in the “OnMouseUp” property.
- **“Mouse down/up”**: Pressing and releasing the key triggers the input actions that are configured in the “OnMouseDown” property and the “OnMouseUp” property.

**“Shift”**

- ✅: Combination with the Shift key
- Example: [Shift]+[T].

**“Control”**

- ✅: Combination with the Ctrl key
- Example: [Ctrl]+[T].

**“Alt”**

- ✅: Combination with the Alt key
- Example: [Alt]+[T].

---

All keyboard shortcuts and their actions that are configured in the visualization are listed in the “Keyboard configuration” tab.

See also
- ✔ Chapter 1.3.5.19.2.2 “Command ‘Keyboard Configuration’” on page 1558
- ✔ Chapter 1.3.5.19.3.6 “Dialog ‘Input Configuration’” on page 1586

**Element property ‘Access rights’**

Requirement: User management is set up for the visualization.

**“Access rights”**

- Opens the “Access rights” dialog. There you can edit the access privileges for the element.
- Status messages:
  - **“Not set. Full rights.”**: Access rights for all user groups: “operable”
  - **“Rights are set: Limited rights”**: Access is restricted for at least one group.

See also
- ✔ Chapter 1.3.5.19.3.1 “Dialog ‘Access Rights’” on page 1583

See also
- ✔

**Visualization element ‘Checkbox’**

Symbol:

Tag: “Common controls”

The element is used for setting and resetting a Boolean variable. The set state is represented by a check mark.
**Element properties**

<table>
<thead>
<tr>
<th>&quot;Element name&quot;</th>
<th>Example: <code>signal_tone_for_parts_deficit</code></th>
</tr>
</thead>
<tbody>
<tr>
<td>Optional</td>
<td></td>
</tr>
<tr>
<td>Hint: Assign individual names for elements so that they are found faster in the element list.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>&quot;Type of element&quot;</th>
<th>&quot;Checkbox&quot;</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>&quot;Text ID&quot;</th>
<th>ID for the text in the &quot;GlobalTextList&quot;</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example: 22</td>
<td></td>
</tr>
<tr>
<td>The text ID cannot be modified. As soon as you specify and save a text in &quot;Texts&quot; - &quot;Text&quot;, CODESYS automatically creates an entry in the &quot;GlobalTextList&quot; and displays the ID here.</td>
<td></td>
</tr>
</tbody>
</table>

**Element property 'Position'**

The position defines the location and size of the element in the visualization window. These are based on the Cartesian coordinate system. The origin is located at the upper left corner of the window. The positive horizontal x-axis runs to the right. The positive vertical y-axis runs downwards.

<table>
<thead>
<tr>
<th>&quot;X&quot;</th>
<th>X coordinate of the upper left corner of the element</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Specified in pixels.</td>
</tr>
<tr>
<td></td>
<td>Example: 10.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>&quot;Y&quot;</th>
<th>Y coordinate of the upper left corner of the element</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Specified in pixels.</td>
</tr>
<tr>
<td></td>
<td>Example: 10.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>&quot;Width&quot;</th>
<th>Specified in pixels.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Example: 150</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>&quot;Height&quot;</th>
<th>Specified in pixels.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Example: 30</td>
</tr>
</tbody>
</table>

You can also change the values by dragging the box symbols (□) to other positions in the editor.

See also

- § Chapter 1.3.5.3.2 “Positioning elements, adapting size and layer” on page 1113

**Element property 'Center'**

The properties contain fixed values for the coordinates of the point of rotation. This point of rotation is shown as the ♦ symbol. The point is used as the center for rotating and scaling.

<table>
<thead>
<tr>
<th>&quot;X&quot;</th>
<th>X-coordinate of the point of rotation</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>&quot;Y&quot;</th>
<th>Y-coordinate of the point of rotation</th>
</tr>
</thead>
</table>
You can also change the values by dragging the symbols (aceutical) to other positions in the editor.

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Variable”</td>
<td>Variable of type BOOL.</td>
<td>“PLC_PRG.xIsTrue”</td>
</tr>
<tr>
<td>“Frame size”</td>
<td>Distance of the element to the edge</td>
<td>“From style”</td>
</tr>
</tbody>
</table>

**Element property ’Texts’**

The properties contain character strings for labeling the element. CODESYS accepts the specified texts automatically into the “GlobalTextList” text list. Therefore, these texts can be localized.

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Text”</td>
<td>Character string (without single straight quotation marks) for the labeling the element.</td>
<td>Axis 1</td>
</tr>
<tr>
<td>“Tooltip”</td>
<td>Character string (without single straight quotation marks) that is displayed as the tooltip of an element.</td>
<td>Parameters of Axis 1</td>
</tr>
</tbody>
</table>

See also

- [The properties contain fixed values for the text properties.](#)

**“Usage of”**

- “Default style values”: The values of the visualization style are used.
- “Individual settings”: The "Individual text properties" property group is shown. The values can be customized here.

**“Individual text properties”**

Requirement: The "Individual settings" text property is defined.

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Horizontal alignment”</td>
<td>Horizontal alignment of the text within the element.</td>
<td></td>
</tr>
<tr>
<td>“Vertical alignment”</td>
<td>Vertical alignment of the text within the element.</td>
<td></td>
</tr>
<tr>
<td>“Text format”</td>
<td>Definition for displaying texts that are too long</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- “Default”: The long text is truncated.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- “Line break”: The text is split into parts.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- “Ellipsis”: The visible text ends with “…” indicating that it is not complete.</td>
<td></td>
</tr>
<tr>
<td>“Font”</td>
<td>Example: “Default”</td>
<td></td>
</tr>
<tr>
<td><img src="#" alt="Font" /></td>
<td>The “Font” dialog box opens.</td>
<td></td>
</tr>
<tr>
<td><img src="#" alt="Font" /></td>
<td>The “Font” dialog box opens.</td>
<td></td>
</tr>
<tr>
<td><img src="#" alt="Font" /></td>
<td>The “Font” dialog box opens.</td>
<td></td>
</tr>
<tr>
<td><img src="#" alt="Font" /></td>
<td>The “Font” dialog box opens.</td>
<td></td>
</tr>
<tr>
<td><img src="#" alt="Font" /></td>
<td>The “Font” dialog box opens.</td>
<td></td>
</tr>
</tbody>
</table>

---

1788 3ADR010583, 1, en_US 2020/12/10
**“Font color”**

Example: “Black”

![The “Color” dialog box opens.](image)

![Drop-down list with style colors.](image)

**“Transparency”**

Whole number (value range from 0 to 255). This determines the transparency of the respective color.

Example: 255: The color is opaque.

0: The color is completely transparent.

Please note: If the color is a style color and already has a transparency value, then this property is write-protected.

---

**Element properties 'Absolute movement’**

The properties contain IEC variables for controlling the position of the element dynamically. The reference point is the upper left corner of the element. In runtime mode, the entire element is moved.

---

**“Movement”**

**“X”**

Variable (numeric data type). Defines the X position (in pixels).

Example: PLC_PRG.iPos_X.

Increasing this value in runtime mode moves the element to the right.

**“Y”**

Variable (numeric data type). Defines the Y position (in pixels).

Example: PLC_PRG.iPos_Y.

Increasing this value in runtime mode moves the element downwards.

**“Rotation”**

Variable (numeric data type). Defines the angle of rotation (in degrees).

Example: PLC_PRG.iAngle1.

The midpoint of the element rotates at the “Center” point. This rotation point is shown as the symbol.

In runtime mode, the alignment of the element remains the same with respect to the coordinate system of the visualization. Increasing the value rotates the element to the right.

**“Interior rotation”**

Variable (numeric data type). Defines the angle of rotation (in degrees).

Example: PLC_PRG.iAngle2.

In runtime mode, the element rotates about the point of rotation specified in “Center” according to the value of the variable. In addition, the alignment of the element rotates according to the coordinate system of the visualization. Increasing the value in the code rotates clockwise.

The rotation point is shown as the symbol.

Note: If a static angle of rotation is specified in the “Position ➔ Angle” property, then the static angle of rotation is added to the variable angle of rotation (offset) when the visualization is executed.
You can link the variables to a unit conversion.

The "X", "Y", "Rotation", and "Interior rotation" properties are supported by the "Client Animation" functionality.

See also
- Unit conversion
- 

Element property 'State variables'

The variables control the element behavior dynamically.

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Invisible&quot;</td>
<td>Variable (BOOL). Toggles the visibility of the element. TRUE: The element is not visible at runtime.</td>
</tr>
<tr>
<td>&quot;Deactivate inputs&quot;</td>
<td>Variable (BOOL). Toggles the operability of the element. TRUE: User inputs do not have any effect in runtime more. The element is shown as deactivated.</td>
</tr>
</tbody>
</table>

The "Invisible" property is supported by the "Client Animation" functionality.

See also
- 

These properties are available only when you select the "Preview: Support client animations and overlay of native elements" option in the Visualization Manager.

You can animate a movement, a rotation, and the visibility of a visualization element.

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Animation duration&quot;</td>
<td>Time that the element executes an animation (in milliseconds) Example: 500</td>
</tr>
<tr>
<td>&quot;Move to the foreground&quot;</td>
<td>Property value (BOOL) TRUE: At runtime, the element is displayed in the foreground. FALSE: At runtime, the element is displayed in the layer where it was inserted in the visualization editor.</td>
</tr>
</tbody>
</table>

See also
- % Chapter 1.3.5.21.5 “Using client animation” on page 1962

Element property 'Access rights'

Requirement: User management is set up for the visualization.
"Access rights" Opens the "Access rights" dialog. There you can edit the access privileges for the element.
Status messages:
- "Not set. Full rights.": Access rights for all user groups: "operable"
- "Rights are set: Limited rights": Access is restricted for at least one group.

See also
- Chapter 1.3.5.19.3.1 "Dialog ‘Access Rights’" on page 1583

Visualization element 'Progress bar'
Symbol:

Tag: “Common controls”
The element represents the value of a variable as a progress bar.

**Element properties**

| “Element name” | Optional
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Hint: Assign individual names for elements so that they are found faster in the element list. Example: Progress_Data_Transfer</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>“Type of element”</th>
<th>“Progress bar”</th>
</tr>
</thead>
</table>

| “Text ID” | Identification of the global text list
| Requirement: Text is configured in the property “Texts ➔ Text”. |

| “Variable” | Variable (numeric data type). Represents the length of the progress bar. |

| “Minimum value” | Value range of the variable |

| “Maximum value” | |

| “Style” | ● “Blocks”
| ● “Bar” |

**Element property 'Position'**
The position defines the location and size of the element in the visualization window. These are based on the Cartesian coordinate system. The origin is located at the upper left corner of the window. The positive horizontal x-axis runs to the right. The positive vertical y-axis runs downwards.
“X”  X coordinate of the upper left corner of the element
specified in pixels.
Example: 10.

“Y”  Y coordinate of the upper left corner of the element
specified in pixels.
Example: 10.

“Width”  Specified in pixels.
Example: 150

“Height”  Specified in pixels.
Example: 30

You can also change the values by dragging the box symbols (_dragArrow_) to other positions in the editor.

See also

- § Chapter 1.3.5.3.2 “Positioning elements, adapting size and layer” on page 1113

Element property ‘Center’

The properties contain fixed values for the coordinates of the point of rotation. This point of rotation is shown as the symbol. The point is used as the center for rotating and scaling.

“X”  X-coordinate of the point of rotation

“Y”  Y-coordinate of the point of rotation

You can also change the values by dragging the symbols (_dragArrow_) to other positions in the editor.

Element property ‘Texts’

“Text”  String label for the element.
Example: Zoom

Element property 'Absolute movement'

The properties contain IEC variables for controlling the position of the element dynamically. The reference point is the upper left corner of the element. In runtime mode, the entire element is moved.

“Movement”

“X”  Variable (numeric data type). Defines the X position (in pixels).
Example: PLC_PRG.iPos_X.
Increasing this value in runtime mode moves the element to the right.
"Y" Variable (numeric data type). Defines the Y position (in pixels).
Example: PLC_PRG.iPos_Y.
Increasing this value in runtime mode moves the element downwards.

"Rotation" Variable (numeric data type). Defines the angle of rotation (in degrees).
Example: PLC_PRG.iAngle1.
The midpoint of the element rotates at the "Center" point. This rotation point is shown as the symbol.
In runtime mode, the alignment of the element remains the same with respect to the coordinate system of the visualization. Increasing the value rotates the element to the right.

"Interior rotation" Variable (numeric data type). Defines the angle of rotation (in degrees).
Example: PLC_PRG.iAngle2.
In runtime mode, the element rotates about the point of rotation specified in "Center" according to the value of the variable. In addition, the alignment of the element rotates according to the coordinate system of the visualization. Increasing the value in the code rotates clockwise.
The rotation point is shown as the symbol.
Note: If a static angle of rotation is specified in the "Position Angle" property, then the static angle of rotation is added to the variable angle of rotation (offset) when the visualization is executed.

You can link the variables to a unit conversion.

The "X", "Y", "Rotation", and "Interior rotation" properties are supported by the "Client Animation" functionality.

See also
- Unit conversion
- 

Element property 'State variables'
The variables control the element behavior dynamically.

"Invisible" Variable (BOOL). Toggles the visibility of the element.
TRUE: The element is not visible at runtime.
The “Invisible” property is supported by the "Client Animation" functionality.

See also

These properties are available only when you select the “Preview: Support client animations and overlay of native elements” option in the Visualization Manager.

You can animate a movement, a rotation, and the visibility of a visualization element.

<table>
<thead>
<tr>
<th>“Animation duration”</th>
<th>Time that the element executes an animation (in milliseconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Example: 500</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>“Move to the foreground”</th>
<th>Property value (BOOL)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>TRUE: At runtime, the element is displayed in the foreground.</td>
</tr>
<tr>
<td></td>
<td>FALSE: At runtime, the element is displayed in the layer where it was inserted in the visualization editor.</td>
</tr>
</tbody>
</table>

See also

- Chapter 1.3.5.21.5 “Using client animation” on page 1962

Element property ‘Access rights’

Requirement: User management is set up for the visualization.

<table>
<thead>
<tr>
<th>“Access rights”</th>
<th>Opens the “Access rights” dialog. There you can edit the access privileges for the element.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Status messages:</td>
</tr>
<tr>
<td></td>
<td>&quot;Not set. Full rights.”: Access rights for all user groups : “operable”</td>
</tr>
<tr>
<td></td>
<td>&quot;Rights are set: Limited rights“: Access is restricted for at least one group.</td>
</tr>
</tbody>
</table>

See also

- Chapter 1.3.5.19.3.1 “Dialog ‘Access Rights’” on page 1583

See also

-  

Visualization element 'Radio button'

Symbol:

Tag: “Common controls”

The element provides a field with any number of options.
**Element properties**

| Element name | Optional
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Hint: Assign individual names for elements so that they are found faster in the element list. Example: early_shift</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Type of element</th>
<th>Radio button</th>
</tr>
</thead>
</table>

**Element property 'Position'**

The position defines the location and size of the element in the visualization window. These are based on the Cartesian coordinate system. The origin is located at the upper left corner of the window. The positive horizontal x-axis runs to the right. The positive vertical y-axis runs downwards.

| X | X coordinate of the upper left corner of the element
Specified in pixels.
Example: 10 |
|---|---|
| Y | Y coordinate of the upper left corner of the element
Specified in pixels.
Example: 10 |
| Width | Specified in pixels.
Example: 150 |
| Height | Specified in pixels.
Example: 30 |

You can also change the values by dragging the box symbols ( hè ) to other positions in the editor.

See also

- Chapter 1.3.5.3.2 “Positioning elements, adapting size and layer” on page 1113

| Variable | Variable (integer data type) that gives the index of the radio button that the visualization user has activated
Example: PLC_PRG.iNrOfActivatedRadioButton |
| Number of columns | Definition of the number of list boxes displayed in a row
Example: 2 |
| Radio button order | “Left to right”: The radio buttons are aligned by rows until the number of columns is reached.
“Top to bottom”: The radio buttons are aligned row by columns until the number of columns is reached. |
| Frame size | Defines the distance from the list boxes to the edge (in pixels). |
| Row height | Height of the row (in pixels). Modifying the height of the row also changes the size of the list box. |
Element property 'Text properties'

The properties contain fixed values for the text properties.

| “Usage of”                  | • “Default style values”: The values of the visualization style are used.  
|                            | • “Individual settings”: The “Individual text properties” property group is shown  
|                            | The values can be customized here. |

| “Individual text properties”                 |
| Requirement: The “Individual settings” text property is defined. |

| “Horizontal alignment”    | Horizontal alignment of the text within the element. |
| “Vertical alignment”      | Vertical alignment of the text within the element. |

| “Text format”             | Definition for displaying texts that are too long  
|                            | • “Default”: The long text is truncated.  
|                            | • “Line break”: The text is split into parts.  
|                            | • “Ellipsis”: The visible text ends with ”...” indicating that it is not complete. |

| “Font”                    | Example: “Default”  
|                          | 🔗: The “Font” dialog box opens.  
|                          | ▼: Drop-down list with style fonts. |

| “Font color”              | Example: “Black”  
|                          | 🔗: The “Color” dialog box opens.  
|                          | ▼: Drop-down list with style colors. |

| “Transparency”            | Whole number (value range from 0 to 255). This determines the transparency of the respective color.  
| Example: 255: The color is opaque.  
| 0: The color is completely transparent. |
| Please note: If the color is a style color and already has a transparency value, then this property is write-protected. |

Element property 'Absolute movement'

The properties contain IEC variables for controlling the position of the element dynamically. The reference point is the upper left corner of the element. In runtime mode, the entire element is moved.

| “Movement”                |
| “X”                       | Variable (numeric data type). Defines the X position (in pixels).  
| Example: PLC_PRG.iPos_X.  
| Increasing this value in runtime mode moves the element to the right. |

| “Y”                       | Variable (numeric data type). Defines the Y position (in pixels).  
| Example: PLC_PRG.iPos_Y.  
| Increasing this value in runtime mode moves the element downwards. |
“Rotation” Variable (numeric data type). Defines the angle of rotation (in degrees).

Example: PLC_PRG.iAngle1.

The midpoint of the element rotates at the “Center” point. This rotation point is shown as the symbol.

In runtime mode, the alignment of the element remains the same with respect to the coordinate system of the visualization. Increasing the value rotates the element to the right.

“Interior rotation” Variable (numeric data type). Defines the angle of rotation (in degrees).

Example: PLC_PRG.iAngle2.

In runtime mode, the element rotates about the point of rotation specified in “Center” according to the value of the variable. In addition, the alignment of the element rotates according to the coordinate system of the visualization. Increasing the value in the code rotates clockwise.

The rotation point is shown as the symbol.

Note: If a static angle of rotation is specified in the “Position Angle” property, then the static angle of rotation is added to the variable angle of rotation (offset) when the visualization is executed.

You can link the variables to a unit conversion.

The “X”, “Y”, “Rotation”, and “Interior rotation” properties are supported by the “Client Animation” functionality.

See also
- Unit conversion
- 

Element property ‘State variables’ The variables control the element behavior dynamically.

| “Invisible” | Variable (BOOL). Toggles the visibility of the element. TRUE: The element is not visible at runtime. |
| “Deactivate inputs” | Variable (BOOL). Toggles the operability of the element. TRUE: User inputs do not have any effect in runtime more. The element is shown as deactivated. |
The “Invisible” property is supported by the “Client Animation” functionality.

See also

- “Radio button”
  - “Areas”
    - “[<n>]”
“Create new”: Clicking this button creates a new selection button in the editor and lists an additional area in the properties editor. For each radio button, an area is visible that records the settings.
  - “[<n>]”
    - “[<n>]”: This number indicates the area. Clicking “Delete” will delete the associated radio button with its settings “Text”, “Tooltip”, and “Line spacing (in pixels)”.

Areas: [<n>]

- “Text”
  The button name is specified here. Default value: “Radio_button”
- “Tooltip”
  Text is specified here that is displayed in a tooltip.
- “Line spacing (in pixels)”
  The distance (in pixels) to the upper button can be specified here.

These properties are available only when you select the “Preview: Support client animations and overlay of native elements” option in the Visualization Manager.

You can animate a movement, a rotation, and the visibility of a visualization element.

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Animation duration”</td>
<td>Time that the element executes an animation (in milliseconds)</td>
</tr>
<tr>
<td>Example: 500</td>
<td></td>
</tr>
<tr>
<td>“Move to the foreground”</td>
<td>Property value (BOOL)</td>
</tr>
<tr>
<td>TRUE: At runtime, the element is displayed in the foreground.</td>
<td></td>
</tr>
<tr>
<td>FALSE: At runtime, the element is displayed in the layer where it was inserted in the visualization editor.</td>
<td></td>
</tr>
</tbody>
</table>

See also

- Chapter 1.3.5.21.5 “Using client animation” on page 1962

Requirement: User management is set up for the visualization.

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Access rights”</td>
<td>Opens the “Access rights” dialog. There you can edit the access privileges for the element.</td>
</tr>
<tr>
<td>Status messages:</td>
<td></td>
</tr>
<tr>
<td>“Not set. Full rights.”: Access rights for all user groups : “operable”</td>
<td></td>
</tr>
<tr>
<td>“Rights are set: Limited rights”: Access is restricted for at least one group.</td>
<td></td>
</tr>
</tbody>
</table>

See also

- Chapter 1.3.5.19.3.1 “Dialog ‘Access Rights’” on page 1583
Visualization element 'Alarm table'

Symbol:

Tag: “Alarm manager”

The element displays alarms in a list. In the element properties, you specify which information is shown. You define the appearance of the element and the variables that control the element behavior.

In online mode, you can sort an alarm table by a specific column – even in the classic view. Click into the column header. A small triangle indicates the current sort order (ascending, descending). Clicking the symbol reverses the order.

Sorting inside the column depends on the type of the contained information. The "Priority" column is sorted numerically, and the "Message" and "Class" columns alphabetically. The "Value" and "Latch" columns may contain different value types. In this case, sorting is first by type (blank, Boolean, numeric value, character string) and then either numerically or alphabetically depending on the type.

<table>
<thead>
<tr>
<th>Element properties</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Element name&quot;</td>
<td>GenElemInst_1</td>
</tr>
<tr>
<td>&quot;Type of element&quot;</td>
<td>&quot;Alarm Table&quot;</td>
</tr>
<tr>
<td>&quot;Data source&quot;</td>
<td>Selection of the device and the application where the data to be visualized and the alarms are generated</td>
</tr>
<tr>
<td></td>
<td>• Remote data source which accesses a remote device, accesses a remote application, and then transfers the data to the alarm configuration</td>
</tr>
<tr>
<td></td>
<td>Example: <code>DataSource_A</code></td>
</tr>
<tr>
<td></td>
<td>Below the (now visible) “Application” property, the remote application is displayed as configured in the data source.</td>
</tr>
<tr>
<td></td>
<td>Example: <code>App_A</code></td>
</tr>
<tr>
<td></td>
<td>Note: If the data source is accessed symbolically by means of a symbol file (CODESYS symbolic), then the required symbol file and the corresponding project have to be saved in the same folder.</td>
</tr>
<tr>
<td></td>
<td>• Local application below which the alarm configuration is located</td>
</tr>
<tr>
<td></td>
<td>Example: <code>&quot;&lt;local application&gt;&quot;</code></td>
</tr>
</tbody>
</table>

See also

• Object 'Data source'
**“Alarm groups”**
Opens the “Select Alarm Group” dialog where you define the alarm groups that you want to display.

**“Priority from”**
Lowest priority for alarm display. (0 to 255).

**“Priority to”**
Highest priority for alarm display. (0 to 255).

**“Alarm classes”**
Opens the “Select Class Group” dialog where you define the alarm classes that you want to display.

**“Filter criterion”**
For the alarm banner element only
- **“Most important”**: The alarm with the highest priority (lowest value) is displayed.
- **“Newest”**: The most recent alarm is displayed.

**“Filter by latch 1”**
The generated alarms (previous and current) can be filtered by the contents of “Latch variable 1”, which is specified in the configuration of the alarm group. In “Filter type”, you define whether or not the filtering is performed by a string value or a numerical value.
- **“Filter variable”**: Indicates what the alarms are filtered by. Possible entries: Application variable of data type STRING or WSTRING, or a literal value directly. Examples: PLC_PRG.strFilterVariable, 'STRING'.
- **“Filter type”**: Integer value that determines by which criteria the latch variable value is used for filtering. Possible entries: Numerical variable from the application (example: PLC_PRG.diFilterType, or a value directly (example: 2).

Possible values:
- 0: No filtering
- 1: Filter by alarms whose latch variable 1 contains the string specified in “Filter variable”. Example: The filter variable contains 'Error 1' which is the latch variable 1 of different alarms of type STRING and has the value 'Error 1' ->. Only these alarms are displayed.
- 2: Filter by alarms whose latch variable 1 contains the typed literal specified in “Filter variable” according to IEC 61131-3. Examples: T#1h2s, DINT#15, REAL#1.5, FALSE
- 3: Filter by alarms whose latch variable 1 contains the LINT literal value specified in “Filter variable”. Therefore, the value of the latch variables has to be in the range of 9,223,372,036,854,775,808 to 9,223,372,036,854,775,807.
- All other values: The behavior is not defined and can change in the future.

See also
- % Chapter 1.3.5.19.3.17 “Dialog ‘Selected Alarm Group’” on page 1605
- % Chapter 1.3.5.19.3.16 “Dialog ‘Selected Alarm Class’” on page 1604
- % Chapter 1.3.5.7 “Visualizing alarm management” on page 1146

**Element property ‘General table configuration’**

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Show row header”</td>
<td>Display of the row number at the beginning of the row.</td>
</tr>
<tr>
<td>“Show column header”</td>
<td>Display of the column heading as defined in “Column heading”.</td>
</tr>
<tr>
<td>“Row height”</td>
<td>Height of the table rows (in pixels).</td>
</tr>
<tr>
<td>“Row header width”</td>
<td>Width of the line header (in pixels).</td>
</tr>
</tbody>
</table>
**Scrollbar size**

Width of the scrollbar when it runs vertically. Width of the scrollbar when it runs horizontally. Specified in pixels.

**Automatic line break for alarm message**

☑: The message text is truncated at the end of the line.

☐: The message text is truncated at the end of the column, if the text is too long.

By default, columns [0] and [1] are configured: “Time stamp” and “Message text”. You can create more columns by clicking the “Create new”, and remove columns by clicking “Delete”.

Animations (dynamic text, font variables), text, and tooltip are not supported.

**Column header**

The standard header is set and changed here by specifying a new text.

**Use text alignment in title**

☑: The text in the column header is aligned according to the current definition in “Text alignment”.

☐: The text in the column header is centered.

**Width**

Width of the column (in pixels).

**Data type**

Notice about time stamps: For use in a TargetVisu or WebVisu, you can control the date and time format with the help of the global string variables from the library Alarmmanager.library: AlarmGlobals.g_sDateFormat (example: AlarmGlobals.g_sDateFormat := 'MM.yyyy') and AlarmGlobals.g_sTimeFormat (example: AlarmGlobals.g_sTimeFormat := 'HH:mm').

Define the information to be displayed in the column.

- “Symbol”
- “Time stamp”: Date and time of the last status change of the alarm.
- “Time stamp active”: Date and time of the last activation of the alarm.
- “Time stamp inactive”: Date and time of the last deactivation of the alarm.
- “Time stamp acknowledge”: Date and time of the last acknowledgment.
- “Value”: Current value of the printout
- “Message text”: Output of the message text
- “Priority”: Alarm priority
- “Class”: Alarm class
- “State”: Alarm state
- “Latch Variable <n>”: Value of the selected latch variables

**Text alignment**

Alignment of the text in this column

- “Left”
- “Centered”
- “Right”

**Color settings**

- “Activate color settings”: Boolean variable for activating and deactivating the color settings defined here. Example: PLC_PRG.bColorSettings
- “Cell fill color”:
  - “Color variable”: Variable for the cell fill color, example: dwCellColor (hexadecimal color definition: 16#TTRRGGBB)
  - “Use color also for column header”: ☑: The color defined via “Color variable” is used in the column header as well.
- “Text color”:
  - “Color variable”: Variable for the definition of the text color in the column, example: dwTextColor (hexadecimal color definition: 16#TTRRGGBB)
  - “Use color also for column header”: ☐: The color defined via “Color variable” is used in the column header as well.

See also

- Chapter 1.3.5.8.3 “Animating a color display” on page 1152
Element property 'Position'
The position defines the location and size of the element in the visualization window. These are based on the Cartesian coordinate system. The origin is located at the upper left corner of the window. The positive horizontal x-axis runs to the right. The positive vertical y-axis runs downwards.

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>“X”</td>
<td>X coordinate of the upper left corner of the element Specified in pixels. Example: 10.</td>
</tr>
<tr>
<td>“Y”</td>
<td>Y coordinate of the upper left corner of the element Specified in pixels. Example: 10.</td>
</tr>
<tr>
<td>“Width”</td>
<td>Specified in pixels. Example: 150</td>
</tr>
<tr>
<td>“Height”</td>
<td>Specified in pixels. Example: 30</td>
</tr>
</tbody>
</table>

You can also change the values by dragging the box symbols (_draged_box_ to other positions in the editor.

See also
- Chapter 1.3.5.3.2 “Positioning elements, adapting size and layer” on page 1113

Element property 'Center'
The properties contain fixed values for the coordinates of the point of rotation. This point of rotation is shown as the _center_ symbol. The point is used as the center for rotating and scaling.

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>“X”</td>
<td>X-coordinate of the point of rotation</td>
</tr>
<tr>
<td>“Y”</td>
<td>Y-coordinate of the point of rotation</td>
</tr>
</tbody>
</table>

You can also change the values by dragging the symbols (_draged_circle_ to other positions in the editor.

Element property 'Text properties'
The properties contain fixed values for the text properties.
### “Font”

Example: “Default”

- The “Font” dialog box opens.
- Drop-down list with style fonts.

### “Font color”

Example: “Black”

- The “Color” dialog box opens.
- Drop-down list with style colors.

### “Transparency”

Whole number (value range from 0 to 255). This determines the transparency of the respective color.

Example: 255: The color is opaque.
0: The color is completely transparent.

Please note: If the color is a style color and already has a transparency value, then this property is write-protected.

---

#### Element property 'Selection'

<table>
<thead>
<tr>
<th>“Background color on selection”</th>
<th>Fill color of the selected row.</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Selection font color”</td>
<td>Font color of the selected row.</td>
</tr>
<tr>
<td>“Frame around selected cells”</td>
<td>A frame is drawn around the selected cells at runtime.</td>
</tr>
<tr>
<td>“Variable for selected alarm group”</td>
<td>Name of the affected alarm group; type: STRING, WSTRING.</td>
</tr>
<tr>
<td>“Variable for selected alarm ID”</td>
<td>Alarm ID of the affected alarm group; type: STRING, WSTRING.</td>
</tr>
<tr>
<td>“Variable for selected line”</td>
<td>Index of the selected alarm line (0-based). The index can be read and written; integer data type</td>
</tr>
<tr>
<td>“Variable for valid selection”</td>
<td>TRUE: An alarm line is selected. FALSE: The selection is invalid. For example, for an empty alarm table or when an alarm is not selected yet.</td>
</tr>
</tbody>
</table>

---

#### Element property 'Control variables'

Boolean variables are defined here for executing specific actions in the table can be executed at runtime.

<table>
<thead>
<tr>
<th>“Acknowledge selected”</th>
<th>Variable (BOOL)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example: PLC_PRG.bAckSelectedAlarms</td>
<td></td>
</tr>
<tr>
<td>If the assigned variable is TRUE, then the selected alarm is acknowledged.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>“Acknowledge all visible”</th>
<th>Variable (BOOL)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example: PLC_PRG.bAckVisibleAlarms</td>
<td></td>
</tr>
<tr>
<td>If the assigned variable is TRUE, then all alarms are acknowledged that are visible in the alarm table.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>“History”</th>
<th>Variable (BOOL)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example: PLC_PRG.bShowHistory</td>
<td></td>
</tr>
<tr>
<td>If the assigned variable is TRUE, then the history alarms are displayed in addition to the active alarms. In the classic view, the same sort options apply as in normal mode. Note: Acknowledgment is not possible in this view.</td>
<td></td>
</tr>
</tbody>
</table>
### “Freeze scroll position” Variable (BOOL)

**Example:** PLC_PRG.bFreezeScrollPosition

If the assigned variable is TRUE, then the scroll position set in the “History” view is retained, even if a new alarm is active. If not, then the scroll position jumps to the first table row (the newest alarm).

### “Count alarms” Variable (integer data type)

**Example:** PLC_PRG.iNumberOfAlarms.

Number of alarms that are currently displayed in the alarm table. Defined by the alarm table.

### “Count visible rows” Variable (integer data type)

**Example:** PLC_PRG.iNumberOfVisibleLines

Number of alarms that can be displayed on one page of the alarm table. Defined by the alarm table.

### “Current scroll index” Variable (integer data type)

**Example:** PLC_PRG.iScrollIndex

The index of the first visible row if the alarm table (0-based). The variable can be read and written.

### “Current column sorting” Variable (integer data type)

**Example:** PLC_PRG.iColSort

The variable contains a value of the enumeration "VisuElemsAlarm.VisuEnumAlarmDataType". This value determines the column that sorts the alarm table.

### “Variable for sorting direction” Variable (BOOL)

**Example:** PLC_PRG.xSortAscending

The variable determines the sort order for the entries in the alarm table (TRUE: ascending; FALSE: descending).

---

You can also use the “Insert elements for acknowledging alarms” command to define buttons with predefined control variables.

See also

- ☐ Chapter 1.3.5.19.2.23 “Command ‘Add Elements for Alarm Acknowledgement’” on page 1582

---

**Element property 'Absolute movement'**

The properties contain IEC variables for controlling the position of the element dynamically. The reference point is the upper left corner of the element. In runtime mode, the entire element is moved.

#### “Movement”

**“X”** Variable (numeric data type). Defines the X position (in pixels).

**Example:** PLC_PRG.iPos_X.

Increasing this value in runtime mode moves the element to the right.

**“Y”** Variable (numeric data type). Defines the Y position (in pixels).

**Example:** PLC_PRG.iPos_Y.

Increasing this value in runtime mode moves the element downwards.
**“Rotation”**  Variable (numeric data type). Defines the angle of rotation (in degrees).

Example: PLC_PRG.iAngle1.

The midpoint of the element rotates at the “Center” point. This rotation point is shown as the symbol.

In runtime mode, the alignment of the element remains the same with respect to the coordinate system of the visualization. Increasing the value rotates the element to the right.

**“Interior rotation”**  Variable (numeric data type). Defines the angle of rotation (in degrees).

Example: PLC_PRG.iAngle2.

In runtime mode, the element rotates about the point of rotation specified in “Center” according to the value of the variable. In addition, the alignment of the element rotates according to the coordinate system of the visualization. Increasing the value in the code rotates clockwise.

The rotation point is shown as the symbol.

Note: If a static angle of rotation is specified in the “Position” property, then the static angle of rotation is added to the variable angle of rotation (offset) when the visualization is executed.

You can link the variables to a unit conversion.

The “X”, “Y”, “Rotation”, and “Interior rotation” properties are supported by the “Client Animation” functionality.

See also

- Unit conversion

Element property ‘State variables’

The variables control the element behavior dynamically.

**“Invisible”**  Variable (BOOL). Toggles the visibility of the element.

TRUE: The element is not visible at runtime.

The “Invisible” property is supported by the “Client Animation” functionality.
These properties are available only when you select the “Preview: Support client animations and overlay of native elements” option in the Visualization Manager.

You can animate a movement, a rotation, and the visibility of a visualization element.

<table>
<thead>
<tr>
<th>“Animation duration”</th>
<th>Time that the element executes an animation (in milliseconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example: 500</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>“Move to the foreground”</th>
<th>Property value (BOOL)</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRUE: At runtime, the element is displayed in the foreground.</td>
<td></td>
</tr>
<tr>
<td>FALSE: At runtime, the element is displayed in the layer where it was inserted in the visualization editor.</td>
<td></td>
</tr>
</tbody>
</table>

Requirement: User management is set up for the visualization.

<table>
<thead>
<tr>
<th>“Access rights”</th>
<th>Opens the “Access rights” dialog. There you can edit the access privileges for the element.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Status messages:</td>
<td></td>
</tr>
<tr>
<td>“Not set. Full rights.”</td>
<td>Access rights for all user groups : “operable”</td>
</tr>
<tr>
<td>“Rights are set: Limited rights”</td>
<td>Access is restricted for at least one group.</td>
</tr>
</tbody>
</table>

See also

- Chapter 1.3.5.21.5 “Using client animation” on page 1962

Visualization element ‘Alarm banner’

Symbol:

Tag: “Alarm manager”

The element is a simplified version of the alarm table. It visualizes a single alarm only. In the element properties, you specify which information is shown. You define the appearance of the element and the variables that control the element behavior.
The alarm banner displays active alarms only. If the alarm is acknowledged, then it disappears from the alarm banner.

### Element properties

<table>
<thead>
<tr>
<th>“Element name”</th>
<th>Example: GenElemInst_1</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Type of element”</td>
<td>“Alarm banner”</td>
</tr>
<tr>
<td>“Data source”</td>
<td>If you intend to use a remote alarm configuration, then you have to specify the name of the remote application here. If you do not specify anything, the alarm configuration will be located locally.</td>
</tr>
</tbody>
</table>

### Element property 'Alarm configuration’

| “Alarm groups” | Opens the “Select Alarm Group” dialog where you define the alarm groups that you want to display. |
| “Priority from” | Lowest priority for alarm display. (0 to 255). |
| “Priority to” | Highest priority for alarm display. (0 to 255). |
| “Alarm classes” | Opens the “Select Class Group” dialog where you define the alarm classes that you want to display. |
### Filter criterion

For the alarm banner element only
- **“Most important”:** The alarm with the highest priority (lowest value) is displayed.
- **“Newest”:** The most recent alarm is displayed.

### Filter by latch 1

The generated alarms (previous and current) can be filtered by the contents of “Latch variable 1”, which is specified in the configuration of the alarm group. In “Filter type”, you define whether or not the filtering is performed by a string value or a numerical value.

- **“Filter variable”:** Indicates what the alarms are filtered by. Possible entries: Application variable of data type STRING or WSTRING, or a literal value directly. Examples: PLC_PRG.strFilterVariable, 'STRING'.
- **“Filter type”:** Integer value that determines by which criteria the latch variable value is used for filtering. Possible entries: Numerical variable from the application (example: PLC_PRG.diFilterType, or a value directly (example: 2).

Possible values:
- 0: No filtering
- 1: Filter by alarms whose latch variable 1 contains the string specified in “Filter variable”. Example: The filter variable contains ‘Error 1’ which is the latch variable 1 of different alarms of type STRING and has the value ‘Error 1’ ->. Only these alarms are displayed.
- 2: Filter by alarms whose latch variable 1 contains the typed literal specified in “Filter variable” according to IEC 61131-3. Examples: T#1h2s, DINT#15, REAL#1.5, FALSE
- 3: Filter by alarms whose latch variable 1 contains the LINT literal value specified in “Filter variable”. Therefore, the value of the latch variables has to be in the range of 9,223,372,036,854,775,808 to 9,223,372,036,854,775,807.
- All other values: The behavior is not defined and can change in the future.

### See also

- % Chapter 1.3.5.19.3.17 “Dialog ‘Selected Alarm Group’” on page 1605
- % Chapter 1.3.5.19.3.16 “Dialog ‘Selected Alarm Class’” on page 1604
- % Chapter 1.3.5.7 “Visualizing alarm management” on page 1146

### Element property ‘Columns: Column [<n>]

By default, columns [0] and [1] are preconfigured: “Time stamp” and “Message text”. You create more columns by clicking “Create new”. You remove columns by clicking “Delete”.

Animations (dynamic text, font variables), texts, and tooltips are not supported.
### “Width”

Width of the column (in pixels)

### “Type of data”

About time stamps: When used in a TargetVisu or WebVisu, you can control the date and time format by means of the global string variables from the library Alarmmanager.library: AlarmGlobals.g_sDateFormat (example: AlarmGlobals.g_sDateFormat := 'MM.yyyy') and AlarmGlobals.g_sTimeFormat (example: AlarmGlobals.g_sTimeFormat := 'HH:mm').

Here you define the information to be displayed in the column.

- “Bitmap”
- “Time stamp”: Date and time of the last status change of the alarm
- “Time stamp active”: Date and time of the last activation of the alarm
- “Time stamp inactive”: Date and time of the last deactivation of the alarm
- “Time stamp acknowledge”: Date and time of the last acknowledgement
- “Value”: Actual value of the expression
- “Message”: Output of the message text
- “Priority”: Alarm priority
- “Class”: Alarm class
- “State”: Alarm state
- “Latch Variable <n>”: Value of the selected latch variables

### “Text alignment”

Alignment of the contents in the column

- “Left”
- “Centered”
- “Right”

---

**Element property 'Position'**

The position defines the location and size of the element in the visualization window. These are based on the Cartesian coordinate system. The origin is located at the upper left corner of the window. The positive horizontal x-axis runs to the right. The positive vertical y-axis runs downwards.

<table>
<thead>
<tr>
<th>“X”</th>
<th>X coordinate of the upper left corner of the element</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Specified in pixels.</td>
</tr>
<tr>
<td>Example:</td>
<td>10</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>“Y”</th>
<th>Y coordinate of the upper left corner of the element</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Specified in pixels.</td>
</tr>
<tr>
<td>Example:</td>
<td>10</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>“Width”</th>
<th>Specified in pixels.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example:</td>
<td>150</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>“Height”</th>
<th>Specified in pixels.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example:</td>
<td>30</td>
</tr>
</tbody>
</table>

You can also change the values by dragging the box symbols (_drag_ ) to other positions in the editor.

See also

- % Chapter 1.3.5.3.2 “Positioning elements, adapting size and layer” on page 1113
Element property 'Center': The properties contain fixed values for the coordinates of the point of rotation. This point of rotation is shown as the symbol. The point is used as the center for rotating and scaling.

<table>
<thead>
<tr>
<th>“X”</th>
<th>X-coordinate of the point of rotation</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Y”</td>
<td>Y-coordinate of the point of rotation</td>
</tr>
</tbody>
</table>

You can also change the values by dragging the symbols (φ) to other positions in the editor.

Element property 'Text properties': The properties contain fixed values for the text properties.

<table>
<thead>
<tr>
<th>“Font”</th>
<th>Example: “Default”</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>☀️: The “Font” dialog box opens.</td>
</tr>
<tr>
<td></td>
<td>⊖: Drop-down list with style fonts.</td>
</tr>
<tr>
<td>“Font color”</td>
<td>Example: “Black”</td>
</tr>
<tr>
<td></td>
<td>☀️: The “Color” dialog box opens.</td>
</tr>
<tr>
<td></td>
<td>⊖: Drop-down list with style colors.</td>
</tr>
<tr>
<td>“Transparency”</td>
<td>Whole number (value range from 0 to 255). This determines the transparency of the respective color.</td>
</tr>
<tr>
<td></td>
<td>Example: 255: The color is opaque.</td>
</tr>
<tr>
<td></td>
<td>0: The color is completely transparent.</td>
</tr>
<tr>
<td></td>
<td>Please note: If the color is a style color and already has a transparency value, then this property is write-protected.</td>
</tr>
</tbody>
</table>

| “Acknowledge variable” | A rising edge of this variable acknowledges the alarm. |

Handling of multiple active alarms

<table>
<thead>
<tr>
<th>“Automatic switch”</th>
<th>☑️: The display in the alarm banner is switched automatically according to the time to the next alarm as configured in “Every N second”.</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Every N second”</td>
<td>Time period until the next switching. Available only if “Automatic switch” is selected.</td>
</tr>
<tr>
<td>“Next alarm”</td>
<td>Variable for switching to the next alarm. Available only if “Automatic switch” is not selected.</td>
</tr>
<tr>
<td>“Previous alarm”</td>
<td>Variable for switching to the previous alarm. Available only if “Automatic switch” is not selected.</td>
</tr>
<tr>
<td>“Multiple alarms active”</td>
<td>Variable that has the value TRUE if multiple alarms are active.</td>
</tr>
</tbody>
</table>

Element property 'Absolute movement': The properties contain IEC variables for controlling the position of the element dynamically. The reference point is the upper left corner of the element. In runtime mode, the entire element is moved.
### “Movement”

| **“X”** | Variable (numeric data type). Defines the X position (in pixels).  
Example: PLC_PRG.iPos_X.  
Increasing this value in runtime mode moves the element to the right. |
| --- | --- |
| **“Y”** | Variable (numeric data type). Defines the Y position (in pixels).  
Example: PLC_PRG.iPos_Y.  
Increasing this value in runtime mode moves the element downwards. |
| **“Rotation”** | Variable (numeric data type). Defines the angle of rotation (in degrees).  
Example: PLC_PRG.iAngle1.  
The midpoint of the element rotates at the “Center” point. This rotation point is shown as the symbol.  
In runtime mode, the alignment of the element remains the same with respect to the coordinate system of the visualization. Increasing the value rotates the element to the right. |
| **“Interior rotation”** | Variable (numeric data type). Defines the angle of rotation (in degrees).  
Example: PLC_PRG.iAngle2.  
In runtime mode, the element rotates about the point of rotation specified in “Center” according to the value of the variable. In addition, the alignment of the element rotates according to the coordinate system of the visualization. Increasing the value in the code rotates clockwise.  
The rotation point is shown as the symbol.  
Note: If a static angle of rotation is specified in the “Position ➔ Angle” property, then the static angle of rotation is added to the variable angle of rotation (offset) when the visualization is executed. |

You can link the variables to a unit conversion.

The “X”, “Y”, “Rotation”, and “Interior rotation” properties are supported by the “Client Animation” functionality.

See also
- Unit conversion
The variables control the element behavior dynamically.

**"Invisible"**

<table>
<thead>
<tr>
<th>Variable (BOOL)</th>
<th>Toggles the visibility of the element.</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRUE</td>
<td>The element is not visible at runtime.</td>
</tr>
</tbody>
</table>

The "Invisible" property is supported by the "Client Animation" functionality.

See also

- 

These properties are available only when you select the "Preview: Support client animations and overlay of native elements" option in the Visualization Manager.

You can animate a movement, a rotation, and the visibility of a visualization element.

<table>
<thead>
<tr>
<th>&quot;Animation duration&quot;</th>
<th>Time that the element executes an animation (in milliseconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example: 500</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>&quot;Move to the foreground&quot;</th>
<th>Property value (BOOL)</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRUE</td>
<td>At runtime, the element is displayed in the foreground.</td>
</tr>
<tr>
<td>FALSE</td>
<td>At runtime, the element is displayed in the layer where it was inserted in the visualization editor.</td>
</tr>
</tbody>
</table>

See also

- Chapter 1.3.5.21.5 “Using client animation” on page 1962

**Element property 'Access rights'**

Requirement: User management is set up for the visualization.

<table>
<thead>
<tr>
<th>&quot;Access rights&quot;</th>
<th>Opens the &quot;Access rights&quot; dialog. There you can edit the access privileges for the element.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Status messages:</td>
<td>- &quot;Not set. Full rights.&quot;: Access rights for all user groups: &quot;operable&quot;</td>
</tr>
<tr>
<td></td>
<td>- &quot;Rights are set: Limited rights&quot;: Access is restricted for at least one group.</td>
</tr>
</tbody>
</table>

See also

- Chapter 1.3.5.19.3.1 “Dialog 'Access Rights’” on page 1583

See also

- Chapter 1.3.5.7 “Visualizing alarm management” on page 1146

**Visualization element 'Bar display'**

Symbol:
Tag: “Measurement controls”
The element displays the value of a variable.

See also

- Element properties

<table>
<thead>
<tr>
<th>“Element name”</th>
<th>Example: GenElemInst_2</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Type of element”</td>
<td>“Bar display”</td>
</tr>
<tr>
<td>“Value”</td>
<td>Variable (numeric data type)</td>
</tr>
<tr>
<td></td>
<td>The value of the variable is displayed as a bar length.</td>
</tr>
</tbody>
</table>

**Element property ‘Center’**

The properties contain fixed values for the coordinates of the point of rotation. This point of rotation is shown as the symbol. The point is used as the center for rotating and scaling.

<table>
<thead>
<tr>
<th>“X”</th>
<th>X-coordinate of the point of rotation</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Y”</td>
<td>Y-coordinate of the point of rotation</td>
</tr>
</tbody>
</table>

You can also change the values by dragging the symbols (⃝) to other positions in the editor.

These properties are available only when you select the “Preview: Support client animations and overlay of native elements” option in the Visualization Manager.

You can animate a movement, a rotation, and the visibility of a visualization element.

<table>
<thead>
<tr>
<th>“Animation duration”</th>
<th>Time that the element executes an animation (in milliseconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example: 500</td>
<td></td>
</tr>
<tr>
<td>“Move to the foreground”</td>
<td>Property value (BOOL)</td>
</tr>
<tr>
<td>TRUE: At runtime, the element is displayed in the foreground.</td>
<td></td>
</tr>
<tr>
<td>FALSE: At runtime, the element is displayed in the layer where it was inserted in the visualization editor.</td>
<td></td>
</tr>
</tbody>
</table>

See also

- § Chapter 1.3.5.21.5 “Using client animation” on page 1962

**Element property ‘Position’**

The position defines the location and size of the element in the visualization window. These are based on the Cartesian coordinate system. The origin is located at the upper left corner of the window. The positive horizontal x-axis runs to the right. The positive vertical y-axis runs downwards.
### Element property 'Background'

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>X coordinate of the upper left corner of the element. Specified in pixels. Example: 10.</td>
</tr>
<tr>
<td>Y</td>
<td>Y coordinate of the upper left corner of the element. Specified in pixels. Example: 10.</td>
</tr>
<tr>
<td>Width</td>
<td>Specified in pixels. Example: 150</td>
</tr>
<tr>
<td>Height</td>
<td>Specified in pixels. Example: 30</td>
</tr>
</tbody>
</table>

You can also change the values by dragging the box symbols (_drag_ symbol) to other positions in the editor.

See also
- Chapter 1.3.5.3.2 “Positioning elements, adapting size and layer” on page 1113

#### Background color
Drop-down list with background colors.

Note: This property depends on the style. For example, there are no heterochromatic background images for "FlatStyle" and "Whitestyle".

#### Own image
- "image": Image ID of the background image. You select the background image from an image pool by clicking the image symbol.
  - Info: If you specify the "<default>" value or select the image from the "Default" category in the input assistant, then the original element background image is used.
- "Transparent color": Color of pixels that are displayed as transparent. Selection from drop-down list or input assistant.

#### Optimized drawing
- [✓]: The background image is drawn one time. If there is a change in the foreground, then only the affected part of the image is redrawn.
- [☐]: The background image is redrawn in cycles.

Note: Deactivating this option is sensible only in certain exceptional cases.

### Element property 'Bar'

#### Diagram type
- "Scale besides bar"
- "Scale in bar"
- "Bar in scale"
- "No scale"

#### Orientation
Orientation depending on the ratio of width to height of the bar display:
- "Horizontal"
- "Vertical"
### “Running direction”
Direction the values are increased.  
Drop-down list for “Orientation Horizontal”:  
- “Left to right”  
- “Right to left”  
Drop-down list for “Orientation Vertical”:  
- “Bottom to top”  
- “Top to bottom”

### “Optimum size for bar”
- The bar width requires the majority of the element surface.  
Note: This property depends on the style. It is not provided for “FlatStyle” or “WhiteStyle”.

---

#### Element property 'Scale'

### “Scale start”
Least value of the scale and the lower limit of the value range for the element.  
Example: 0  
- The property “Variable” is shown below.

### “Variable”
Variable (integer data type). Contains the scale start.  
Example: PLC_PRG.iScaleStart  
Declaration:  
```
PROGRAM PLC_PRG
VAR
  iScaleStart : INT := 0;
END_VAR
```

### “Scale end”
Greatest value of the scale and the upper limit of the value range for the element.  
Example: 100  
- The property “Variable” is shown below.

### “Variable”
Variable (integer data type). Contains the scale end.  
Example: PLC_PRG.iScaleEnd  
Declaration:  
```
PROGRAM PLC_PRG
VAR
  iScaleEnd : INT := 120;
END_VAR
```

### “Main scale”
Distance between 2 values on the rough scale.  
Example: 10  
- The property “Variable” is shown below.

### “Variable”
Variable (integer data type). Contains the distance.  
Example: PLC_PRG.iMainScale  
Declaration:  
```
PROGRAM PLC_PRG
VAR
  iMainScale : INT := 20;
END_VAR
```
**Subscale**

Distance between 2 values on the fine scale.
You can hide the fine scale by setting the value to 0.
Example: 2
φ: The property “Variable” is shown below.

**Variable**

Variable (integer data type). Contains the spacing.
Example: PLC_PRG.iSubScale

Declaration:

```plaintext
PROGRAM PLC_PRG
VAR
   iSubScale : INT := 5;
END_VAR
```

**Scale line width**

Specified in pixels.
Example: 3

**Scale color**

Color of scale lines
- ⚫️: The “Color” dialog box opens.
- ⬇️: A drop-down list with color names opens.

**Scale in 3D**

☑️: Tick marks are displayed with slight 3D shadowing.
Note: This property depends on the style. Not available for “FlatStyle”.

**Element frame**

☑️: A frame is drawn around the element.

---

**Element property 'Label'**

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Unit</strong></td>
<td>Text that is displayed in the element. Example: Units displayed in m/s.</td>
</tr>
<tr>
<td><strong>Font</strong></td>
<td>Font for labels (example: scale numbering). Selection from the drop-down list or by clicking the “…” button.</td>
</tr>
</tbody>
</table>
| **Scale format (C Syntax)”** | Values scaled in "printf" syntax  
Example: %d, %5.2f |
| **Max. text width of labels”** | (optional) Value that redefines the maximum width of the scale label. The correct value is normally set automatically.  
Note: Change this value only if the automatic adjustment does not yield the expected result. |
| **Text height of labels”** | (optional) Value that redefines the maximum height of the scale label. The correct value is normally set automatically.  
Note: Change this value only if the automatic adjustment does not yield the expected result. |
| **Font color”**   | Selection from the drop-down list or by clicking the “…” button. |

---

**Element property 'Positioning'**
### Element properties 'Colors'

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>“Graph color”</strong></td>
<td>Color of the bar</td>
</tr>
<tr>
<td><strong>“Bar background”</strong></td>
<td>- ![Black]: The background of the bar is black.</td>
</tr>
<tr>
<td></td>
<td>- ![White]: The background of the bar is white.</td>
</tr>
<tr>
<td><strong>“Frame color”</strong></td>
<td>Color that the frames are drawn.</td>
</tr>
<tr>
<td><strong>“Switch whole color”</strong></td>
<td>- ![Switched]: The total color of the bar is switched to the color of the color area of the current value.</td>
</tr>
<tr>
<td><strong>“Use gradient color for bar”</strong></td>
<td>- ![Gradient]: Bar is displayed with a gradient.</td>
</tr>
<tr>
<td><strong>“Color range markers”</strong></td>
<td>The color areas can be separated from each other inside the bar with a vertical mark.</td>
</tr>
<tr>
<td></td>
<td>- “No markers”: No display.</td>
</tr>
<tr>
<td></td>
<td>- “Marker forwards”: The color of the vertical mark corresponds to the color of the previous color area.</td>
</tr>
<tr>
<td></td>
<td>- “Marker backwards”: The color of the vertical mark corresponds to the color of the next color area.</td>
</tr>
<tr>
<td><strong>“Color areas”</strong></td>
<td></td>
</tr>
<tr>
<td><strong>“Create new”</strong></td>
<td>A new color area is added.</td>
</tr>
<tr>
<td><strong>“Delete”</strong></td>
<td>The color area is removed from the list.</td>
</tr>
<tr>
<td><strong>“Begin of area”</strong></td>
<td>Start value of the color area</td>
</tr>
<tr>
<td><strong>“End of area”</strong></td>
<td>End value of the color area</td>
</tr>
<tr>
<td><strong>“Color”</strong></td>
<td>Color that is used for displaying the area.</td>
</tr>
</tbody>
</table>

### Element property 'Absolute movement'

The properties contain IEC variables for controlling the position of the element dynamically. The reference point is the upper left corner of the element. In runtime mode, the entire element is moved.
“Movement”

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>“X”</td>
<td>Variable (numeric data type). Defines the X position (in pixels).</td>
</tr>
<tr>
<td></td>
<td>Example: PLC_PRG.iPos_X.</td>
</tr>
<tr>
<td></td>
<td>Increasing this value in runtime mode moves the element to the right.</td>
</tr>
<tr>
<td>“Y”</td>
<td>Variable (numeric data type). Defines the Y position (in pixels).</td>
</tr>
<tr>
<td></td>
<td>Example: PLC_PRG.iPos_Y.</td>
</tr>
<tr>
<td></td>
<td>Increasing this value in runtime mode moves the element downwards.</td>
</tr>
<tr>
<td>“Rotation”</td>
<td>Variable (numeric data type). Defines the angle of rotation (in degrees).</td>
</tr>
<tr>
<td></td>
<td>Example: PLC_PRG.iAngle1.</td>
</tr>
<tr>
<td></td>
<td>The midpoint of the element rotates at the “Center” point.</td>
</tr>
<tr>
<td></td>
<td>This rotation point is shown as the symbol.</td>
</tr>
<tr>
<td></td>
<td>In runtime mode, the alignment of the element remains the same with respect</td>
</tr>
<tr>
<td></td>
<td>to the coordinate system of the visualization. Increasing the value</td>
</tr>
<tr>
<td></td>
<td>rotates the element to the right.</td>
</tr>
<tr>
<td>“Interior rotation”</td>
<td>Variable (numeric data type). Defines the angle of</td>
</tr>
<tr>
<td></td>
<td>Example: PLC_PRG.iAngle2.</td>
</tr>
<tr>
<td></td>
<td>In runtime mode, the element rotates about the point of rotation specified</td>
</tr>
<tr>
<td></td>
<td>in “Center” according to the value of the variable. In addition, the</td>
</tr>
<tr>
<td></td>
<td>alignment of the element rotates according to the coordinate system of the</td>
</tr>
<tr>
<td></td>
<td>visualization. Increasing the value in the code rotates clockwise.</td>
</tr>
<tr>
<td></td>
<td>The rotation point is shown as the symbol.</td>
</tr>
<tr>
<td></td>
<td>Note: If a static angle of rotation is specified in the</td>
</tr>
<tr>
<td></td>
<td>“Position ➔ Angle” property, then the static angle of rotation is added to</td>
</tr>
<tr>
<td></td>
<td>the variable angle of rotation (offset) when the visualization is executed.</td>
</tr>
</tbody>
</table>

You can link the variables to a unit conversion.

The “X”, “Y”, “Rotation”, and “Interior rotation” properties are supported by the “Client Animation” functionality.

See also
- Unit conversion
**Element property 'State variables'**

The variables control the element behavior dynamically.

| “Deactivate inputs” | Variable (BOOL). Toggles the operability of the element. TRUE: User inputs do not have any effect in runtime more. The element is shown as deactivated. |

**Element property 'Access rights'**

Requirement: User management is set up for the visualization.

| “Access rights” | Opens the “Access rights” dialog. There you can edit the access privileges for the element. Status messages: ● “Not set. Full rights.”: Access rights for all user groups: “operable” ● “Rights are set: Limited rights”: Access is restricted for at least one group. |

See also
- % Chapter 1.3.5.19.3.1 “Dialog 'Access Rights'” on page 1583

**Visualization element 'Meter 90°'**

Symbol:

![Meter 90° symbol]

Tag: “Measurement controls”

The element displays the value of a variable. The needle is positioned according to the value of the assigned variable. A meter is used to represent a tachometer, for example.

**Element properties**

| “Element name” | Example: GenElemInst_1 |
| "Type of element" | “Meter 90°” |
| "Value" | Variable (numeric data type) The variable value determines the pointer movement of the element. |

These properties are available only when you select the “Preview: Support client animations and overlay of native elements” option in the Visualization Manager.

You can animate a movement, a rotation, and the visibility of a visualization element.
### Animation duration

Time that the element executes an animation (in milliseconds)

**Example:** 500

### Move to the foreground

Property value (BOOL)

- **TRUE:** At runtime, the element is displayed in the foreground.
- **FALSE:** At runtime, the element is displayed in the layer where it was inserted in the visualization editor.

### See also

- Chapter 1.3.5.21.5 “Using client animation” on page 1962

### Element property 'Position'

The position defines the location and size of the element in the visualization window. These are based on the Cartesian coordinate system. The origin is located at the upper left corner of the window. The positive horizontal x-axis runs to the right. The positive vertical y-axis runs downwards.

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
</table>
| **X** | X coordinate of the upper left corner of the element  
Specified in pixels.  
**Example:** 10. |
| **Y** | Y coordinate of the upper left corner of the element  
Specified in pixels.  
**Example:** 10. |
| **Width** | Specified in pixels.  
**Example:** 150 |
| **Height** | Specified in pixels.  
**Example:** 30 |

You can also change the values by dragging the box symbols to other positions in the editor.

### See also

- Chapter 1.3.5.3.2 “Positioning elements, adapting size and layer” on page 1113

### Element property 'Background'

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Image color</strong></td>
<td>List box containing background colors</td>
</tr>
<tr>
<td><strong>Own image</strong></td>
<td></td>
</tr>
</tbody>
</table>
- **Image**: ID of the background image.  
  You select the background image from an image pool by clicking  
  Info: If you specify the value "<default>" or select the image from the "Default" category in the Input Assistant, then the original element background image is used.  
- **Transparency color**: Selection from list box or Input Assistant. |
### Element property 'Arrow'

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>&quot;Hand style&quot;</strong></td>
<td>Drop-down list with different arrow types</td>
</tr>
<tr>
<td><strong>&quot;Color&quot;</strong></td>
<td>- [ ] The &quot;Color&quot; dialog box opens.</td>
</tr>
<tr>
<td></td>
<td>- ▼: Drop-down list with color names</td>
</tr>
<tr>
<td><strong>&quot;Angle range&quot;</strong></td>
<td>Drop-down list for the alignment of the element</td>
</tr>
<tr>
<td><strong>&quot;Additional arrow&quot;</strong></td>
<td>[ ]: An additional arrow is shown inside the scale.</td>
</tr>
</tbody>
</table>

### Element property 'Scale'

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>&quot;Sub scale position&quot;</strong></td>
<td>- &quot;Outside&quot;: The subscale is displayed on the outer scale ring. (&quot;Frame outside&quot;)</td>
</tr>
<tr>
<td></td>
<td>- &quot;Inside&quot;: The subscale is displayed on the inner scale ring. (&quot;Frame inside&quot;)</td>
</tr>
<tr>
<td><strong>&quot;Scale type&quot;</strong></td>
<td>Type of scale</td>
</tr>
<tr>
<td></td>
<td>- &quot;Lines&quot;</td>
</tr>
<tr>
<td></td>
<td>- &quot;Dots&quot;</td>
</tr>
<tr>
<td></td>
<td>- &quot;Squares&quot;</td>
</tr>
<tr>
<td><strong>&quot;Scale start&quot;</strong></td>
<td>Least value of the scale and the lower limit of the value range for the element</td>
</tr>
<tr>
<td></td>
<td>Example: 0</td>
</tr>
<tr>
<td></td>
<td>♦: The &quot;Variable&quot; property is displayed in the line below this.</td>
</tr>
<tr>
<td><strong>&quot;Variable&quot;</strong></td>
<td>Variable (integer data type). Contains the scale start</td>
</tr>
<tr>
<td></td>
<td>Example: PLC_PRG.iScaleStart</td>
</tr>
<tr>
<td></td>
<td>Declaration: PROGRAM PLC_PRG</td>
</tr>
<tr>
<td></td>
<td>VAR</td>
</tr>
<tr>
<td></td>
<td>iScaleStart : INT := 0;</td>
</tr>
<tr>
<td></td>
<td>END_VAR</td>
</tr>
<tr>
<td><strong>&quot;Scale end&quot;</strong></td>
<td>Greatest value of the scale and the upper limit of the value range for the element</td>
</tr>
<tr>
<td></td>
<td>Example: 100</td>
</tr>
<tr>
<td></td>
<td>♦: The &quot;Variable&quot; property is shown below this.</td>
</tr>
<tr>
<td><strong>&quot;Variable&quot;</strong></td>
<td>Variable (integer data type). Contains the scale end</td>
</tr>
<tr>
<td></td>
<td>Example: PLC_PRG.iScaleEnd</td>
</tr>
<tr>
<td></td>
<td>Declaration: PROGRAM PLC_PRG</td>
</tr>
<tr>
<td></td>
<td>VAR</td>
</tr>
<tr>
<td></td>
<td>iScaleEnd : INT := 120;</td>
</tr>
<tr>
<td></td>
<td>END_VAR</td>
</tr>
<tr>
<td><strong>&quot;Main scale&quot;</strong></td>
<td>Distance between two values on the main scale</td>
</tr>
<tr>
<td></td>
<td>Example: 10</td>
</tr>
<tr>
<td></td>
<td>♦: The &quot;Variable&quot; property is shown below.</td>
</tr>
</tbody>
</table>
### Variable

Variable (integer data type) Contains the distance between two values on the main scale

**Example:** PLC_PRG.iMainScale

**Declaration:**

```
PROGRAM PLC_PRG
VAR
  iMainScale : INT := 20;
END_VAR
```

### Sub scale

Distance between two values on the fine scale

You can hide the fine scale by setting the value to 0.

**Example:** 2

*The “Variable” property is shown below this.

### Variable

Variable (integer data type) Contains the distance between two values on the fine scale

**Example:** PLC_PRG.iSubScale

**Declaration:**

```
PROGRAM PLC_PRG
VAR
  iSubScale : INT := 5;
END_VAR
```

### Scale line width

Specified in pixels

**Example:** 3

### Scale color

Color of scale lines

- [ ] The “Color” dialog opens.
- [ ] A list box with style colors opens.

### Scale in 3D

*Scale lines are displayed with soft 3D shadowing.*

Note: This property is not displayed in “FlatStyle”.

### Show scale

*The scale is displayed.*

### Frame inside

* A frame is drawn at the inner end of the scale.

### Frame outside

* A frame is drawn at the outer end of the scale.

---

**Element property 'Label'**

### Label

Selection list

- “Outside”: Scale values are placed outside of the scale.
- “Inside”: Scale values are placed inside of the scale.

### Unit

Text that is displayed in the element.

**Example:** Units displayed in m/s.

### Font

Font for labels (example: scale numbering).

Selection from the drop-down list or by clicking the “” button.

### Scale format (C Syntax)

Values scaled in "printf" syntax

**Examples:** %d, %5.2f
### “Max. text width of labels”

(ground) Value that redefines the maximum width of the scale label. The correct value is normally set automatically.

Note: Change this value only if the automatic adjustment does not yield the expected result.

### “Text height of labels”

(ground) Value that redefines the maximum height of the scale label. The correct value is normally set automatically.

Note: Change this value only if the automatic adjustment does not yield the expected result.

### “Font color”

Selection from the drop-down list or by clicking the button.

---

**Element property ‘Positioning’**

| “Usage of”            |  ● “Preset style values”: Values from the current style  
<table>
<thead>
<tr>
<th></th>
<th>● “User-defined settings”: The subnode “Positioning” appears.</th>
</tr>
</thead>
</table>

**“Positioning”**

Requirement: “User-defined settings” is selected as “Usage of”.

The displayed positioning settings depend on the type of needle instrument and potentiometer, and partially on whether a custom background image is selected. The following settings are used for achieving the exact position relative to the background image.

<table>
<thead>
<tr>
<th>“Needle movement”</th>
<th>Length of the needle (in pixels)</th>
</tr>
</thead>
</table>
| “Scale movement”  | Distance from the tick marks to the center (in pixels)  
| Requirement: A customer image is selected as “Background”.  |
| “Scale length”    | Length of the tick marks (in pixels)  
| Requirement: A customer image is selected as “Background”.  |
| “Label offset”    | Distance from the labels to the tick marks (in pixels) |
| “Unit offset”     | Distance of the unit text “Label ➔ Unit” from the upper scale edge (in pixels) |
| “Origin offset”   | Offset of the element (in pixels)  
| Requirement: For the elements “Meter 180°” and “Meter 90°”, this property is displayed only if a custom image is selected as “Background”.  |

---

**Element property ‘Colors’**

| “Color areas”       |  }
|---------------------|---------------------------------------------------------------------------------------------------|
| “Durable color areas” |  □: All color areas are visible, regardless of the current value.  
|                     |  □: Only the color area is visible that includes the current value.  |
| “Use colors for scale” |  □: Colors in the color area are used only for the scale and frame.  |

| “Color areas”       |  }
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>“Create new”</td>
<td>A new color area is added to the “Elements” view.</td>
</tr>
<tr>
<td>“Delete”</td>
<td>The color area is removed from the list and the list is refreshed.</td>
</tr>
</tbody>
</table>
| “Begin of area”     | Start value of the color area  
| Example: 20         |  ▶: The property “Variable” is shown below.  |
**“Variable”**  
Variable (integer data type). Contains the start value.  
Example: PLC_PRG.iColorAreaStart0  
Declaration:  
```plaintext
PROGRAM PLC_PRG
VAR
  iColorAreaStart0 : INT := 80;
END_VAR
```

**“End of area”**  
End value of the color area  
Example: 120  
♦️: The property “Variable” is shown below.

**“Variable”**  
Variable (integer data type). Contains the end value.  
Example: iColorAreaEnd0  
Declaration:  
```plaintext
PROGRAM PLC_PRG
VAR
  iColorAreaEnd0 : INT := 100;
END_VAR
```

**“Color”**  
Color that is used for displaying the area.

---

**Element property 'Absolute movement'**  
The properties contain IEC variables for controlling the position of the element dynamically. The reference point is the upper left corner of the element. In runtime mode, the entire element is moved.

**“Movement”**

**“X”**  
Variable (numeric data type). Defines the X position (in pixels).  
Example: PLC_PRG.iPos_X.  
Increasing this value in runtime mode moves the element to the right.

**“Y”**  
Variable (numeric data type). Defines the Y position (in pixels).  
Example: PLC_PRG.iPos_Y.  
Increasing this value in runtime mode moves the element downwards.

**“Rotation”**  
Variable (numeric data type). Defines the angle of rotation (in degrees).  
Example: PLC_PRG.iAngle1.  
The midpoint of the element rotates at the “Center” point. This rotation point is shown as the ♦️ symbol.  
In runtime mode, the alignment of the element remains the same with respect to the coordinate system of the visualization. Increasing the value rotates the element to the right.
"Interior rotation" Variable (numeric data type). Defines the angle of rotation (in degrees).

Example: PLC_PRG.iAngle2.

In runtime mode, the element rotates about the point of rotation specified in "Center" according to the value of the variable. In addition, the alignment of the element rotates according to the coordinate system of the visualization. Increasing the value in the code rotates clockwise.

The rotation point is shown as the symbol.

Note: If a static angle of rotation is specified in the "Position ➔ Angle" property, then the static angle of rotation is added to the variable angle of rotation (offset) when the visualization is executed.

You can link the variables to a unit conversion.

The “X”, “Y”, “Rotation”, and “Interior rotation” properties are supported by the "Client Animation" functionality.

See also
- Unit conversion

Element property 'Access rights'

Requirement: User management is set up for the visualization.

"Access rights" Opens the “Access rights” dialog. There you can edit the access privileges for the element.

Status messages:
- “Not set. Full rights.”: Access rights for all user groups : "operable"
- “Rights are set: Limited rights”: Access is restricted for at least one group.

See also
- % Chapter 1.3.5.19.3.1 “Dialog ‘Access Rights’” on page 1583

Visualization element 'Meter 180°'

Symbol:

Tag: “Measurement controls”
The element displays the value of a variable. The needle is positioned according to the value of the assigned variable on a scale. A meter is used to represent a tachometer, for example.

**Element properties**

<table>
<thead>
<tr>
<th>“Element name”</th>
<th>Example: GenElemInst_1</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Type of element”</td>
<td>“Meter 180°”</td>
</tr>
<tr>
<td>“Value”</td>
<td>Variable (numeric data type)</td>
</tr>
<tr>
<td></td>
<td>The variable value determines the pointer movement of the element.</td>
</tr>
</tbody>
</table>

**Element property ‘Center’**

The properties contain fixed values for the coordinates of the point of rotation. This point of rotation is shown as the + symbol. The point is used as the center for rotating and scaling.

<table>
<thead>
<tr>
<th>“X”</th>
<th>X-coordinate of the point of rotation</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Y”</td>
<td>Y-coordinate of the point of rotation</td>
</tr>
</tbody>
</table>

You can also change the values by dragging the symbols (+) to other positions in the editor.

These properties are available only when you select the “Preview: Support client animations and overlay of native elements” option in the Visualization Manager.

You can animate a movement, a rotation, and the visibility of a visualization element.

<table>
<thead>
<tr>
<th>“Animation duration”</th>
<th>Time that the element executes an animation (in milliseconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Example: 500</td>
</tr>
<tr>
<td>“Move to the foreground”</td>
<td>Property value (BOOL)</td>
</tr>
<tr>
<td></td>
<td>TRUE: At runtime, the element is displayed in the foreground.</td>
</tr>
<tr>
<td></td>
<td>FALSE: At runtime, the element is displayed in the layer where it was inserted in the visualization editor.</td>
</tr>
</tbody>
</table>

See also

- Chapter 1.3.5.21.5 “Using client animation” on page 1962

**Element property ‘Position’**

The position defines the location and size of the element in the visualization window. These are based on the Cartesian coordinate system. The origin is located at the upper left corner of the window. The positive horizontal x-axis runs to the right. The positive vertical y-axis runs downwards.
“X”
X coordinate of the upper left corner of the element
Specified in pixels.
Example: 10.

“Y”
Y coordinate of the upper left corner of the element
Specified in pixels.
Example: 10.

“Width”
Specified in pixels.
Example: 150

“Height”
Specified in pixels.
Example: 30

You can also change the values by dragging the box symbols (_drag_ ) to other positions in the editor.

See also
● Chapter 1.3.5.3.2 “Positioning elements, adapting size and layer” on page 1113

Element property ‘Background’

“Image color”
List box containing background colors

“Own image”
● “Image”: ID of the background image.
  You select the background image from an image pool by clicking .
  Info: If you specify the value “<default>” or select the image from the “Default” category in the Input Assistant, then the original element background image is used.
● “Transparency color”: Selection from list box or Input Assistant.

Element property ‘Arrow’

“Hand style”
Drop-down list with different arrow types

“Color”
● _[ ]_: The “Color” dialog box opens.
● _[ ]_: Drop-down list with color names

“Angle range”
Drop-down list for the alignment of the element

“Additional arrow”
☑ An additional arrow is shown inside the scale.

Element property ‘Scale’
<table>
<thead>
<tr>
<th><strong>Sub scale position</strong></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>● “Outside”: The subscale is displayed on the outer scale ring. (“Frame outside”)</td>
<td></td>
</tr>
<tr>
<td>● “Inside”: The subscale is displayed on the inner scale ring. (“Frame inside”)</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Scale type</strong></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Type of scale</td>
<td></td>
</tr>
<tr>
<td>● “Lines”</td>
<td></td>
</tr>
<tr>
<td>● “Dots”</td>
<td></td>
</tr>
<tr>
<td>● “Squares”</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Scale start</strong></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Least value of the scale and the lower limit of the value range for the element</td>
<td></td>
</tr>
<tr>
<td>Example: 0</td>
<td></td>
</tr>
</tbody>
</table>

*The “Variable” property is displayed in the line below this.*

<table>
<thead>
<tr>
<th><strong>Variable</strong></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Variable (integer data type). Contains the scale start</td>
<td></td>
</tr>
<tr>
<td>Example: PLC_PRG.iScaleStart</td>
<td></td>
</tr>
</tbody>
</table>

**Declaration:**

```
PROGRAM PLC_PRG
VAR
  iScaleStart : INT := 0;
END_VAR
```

<table>
<thead>
<tr>
<th><strong>Scale end</strong></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Greatest value of the scale and the upper limit of the value range for the element</td>
<td></td>
</tr>
<tr>
<td>Example: 100</td>
<td></td>
</tr>
</tbody>
</table>

*The “Variable” property is shown below this.*

<table>
<thead>
<tr>
<th><strong>Variable</strong></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Variable (integer data type). Contains the scale end</td>
<td></td>
</tr>
<tr>
<td>Example: PLC_PRG.iScaleEnd</td>
<td></td>
</tr>
</tbody>
</table>

**Declaration:**

```
PROGRAM PLC_PRG
VAR
  iScaleEnd : INT := 120;
END_VAR
```

<table>
<thead>
<tr>
<th><strong>Main scale</strong></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Distance between two values on the main scale</td>
<td></td>
</tr>
<tr>
<td>Example: 10</td>
<td></td>
</tr>
</tbody>
</table>

*The “Variable” property is shown below.*

<table>
<thead>
<tr>
<th><strong>Variable</strong></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Variable (integer data type) Contains the distance between two values on the main scale</td>
<td></td>
</tr>
<tr>
<td>Example: PLC_PRG.iMainScale</td>
<td></td>
</tr>
</tbody>
</table>

**Declaration:**

```
PROGRAM PLC_PRG
VAR
  iMainScale : INT := 20;
END_VAR
```

<table>
<thead>
<tr>
<th><strong>Sub scale</strong></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Distance between two values on the fine scale</td>
<td></td>
</tr>
<tr>
<td>You can hide the fine scale by setting the value to 0.</td>
<td></td>
</tr>
<tr>
<td>Example: 2</td>
<td></td>
</tr>
</tbody>
</table>

*The “Variable” property is shown below this.*
Variable

Variable (integer data type) Contains the distance between two values on the fine scale
Example: PLC_PRG.iSubScale
Declaration:

```plaintext
PROGRAM PLC_PRG
VAR
  iSubScale : INT := 5;
END_VAR
```

Scale line width

Specified in pixels
Example: 3

Scale color

Color of scale lines
-  <textarea></textarea>: The “Color” dialog opens.
-   <*>: A list box with style colors opens.

Scale in 3D

☑️: Scale lines are displayed with soft 3D shadowing.
Note: This property is not displayed in “FlatStyle”.

Show scale

☑️: The scale is displayed.

Frame inside

☑️: A frame is drawn at the inner end of the scale.

Frame outside

☑️: A frame is drawn at the outer end of the scale.

Element property 'Label'

Label

Selection list
-  “Outside”: Scale values are placed outside of the scale.
-  “Inside”: Scale values are placed inside of the scale.

Unit

Text that is displayed in the element.
Example: Units displayed in m/s.

Font

Font for labels (example: scale numbering).
Selection from the drop-down list or by clicking the “” button.

Scale format (C Syntax)

Values scaled in "printf" syntax
Examples: %d, %5.2f

Max. text width of labels

(optional) Value that redefines the maximum width of the scale label. The correct value is normally set automatically.
Note: Change this value only if the automatic adjustment does not yield the expected result.

Text height of labels

(optional) Value that redefines the maximum height of the scale label. The correct value is normally set automatically.
Note: Change this value only if the automatic adjustment does not yield the expected result.

Font color

Selection from the drop-down list or by clicking the “” button.
"Usage of"
- "Preset style values": Values from the current style
- "User-defined settings": The subnode "Positioning" appears.

"Positioning"
Requirement: "User-defined settings" is selected as "Usage of".
The displayed positioning settings depend on the type of needle instrument and potentiometer, and partially on whether a custom background image is selected. The following settings are used for achieving the exact position relative to the background image.

<table>
<thead>
<tr>
<th>Setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Needle movement&quot;</td>
<td>Length of the needle (in pixels)</td>
</tr>
<tr>
<td>&quot;Scale movement&quot;</td>
<td>Distance from the tick marks to the center (in pixels) Requirement: A customer image is selected as &quot;Background&quot;.</td>
</tr>
<tr>
<td>&quot;Scale length&quot;</td>
<td>Length of the tick marks (in pixels) Requirement: A customer image is selected as &quot;Background&quot;.</td>
</tr>
<tr>
<td>&quot;Label offset&quot;:</td>
<td>Distance from the labels to the tick marks (in pixels)</td>
</tr>
<tr>
<td>&quot;Unit offset&quot;:</td>
<td>Distance of the unit text &quot;Label ➔ Unit&quot; from the upper scale edge (in pixels)</td>
</tr>
<tr>
<td>&quot;Origin offset&quot;</td>
<td>Offset of the element (in pixels) Requirement: For the elements &quot;Meter 180°&quot; and &quot;Meter 90°&quot;, this property is displayed only if a custom image is selected as &quot;Background&quot;.</td>
</tr>
</tbody>
</table>

Element property 'Colors'

"Color areas"

<table>
<thead>
<tr>
<th>Setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Durable color areas&quot;</td>
<td>☐: All color areas are visible, regardless of the current value.</td>
</tr>
<tr>
<td></td>
<td>☑: Only the color area is visible that includes the current value.</td>
</tr>
<tr>
<td>&quot;Use colors for scale&quot;</td>
<td>☑: Colors in the color area are used only for the scale and frame.</td>
</tr>
<tr>
<td>&quot;Create new&quot;</td>
<td>A new color area is added to the &quot;Elements&quot; view.</td>
</tr>
<tr>
<td>&quot;Delete&quot;</td>
<td>The color area is removed from the list and the list is refreshed.</td>
</tr>
</tbody>
</table>

"Begin of area"
Start value of the color area
Example: 20

**: The property "Variable" is shown below.

"Variable"
Variable (integer data type). Contains the start value.
Example: PLC_PRG.iColorAreaStart0
Declarion:

```plc
PROGRAM PLC_PRG
VAR
  iColorAreaStart0 : INT := 80;
END_VAR
```

"End of area"
End value of the color area
Example: 120

**: The property "Variable" is shown below.
<table>
<thead>
<tr>
<th><strong>Variable</strong></th>
<th>Variable (integer data type). Contains the end value. Example: iColorAreaEnd0 Declaration:</th>
</tr>
</thead>
<tbody>
<tr>
<td>PROGRAM PLC_PRG</td>
<td></td>
</tr>
<tr>
<td>VAR</td>
<td></td>
</tr>
<tr>
<td>iColorAreaEnd0 : INT := 100;</td>
<td></td>
</tr>
<tr>
<td>END_VAR</td>
<td></td>
</tr>
</tbody>
</table>

**Color**

Color that is used for displaying the area.

**Element property 'Absolute movement'**

The properties contain IEC variables for controlling the position of the element dynamically. The reference point is the upper left corner of the element. In runtime mode, the entire element is moved.

**Movement**

<table>
<thead>
<tr>
<th><strong>X</strong></th>
<th>Variable (numeric data type). Defines the X position (in pixels). Example: PLC_PRG.iPos_X. Increasing this value in runtime mode moves the element to the right.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Y</strong></td>
<td>Variable (numeric data type). Defines the Y position (in pixels). Example: PLC_PRG.iPos_Y. Increasing this value in runtime mode moves the element downwards.</td>
</tr>
</tbody>
</table>

**Rotation**


The midpoint of the element rotates at the “Center” point. This rotation point is shown as the ☞ symbol.

In runtime mode, the alignment of the element remains the same with respect to the coordinate system of the visualization. Increasing the value rotates the element to the right.

**Interior rotation**


In runtime mode, the element rotates about the point of rotation specified in “Center” according to the value of the variable. In addition, the alignment of the element rotates according to the coordinate system of the visualization. Increasing the value in the code rotates clockwise.

The rotation point is shown as the ☞ symbol.

Note: If a static angle of rotation is specified in the “Position ☞ Angle” property, then the static angle of rotation is added to the variable angle of rotation (offset) when the visualization is executed.
You can link the variables to a unit conversion.

The “X”, “Y”, “Rotation”, and “Interior rotation” properties are supported by the “Client Animation” functionality.

See also
- Unit conversion

Element property 'State variables'

<table>
<thead>
<tr>
<th>“Invisible”</th>
<th>Variable (BOOL). Toggles the visibility of the element.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>TRUE: The element is not visible at runtime.</td>
</tr>
</tbody>
</table>

The “Invisible” property is supported by the “Client Animation” functionality.

See also
- 

Element property 'Access rights'

Requirement: User management is set up for the visualization.

<table>
<thead>
<tr>
<th>“Access rights”</th>
<th>Opens the “Access rights” dialog. There you can edit the access privileges for the element.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Status messages:</td>
</tr>
<tr>
<td></td>
<td>- “Not set. Full rights.”: Access rights for all user groups: “operable”</td>
</tr>
<tr>
<td></td>
<td>- “Rights are set: Limited rights”: Access is restricted for at least one group.</td>
</tr>
</tbody>
</table>

See also
- ☞ Chapter 1.3.5.19.3.1 “Dialog ‘Access Rights’” on page 1583

See also
- 

Visualization element 'Meter'

Symbol:
Tag: “Measurement controls”

The element displays the value of a variable. The needle is positioned according to the value of the assigned variable. A meter is used to represent a tachometer, for example.

### Element properties

<table>
<thead>
<tr>
<th><strong>“Element name”</strong></th>
<th>Example: GenElemInst_1</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>“Type of element”</strong></td>
<td>“Meter”</td>
</tr>
<tr>
<td><strong>“Value”</strong></td>
<td>Variable (numeric data type).&lt;br&gt;The variable value determines the pointer movement of the element.</td>
</tr>
</tbody>
</table>

### Element property 'Center'

The properties contain fixed values for the coordinates of the point of rotation. This point of rotation is shown as the symbol. The point is used as the center for rotating and scaling.

<table>
<thead>
<tr>
<th><strong>“X”</strong></th>
<th>X-coordinate of the point of rotation</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>“Y”</strong></td>
<td>Y-coordinate of the point of rotation</td>
</tr>
</tbody>
</table>

You can also change the values by dragging the symbols (榉) to other positions in the editor.

These properties are available only when you select the “Preview: Support client animations and overlay of native elements” option in the Visualization Manager.

You can animate a movement, a rotation, and the visibility of a visualization element.

<table>
<thead>
<tr>
<th><strong>“Animation duration”</strong></th>
<th>Time that the element executes an animation (in milliseconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Example:</strong> 500</td>
<td></td>
</tr>
<tr>
<td><strong>“Move to the foreground”</strong></td>
<td>Property value (BOOL)&lt;br&gt;<strong>TRUE:</strong> At runtime, the element is displayed in the foreground.&lt;br&gt;<strong>FALSE:</strong> At runtime, the element is displayed in the layer where it was inserted in the visualization editor.</td>
</tr>
</tbody>
</table>

See also

- ¶ Chapter 1.3.5.21.5 “Using client animation” on page 1962

### Element property 'Position'

The position defines the location and size of the element in the visualization window. These are based on the Cartesian coordinate system. The origin is located at the upper left corner of the window. The positive horizontal x-axis runs to the right. The positive vertical y-axis runs downwards.
### Element property 'Background'

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Image color&quot;</td>
<td>List box containing background colors</td>
</tr>
<tr>
<td>&quot;Own image&quot;</td>
<td></td>
</tr>
</tbody>
</table>
  - "Image": ID of the background image.  
  - Info: If you specify the value "<default>" or select the image from the "Default" category in the Input Assistant, then the original element background image is used.  
  - "Transparency color": Selection from list box or Input Assistant. |

### Element property 'Arrow'

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Hand style&quot;</td>
<td>Drop-down list with different arrow types</td>
</tr>
<tr>
<td>&quot;Color&quot;</td>
<td></td>
</tr>
</tbody>
</table>
  - ▲: The "Color" dialog box opens.  
  - △: Drop-down list with color names                                           |
| "Arrow start"    | Angle (in degrees) between the scale start and the horizontal axis          |
| "Arrow end"      | Angle (in degrees) between the right edge of the pointer instrument and the horizontal axis |
| "Additional arrow" | ✅: An additional arrow is shown inside the scale.                         |

### Element property 'Scale'

You can also change the values by dragging the box symbols (_drag_ to other positions in the editor.

See also

- ☀ Chapter 1.3.5.3.2 “Positioning elements, adapting size and layer” on page 1113
### “Sub scale position”
- “Outside”: The subscale is displayed on the outer scale ring. (“Frame outside”)
- “Inside”: The subscale is displayed on the inner scale ring. (“Frame inside”)

### “Scale type”
Type of scale
- “Lines”
- “Dots”
- “Squares”

### “Scale start”
Least value of the scale and the lower limit of the value range for the element
Example: 0
*: The “Variable” property is displayed in the line below this.

### “Variable”
Variable (integer data type). Contains the scale start
Example: PLC_PRG.iScaleStart
Declarations:

```plaintext
PROGRAM PLC_PRG
VAR
  iScaleStart : INT := 0;
END_VAR
```

### “Scale end”
Greatest value of the scale and the upper limit of the value range for the element
Example: 100
*: The “Variable” property is shown below this.

### “Variable”
Variable (integer data type). Contains the scale end
Example: PLC_PRG.iScaleEnd
Declarations:

```plaintext
PROGRAM PLC_PRG
VAR
  iScaleEnd : INT := 120;
END_VAR
```

### “Main scale”
Distance between two values on the main scale
Example: 10
*: The “Variable” property is shown below.

### “Variable”
Variable (integer data type) Contains the distance between two values on the main scale
Example: PLC_PRG.iMainScale
Declarations:

```plaintext
PROGRAM PLC_PRG
VAR
  iMainScale : INT := 20;
END_VAR
```

### “Sub scale”
Distance between two values on the fine scale
You can hide the fine scale by setting the value to 0.
Example: 2
*: The “Variable” property is shown below this.
"Variable"  
Variable (integer data type) Contains the distance between two values on the fine scale  
Example: PLC_PRG.iSubScale  
Declarations:

```plaintext
PROGRAM PLC_PRG
VAR
  iSubScale : INT := 5;
END_VAR
```

"Scale line width"  
Specified in pixels  
Example: 3

"Scale color"  
Color of scale lines  
- [ ] : The “Color” dialog opens.  
- ▼ : A list box with style colors opens.

"Scale in 3D"  
☑️: Scale lines are displayed with soft 3D shadowing.  
Note: This property is not displayed in "FlatStyle".

"Show scale"  
☑️: The scale is displayed.

"Frame inside"  
☑️: A frame is drawn at the inner end of the scale.

"Frame outside"  
☑️: A frame is drawn at the outer end of the scale.

Element property 'Label'

"Label"  
Selection list  
- "Outside": Scale values are placed outside of the scale.  
- "Inside": Scale values are placed inside of the scale.

"Unit"  
Text that is displayed in the element.  
Example: Units displayed in m/s.

"Font"  
Font for labels (example: scale numbering).  
Selection from the drop-down list or by clicking the "" button.

"Scale format (C Syntax)"  
Values scaled in "printf" syntax  
Examples: %d, %.2f

"Max. text width of labels"  
(optional) Value that redefines the maximum width of the scale label. The correct value is normally set automatically.  
Note: Change this value only if the automatic adjustment does not yield the expected result.

"Text height of labels"  
(optional) Value that redefines the maximum height of the scale label. The correct value is normally set automatically.  
Note: Change this value only if the automatic adjustment does not yield the expected result.

"Font color"  
Selection from the drop-down list or by clicking the """ button.

Element property 'Positioning'
“Usage of”

- “Preset style values”: Values from the current style
- “User-defined settings”: The subnode “Positioning” appears.

“Positioning”

Requirement: “User-defined settings” is selected as “Usage of”.

The displayed positioning settings depend on the type of needle instrument and potentiometer, and partially on whether a custom background image is selected. The following settings are used for achieving the exact position relative to the background image.

<table>
<thead>
<tr>
<th>“Needle movement”</th>
<th>Length of the needle (in pixels)</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Scale movement”</td>
<td>Distance from the tick marks to the center (in pixels)</td>
</tr>
</tbody>
</table>

Requirement: A customer image is selected as “Background”.

<table>
<thead>
<tr>
<th>“Scale length”</th>
<th>Length of the tick marks (in pixels)</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Label offset”</td>
<td>Distance from the labels to the tick marks (in pixels)</td>
</tr>
<tr>
<td>“Unit offset”</td>
<td>Distance of the unit text “Label ➔ Unit” from the upper scale edge (in pixels)</td>
</tr>
<tr>
<td>“Origin offset”</td>
<td>Offset of the element (in pixels)</td>
</tr>
</tbody>
</table>

Requirement: For the elements “Meter 180°” and “Meter 90°”, this property is displayed only if a custom image is selected as “Background”.

Element property ‘Colors’

“Color areas”

<table>
<thead>
<tr>
<th>“Durable color areas”</th>
<th>All color areas are visible, regardless of the current value.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Only the color area is visible that includes the current value.</td>
</tr>
</tbody>
</table>

“Use colors for scale”

Colors in the color area are used only for the scale and frame.

“Color areas”

“Create new”

A new color area is added to the “Elements” view.

“Delete”

The color area is removed from the list and the list is refreshed.

“Begin of area”

Start value of the color area

Example: 20

✧: The property “Variable” is shown below.

“Variable”

Variable (integer data type). Contains the start value.

Example: PLC_PRG.iColorAreaStart0

Declaration:

```plaintext
PROGRAM PLC_PRG
VAR
  iColorAreaStart0 : INT := 80;
END_VAR
```

“End of area”

End value of the color area

Example: 120

✧: The property “Variable” is shown below.
### Variable

Variable (integer data type). Contains the end value.

**Example:** `iColorAreaEnd0`

**Declaration:**

```plaintext
PROGRAM PLC_PRG
VAR
  iColorAreaEnd0 : INT := 100;
END_VAR
```

### Color

Color that is used for displaying the area.

---

**Element property 'Absolute movement'**

The properties contain IEC variables for controlling the position of the element dynamically. The reference point is the upper left corner of the element. In runtime mode, the entire element is moved.

---

### Movement

| **“X”** | Variable (numeric data type). Defines the X position (in pixels).  
**Example:** PLC_PRG.iPos_X.  
Increasing this value in runtime mode moves the element to the right. |
| --- | --- |

| **“Y”** | Variable (numeric data type). Defines the Y position (in pixels).  
**Example:** PLC_PRG.iPos_Y.  
Increasing this value in runtime mode moves the element downwards. |
| --- | --- |

| **“Rotation”** | Variable (numeric data type). Defines the angle of rotation (in degrees).  
**Example:** PLC_PRG.iAngle1.  
The midpoint of the element rotates at the “Center” point. This rotation point is shown as the symbol.  
In runtime mode, the alignment of the element remains the same with respect to the coordinate system of the visualization. Increasing the value rotates the element to the right. |
| --- | --- |

| **“Interior rotation”** | Variable (numeric data type). Defines the angle of rotation (in degrees).  
**Example:** PLC_PRG.iAngle2.  
In runtime mode, the element rotates about the point of rotation specified in “Center” according to the value of the variable. In addition, the alignment of the element rotates according to the coordinate system of the visualization. Increasing the value in the code rotates clockwise.  
The rotation point is shown as the symbol.  
Note: If a static angle of rotation is specified in the “Position ➔ Angle” property, then the static angle of rotation is added to the variable angle of rotation (offset) when the visualization is executed. |
| --- | --- |
You can link the variables to a unit conversion.

The “X”, “Y”, “Rotation”, and “Interior rotation” properties are supported by the “Client Animation” functionality.

See also
- Unit conversion

Element property 'State variables'
The variables control the element behavior dynamically.

| "Invisible" | Variable (BOOL). Toggles the visibility of the element. TRUE: The element is not visible at runtime. |

The “Invisible” property is supported by the "Client Animation" functionality.

See also
- 

Element property 'Access rights'
Requirement: User management is set up for the visualization.

<table>
<thead>
<tr>
<th>&quot;Access rights&quot;</th>
<th>Opens the “Access rights” dialog. There you can edit the access privileges for the element. Status messages:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>&quot;Not set. Full rights.&quot;: Access rights for all user groups : &quot;operable&quot;</td>
</tr>
<tr>
<td></td>
<td>&quot;Rights are set: Limited rights&quot;: Access is restricted for at least one group.</td>
</tr>
</tbody>
</table>

See also
- ☀ Chapter 1.3.5.19.3.1 “Dialog 'Access Rights'” on page 1583

See also
- 

Visualization element 'Potentiometer'
Symbol: Tag: "Measurement controls"
The element displays the value of a variable as a setting on the potentiometer. A visualization user can modify the value by dragging the pointer to another position.

See also

**Element properties**

<table>
<thead>
<tr>
<th>“Element name”</th>
<th>Example: GenElemInst_1</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Type of element”</td>
<td>“Potentiometer”</td>
</tr>
<tr>
<td>“Variable”</td>
<td>Variable (numeric data type). Contains the position of the pointer for the potentiometer. A visualization user can modify the value by dragging the pointer to another position.</td>
</tr>
</tbody>
</table>

**Element property ‘Center’**

The properties contain fixed values for the coordinates of the point of rotation. This point of rotation is shown as the symbol. The point is used as the center for rotating and scaling.

<table>
<thead>
<tr>
<th>“X”</th>
<th>X-coordinate of the point of rotation</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Y”</td>
<td>Y-coordinate of the point of rotation</td>
</tr>
</tbody>
</table>

You can also change the values by dragging the symbols ( здоровья) to other positions in the editor.

These properties are available only when you select the “Preview: Support client animations and overlay of native elements” option in the Visualization Manager.

You can animate a movement, a rotation, and the visibility of a visualization element.

<table>
<thead>
<tr>
<th>“Animation duration”</th>
<th>Time that the element executes an animation (in milliseconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example: 500</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>“Move to the foreground”</th>
<th>Property value (BOOL)</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRUE: At runtime, the element is displayed in the foreground.</td>
<td></td>
</tr>
<tr>
<td>FALSE: At runtime, the element is displayed in the layer where it was inserted in the visualization editor.</td>
<td></td>
</tr>
</tbody>
</table>

See also

- Chapter 1.3.5.21.5 “Using client animation” on page 1962
**Element property 'Position'**

The position defines the location and size of the element in the visualization window. These are based on the Cartesian coordinate system. The origin is located at the upper left corner of the window. The positive horizontal x-axis runs to the right. The positive vertical y-axis runs downwards.

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;X&quot;</td>
<td>X coordinate of the upper left corner of the element</td>
</tr>
<tr>
<td></td>
<td>Specified in pixels.</td>
</tr>
<tr>
<td></td>
<td>Example: 10.</td>
</tr>
<tr>
<td>&quot;Y&quot;</td>
<td>Y coordinate of the upper left corner of the element</td>
</tr>
<tr>
<td></td>
<td>Specified in pixels.</td>
</tr>
<tr>
<td></td>
<td>Example: 10.</td>
</tr>
<tr>
<td>&quot;Width&quot;</td>
<td>Specified in pixels.</td>
</tr>
<tr>
<td></td>
<td>Example: 150</td>
</tr>
<tr>
<td>&quot;Height&quot;</td>
<td>Specified in pixels.</td>
</tr>
<tr>
<td></td>
<td>Example: 30</td>
</tr>
</tbody>
</table>

You can also change the values by dragging the box symbols (_drag box_) to other positions in the editor.

See also
- § Chapter 1.3.5.3.2 “Positioning elements, adapting size and layer” on page 1113

**Element property 'Background'**

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Image color&quot;</td>
<td>List box containing background colors</td>
</tr>
<tr>
<td>&quot;Own image&quot;</td>
<td>• &quot;Image&quot;: ID of the background image. You select the background image from an image pool by clicking ![image pool]. Info: If you specify the value &quot;&lt;default&gt;&quot; or select the image from the &quot;Default&quot; category in the Input Assistant, then the original element background image is used. • &quot;Transparency color&quot;: Selection from list box or Input Assistant.</td>
</tr>
</tbody>
</table>

**Element property 'Arrow'**

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Hand style&quot;</td>
<td>Drop-down list with different arrow types</td>
</tr>
<tr>
<td>&quot;Color&quot;</td>
<td>• ![color]: The &quot;Color&quot; dialog box opens.  • ![color names]: Drop-down list with color names</td>
</tr>
<tr>
<td>&quot;Arrow start&quot;</td>
<td>Angle (in degrees) between the left edge of the element and the horizontal axis</td>
</tr>
<tr>
<td>&quot;Arrow end&quot;</td>
<td>Angle (in degrees) between the right edge of the element and the horizontal axis</td>
</tr>
</tbody>
</table>

**Element property 'Scale'**
### “Sub scale position”
- **“Outside”**: The subscale is displayed on the outer scale ring. (“Frame outside”)
- **“Inside”**: The subscale is displayed on the inner scale ring. (“Frame inside”)

### “Scale type”
Type of scale
- **“Lines”**
- **“Dots”**
- **“Squares”**

### “Scale start”
Least value of the scale and the lower limit of the value range for the element

**Example:** 0

- The “Variable” property is displayed in the line below this.

### “Variable”
Variable (integer data type). Contains the scale start

**Example:** PLC_PRG.iScaleStart

**Declaration:**
```plaintext
PROGRAM PLC_PRG
VAR
  iScaleStart : INT := 0;
END_VAR
```

### “Scale end”
Greatest value of the scale and the upper limit of the value range for the element

**Example:** 100

- The “Variable” property is shown below this.

### “Variable”
Variable (integer data type). Contains the scale end

**Example:** PLC_PRG.iScaleEnd

**Declaration:**
```plaintext
PROGRAM PLC_PRG
VAR
  iScaleEnd : INT := 120;
END_VAR
```

### “Main scale”
Distance between two values on the main scale

**Example:** 10

- The “Variable” property is shown below.

### “Variable”
Variable (integer data type) Contains the distance between two values on the main scale

**Example:** PLC_PRG.iMainScale

**Declaration:**
```plaintext
PROGRAM PLC_PRG
VAR
  iMainScale : INT := 20;
END_VAR
```

### “Sub scale”
Distance between two values on the fine scale

You can hide the fine scale by setting the value to 0.

**Example:** 2

- The “Variable” property is shown below this.
**“Variable”**

Variable (integer data type) Contains the distance between two values on the fine scale

Example: PLC_PRG.iSubScale

Declaration:

```plaintext
PROGRAM PLC_PRG
VAR
  iSubScale : INT := 5;
END_VAR
```

**“Scale line width”**

Specified in pixels

Example: 3

**“Scale color”**

Color of scale lines

- □: The “Color” dialog opens.
- ▼: A list box with style colors opens.

**“Scale in 3D”**

☑: Scale lines are displayed with soft 3D shadowing.

Note: This property is not displayed in “FlatStyle”.

**“Show scale”**

☑: The scale is displayed.

**“Frame inside”**

☑: A frame is drawn at the inner end of the scale.

**“Frame outside”**

☑: A frame is drawn at the outer end of the scale.

---

**Element property 'Label'**

<table>
<thead>
<tr>
<th>“Label”</th>
<th>Selection list</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Outside”</td>
<td>Scale values are placed outside of the scale.</td>
</tr>
<tr>
<td>“Inside”</td>
<td>Scale values are placed inside of the scale.</td>
</tr>
</tbody>
</table>

**“Unit”**

Text that is displayed in the element.

Example: Units displayed in m/s.

**“Font”**

Font for labels (example: scale numbering).

Selection from the drop-down list or by clicking the “” button.

**“Scale format (C Syntax)”**

Values scaled in “printf” syntax

Examples: %d, %5.2f

**“Max. text width of labels”**

(optional) Value that redefines the maximum width of the scale label. The correct value is normally set automatically.

Note: Change this value only if the automatic adjustment does not yield the expected result.

**“Text height of labels”**

(optional) Value that redefines the maximum height of the scale label. The correct value is normally set automatically.

Note: Change this value only if the automatic adjustment does not yield the expected result.

**“Font color”**

Selection from the drop-down list or by clicking the “” button.

---

---

PLC Automation with V3 CPUs
Programming with IEC 61131-3 editor > CODESYS Visualization

2020/12/10

3ADR010583, 1, en_US

1843
"Usage of"

- "Preset style values": Values from the current style
- "User-defined settings": The subnode "Positioning" appears.

"Positioning"

Requirement: "User-defined settings" is selected as "Usage of".

The displayed positioning settings depend on the type of needle instrument and potentiometer, and partially on whether a custom background image is selected. The following settings are used for achieving the exact position relative to the background image.

<table>
<thead>
<tr>
<th>Setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Needle movement&quot;</td>
<td>Length of the needle (in pixels)</td>
</tr>
<tr>
<td>&quot;Scale movement&quot;</td>
<td>Distance from the tick marks to the center (in pixels)</td>
</tr>
<tr>
<td>Requirement:</td>
<td>A customer image is selected as &quot;Background&quot;.</td>
</tr>
<tr>
<td>&quot;Scale length&quot;</td>
<td>Length of the tick marks (in pixels)</td>
</tr>
<tr>
<td>Requirement:</td>
<td>A customer image is selected as &quot;Background&quot;.</td>
</tr>
<tr>
<td>&quot;Label offset&quot;</td>
<td>Distance from the labels to the tick marks (in pixels)</td>
</tr>
<tr>
<td>&quot;Unit offset&quot;</td>
<td>Distance of the unit text &quot;Label ➔ Unit&quot; from the upper scale edge (in pixels)</td>
</tr>
<tr>
<td>&quot;Origin offset&quot;</td>
<td>Offset of the element (in pixels)</td>
</tr>
<tr>
<td>Requirement: For</td>
<td>The elements &quot;Meter 180°&quot; and &quot;Meter 90°&quot;, this property is displayed only if a custom image is selected as &quot;Background&quot;.</td>
</tr>
</tbody>
</table>

"Color areas"

- "Durable color areas": All color areas are visible, regardless of the current value.
- "Use colors for scale": Only the color area is visible that includes the current value.

"Color areas"

- "Create new": A new color area is added to the "Elements" view.
- "Delete": The color area is removed from the list and the list is refreshed.

"Begin of area"

Start value of the color area
Example: 20

★ The property "Variable" is shown below.

"Variable"

Variable (integer data type). Contains the start value.
Example: PLC_PRG.iColorAreaStart0
Declaration:

```
PROGRAM PLC_PRG
VAR
   iColorAreaStart0 : INT := 80;
END_VAR
```

"End of area"

End value of the color area
Example: 120

★ The property "Variable" is shown below.
**“Variable”**

Variable (integer data type). Contains the end value.

Example: `IColorAreaEnd0`

Declaration:

```
PROGRAM PLC_PRG
VAR
  iColorAreaEnd0 : INT := 100;
END_VAR
```

**“Color”**

Color that is used for displaying the area.

---

**Element property 'State variables'**

The variables control the element behavior dynamically.

<table>
<thead>
<tr>
<th><strong>“Invisible”</strong></th>
<th>Variable (BOOL). Toggles the visibility of the element. TRUE: The element is not visible at runtime.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>“Deactivate inputs”</strong></td>
<td>Variable (BOOL). Toggles the operability of the element. TRUE: User inputs do not have any effect in runtime more. The element is shown as deactivated.</td>
</tr>
</tbody>
</table>

The “Invisible” property is supported by the “Client Animation” functionality.

See also

- 

---

**Element property 'Absolute movement'**

The properties contain IEC variables for controlling the position of the element dynamically. The reference point is the upper left corner of the element. In runtime mode, the entire element is moved.

| **“Movement”** | 
|-----------------|--------------------------------------------------------------------------------------------------|
| **“X”** | Variable (numeric data type). Defines the X position (in pixels). Example: `PLC_PRG.iPos_X`. Increasing this value in runtime mode moves the element to the right. |
| **“Y”** | Variable (numeric data type). Defines the Y position (in pixels). Example: `PLC_PRG.iPos_Y`. Increasing this value in runtime mode moves the element downwards. |
“Rotation” Variable (numeric data type). Defines the angle of rotation (in degrees).

Example: PLC_PRG.iAngle1.

The midpoint of the element rotates at the “Center” point. This rotation point is shown as the symbol.

In runtime mode, the alignment of the element remains the same with respect to the coordinate system of the visualization. Increasing the value rotates the element to the right.

“Interior rotation” Variable (numeric data type). Defines the angle of rotation (in degrees).

Example: PLC_PRG.iAngle2.

In runtime mode, the element rotates about the point of rotation specified in “Center” according to the value of the variable. In addition, the alignment of the element rotates according to the coordinate system of the visualization. Increasing the value in the code rotates clockwise.

The rotation point is shown as the symbol.

Note: If a static angle of rotation is specified in the “Position Angle” property, then the static angle of rotation is added to the variable angle of rotation (offset) when the visualization is executed.

You can link the variables to a unit conversion.

The “X”, “Y”, “Rotation”, and “Interior rotation” properties are supported by the “Client Animation” functionality.

See also
- Unit conversion

Element property ‘Access rights’ Requirement: User management is set up for the visualization.

“Access rights” Opens the “Access rights” dialog. There you can edit the access privileges for the element.

Status messages:
- “Not set. Full rights.”: Access rights for all user groups: “operable”
- “Rights are set: Limited rights”: Access is restricted for at least one group.

See also
- Chapter 1.3.5.19.3.1 “Dialog ‘Access Rights’” on page 1583
Visualization element 'Histogram'

Symbol:

Tag: “Measurement controls”
The element displays the data of a one-dimensional array as a histogram. You can define specific colors for certain value ranges.
See also
● Chapter 1.3.5.21.2 “Displaying array data in histograms” on page 1958

Element properties

<table>
<thead>
<tr>
<th>“Element name”</th>
<th>Example: GenElemInst_35</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Type of element”</td>
<td>“Histogram”</td>
</tr>
<tr>
<td>“Data array”</td>
<td>One-dimensional array with data displayed in this histogram.</td>
</tr>
<tr>
<td>Example: PLC_PRG.arr1</td>
<td></td>
</tr>
</tbody>
</table>

Element property 'Subrange of array'

| “Use subrange” | ☑: Only part of the array is displayed in the histogram. |
| “Start index” | First array index with a displayed value. |
| Requirement: “Use subrange” is activated. |
| “End index” | Last array index with a displayed value. |
| Requirement: “Use subrange” is activated. |

| “Display type” | ● “Bars”: Data is displayed as bars. |
| ● “Lines”: Data is displayed as lines. |
| ● “Curve”: Interpolation of data into a curve. |

| “Line width” | Specified in pixels |
| Requirement: “Curve” is selected as the “Display type”. |

| “Show horizontal lines” | ☑: Horizontal lines are drawn on the main scale. |
| Note: Not all visualization styles have this property. This element property is not available for visualization styles that have striped backgrounds (example: “Flat style”). |

| “Relative bar width” | Integer value between 1 and 100 |
| ● 1: The bars are drawn as lines. |
| ● 100: The complete width of the histogram is filled with the bars. |

Element property ‘Position’
The position defines the location and size of the element in the visualization window. These are based on the Cartesian coordinate system. The origin is located at the upper left corner of the window. The positive horizontal x-axis runs to the right. The positive vertical y-axis runs downwards.
<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>“X”</td>
<td>X coordinate of the upper left corner of the element. Specified in pixels.</td>
</tr>
<tr>
<td></td>
<td>Example: 10.</td>
</tr>
<tr>
<td>“Y”</td>
<td>Y coordinate of the upper left corner of the element. Specified in pixels.</td>
</tr>
<tr>
<td></td>
<td>Example: 10.</td>
</tr>
<tr>
<td>“Width”</td>
<td>Specified in pixels.</td>
</tr>
<tr>
<td></td>
<td>Example: 150.</td>
</tr>
<tr>
<td>“Height”</td>
<td>Specified in pixels.</td>
</tr>
<tr>
<td></td>
<td>Example: 30</td>
</tr>
</tbody>
</table>

You can also change the values by dragging the box symbols (.masks) to other positions in the editor.

**See also**
- Chapter 1.3.5.3.2 “Positioning elements, adapting size and layer” on page 1113

**Element property ‘Center’**
The properties contain fixed values for the coordinates of the point of rotation. This point of rotation is shown as the symbol. The point is used as the center for rotating and scaling.

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>“X”</td>
<td>X-coordinate of the point of rotation</td>
</tr>
<tr>
<td>“Y”</td>
<td>Y-coordinate of the point of rotation</td>
</tr>
</tbody>
</table>

You can also change the values by dragging the symbols (.) to other positions in the editor.

**Element property ‘Scale’**

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Scale start”</td>
<td>Least value of the scale and the lower limit of the value range for the element.</td>
</tr>
<tr>
<td></td>
<td>Example: 0</td>
</tr>
<tr>
<td></td>
<td>♦: The property “Variable” is shown below.</td>
</tr>
<tr>
<td>“Variable”</td>
<td>Variable (integer data type). Contains the scale start.</td>
</tr>
<tr>
<td></td>
<td>Example: PLC_PRG.iScaleStart</td>
</tr>
<tr>
<td></td>
<td>Declaration:</td>
</tr>
<tr>
<td></td>
<td>PROGRAM PLC_PRG</td>
</tr>
<tr>
<td></td>
<td>VAR</td>
</tr>
<tr>
<td></td>
<td>iScaleStart : INT := 0;</td>
</tr>
<tr>
<td></td>
<td>END_VAR</td>
</tr>
</tbody>
</table>

You can also change the values by dragging the box symbols (.) to other positions in the editor.
| **Scale end** | Greatest value of the scale and the upper limit of the value range for the element.  
Example: 100  
✦: The property “Variable” is shown below. |
| **Variable** | Variable (integer data type). Contains the scale end.  
Example: PLC_PRG.iScaleEnd  
Declaration:  
```plaintext
PROGRAM PLC_PRG  
VAR  
iScaleEnd : INT := 120;  
END_VAR
``` |
| **Main scale** | Distance between 2 values on the rough scale.  
Example: 10  
✦: The property “Variable” is shown below. |
| **Variable** | Variable (integer data type). Contains the distance.  
Example: PLC_PRG.iMainScale  
Declaration:  
```plaintext
PROGRAM PLC_PRG  
VAR  
iMainScale : INT := 20;  
END_VAR
``` |
| **Subscale** | Distance between 2 values on the fine scale.  
You can hide the fine scale by setting the value to 0.  
Example: 2  
✦: The property “Variable” is shown below. |
| **Variable** | Variable (integer data type). Contains the spacing.  
Example: PLC_PRG.iSubScale  
Declaration:  
```plaintext
PROGRAM PLC_PRG  
VAR  
iSubScale : INT := 5;  
END_VAR
``` |
| **Scale color** | Color of scale lines  
• ![Color](#): The “Color” dialog box opens.  
• ![Color](#): A drop-down list with color names opens. |
| **Base line** | Value of the main scale where the horizontal base line of the histogram is located.  
The drawing of the bar starts at the base line. |

Element property ‘Label’
| “Unit”        | Text that is displayed in the element.  
|              | Example: Units displayed in m/s.       |
|              |                                         |
| “Font”       | Font for labels (example: scale numbering).  
|              | Selection from the drop-down list or by clicking the “…” button. |
| “Scale format (C Syntax)” | Values scaled in "printf" syntax  
|              | Examples: %d, %5.2f                        |
| “Max. text width of labels” | Optional value that defines the maximum width of the scale label.  
|              | Note: Change this value only if the automatic adjustment does not yield the expected result. |
| “Text height of labels” | Optional value that defines the maximum height of the scale label.  
|              | Note: Change this value only if the automatic adjustment does not yield the expected result. |
| “Font color” | Selection from the drop-down list or by clicking the "…” button. |

**Element property 'Colors’**

| “Graph color” | Color of the bar in normal state.  
|              | Note: The normal state is in effect when the current value of the array component does not fulfill the alarm condition. |
| “Alarm value” | Threshold for the alarm |
| “Alarm condition” | If the current value of the array component fulfills the alarm condition, then the alarm condition is set.  
|              | ● “Less”: The current value is less than the “Alarm value”  
|              | ● “More”: The current value is greater than the “Alarm value” |
| “Alarm color” | Color of the bar in alarm state. |
| “Use color areas” | ☑: The color areas defined in this element are used.  
| “Color areas” |  |
| “Create new” | A new color area is added. |
| “Delete” | The color area is removed from the list. |
| “Begin of area” | The start value on the “Scale” of the histogram where the color area begins.  
| “End of area” | The end value on the “Scale” of the histogram where the color area ends.  
| “Color” | Color that is used for displaying the area. |

**Element property 'Absolute movement'**

The properties contain IEC variables for controlling the position of the element dynamically. The reference point is the upper left corner of the element. In runtime mode, the entire element is moved.

| “Movement” |  |
| “X” | Variable (numeric data type). Defines the X position (in pixels).  
| | Example: PLC_PRG.iPos_X.  
<p>| | Increasing this value in runtime mode moves the element to the right. |</p>
<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
<th>Example</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Y”</td>
<td>Variable (numeric data type). Defines the Y position (in pixels).</td>
<td>PLC_PRG.iPos_Y</td>
<td>Increasing this value in runtime mode moves the element downwards.</td>
</tr>
<tr>
<td>“Rotation”</td>
<td>Variable (numeric data type). Defines the angle of rotation (in degrees).</td>
<td>PLC_PRG.iAngle1</td>
<td>The midpoint of the element rotates at the “Center” point. This rotation point is shown as the symbol.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>In runtime mode, the alignment of the element remains the same with respect to the coordinate system of the visualization. Increasing the value rotates the element to the right.</td>
</tr>
<tr>
<td>“Interior rotation”</td>
<td>Variable (numeric data type). Defines the angle of rotation (in degrees).</td>
<td>PLC_PRG.iAngle2</td>
<td>In runtime mode, the element rotates about the point of rotation specified in “Center” according to the value of the variable.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>In addition, the alignment of the element rotates according to the coordinate system of the visualization. Increasing the value in the code rotates clockwise.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The rotation point is shown as the symbol.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Note: If a static angle of rotation is specified in the “Position ➔ Angle” property, then the static angle of rotation is added to the variable angle of rotation (offset) when the visualization is executed.</td>
</tr>
</tbody>
</table>

You can link the variables to a unit conversion.

The “X”, “Y”, “Rotation”, and “Interior rotation” properties are supported by the “Client Animation” functionality.

See also

- Unit conversion

These properties are available only when you select the “Preview: Support client animations and overlay of native elements” option in the Visualization Manager.

You can animate a movement, a rotation, and the visibility of a visualization element.
### Animation duration

Time that the element executes an animation (in milliseconds)

**Example:** 500

### Move to the foreground

**Property value** (BOOL)

- **TRUE:** At runtime, the element is displayed in the foreground.
- **FALSE:** At runtime, the element is displayed in the layer where it was inserted in the visualization editor.

**See also**

- Chapter 1.3.5.21.5 “Using client animation” on page 1962

### Element property 'Access rights'

Requirement: User management is set up for the visualization.

**“Access rights”**

Opens the “Access rights” dialog. There you can edit the access privileges for the element.

**Status messages:**

- **“Not set. Full rights.”:** Access rights for all user groups: “operable”
- **“Rights are set: Limited rights.”:** Access is restricted for at least one group.

**See also**

- Chapter 1.3.5.19.3.1 “Dialog ‘Access Rights’” on page 1583

### Visualization element 'Image switcher'

**Symbol:**

![Image switcher symbol]

**Tag:** "Lamps/Switches/Bitmaps"

The element displays one of three referenced images. Mouse actions change the displayed image. The images are defined in the "Image settings" element properties. The effects of mouse clicks are defined in the "Element behavior" property.

### Element properties

#### "Element name"

Optional

**Hint:** Assign individual names for elements so that they are found faster in the element list.

**Example:** ImageSwitcher_1

#### "Type of element"

"Image switcher"

### Element property 'Position'

The position defines the location and size of the element in the visualization window. These are based on the Cartesian coordinate system. The origin is located at the upper left corner of the window. The positive horizontal x-axis runs to the right. The positive vertical y-axis runs downwards.
| **X** | X coordinate of the upper left corner of the element  
Specified in pixels.  
Example: 10. |
|-------|--------------------------------------------------|
| **Y** | Y coordinate of the upper left corner of the element  
Specified in pixels.  
Example: 10. |
| **Width** | Specified in pixels.  
Example: 150 |
| **Height** | Specified in pixels.  
Example: 30 |

You can also change the values by dragging the box symbols ( يبدو) to other positions in the editor.

See also
- Chapter 1.3.5.3.2 “Positioning elements, adapting size and layer” on page 1113

| **Variable** | Variable (BOOL).  
The value of the variable changes according to user input and it is independent of the “Element behavior” element property. |
|--------------|--------------------------------------------------|

**Image settings**

| **Image "on"** | Image ID from an image pool. The image can be selected using the input assistant.  
The image is used if the variable of the “Variable” property has the value TRUE. |
|----------------|--------------------------------------------------------------------------------|
| **Image "off"** | Image ID from an image pool. The image can be selected using the input assistant.  
The image is used if the variable of the “Variable” property has the value FALSE. |
| **Image "clicked"** | Image ID from an image pool. The image is selected using the input assistant.  
In runtime mode, the visualization displays the referenced image when the element is clicked (and the mouse button is held down).  
Requirement: The “Element behavior” is “Image toggler”. |
| **Transparency** | ☑: The “Transparent color” is selected. |
| **Transparent color** | The image pixels that have the transparent color are displayed as transparent.  
Requirement: “Transparency” is activated.  
- ☑️ The “Color” dialog box opens.  
- ☟: A drop-down list with color names opens. |
"Scaling type"  Defines how an image fits in the element frame.
- "Fixed": The original size of the image is retained, regardless of the dimensions of the element.
- "Isotropic": The entire image is shown in the element frame, either larger or smaller. As a result, the proportion of height and width are retained.
- "Anisotropic": The image resizes automatically to the dimensions of the element frame, filling the entire element frame. As a result, the proportions are not retained.

"Horizontal alignment"  Horizontal alignment of the image within the element frame or element
- Left
- Centered
- Right
Requirement: "Scaling type" is "Isotropic".

"Vertical alignment"  Vertical alignment of the image within the element frame or element
- Top
- Centered
- Bottom
Requirement: "Scaling type" is "Isotropic".

"Element behavior"  
- "Image toggler": Every mouse click switches the image.
- "Image tapper": While a visualization user holds down the mouse button, the image of the "Image on" property is displayed. At the same time, the value TRUE is assigned to the "Variable" property.

"Tap FALSE"  
☑: While the mouse button is pressed, the image of the "Image" property is displayed and the "Variable" property gets the value FALSE instead of the value TRUE, and back.
Requirement: "Image tapper" is selected in the "Element behavior" property.

Element property 'Center'  
The properties contain fixed values for the coordinates of the point of rotation. This point of rotation is shown as the symbol. The point is used as the center for rotating and scaling.

"X"  
X-coordinate of the point of rotation

"Y"  
Y-coordinate of the point of rotation

You can also change the values by dragging the symbols (_symbol) to other positions in the editor.

Element property 'Texts'

"Tooltip"  
String display as tooltip for the element
Example: Valid access.
The properties contain IEC variables for controlling the position of the element dynamically. The reference point is the upper left corner of the element. In runtime mode, the entire element is moved.

**"Movement"**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
<th>Example</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;X&quot;</td>
<td>Variable (numeric data type). Defines the X position (in pixels).</td>
<td>PLC_PRG.iPos_X</td>
<td>Increasing this value in runtime mode moves the element to the right.</td>
</tr>
<tr>
<td>&quot;Y&quot;</td>
<td>Variable (numeric data type). Defines the Y position (in pixels).</td>
<td>PLC_PRG.iPos_Y</td>
<td>Increasing this value in runtime mode moves the element downwards.</td>
</tr>
<tr>
<td>&quot;Rotation&quot;</td>
<td>Variable (numeric data type). Defines the angle of rotation (in degrees).</td>
<td>PLC_PRG.iAngle1</td>
<td>The midpoint of the element rotates at the “Center” point. This rotation point is shown as the symbol. In runtime mode, the alignment of the element remains the same with respect to the coordinate system of the visualization. Increasing the value rotates the element to the right.</td>
</tr>
<tr>
<td>&quot;Interior rotation&quot;</td>
<td>Variable (numeric data type). Defines the angle of rotation (in degrees).</td>
<td>PLC_PRG.iAngle2</td>
<td>In runtime mode, the element rotates about the point of rotation specified in “Center” according to the value of the variable. In addition, the alignment of the element rotates according to the coordinate system of the visualization. Increasing the value in the code rotates clockwise. The rotation point is shown as the symbol. Note: If a static angle of rotation is specified in the &quot;Position ➔ Angle&quot; property, then the static angle of rotation is added to the variable angle of rotation (offset) when the visualization is executed.</td>
</tr>
</tbody>
</table>

You can link the variables to a unit conversion.

The "X", "Y", "Rotation", and "Interior rotation" properties are supported by the "Client Animation" functionality.
See also
- Unit conversion

Element property 'State variables'
The variables control the element behavior dynamically.

| "Invisible" | Variable (BOOL). Toggles the visibility of the element. TRUE: The element is not visible at runtime. |
| "Deactivate inputs" | Variable (BOOL). Toggles the operability of the element. TRUE: User inputs do not have any effect in runtime more. The element is shown as deactivated. |

The "Invisible" property is supported by the "Client Animation" functionality.

See also
- Ä Chapter 1.3.5.21.5 "Using client animation" on page 1962

These properties are available only when you select the "Preview: Support client animations and overlay of native elements" option in the Visualization Manager.

You can animate a movement, a rotation, and the visibility of a visualization element.

| "Animation duration" | Time that the element executes an animation (in milliseconds) Example: 500 |
| "Move to the foreground" | Property value (BOOL) TRUE: At runtime, the element is displayed in the foreground. FALSE: At runtime, the element is displayed in the layer where it was inserted in the visualization editor. |

See also
- Ä Chapter 1.3.5.21.5 "Using client animation" on page 1962

Element property 'Access rights'
Requirement: User management is set up for the visualization.

| "Access rights" | Opens the "Access rights" dialog. There you can edit the access privileges for the element. Status messages: • "Not set. Full rights.": Access rights for all user groups: "operable" • "Rights are set: Limited rights": Access is restricted for at least one group. |

See also
- Ä Chapter 1.3.5.19.3.1 "Dialog 'Access Rights'" on page 1583
Visualization element 'Lamp'

Symbol:

Tag: “Lamps/Switches/Bitmaps”
The element shows the value of a variable, and the element is displayed as illuminated or not.

Element properties

<table>
<thead>
<tr>
<th>“Element name”</th>
<th>Optional</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hint: Assign individual names for elements so that they are found faster in the element list.</td>
<td></td>
</tr>
<tr>
<td>Example: Lamp_green</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>“Type of element”</th>
<th>“Lamp”</th>
</tr>
</thead>
</table>

Element property 'Position'
The position defines the location and size of the element in the visualization window. These are based on the Cartesian coordinate system. The origin is located at the upper left corner of the window. The positive horizontal x-axis runs to the right. The positive vertical y-axis runs downwards.

<table>
<thead>
<tr>
<th>“X”</th>
<th>X coordinate of the upper left corner of the element</th>
</tr>
</thead>
<tbody>
<tr>
<td>Specified in pixels.</td>
<td></td>
</tr>
<tr>
<td>Example: 10.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>“Y”</th>
<th>Y coordinate of the upper left corner of the element</th>
</tr>
</thead>
<tbody>
<tr>
<td>Specified in pixels.</td>
<td></td>
</tr>
<tr>
<td>Example: 10.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>“Width”</th>
<th>Specified in pixels.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example: 150</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>“Height”</th>
<th>Specified in pixels.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example: 30</td>
<td></td>
</tr>
</tbody>
</table>

You can also change the values by dragging the box symbols (_drag-) to other positions in the editor.

See also

- § Chapter 1.3.5.3.2 “Positioning elements, adapting size and layer” on page 1113
### Image settings

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Variable</strong></td>
<td>Variable (BOOL). The variable value is displayed as a lamp that goes on (TRUE) or off (FALSE).</td>
</tr>
<tr>
<td><strong>Transparency</strong></td>
<td>The “Transparent color” property is selected. Pixels in this color are displayed as transparent. Requirement: “Transparency” is activated.</td>
</tr>
<tr>
<td><strong>Scaling type</strong></td>
<td>Reaction of the element when the dimension of the “Frame” element is changed:</td>
</tr>
<tr>
<td></td>
<td>- “Isotropic”: The height and width of the image are resized proportionally to the “Frame”. Please note: To retain the alignment of elements also within a scaled “Frame” element, define the “Horizontal alignment” or “Vertical alignment” explicitly with “Centered”.</td>
</tr>
<tr>
<td></td>
<td>- “Anisotropic”: The image fills the entire “Frame” regardless of its proportions.</td>
</tr>
<tr>
<td><strong>Horizontal alignment</strong></td>
<td>Horizontal alignment of the image within the element frame or element</td>
</tr>
<tr>
<td></td>
<td>- Left</td>
</tr>
<tr>
<td></td>
<td>- Centered</td>
</tr>
<tr>
<td></td>
<td>- Right</td>
</tr>
<tr>
<td><strong>Vertical alignment</strong></td>
<td>Vertical alignment of the image within the element frame or element</td>
</tr>
<tr>
<td></td>
<td>- Top</td>
</tr>
<tr>
<td></td>
<td>- Centered</td>
</tr>
<tr>
<td></td>
<td>- Bottom</td>
</tr>
</tbody>
</table>

### Element property 'Center'

The properties contain fixed values for the coordinates of the point of rotation. This point of rotation is shown as the ⠶ symbol. The point is used as the center for rotating and scaling.

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>X</strong></td>
<td>X-coordinate of the point of rotation</td>
</tr>
<tr>
<td><strong>Y</strong></td>
<td>Y-coordinate of the point of rotation</td>
</tr>
</tbody>
</table>

You can also change the values by dragging the symbols (⠶) to other positions in the editor.

### Element property 'Texts'

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Tooltip</strong></td>
<td>String display as tooltip for the element</td>
</tr>
<tr>
<td></td>
<td>Example: Valid access.</td>
</tr>
</tbody>
</table>
**Element property 'Absolute movement'**

The properties contain IEC variables for controlling the position of the element dynamically. The reference point is the upper left corner of the element. In runtime mode, the entire element is moved.

<table>
<thead>
<tr>
<th><strong>“Movement”</strong></th>
<th></th>
</tr>
</thead>
</table>
| **“X”** | Variable (numeric data type). Defines the X position (in pixels).  
Example: PLC_PRG.iPos_X.  
Increasing this value in runtime mode moves the element to the right. |
| **“Y”** | Variable (numeric data type). Defines the Y position (in pixels).  
Example: PLC_PRG.iPos_Y.  
Increasing this value in runtime mode moves the element downwards. |
| **“Rotation”** | Variable (numeric data type). Defines the angle of rotation (in degrees).  
Example: PLC_PRG.iAngle1.  
The midpoint of the element rotates at the “Center” point. This rotation point is shown as the symbol.  
In runtime mode, the alignment of the element remains the same with respect to the coordinate system of the visualization. Increasing the value rotates the element to the right. |
| **“Interior rotation”** | Variable (numeric data type). Defines the angle of rotation (in degrees).  
Example: PLC_PRG.iAngle2.  
In runtime mode, the element rotates about the point of rotation specified in “Center” according to the value of the variable. In addition, the alignment of the element rotates according to the coordinate system of the visualization. Increasing the value in the code rotates clockwise.  
The rotation point is shown as the symbol.  
Note: If a static angle of rotation is specified in the “Position ➔ Angle” property, then the static angle of rotation is added to the variable angle of rotation (offset) when the visualization is executed. |

You can link the variables to a unit conversion.

The “X”, “Y”, “Rotation”, and “Interior rotation” properties are supported by the “Client Animation” functionality.
See also

- Unit conversion

**Element property 'State variables'**

The variables control the element behavior dynamically.

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>&quot;Invisible&quot;</strong></td>
<td>Variable (BOOL). Toggles the visibility of the element. TRUE: The element is not visible at runtime.</td>
</tr>
</tbody>
</table>

*The "Invisible" property is supported by the "Client Animation" functionality.*

See also

- 

These properties are available only when you select the “Preview: Support client animations and overlay of native elements” option in the Visualization Manager.

You can animate a movement, a rotation, and the visibility of a visualization element.

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>&quot;Animation duration&quot;</strong></td>
<td>Time that the element executes an animation (in milliseconds) &lt;br&gt;Example: 500</td>
</tr>
<tr>
<td><strong>&quot;Move to the foreground&quot;</strong></td>
<td>Property value (BOOL) &lt;br&gt;TRUE: At runtime, the element is displayed in the foreground. FALSE: At runtime, the element is displayed in the layer where it was inserted in the visualization editor.</td>
</tr>
</tbody>
</table>

See also

- Chapter 1.3.5.21.5 “Using client animation” on page 1962

**Element property 'Background'**

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>&quot;Image&quot;</strong></td>
<td>Drop-down list with background colors &lt;br&gt;Depends on the visualization style</td>
</tr>
</tbody>
</table>

**Element property 'Access rights'**

Requirement: User management is set up for the visualization.

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>&quot;Access rights&quot;</strong></td>
<td>Opens the “Access rights” dialog. There you can edit the access privileges for the element. Status messages: &lt;br&gt;- &quot;Not set. Full rights.&quot;: Access rights for all user groups: &quot;operable&quot; &lt;br&gt;- &quot;Rights are set: Limited rights&quot;: Access is restricted for at least one group.</td>
</tr>
</tbody>
</table>
See also

- Chapter 1.3.5.19.3.1 “Dialog ‘Access Rights’” on page 1583

See also

- 

Visualization element 'Dip switch', 'Power switch', 'Push switch', 'Push switch LED', 'Rocker switch'

Symbols:

Tag: “Lamps/Switches/Bitmaps”

The element assigns a value to a Boolean variable. The switch position "on" the value TRUE to the variable, and the switch position "off" assigns the value FALSE. Use the mouse to change the switch position.

**Element properties**

<table>
<thead>
<tr>
<th>“Element name”</th>
<th>Example: Operating_Switch</th>
</tr>
</thead>
<tbody>
<tr>
<td>Optional</td>
<td>Hint: Assign individual names for elements so that they are found faster in the element list.</td>
</tr>
<tr>
<td>“Type of element”</td>
<td>Depending on the element: “Dip switch”, “Power switch”, “Push switch”, “Push switch LED”, or “Rocker switch”</td>
</tr>
</tbody>
</table>

**Element property 'Position'**

The position defines the location and size of the element in the visualization window. These are based on the Cartesian coordinate system. The origin is located at the upper left corner of the window. The positive horizontal x-axis runs to the right. The positive vertical y-axis runs downwards.

<table>
<thead>
<tr>
<th>“X”</th>
<th>X coordinate of the upper left corner of the element</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Specified in pixels.</td>
</tr>
<tr>
<td></td>
<td>Example: 10.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>“Y”</th>
<th>Y coordinate of the upper left corner of the element</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Specified in pixels.</td>
</tr>
<tr>
<td></td>
<td>Example: 10.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>“Width”</th>
<th>Specified in pixels.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Example: 150</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>“Height”</th>
<th>Specified in pixels.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Example: 30</td>
</tr>
</tbody>
</table>
You can also change the values by dragging the box symbols (_drag_ to other positions in the editor.

See also

- Chapter 1.3.5.3.2 “Positioning elements, adapting size and layer” on page 1113

<table>
<thead>
<tr>
<th>“Variable”</th>
<th>Variable (BOOL)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>The value of the variables TRUE and FALSE indicates the switch position on/off.</td>
</tr>
</tbody>
</table>

### Image settings

<table>
<thead>
<tr>
<th>“Transparency”</th>
<th>☑: The “Transparent color” property is selected.</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Transparent color”</td>
<td>Pixels in this color are displayed as transparent. Requirement: “Transparency” is activated.</td>
</tr>
<tr>
<td></td>
<td>• [ ]: The “Color” dialog box opens.</td>
</tr>
<tr>
<td></td>
<td>• [ ]: A drop-down list with style colors opens.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>“Scaling type”</th>
<th>Reaction of the element when the dimension of the “Frame” element is changed:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>• “Isotropic”: The height and width of the image are resized proportionally to the “Frame”.</td>
</tr>
<tr>
<td></td>
<td>Please note: To retain the alignment of elements also within a scaled “Frame” element, define the “Horizontal alignment” or “Vertical alignment” explicitly with “Centered”.</td>
</tr>
<tr>
<td></td>
<td>• “Anisotropic”: The image fills the entire “Frame” regardless of its proportions.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>“Horizontal alignment”</th>
<th>Horizontal alignment of the image within the element frame or element</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>• Left</td>
</tr>
<tr>
<td></td>
<td>• Centered</td>
</tr>
<tr>
<td></td>
<td>• Right</td>
</tr>
<tr>
<td>Requirement: “Scaling type” is “Isotropic”.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>“Vertical alignment”</th>
<th>Vertical alignment of the image within the element frame or element</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>• Top</td>
</tr>
<tr>
<td></td>
<td>• Centered</td>
</tr>
<tr>
<td></td>
<td>• Bottom</td>
</tr>
<tr>
<td>Requirement: “Scaling type” is “Isotropic”.</td>
<td></td>
</tr>
</tbody>
</table>

| “Element behavior” | “Image toggler”: Every mouse click changes the switch and the “Variable” value. |
|                   | “Image tapper”: The switch is “on” and the “Variable” value is TRUE while the mouse button is pressed. |

| “Tap FALSE” | ☑: The value TRUE is assigned to the “Variable” property instead of the value FALSE, and back. |
|             | Requirement: “Image tapper” is selected in the “Element behavior” property. |

**Element property ‘Center’** The properties contain fixed values for the coordinates of the point of rotation. This point of rotation is shown as the _ symbol. The point is used as the center for rotating and scaling.
<table>
<thead>
<tr>
<th>“X”</th>
<th>X-coordinate of the point of rotation</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Y”</td>
<td>Y-coordinate of the point of rotation</td>
</tr>
</tbody>
</table>

You can also change the values by dragging the symbols (⊕) to other positions in the editor.

**Element property 'Texts'**

<table>
<thead>
<tr>
<th>“Tooltip”</th>
<th>String display as tooltip for the element</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example:</td>
<td>Valid access.</td>
</tr>
</tbody>
</table>

**Element property 'Absolute movement'**

The properties contain IEC variables for controlling the position of the element dynamically. The reference point is the upper left corner of the element. In runtime mode, the entire element is moved.

**“Movement”**

<table>
<thead>
<tr>
<th>“X”</th>
<th>Variable (numeric data type). Defines the X position (in pixels).</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example:</td>
<td>PLC_PRG.iPos_X.</td>
</tr>
<tr>
<td></td>
<td>Increasing this value in runtime mode moves the element to the right.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>“Y”</th>
<th>Variable (numeric data type). Defines the Y position (in pixels).</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example:</td>
<td>PLC_PRG.iPos_Y.</td>
</tr>
<tr>
<td></td>
<td>Increasing this value in runtime mode moves the element downwards.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>“Rotation”</th>
<th>Variable (numeric data type). Defines the angle of rotation (in degrees).</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example:</td>
<td>PLC_PRG.iAngle1.</td>
</tr>
<tr>
<td></td>
<td>The midpoint of the element rotates at the “Center” point. This rotation point is shown as the ⊕ symbol.</td>
</tr>
<tr>
<td></td>
<td>In runtime mode, the alignment of the element remains the same with respect to the coordinate system of the visualization. Increasing the value rotates the element to the right.</td>
</tr>
</tbody>
</table>
### "Interior rotation"

Variable (numeric data type). Defines the angle of rotation (in degrees).

**Example:** PLC_PRG.iAngle2.

In runtime mode, the element rotates about the point of rotation specified in "Center" according to the value of the variable. In addition, the alignment of the element rotates according to the coordinate system of the visualization. Increasing the value in the code rotates clockwise.

The rotation point is shown as the \( \circ \) symbol.

Note: If a static angle of rotation is specified in the "Position \( \rightarrow \) Angle" property, then the static angle of rotation is added to the variable angle of rotation (offset) when the visualization is executed.

---

**You can link the variables to a unit conversion.**

---

**The "X", "Y", "Rotation", and "Interior rotation" properties are supported by the "Client Animation" functionality.**

See also
- [Unit conversion](#)

---

**Element property 'State variables'**

The variables control the element behavior dynamically.

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Invisible&quot;</td>
<td>Variable [BOOL]. Toggles the visibility of the element. TRUE: The element is not visible at runtime.</td>
</tr>
<tr>
<td>&quot;Deactivate inputs&quot;</td>
<td>Variable [BOOL]. Toggles the operability of the element. TRUE: User inputs do not have any effect in runtime more. The element is shown as deactivated.</td>
</tr>
</tbody>
</table>

---

**The "Invisible" property is supported by the "Client Animation" functionality.**

See also
- [Unit conversion](#)

---

These properties are available only when you select the "Preview: Support client animations and overlay of native elements" option in the Visualization Manager.

You can animate a movement, a rotation, and the visibility of a visualization element.
**Animation duration**
Time that the element executes an animation (in milliseconds)
Example: 500

**Move to the foreground**
Property value (BOOL)

- TRUE: At runtime, the element is displayed in the foreground.
- FALSE: At runtime, the element is displayed in the layer where it was inserted in the visualization editor.

See also
- Chapter 1.3.5.21.5 “Using client animation” on page 1962

**Image**
Drop-down list with background colors
Depends on the visualization style

**Access rights**
Requirement: User management is set up for the visualization.

<table>
<thead>
<tr>
<th><strong>Access rights</strong></th>
<th>Opens the “Access rights” dialog. There you can edit the access privileges for the element. Status messages:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>- “Not set. Full rights.”: Access rights for all user groups: “operable”</td>
</tr>
<tr>
<td></td>
<td>- “Rights are set: Limited rights”: Access is restricted for at least one group.</td>
</tr>
</tbody>
</table>

See also
- Chapter 1.3.5.19.3.1 “Dialog ‘Access Rights’” on page 1583

See also
- 

**Visualization element ‘Rotary switch’**
Symbol:

Tag: “Lamps/Switches/Bitmaps”
The element assigns a value to a Boolean variable. The switch position "on" the value TRUE to the variable, and the switch position "off" assigns the value FALSE. Use the mouse to change the switch position.
### Element property 'Position'

The position defines the location and size of the element in the visualization window. These are based on the Cartesian coordinate system. The origin is located at the upper left corner of the window. The positive horizontal x-axis runs to the right. The positive vertical y-axis runs downwards.

<table>
<thead>
<tr>
<th>“Element name”</th>
<th>Example: Operating_Switch</th>
</tr>
</thead>
<tbody>
<tr>
<td>Optional</td>
<td></td>
</tr>
<tr>
<td>Hint: Assign individual names for elements so that they are found faster in the element list.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>“Type of element”</th>
<th>“Rotary switch”</th>
</tr>
</thead>
</table>

#### "X"

X coordinate of the upper left corner of the element

- Specified in pixels.
- Example: 10.

#### "Y"

Y coordinate of the upper left corner of the element

- Specified in pixels.
- Example: 10.

#### "Width"

- Specified in pixels.
- Example: 150

#### "Height"

- Specified in pixels.
- Example: 30

You can also change the values by dragging the box symbols ( воды) to other positions in the editor.

See also

- ☒ Chapter 1.3.5.3.2 “Positioning elements, adapting size and layer” on page 1113

<table>
<thead>
<tr>
<th>“Variable”</th>
<th>Variable (BOOL).</th>
</tr>
</thead>
<tbody>
<tr>
<td>The value of the variables TRUE and FALSE indicates the switch position on/off.</td>
<td></td>
</tr>
</tbody>
</table>

### Image settings

<table>
<thead>
<tr>
<th>“Transparency”</th>
<th>☑: The “Transparent color” property is selected.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>“Transparent color”</th>
<th>Pixels in this color are displayed as transparent.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Requirement: &quot;Transparency&quot; is activated.</td>
<td></td>
</tr>
<tr>
<td>☐ The “Color” dialog box opens.</td>
<td></td>
</tr>
<tr>
<td>☐ A drop-down list with style colors opens.</td>
<td></td>
</tr>
</tbody>
</table>
### “Scaling type”

Reaction of the element when the dimension of the “Frame” element is changed:
- **“Isotropic”**: The height and width of the image are resized proportionally to the “Frame”.
  
  Please note: To retain the alignment of elements also within a scaled “Frame” element, define the “Horizontal alignment” or “Vertical alignment” explicitly with “Centered”.
- **“Anisotropic”**: The image fills the entire “Frame” regardless of its proportions.

### “Horizontal alignment”

Horizontal alignment of the image within the element frame or element

- Left
- Centered
- Right

Requirement: “Scaling type” is “Isotropic”.

### “Vertical alignment”

Vertical alignment of the image within the element frame or element

- Top
- Centered
- Bottom

Requirement: “Scaling type” is “Isotropic”.

### “Element behavior”

- **“Image toggler”**: Every mouse click changes the switch and the “Variable” value.
- **“Image tapper”**: The switch is "on" and the “Variable” value is TRUE while the mouse button is pressed.

### “Orientation”

- **“At top”**: The rotary switch turns from the top right to the top left.
- **“At side”**: The rotary switch turns from the top right to the bottom right.

### “Color change”

- ☑: The element changes in color when “Variable” is TRUE.

---

**Element property ‘Center’**

The properties contain fixed values for the coordinates of the point of rotation. This point of rotation is shown as the ☞ symbol. The point is used as the center for rotating and scaling.

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>X-coordinate of the point of rotation</td>
</tr>
<tr>
<td>Y</td>
<td>Y-coordinate of the point of rotation</td>
</tr>
</tbody>
</table>

You can also change the values by dragging the symbols (☞) to other positions in the editor.

---

**Element property ‘Texts’**

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tooltip</td>
<td>String display as tooltip for the element</td>
</tr>
</tbody>
</table>

Example: Valid access.

---

**Element property ‘Absolute movement’**

The properties contain IEC variables for controlling the position of the element dynamically. The reference point is the upper left corner of the element. In runtime mode, the entire element is moved.
## "Movement"

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
<th>Example</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;X&quot;</td>
<td>Variable (numeric data type). Defines the X position (in pixels).</td>
<td>PLC_PRG.iPos_X</td>
<td>Increasing this value in runtime mode moves the element to the right.</td>
</tr>
<tr>
<td>&quot;Y&quot;</td>
<td>Variable (numeric data type). Defines the Y position (in pixels).</td>
<td>PLC_PRG.iPos_Y</td>
<td>Increasing this value in runtime mode moves the element downwards.</td>
</tr>
<tr>
<td>&quot;Rotation&quot;</td>
<td>Variable (numeric data type). Defines the angle of rotation (in degrees).</td>
<td>PLC_PRG.iAngle1</td>
<td>The midpoint of the element rotates at the &quot;Center&quot; point. This rotation point is shown as the symbol. In runtime mode, the alignment of the element remains the same with respect to the coordinate system of the visualization. Increasing the value rotates the element to the right.</td>
</tr>
<tr>
<td>&quot;Interior rotation&quot;</td>
<td>Variable (numeric data type). Defines the angle of rotation (in degrees).</td>
<td>PLC_PRG.iAngle2</td>
<td>In runtime mode, the element rotates about the point of rotation specified in &quot;Center&quot; according to the value of the variable. In addition, the alignment of the element rotates according to the coordinate system of the visualization. Increasing the value in the code rotates clockwise. The rotation point is shown as the symbol. Note: If a static angle of rotation is specified in the &quot;Position ➔ Angle&quot; property, then the static angle of rotation is added to the variable angle of rotation (offset) when the visualization is executed.</td>
</tr>
</tbody>
</table>

You can link the variables to a unit conversion.

The "X", "Y", "Rotation", and "Interior rotation" properties are supported by the "Client Animation" functionality.

See also
- Unit conversion
Element property 'State variables'
The variables control the element behavior dynamically.

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Invisible&quot;</td>
<td>Variable (BOOL). Toggles the visibility of the element. TRUE: The element is not visible at runtime.</td>
</tr>
<tr>
<td>&quot;Deactivate inputs&quot;</td>
<td>Variable (BOOL). Toggles the operability of the element. TRUE: User inputs do not have any effect in runtime more. The element is shown as deactivated.</td>
</tr>
</tbody>
</table>

The "Invisible" property is supported by the "Client Animation" functionality.

See also

- These properties are available only when you select the “Preview: Support client animations and overlay of native elements” option in the Visualization Manager.

You can animate a movement, a rotation, and the visibility of a visualization element.

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Animation duration&quot;</td>
<td>Time that the element executes an animation (in milliseconds)</td>
</tr>
<tr>
<td>Example: 500</td>
<td></td>
</tr>
<tr>
<td>&quot;Move to the foreground&quot;</td>
<td>Property value (BOOL)</td>
</tr>
<tr>
<td>TRUE: At runtime, the element is displayed in the foreground.</td>
<td></td>
</tr>
<tr>
<td>FALSE: At runtime, the element is displayed in the layer where it was inserted in the visualization editor.</td>
<td></td>
</tr>
</tbody>
</table>

See also

- § Chapter 1.3.5.21.5 “Using client animation” on page 1962

Element property 'Background'

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Image&quot;</td>
<td>Drop-down list with background colors</td>
</tr>
<tr>
<td></td>
<td>Depends on the visualization style</td>
</tr>
</tbody>
</table>

Element property 'Access rights'
Requirement: User management is set up for the visualization.

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Access rights&quot;</td>
<td>Opens the &quot;Access rights&quot; dialog. There you can edit the access privileges for the element.</td>
</tr>
<tr>
<td>Status messages:</td>
<td></td>
</tr>
<tr>
<td>&quot;Not set. Full rights.&quot;: Access rights for all user groups: &quot;operable&quot;</td>
<td></td>
</tr>
<tr>
<td>&quot;Rights are set: Limited rights&quot;: Access is restricted for at least one group.</td>
<td></td>
</tr>
</tbody>
</table>

See also

- § Chapter 1.3.5.19.3.1 "Dialog 'Access Rights'" on page 1583
See also

**Visualization element 'Trace'**

Symbol: 

Tag: “Special controls”

The element displays the graphical curve of variable values. In addition, variables can be configured for controlling the view.

See also

- Chapter 1.3.5.10 “Displaying data curve with trace” on page 1162
- Chapter 1.3.5.19.2.13 “Command ‘Configure Trace’” on page 1572

<table>
<thead>
<tr>
<th><strong>Element properties</strong></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Element name&quot;</td>
<td>Example: Velocity</td>
</tr>
<tr>
<td>&quot;Datasource&quot;</td>
<td>Location where the trace data is buffered.</td>
</tr>
<tr>
<td></td>
<td>• &quot;&lt;local application&gt;&quot;: The trace record is listed below the local application. The visualization that contains the trace is located below this application. When the application is downloaded, the trace configuration is downloaded to the local device. During execution, the data is stored locally in the trace buffer.</td>
</tr>
<tr>
<td></td>
<td>• &quot;&lt;data source name&gt;&quot;: Data source that identifies the remote device where the trace record is created. When the local application is downloaded with the visualization, the trace configuration is downloaded to the remote device. During execution, the trace buffer is filled, and the trace data is transferred and then displayed in the local visualization as HMI.</td>
</tr>
<tr>
<td></td>
<td>Example: DataSoure_PLC_A</td>
</tr>
<tr>
<td></td>
<td>Note: The trace buffer is filled only if the remote application is being executed. The data recording is started when the local visualization is started.</td>
</tr>
<tr>
<td>&quot;Application&quot;</td>
<td>Application where data was recorded.</td>
</tr>
<tr>
<td></td>
<td>• Lists all applications that are present below the data source.</td>
</tr>
<tr>
<td></td>
<td>Requirement: A remote data source (not &quot;&lt;local application&gt;&quot;) is referenced in the &quot;Datasource&quot; property.</td>
</tr>
<tr>
<td>&quot;Type of element&quot;</td>
<td>“Trace”</td>
</tr>
<tr>
<td>&quot;Trace&quot;</td>
<td>&quot;&lt;name of trace configuration&gt;&quot;: Opens the “Trace Configuration” dialog where you can modify the trace configuration.</td>
</tr>
</tbody>
</table>

See also

- Chapter 1.3.5.19.2.13 “Command ‘Configure Trace’” on page 1572
- Data Source Manager
**Element property 'Position'**

The position defines the location and size of the element in the visualization window. This is based on the Cartesian coordinate system. The origin is located at the upper left corner of the window. The positive horizontal x-axis runs to the right. The positive vertical y-axis runs downwards.

<table>
<thead>
<tr>
<th><strong>X</strong></th>
<th>The x-coordinate of the upper left corner of the element Specified in pixels Example: 10</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Y</strong></td>
<td>The y-coordinate of the upper left corner of the element Specified in pixels Example: 10</td>
</tr>
<tr>
<td><strong>Width</strong></td>
<td>Specified in pixels Example: 150</td>
</tr>
<tr>
<td><strong>Height</strong></td>
<td>Specified in pixels Example: 30</td>
</tr>
</tbody>
</table>

Tip: You can change the values in “X”, “Y”, “Width”, and “Height” by dragging the corresponding symbols to another position in the editor.

<table>
<thead>
<tr>
<th><strong>Angle</strong></th>
<th>Static angle of rotation (in degrees) Example: 35</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>The element is displayed rotated in the editor. The point of rotation is the center of the element. A positive value rotates clockwise. Tip: You can change the value in the editor by focusing the element to the handle. When the cursor is displayed as a rotating arrow, you can rotate the element about its center as a handle.</td>
</tr>
</tbody>
</table>

(1): Handle

Note: If a dynamic angle of rotation is also configured in the property “Absolute movement ➔ Internal rotation”, then the static and dynamic angles of rotation are added in runtime mode. The static angle of rotation acts as an offset.

See also

- ☞ Chapter 1.3.5.3.2 “Positioning elements, adapting size and layer” on page 1113

<table>
<thead>
<tr>
<th><strong>Show cursor</strong></th>
<th>☑: A cursor (vertical line) is displayed at the mouse position. The trigger and variable values where the cursor points are displayed as a tooltip.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Overwrite existing trace on PLC</strong></td>
<td>☑: If a trace with the same name is on the PLC, then it is overwritten at download with the configuration that is defined here.</td>
</tr>
<tr>
<td><strong>Number format</strong></td>
<td>Number format of values in the tooltip in printf syntax (example: %d, %5.2f).</td>
</tr>
</tbody>
</table>

**Element property 'Control variables'**

The control variables are assigned automatically when you click "Insert elements for controlling trace".
### “Reset Trigger”
Variable (BOOL).
Standard control variable: bResetTrigger
**TRUE**: Resets the triggering. After the action is executed, the variable is set automatically to **FALSE**.

### “Start Trace”
Variable (BOOL).
Standard control variable: bStart
**TRUE**: Starts the trace. After the action is executed, the variable is set automatically to **FALSE**.

### “Stop Trace”
Variable (BOOL).
Standard control variable: bStop
**TRUE**: Stops the trace. After the action is executed, the variable is set automatically to **FALSE**.

### “Save trace to a file”

### “Save Trace”
Variable (BOOL).
Standard control variable: bStore
**TRUE**: Saves the current trace configuration and the data that is stored in the development system to a file. When the action is ended, the variable is set automatically to **FALSE**.

### “File name”
Variable (STRING) that contains the file name of the file to be saved.
Standard control variable: sStoreFilename

### “Load trace from file”

### “Load Trace”
Variable (BOOL).
Standard control variable: bRestore
**TRUE**: Reads the file specified below and loads its contents into the trace editor. The file contains a trace configuration and possibly also trace data. To do this, the stored trace configuration must match the application where the trace configuration is located. When the action is ended, the variable is set automatically to **FALSE**.

**Note**: A trace configuration can be loaded from a file only under special circumstances. The file must have been created with exactly the same (running) application with which it will then be loaded. The consequence of changing the running application (for example by downloading again) is that a file which was previously created from the application cannot no longer be read into the application. Even external manual changes to the file can cause this. You should edit only those configuration settings that have an effect on displaying the variables. If you change variable definitions directly in the file (for example by replacing variable x with v y), then the file cannot be loaded.

### “File name”
Variable (STRING) that contains the file name of the file to be read.
Standard variable: sRestoreFilename

---

**See also**
- § Chapter 1.3.5.19.2.15 “Command 'Insert Elements for Controlling Trace’” on page 1575

**Element property ‘Center’**
The properties contain fixed values for the coordinates of the point of rotation. This point of rotation is shown as the ⪪ symbol. The point is used as the center for rotating and scaling.
The properties contain IEC variables for controlling the position of the element dynamically. The reference point is the upper left corner of the element. In runtime mode, the entire element is moved.

**Element property 'Absolute movement'**

The properties contain IEC variables for controlling the position of the element dynamically. The reference point is the upper left corner of the element. In runtime mode, the entire element is moved.

**Movement**

<table>
<thead>
<tr>
<th>“X”</th>
<th>X-coordinate of the point of rotation</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Y”</td>
<td>Y-coordinate of the point of rotation</td>
</tr>
</tbody>
</table>

You can also change the values by dragging the symbols (ฤษขอ) to other positions in the editor.

**“Movement”**

<table>
<thead>
<tr>
<th>“X”</th>
<th>Variable (numeric data type). Defines the X position (in pixels).</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Example: PLC_PRG.iPos_X.</td>
</tr>
<tr>
<td></td>
<td>Increasing this value in runtime mode moves the element to the right.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>“Y”</th>
<th>Variable (numeric data type). Defines the Y position (in pixels).</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Example: PLC_PRG.iPos_Y.</td>
</tr>
<tr>
<td></td>
<td>Increasing this value in runtime mode moves the element downwards.</td>
</tr>
</tbody>
</table>

**“Rotation”**

Variable (numeric data type). Defines the angle of rotation (in degrees).

Example: PLC_PRG.iAngle1.

The midpoint of the element rotates at the “Center” point. This rotation point is shown as the ⟨rangle symbol.

In runtime mode, the alignment of the element remains the same with respect to the coordinate system of the visualization. Increasing the value rotates the element to the right.

**“Interior rotation”**

Variable (numeric data type). Defines the angle of rotation (in degrees).

Example: PLC_PRG.iAngle2.

In runtime mode, the element rotates about the point of rotation specified in “Center” according to the value of the variable. In addition, the alignment of the element rotates according to the coordinate system of the visualization. Increasing the value in the code rotates clockwise.

The rotation point is shown as the ⟨rangle symbol.

Note: If a static angle of rotation is specified in the “Position ➔ Angle” property, then the static angle of rotation is added to the variable angle of rotation (offset) when the visualization is executed.
You can link the variables to a unit conversion.

The "X", "Y", "Rotation", and "Interior rotation" properties are supported by the "Client Animation" functionality.

See also
- Unit conversion

Element property 'State variables'
The variables control the element behavior dynamically.

<table>
<thead>
<tr>
<th>&quot;Invisible&quot;</th>
<th>Variable (BOOL). Toggles the visibility of the element.</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRUE:</td>
<td>The element is not visible at runtime.</td>
</tr>
</tbody>
</table>

The "Invisible" property is supported by the "Client Animation" functionality.

See also
- 

These properties are available only when you select the “Preview: Support client animations and overlay of native elements” option in the Visualization Manager.

You can animate a movement, a rotation, and the visibility of a visualization element.

<table>
<thead>
<tr>
<th>&quot;Animation duration&quot;</th>
<th>Time that the element executes an animation (in milliseconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example: 500</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>&quot;Move to the foreground&quot;</th>
<th>Property value (BOOL)</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRUE:</td>
<td>At runtime, the element is displayed in the foreground.</td>
</tr>
<tr>
<td>FALSE:</td>
<td>At runtime, the element is displayed in the layer where it was inserted in the visualization editor.</td>
</tr>
</tbody>
</table>

See also
- $Chapter 1.3.5.21.5 “Using client animation” on page 1962

Element property 'Access rights'
Requirement: User management is set up for the visualization.
"Access rights" Opens the "Access rights" dialog. There you can edit the access privileges for the element.
Status messages:
● "Not set. Full rights": Access rights for all user groups: "operable"
● "Rights are set: Limited rights": Access is restricted for at least one group.

See also
● ° Chapter 1.3.5.19.3.1 "Dialog 'Access Rights'" on page 1583

Visualization element 'Trend'
Symbol:

Tag: "Special controls"
The element displays the graphical curve of variable values over a longer period of time. It provides additional controls for the view.

Element properties

<table>
<thead>
<tr>
<th>&quot;Element name&quot;</th>
<th>Example: Velocity</th>
</tr>
</thead>
</table>
| "Data source" | Selection of the device and the application where the data to be visualized and the trend recording are generated
● Remote data source which accesses a remote device, accesses a remote application, and then transfers the data from there
Example: :DataSource_A
Below the (now visible) "Application" property, the remote application is displayed as configured in the data source.
Example: :App_A
Note: If the data source is accessed symbolically by means of a symbol file (CODESYS symbolic), then the required symbol file and the corresponding project have to be saved in the same folder.
● Local application below which the alarm configuration is located
Example: :<local application>"
Example: :Device_A.App_A

<table>
<thead>
<tr>
<th>&quot;Type of element&quot;</th>
<th>&quot;Trend&quot;</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Trend recording&quot;</td>
<td>: Trend recording which is displayed in the visualization and which is located on the device as specified in &quot;Data source&quot;</td>
</tr>
</tbody>
</table>

| "Display Settings" | : The "Display Settings" dialog opens. |

See also
● ° Chapter 1.3.5.19.2.12 "Command 'Configure Display Settings of the Trend'" on page 1570
● Object 'Data source'

Element property 'Position'
The position defines the location and size of the element in the visualization window. This is based on the Cartesian coordinate system. The origin is located at the upper left corner of the window. The positive horizontal x-axis runs to the right. The positive vertical y-axis runs downwards.
### “X”
The x-coordinate of the upper left corner of the element
Specified in pixels
Example: 10

### “Y”
The y-coordinate of the upper left corner of the element
Specified in pixels
Example: 10

### “Width”
Specified in pixels
Example: 150

### “Height”
Specified in pixels
Example: 30

Tip: You can change the values in “X”, “Y”, “Width”, and “Height” by dragging the corresponding symbols to another position in the editor.

### “Angle”
Static angle of rotation (in degrees)
Example: 35
The element is displayed rotated in the editor. The point of rotation is the center of the element. A positive value rotates clockwise.

Tip: You can change the value in the editor by focusing the element to the handle. When the cursor is displayed as a rotating arrow, you can rotate the element about its center as a handle.

Note: If a dynamic angle of rotation is also configured in the property “Absolute movement → Internal rotation”, then the static and dynamic angles of rotation are added in runtime mode. The static angle of rotation acts as an offset.

See also
- § Chapter 1.3.5.3.2 “Positioning elements, adapting size and layer” on page 1113

### “Show cursor”
☑️: A cursor (vertical line) is displayed at the mouse position.

### “Show tooltip”
Requirement: “Show cursor” is activated.
☑️: The information about the variable is displayed in a tooltip at the cursor position.

### “Show frame”
☑️: The visualization element is drawn with a frame.

### “Number format”
Number format of values in the tooltip and in the legend in printf syntax (example: %d, %5.2f).
### Time stamp

"Absolute time stamps": The values of the time stamps are displayed in absolute time. Example: 18.03.2016, 14:05:30:000

The display formatting can be adjusted in the property "Internationalization (format strings)".

### Two-line labelling

- The time stamps are displayed in two lines. The date is displayed in the first line and the time is displayed in the second line.
- Time stamp is displayed in one line. The date and time can also be displayed in one line depending on the formatting.

### Omit irrelevant information in time stamp

- The time stamp has a shorter form. For example, the date is displayed only for the first tick mark, and only the time for the following tick marks. The settings in "Internationalization (format strings)" are ignored for this setting.
- All information is displayed for each time stamp.

### Internationalization (format strings)

Only active when the parameter "Omit irrelevant information in timestamps" is deactivated.

### Date

Definition of the date format. The default setting is taken from the Windows control panel.

### Time

Definition of the time format. The default setting is taken from the Windows control panel.

---

**Element property 'Assigned control elements'**

These elements are created automatically when the control elements are added with the command "Insert elements for controlling trend".

- **Date range picker**: Control element for changing the date and time of the displayed data sets. With all elements that have implemented the interface IDateRangeSelector are available.
- **Time picker**: Control element for changing the time of the displayed data sets. With all elements that have implemented the interface ITimeSelector are available.
- **Legend**: Control element for displaying a legend for the graphs. With all elements that have implemented the interface ILegendDisplayer are available.

See also

- § Chapter 1.3.5.19.2.18 “Command 'Insert Elements for Controlling the Trend'” on page 1577

**Element property 'Center'**

The properties contain fixed values for the coordinates of the point of rotation. This point of rotation is shown as the symbol. The point is used as the center for rotating and scaling.

- **X**: X-coordinate of the point of rotation
- **Y**: Y-coordinate of the point of rotation

---

You can also change the values by dragging the symbols to other positions in the editor.
**Element property 'Absolute movement'**
The properties contain IEC variables for controlling the position of the element dynamically. The reference point is the upper left corner of the element. In runtime mode, the entire element is moved.

**“Movement”**

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
<th>Example</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>“X”</strong></td>
<td>Variable (numeric data type). Defines the X position (in pixels).</td>
<td>PLC_PRG.iPos_X.</td>
<td>Increasing this value in runtime mode moves the element to the right.</td>
</tr>
<tr>
<td><strong>“Y”</strong></td>
<td>Variable (numeric data type). Defines the Y position (in pixels).</td>
<td>PLC_PRG.iPos_Y.</td>
<td>Increasing this value in runtime mode moves the element downwards.</td>
</tr>
<tr>
<td><strong>“Rotation”</strong></td>
<td>Variable (numeric data type). Defines the angle of rotation (in degrees).</td>
<td>PLC_PRG.iAngle1.</td>
<td>The midpoint of the element rotates at the “Center” point. This rotation point is shown as the symbol. In runtime mode, the alignment of the element remains the same with respect to the coordinate system of the visualization. Increasing the value rotates the element to the right.</td>
</tr>
<tr>
<td><strong>“Interior rotation”</strong></td>
<td>Variable (numeric data type). Defines the angle of rotation (in degrees).</td>
<td>PLC_PRG.iAngle2.</td>
<td>In runtime mode, the element rotates about the point of rotation specified in “Center” according to the value of the variable. In addition, the alignment of the element rotates according to the coordinate system of the visualization. Increasing the value in the code rotates clockwise. The rotation point is shown as the symbol. Note: If a static angle of rotation is specified in the “Position → Angle” property, then the static angle of rotation is added to the variable angle of rotation (offset) when the visualization is executed.</td>
</tr>
</tbody>
</table>

You can link the variables to a unit conversion.

The “X”, “Y”, “Rotation”, and “Interior rotation” properties are supported by the “Client Animation” functionality.
Element property 'State variables'

The variables control the element behavior dynamically.

<table>
<thead>
<tr>
<th>&quot;Invisible&quot;</th>
<th>Variable (BOOL). Toggles the visibility of the element.</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRUE: The element is not visible at runtime.</td>
<td></td>
</tr>
</tbody>
</table>

The "Invisible" property is supported by the "Client Animation" functionality.

See also

- [Chapter 1.3.5.21.5 "Using client animation" on page 1962](#)

These properties are available only when you select the "Preview: Support client animations and overlay of native elements" option in the Visualization Manager.

You can animate a movement, a rotation, and the visibility of a visualization element.

<table>
<thead>
<tr>
<th>&quot;Animation duration&quot;</th>
<th>Time that the element executes an animation (in milliseconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example: 500</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>&quot;Move to the foreground&quot;</th>
<th>Property value (BOOL)</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRUE: At runtime, the element is displayed in the foreground.</td>
<td></td>
</tr>
<tr>
<td>FALSE: At runtime, the element is displayed in the layer where it was inserted in the visualization editor.</td>
<td></td>
</tr>
</tbody>
</table>

See also

- [Chapter 1.3.5.21.5 "Using client animation" on page 1962](#)

Element property 'Access rights'

Requirement: User management is set up for the visualization.

<table>
<thead>
<tr>
<th>&quot;Access rights&quot;</th>
<th>Opens the &quot;Access rights&quot; dialog. There you can edit the access privileges for the element.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Status messages:</td>
<td></td>
</tr>
<tr>
<td>- &quot;Not set. Full rights.&quot;. Access rights for all user groups: &quot;operable&quot;</td>
<td></td>
</tr>
<tr>
<td>- &quot;Rights are set: Limited rights&quot;. Access is restricted for at least one group.</td>
<td></td>
</tr>
</tbody>
</table>

See also

- [Chapter 1.3.5.19.3.1 "Dialog 'Access Rights'" on page 1583](#)

Visualization element 'Legend'

Symbol:
Tag: “Special controls”

The element serves as a legend for another element, for example a trend. The legend is assigned in the properties of the other element.

See also

**Element properties**

<table>
<thead>
<tr>
<th>“Element name”</th>
<th>Example: LegendOfTrendA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Optional</td>
<td>Hint: Assign individual names for elements so that they are found faster in the element list.</td>
</tr>
</tbody>
</table>

| “Type of element” | “Legend” |

**Element property ‘Position’**

The position defines the location and size of the element in the visualization window. These are based on the Cartesian coordinate system. The origin is located at the upper left corner of the window. The positive horizontal x-axis runs to the right. The positive vertical y-axis runs downwards.

<table>
<thead>
<tr>
<th>“X”</th>
<th>X coordinate of the upper left corner of the element specified in pixels.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Example: 10.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>“Y”</th>
<th>Y coordinate of the upper left corner of the element specified in pixels.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Example: 10.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>“Width”</th>
<th>Specified in pixels.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Example: 150</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>“Height”</th>
<th>Specified in pixels.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Example: 30</td>
</tr>
</tbody>
</table>

You can also change the values by dragging the box symbols (ठ) to other positions in the editor.

See also

- Chapter 1.3.5.3.2 “Positioning elements, adapting size and layer” on page 1113

**Element property ‘Center’**

The properties contain fixed values for the coordinates of the point of rotation. This point of rotation is shown as the symbol. The point is used as the center for rotating and scaling.

<table>
<thead>
<tr>
<th>“X”</th>
<th>X-coordinate of the point of rotation</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>“Y”</th>
<th>Y-coordinate of the point of rotation</th>
</tr>
</thead>
</table>
You can also change the values by dragging the symbols (🄽) to other positions in the editor.

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Orientation&quot;</td>
<td>Orientation of the element. The value is configured in the assigned element.   ● &quot;Horizontal&quot;   ● &quot;Vertical&quot;</td>
</tr>
<tr>
<td>&quot;Attached element instance&quot;</td>
<td>Example: Element_A</td>
</tr>
<tr>
<td>&quot;Show frame&quot;</td>
<td>![ ]: The element is shown in the class diagram.</td>
</tr>
<tr>
<td>&quot;Number format&quot;</td>
<td>The format of the value in printf syntax (example: %d, %5.2f)</td>
</tr>
</tbody>
</table>

**Element Property 'Layout'**

Defines how many variables can be displayed at a maximum and is calculated from the row and column number.

<table>
<thead>
<tr>
<th>Property</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Max. number of rows&quot;</td>
<td>3</td>
</tr>
<tr>
<td>&quot;Max. number of columns&quot;</td>
<td>2</td>
</tr>
</tbody>
</table>

**Element Property 'Text properties'**

The property affects the text configured in the associated element.

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Text format&quot;</td>
<td>&quot;Default&quot;: The text will be cut and displayed in only the part that fits into the visualization element.</td>
</tr>
<tr>
<td></td>
<td>&quot;Linebreak&quot;: The text will be wrapped in rows.</td>
</tr>
<tr>
<td></td>
<td>&quot;Ellipsis&quot;: The text is cut and ellipsis . . . are added to indicate that something is missing.</td>
</tr>
<tr>
<td>&quot;Font&quot;</td>
<td>Font of the text. The entries of the selection list are defined in the visualization style.</td>
</tr>
<tr>
<td>&quot;Font color&quot;</td>
<td>Text color, for example Grey. The entries of the selection list are defined in the visualization style.</td>
</tr>
<tr>
<td>&quot;Transparency&quot;</td>
<td>Transparency value (255 to 0), which defines the transparency of the corresponding color. Example: 255: The color is opaque. 0: The color is fully transparent.</td>
</tr>
</tbody>
</table>

**Element property 'Absolute movement'**

The properties contain IEC variables for controlling the position of the element dynamically. The reference point is the upper left corner of the element. In runtime mode, the entire element is moved.

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Movement&quot;</td>
<td>&quot;X&quot; Variable (numeric data type). Defines the X position (in pixels). Example: PLC_PRG.iPos_X.</td>
</tr>
<tr>
<td></td>
<td>Increasing this value in runtime mode moves the element to the right.</td>
</tr>
<tr>
<td>Property</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------</td>
<td>--------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>“Y”</td>
<td>Variable (numeric data type). Defines the Y position (in pixels).</td>
</tr>
<tr>
<td>“Rotation”</td>
<td>Variable (numeric data type). Defines the angle of rotation (in degrees).</td>
</tr>
<tr>
<td>“Interior rotation”</td>
<td>Variable (numeric data type). Defines the angle of rotation (in degrees).</td>
</tr>
</tbody>
</table>

You can link the variables to a unit conversion.

The “X”, “Y”, “Rotation”, and “Interior rotation” properties are supported by the “Client Animation” functionality.

See also
- Unit conversion

These properties are available only when you select the “Preview: Support client animations and overlay of native elements” option in the Visualization Manager.

You can animate a movement, a rotation, and the visibility of a visualization element.
**Animation duration**
Time that the element executes an animation (in milliseconds)
**Example:** 500

**Move to the foreground**
Property value (BOOL)
TRUE: At runtime, the element is displayed in the foreground.
FALSE: At runtime, the element is displayed in the layer where it was inserted in the visualization editor.

See also
- [Chapter 1.3.5.21.5 “Using client animation” on page 1962](#)

**Element property 'Access rights'**
Requirement: User management is set up for the visualization.

<table>
<thead>
<tr>
<th><strong>“Access rights”</strong></th>
<th>Opens the “Access rights” dialog. There you can edit the access privileges for the element. Status messages:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>● “Not set. Full rights.”: Access rights for all user groups: “operable”</td>
</tr>
<tr>
<td></td>
<td>● “Rights are set: Limited rights”: Access is restricted for at least one group.</td>
</tr>
</tbody>
</table>

See also
- [Chapter 1.3.5.19.3.1 “Dialog ‘Access Rights’” on page 1583](#)

**Visualization element ‘ActiveX’**
Symbol:

Tag: “Special controls”
The element is used for linking an existing ActiveX element in the visualization. In the element properties of the ActiveX element, you can configure the method calls and their parameters.

In the CODESYS Store, you will find a sample project for using ActiveX components in visualizations.

See also
- [CODESYS Store](#)

**Element properties**

<table>
<thead>
<tr>
<th><strong>“Element name”</strong></th>
<th>Example: GenElemInst_1</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>“Type of element”</strong></td>
<td>“ActiveX element”</td>
</tr>
<tr>
<td><strong>“Element”</strong></td>
<td>Installed ActiveX component that is linked to the visualization. Hint: To avoid typing errors, select the required ActiveX component with the input assistant.</td>
</tr>
</tbody>
</table>
The position defines the location and size of the element in the visualization window. These are based on the Cartesian coordinate system. The origin is located at the upper left corner of the window. The positive horizontal x-axis runs to the right. The positive vertical y-axis runs downwards.

**Element property 'Position'**

<table>
<thead>
<tr>
<th>“X”</th>
<th>X coordinate of the upper left corner of the element Specified in pixels. Example: 10.</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Y”</td>
<td>Y coordinate of the upper left corner of the element Specified in pixels. Example: 10.</td>
</tr>
<tr>
<td>“Width”</td>
<td>Specified in pixels. Example: 150</td>
</tr>
<tr>
<td>“Height”</td>
<td>Specified in pixels. Example: 30</td>
</tr>
</tbody>
</table>

You can also change the values by dragging the box symbols (_drag_ ) to other positions in the editor.

See also

- ¶ Chapter 1.3.5.3.2 “Positioning elements, adapting size and layer” on page 1113

**Element property 'Center'**

The properties contain fixed values for the coordinates of the point of rotation. This point of rotation is shown as the symbol. The point is used as the center for rotating and scaling.

<table>
<thead>
<tr>
<th>“X”</th>
<th>X-coordinate of the point of rotation</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Y”</td>
<td>Y-coordinate of the point of rotation</td>
</tr>
</tbody>
</table>

You can also change the values by dragging the symbols (drag_ ) to other positions in the editor.

**Element property 'Absolute movement'**

The properties contain IEC variables for controlling the position of the element dynamically. The reference point is the upper left corner of the element. In runtime mode, the entire element is moved.

<table>
<thead>
<tr>
<th>“Movement”</th>
<th>Variable (numeric data type). Defines the X position (in pixels). Example: PLC_PRG.iPos_X.</th>
</tr>
</thead>
<tbody>
<tr>
<td>“X”</td>
<td>Increasing this value in runtime mode moves the element to the right.</td>
</tr>
</tbody>
</table>
### “Y”

Variable (numeric data type). Defines the Y position (in pixels).

**Example:** PLC_PRG.iPos_Y.

Increasing this value in runtime mode moves the element downwards.

### “Rotation”

Variable (numeric data type). Defines the angle of rotation (in degrees).

**Example:** PLC_PRG.iAngle1.

The midpoint of the element rotates at the “Center” point. This rotation point is shown as the symbol.

In runtime mode, the alignment of the element remains the same with respect to the coordinate system of the visualization. Increasing the value rotates the element to the right.

### “Scaling”

Variable (integer data type). Causes centric stretching.

**Example:** PLC_PRG.iScaling.

The reference point is the “Center” property.

The value 1 shrinks the element by a factor of 0.001. The value 1000 returns the element to its original size.

### “Interior rotation”

Variable (numeric data type). Defines the angle of rotation (in degrees).

**Example:** PLC_PRG.iAngle2.

In runtime mode, the element rotates about the point of rotation specified in “Center” according to the value of the variable. In addition, the alignment of the element rotates according to the coordinate system of the visualization. Increasing the value in the code rotates clockwise.

The rotation point is shown as the symbol.

Note: If a static angle of rotation is specified in the property “Position ➔ Angle”, then the static angle of rotation is added to the variable angle of rotation (offset) when the visualization is executed.

---

### You can link the variables to a unit conversion.

### The properties “X”, “Y”, “Rotation”, and “Interior rotation” are supported by the “Client Animation” functionality.

See also

- **Unit conversion**
The variables control the element behavior dynamically.

| "Invisible" | Variable (BOOL). Toggles the visibility of the element. TRUE: The element is not visible at runtime. |

The "Invisible" property is supported by the "Client Animation" functionality.

See also

These method calls are executed during initialization. They are executed in the first cycle only.

| "Method calls " | Button "Create new" Creates a subnode below "Methods" with parameters for the method call. |
| "Methods" | "[<number>]" |
|  | ● "Method": Name of the method |
|  | ● "Parameter": Parameter passed at the method call |
|  | ● "Result parameter": Optional variable for the return value of the method |

These method calls are executed in every cycle. They are executed in the refresh rate of the visualization.

| "Method calls " | Button "Create new" Creates a subnode below "Methods" for a method call and its parameters. |
| "Methods" | "[<number>]" |
|  | ● "Method": Name of the method |
|  | ● "Parameter": Parameter passed at the method call |
|  | ● "Result parameter": Optional variable for the return value of the method |

These method calls are executed in the refresh rate of the visualization. You define the call condition in the property "Methods ➔ [<number>] ➔ Call condition".

| "Method calls " | Button "Create new" Creates a subnode below "Methods" with a call condition and parameters for the method call. |
| "Methods" | "[<number>]" |
|  | ● "Method": Name of the method |
|  | ● "Call condition": Variable (BOOL). A rising edge of this variable triggers the call of this method. |
|  | ● "Parameter": Parameter passed at the method call |
|  | ● "Result parameter": Optional variable for the return value of the method |
These properties are available only when you select the “Preview: Support client animations and overlay of native elements” option in the Visualization Manager.

You can animate a movement, a rotation, and the visibility of a visualization element.

<table>
<thead>
<tr>
<th><strong>“Animation duration”</strong></th>
<th>Time that the element executes an animation (in milliseconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Example: 500</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>“Move to the foreground”</strong></th>
<th>Property value (BOOL)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>TRUE: At runtime, the element is displayed in the foreground.</td>
</tr>
<tr>
<td></td>
<td>FALSE: At runtime, the element is displayed in the layer where it was inserted in the visualization editor.</td>
</tr>
</tbody>
</table>

See also

- § Chapter 1.3.5.21.5 “Using client animation” on page 1962

**Element property ‘Access rights’**

Requirement: User management is set up for the visualization.

<table>
<thead>
<tr>
<th><strong>“Access rights”</strong></th>
<th>Opens the “Access rights” dialog. There you can edit the access privileges for the element.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Status messages:</td>
</tr>
<tr>
<td></td>
<td>• “Not set. Full rights.”: Access rights for all user groups: “operable”</td>
</tr>
<tr>
<td></td>
<td>• “Rights are set: Limited rights”: Access is restricted for at least one group.</td>
</tr>
</tbody>
</table>

See also

- § Chapter 1.3.5.19.3.1 “Dialog ‘Access Rights’” on page 1583

**Visualization element ‘Webbrowser’**

Symbol:

Tag: “Special controls”

The element shows a website, PDF file, or video that has a URL.

![NOTICE!](image) The display options of the “Webbrowser” element depend on the operating system and the display variant of the visualization.

Requirement: The software components of the web browser are available in the runtime system and configured accordingly (example: videos to be shown on Linux).

See also

- § Chapter 1.3.5.21.4 “Displaying web content” on page 1961

**Element properties**
<table>
<thead>
<tr>
<th>“Element name”</th>
<th>Example: GenElemInst_59</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Type of element”</td>
<td>“Webbrowser”</td>
</tr>
</tbody>
</table>

**Element property 'Position'**
The position defines the location and size of the element in the visualization window. These are based on the Cartesian coordinate system. The origin is located at the upper left corner of the window. The positive horizontal x-axis runs to the right. The positive vertical y-axis runs downwards.

| “X” | X coordinate of the upper left corner of the element  
Specified in pixels.  
Example: 10. |
| “Y” | Y coordinate of the upper left corner of the element  
Specified in pixels.  
Example: 10. |
| “Width” | Specified in pixels.  
Example: 150 |
| “Height” | Specified in pixels.  
Example: 30 |

You can also change the values by dragging the box symbols (_draged|) to other positions in the editor.

See also

- Chapter 1.3.5.3.2 “Positioning elements, adapting size and layer” on page 1113

**Element property 'Center'**
The properties contain fixed values for the coordinates of the point of rotation. This point of rotation is shown as the symbol. The point is used as the center for rotating and scaling.

| “X” | X-coordinate of the point of rotation |
| “Y” | Y-coordinate of the point of rotation |

You can also change the values by dragging the symbols (_draged|) to other positions in the editor.

**Element property 'Absolute movement'**
The properties contain IEC variables for controlling the position of the element dynamically. The reference point is the upper left corner of the element. In runtime mode, the entire element is moved.
## Movement

<table>
<thead>
<tr>
<th></th>
<th>Description</th>
<th>Example</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>X</strong></td>
<td>Variable (numeric data type). Defines the X position (in pixels).</td>
<td>PLC_PRG.iPos_X.</td>
<td>Increasing this value in runtime mode moves the element to the right.</td>
</tr>
<tr>
<td><strong>Y</strong></td>
<td>Variable (numeric data type). Defines the Y position (in pixels).</td>
<td>PLC_PRG.iPos_Y.</td>
<td>Increasing this value in runtime mode moves the element downwards.</td>
</tr>
<tr>
<td><strong>Rotation</strong></td>
<td>Variable (numeric data type). Defines the angle of rotation (in degrees).</td>
<td>PLC_PRG.iAngle1.</td>
<td>The midpoint of the element rotates at the “Center” point. This rotation point is shown as the + symbol.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><img src="image1.png" alt="Diagram" /></td>
<td>In runtime mode, the alignment of the element remains the same with respect to the coordinate system of the visualization. Increasing the value rotates the element to the right.</td>
</tr>
<tr>
<td><strong>Scaling</strong></td>
<td>Variable (integer data type). Causes centric stretching.</td>
<td>PLC_PRG.iScaling.</td>
<td>The reference point is the “Center” property.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><img src="image2.png" alt="Diagram" /></td>
<td>The value 1 shrinks the element by a factor of 0.001. The value 1000 returns the element to its original size.</td>
</tr>
<tr>
<td><strong>Interior rotation</strong></td>
<td>Variable (numeric data type). Defines the angle of rotation (in degrees).</td>
<td>PLC_PRG.iAngle2.</td>
<td>In runtime mode, the element rotates about the point of rotation specified in “Center” according to the value of the variable. In addition, the alignment of the element rotates according to the coordinate system of the visualization. Increasing the value in the code rotates clockwise.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><img src="image3.png" alt="Diagram" /></td>
<td>The rotation point is shown as the + symbol.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><img src="image4.png" alt="Diagram" /></td>
<td>Note: If a static angle of rotation is specified in the property “Position ➔ Angle”, then the static angle of rotation is added to the variable angle of rotation (offset) when the visualization is executed.</td>
</tr>
</tbody>
</table>

You can link the variables to a unit conversion.

The properties “X”, “Y”, “Rotation”, and “Interior rotation” are supported by the "Client Animation" functionality.
See also

- *Unit conversion*

**Element property 'State variables'**
The variables control the element behavior dynamically.

<table>
<thead>
<tr>
<th>Variable (BOOL)</th>
<th>Toggles the visibility of the element.</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRUE: The element is not visible at runtime.</td>
<td></td>
</tr>
</tbody>
</table>

The "Invisible" property is supported by the "Client Animation" functionality.

See also

- *

**Element property 'Control variables'**

<table>
<thead>
<tr>
<th>Variable (STRING or WSTRING)</th>
<th>URL of the web page that is displayed in the visualization.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example: PLC_PRG.stURL</td>
<td>If the variable contains a rising edge, then the visualization calls the web page given in &quot;URL&quot; and displays its contents in the 'Web browser' visualization element.</td>
</tr>
</tbody>
</table>

| Literal in single straight quotation marks | Example: 'http://de.wikipedia.org' |

<table>
<thead>
<tr>
<th>Variable (BOOL)</th>
<th>Controls the display of the &quot;Web browser&quot; element.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example: PLC_PRG.bSetURL</td>
<td>If the variable contains a rising edge, then the visualization displays the contents of the previously displayed page.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Variable (BOOL)</th>
<th>Controls the back navigation in the &quot;Web browser&quot;.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example: PLC_PRG.bGoBack</td>
<td>If the variable has a rising edge, then the visualization displays the contents of the previously displayed page.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Variable (BOOL)</th>
<th>Controls the forward navigation in the &quot;Web browser&quot;.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example: PLC_PRG.bGoForward</td>
<td>If the variable has a rising edge, then the visualization displays the contents of the previously displayed page.</td>
</tr>
</tbody>
</table>

These properties are available only when you select the "Preview: Support client animations and overlay of native elements" option in the Visualization Manager.

You can animate a movement, a rotation, and the visibility of a visualization element.
“Animation duration”  
Time that the element executes an animation (in milliseconds)  
Example: 500

“Move to the foreground”  
Property value (BOOL)  
TRUE: At runtime, the element is displayed in the foreground.  
FALSE: At runtime, the element is displayed in the layer where it was inserted in the visualization editor.

See also  
● Chapter 1.3.5.21.5 “Using client animation” on page 1962

Element property ‘Access rights’  
Requirement: User management is set up for the visualization.

“Access rights”  
Opens the “Access rights” dialog. There you can edit the access privileges for the element.  
Status messages:  
● “Not set. Full rights.”: Access rights for all user groups: “operable”  
● “Rights are set: Limited rights”: Access is restricted for at least one group.

See also  
● Chapter 1.3.5.19.3.1 “Dialog ‘Access Rights’” on page 1583

Visualization element ‘Waiting symbol cube’

Symbol: 

Tag: “Special controls”

In runtime mode, this element indicates automatically that the runtime system is busy or waiting for data.

Element properties

“Element name”  
Example: Data_Transfer  
Optional  
Hint: Assign individual names for elements so that they are found faster in the element list.

“Type of element”  
Waiting symbol cube

Element property 'Position'  
The position defines the location and size of the element in the visualization window. These are based on the Cartesian coordinate system. The origin is located at the upper left corner of the window. The positive horizontal x-axis runs to the right. The positive vertical y-axis runs downwards.
<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>“X”</td>
<td>X coordinate of the upper left corner of the element. Specified in pixels. Example: 10.</td>
</tr>
<tr>
<td>“Width”</td>
<td>Specified in pixels. Example: 150</td>
</tr>
<tr>
<td>“Height”</td>
<td>Specified in pixels. Example: 30</td>
</tr>
</tbody>
</table>

You can also change the values by dragging the box symbols in the editor.

See also

- Chapter 1.3.5.3.2 “Positioning elements, adapting size and layer” on page 1113

**Element property ‘Center’**

The properties contain fixed values for the coordinates of the point of rotation. This point of rotation is shown as the symbol. The point is used as the center for rotating and scaling.

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>“X”</td>
<td>X-coordinate of the point of rotation</td>
</tr>
<tr>
<td>“Y”</td>
<td>Y-coordinate of the point of rotation</td>
</tr>
</tbody>
</table>

You can also change the values by dragging the symbols in the editor.

**Element property ‘Absolute movement’**

The properties contain IEC variables for controlling the position of the element dynamically. The reference point is the upper left corner of the element. In runtime mode, the entire element is moved.

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Movement”</td>
<td>Variable (numeric data type). Defines the X position (in pixels). Example: PLC_PRG.iPos_X. Increasing this value in runtime mode moves the element to the right.</td>
</tr>
<tr>
<td>“Y”</td>
<td>Variable (numeric data type). Defines the Y position (in pixels). Example: PLC_PRG.iPos_Y. Increasing this value in runtime mode moves the element downwards.</td>
</tr>
</tbody>
</table>
### “Rotation”

Variable (numeric data type). Defines the angle of rotation (in degrees).

**Example:** PLC_PRG.iAngle1.

The midpoint of the element rotates at the “Center” point. This rotation point is shown as the symbol.

In runtime mode, the alignment of the element remains the same with respect to the coordinate system of the visualization. Increasing the value rotates the element to the right.

### “Interior rotation”

Variable (numeric data type). Defines the angle of rotation (in degrees).

**Example:** PLC_PRG.iAngle2.

In runtime mode, the element rotates about the point of rotation specified in “Center” according to the value of the variable. In addition, the alignment of the element rotates according to the coordinate system of the visualization. Increasing the value in the code rotates clockwise.

The rotation point is shown as the symbol.

Note: If a static angle of rotation is specified in the “Position Angle” property, then the static angle of rotation is added to the variable angle of rotation (offset) when the visualization is executed.

You can link the variables to a unit conversion.

The “X”, “Y”, “Rotation”, and “Interior rotation” properties are supported by the “Client Animation” functionality.

See also

- *Unit conversion*

**Element property ‘State variables’**

The variables control the element behavior dynamically.

<table>
<thead>
<tr>
<th>“Invisible”</th>
<th>Variable (BOOL). Toggles the visibility of the element. TRUE: The element is not visible at runtime.</th>
</tr>
</thead>
</table>

The “Invisible” property is supported by the “Client Animation” functionality.
See also

These properties are available only when you select the “Preview: Support client animations and overlay of native elements” option in the Visualization Manager.

You can animate a movement, a rotation, and the visibility of a visualization element.

<table>
<thead>
<tr>
<th>“Animation duration”</th>
<th>Time that the element executes an animation (in milliseconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Example: 500</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>“Move to the foreground”</th>
<th>Property value (BOOL)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>TRUE: At runtime, the element is displayed in the foreground.</td>
</tr>
<tr>
<td></td>
<td>FALSE: At runtime, the element is displayed in the layer where it was inserted in the visualization editor.</td>
</tr>
</tbody>
</table>

See also

- § Chapter 1.3.5.21.5 “Using client animation” on page 1962

Element property 'Access rights'

Requirement: User management is set up for the visualization.

<table>
<thead>
<tr>
<th>“Access rights”</th>
<th>Opens the “Access rights” dialog. There you can edit the access privileges for the element.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Status messages:</td>
</tr>
<tr>
<td></td>
<td>- “Not set. Full rights.”: Access rights for all user groups : “operable”</td>
</tr>
<tr>
<td></td>
<td>- “Rights are set: Limited rights”: Access is restricted for at least one group.</td>
</tr>
</tbody>
</table>

See also

- § Chapter 1.3.5.19.3.1 “Dialog ‘Access Rights’” on page 1583

See also

- §

Visualization element 'Waiting symbol flower'

Symbol:

Tag: “Special controls”

The element indicates that the system is busy or waiting for data.
**“Element name”**

Example: `Data_Transfer`

Optional

Hint: Assign individual names for elements so that they are found faster in the element list.

**“Type of element”**

“Waiting symbol flower”

---

**Element property 'Position’**

The position defines the location and size of the element in the visualization window. These are based on the Cartesian coordinate system. The origin is located at the upper left corner of the window. The positive horizontal x-axis runs to the right. The positive vertical y-axis runs downwards.

| **“X”** | X coordinate of the upper left corner of the element  
Specified in pixels.  
Example: 10. |
|---------|----------------------------------------------------------|
| **“Y”** | Y coordinate of the upper left corner of the element    
Specified in pixels.  
Example: 10. |
| **“Width”** | Specified in pixels.  
Example: 150 |
| **“Height”** | Specified in pixels.  
Example: 30 |

You can also change the values by dragging the box symbols to other positions in the editor.

See also

- Chapter 1.3.5.3.2 “Positioning elements, adapting size and layer” on page 1113

---

**Element property ‘Center’**

The properties contain fixed values for the coordinates of the point of rotation. This point of rotation is shown as the symbol. The point is used as the center for rotating and scaling.

<table>
<thead>
<tr>
<th><strong>“X”</strong></th>
<th>X-coordinate of the point of rotation</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>“Y”</strong></td>
<td>Y-coordinate of the point of rotation</td>
</tr>
</tbody>
</table>

You can also change the values by dragging the symbols to other positions in the editor.

---

**Element property ‘Colors’**

The properties contain fixed values for setting colors.
### “Frame color”

<table>
<thead>
<tr>
<th>“Fill color”</th>
<th>Value (0 to 255) for defining the transparency of the selected color. Example 255: The color is opaque. 0: The color is completely transparent.</th>
</tr>
</thead>
</table>

### “Transparency”

<table>
<thead>
<tr>
<th>“Line width”</th>
<th>Value in pixels Example: 2</th>
</tr>
</thead>
</table>

Note: The values 0 and 1 both result in a line weight of 1 pixel. If no line should be displayed, then the “Line style” property must be set to the option “Invisible”.

### “Fill attributes”

The way in which the element is filled.

- **Filled**: The element is filled with the color from property “Colors ➔ Fill color”.
- **Invisible**: The fill color is invisible.

### “Line style”

Type of line representation

- **Solid**
- **Dashes**
- **Dots**
- **Dash Dot**
- **Dash Dot Dot**
- **not visible**

You can assign variables in the “Appearance variables” property for controlling the appearance dynamically. The fixed values here are overwritten.

See also

- Paragraph 1.3.5.19.3.5 “Dialog ‘Gradient Editor’” on page 1586
- Paragraph 1.3.5.19.3.5 “Dialog ‘Appearance Editor’” on page 1586

### “Symbol color”

Selection of a color for the flower symbol.

### “Line”

Stroke width of the lines (in pixels).

### Element property ‘Appearance’

The properties contain fixed values for setting the look of the element.

### “Movement”

<table>
<thead>
<tr>
<th>“X”</th>
<th>Variable (numeric data type). Defines the X position (in pixels). Example: PLC_PRG.iPos_X. Increasing this value in runtime mode moves the element to the right.</th>
</tr>
</thead>
</table>

### “Y”
Variable (numeric data type). Defines the Y position (in pixels).

**Example:** PLC_PRG.iPos_Y.

Increasing this value in runtime mode moves the element downwards.

### “Rotation”
Variable (numeric data type). Defines the angle of rotation (in degrees).

**Example:** PLC_PRG.iAngle1.

The midpoint of the element rotates at the “Center” point. This rotation point is shown as the symbol.

In runtime mode, the alignment of the element remains the same with respect to the coordinate system of the visualization. Increasing the value rotates the element to the right.

### “Interior rotation”
Variable (numeric data type). Defines the angle of rotation (in degrees).

**Example:** PLC_PRG.iAngle2.

In runtime mode, the element rotates about the point of rotation specified in “Center” according to the value of the variable. In addition, the alignment of the element rotates according to the coordinate system of the visualization. Increasing the value in the code rotates clockwise.

The rotation point is shown as the symbol.

Note: If a static angle of rotation is specified in the “Position ➔ Angle” property, then the static angle of rotation is added to the variable angle of rotation (offset) when the visualization is executed.

---

You can link the variables to a unit conversion.

The “X”, “Y”, “Rotation”, and “Interior rotation” properties are supported by the “Client Animation” functionality.

---

Element property 'State variables'

The variables control the element behavior dynamically.

<table>
<thead>
<tr>
<th>“Invisible”</th>
<th>Variable (BOOL). Toggles the visibility of the element.</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRUE: The element is not visible at runtime.</td>
<td></td>
</tr>
</tbody>
</table>
The “Invisible” property is supported by the “Client Animation” functionality.

See also

- These properties are available only when you select the “Preview: Support client animations and overlay of native elements” option in the Visualization Manager.

You can animate a movement, a rotation, and the visibility of a visualization element.

| “Animation duration” | Time that the element executes an animation (in milliseconds)  
Example: 500 |
|----------------------|---------------------------------------------------------------|
| “Move to the foreground” | Property value (BOOL)  
TRUE: At runtime, the element is displayed in the foreground.  
FALSE: At runtime, the element is displayed in the layer where it was inserted in the visualization editor. |

See also

- Chapter 1.3.5.21.5 “Using client animation” on page 1962

Element property ‘Access rights’

Requirement: User management is set up for the visualization.

| “Access rights” | Opens the “Access rights” dialog. There you can edit the access privileges for the element.  
Status messages:  
- "Not set. Full rights."; Access rights for all user groups : “operable”  
- "Rights are set: Limited rights"; Access is restricted for at least one group. |

See also

- Chapter 1.3.5.19.3.1 “Dialog ‘Access Rights’” on page 1583

See also

- 

Visualization element ‘Text editor’

Symbol:

Tag: “Special controls”

The element shows the contents of text files that are saved on the controller. Files can be encoded in ASCII or Unicode formats.

A visualization user can also edit the text.
Element properties

| "Element name" | Example: GenElemInst_1 |
| "Type of element" | "Text editor" |

Element property 'Position'

The position defines the location and size of the element in the visualization window. These are based on the Cartesian coordinate system. The origin is located at the upper left corner of the window. The positive horizontal x-axis runs to the right. The positive vertical y-axis runs downwards.

| "X" | X coordinate of the upper left corner of the element  
Specified in pixels.  
Example: 10. |
| "Y" | Y coordinate of the upper left corner of the element  
Specified in pixels.  
Example: 10. |
| "Width" | Specified in pixels.  
Example: 150 |
| "Height" | Specified in pixels.  
Example: 30 |

You can also change the values by dragging the box symbols (_drag_box_) to other positions in the editor.

See also

- ☰ Chapter 1.3.5.3.2 “Positioning elements, adapting size and layer” on page 1113

Element property 'Font'

| "Font name" | Non-proportional font used by the visualization to display the contents of the file  
Example: “Courier New” |
| "Size" | Font size  
Example: 12 |

Element property 'Control variables'
### Table 351: Element property “Control variables --> File”

| “Variable” | Variable (STRING). Contains the file names and optionally the location of the file. It is located in the file system of the controller.  
|            | Example: FLC_PRG.strFile: STRING := '/Documentation/Info.txt'; |
| “Open”     | Variable (BOOL). Controls opening the file which is defined in the “Variable” property  
|            | Example: bOpen: BOOL;  
|            | TRUE: The file is opened. |
| “Close”    | Variable (BOOL). Controls closing the file which is defined in the “Variable” property  
|            | Example: bClose: BOOL;  
|            | TRUE: The file is closed. |
| “Save”     | Variable (BOOL). Controls saving the file which is defined in the “Variable” property  
|            | Example: bStore: BOOL;  
|            | TRUE: The file is saved. |
| “New”      | Variable (BOOL). Controls creating a new file. The name is defined in the “Variable” property.  
|            | Example: bCreate: BOOL;  
|            | TRUE: A file is created and opened. |

### Table 352: Element property “Control variables --> Edit”

| “Variable” | Variable (STRING). Contains the string to search for in the file  
|            | Example: strFind: STRING := 'abc'; |
| “Find”     | Variable (BOOL). Controls executing the search for the string in the “Variable” property  
|            | Example: bFind: BOOL;  
|            | TRUE: The search is performed. The variable is automatically reset to FALSE. |
| “Find next”| Variable (BOOL). Controls the location to begin the search in the file  
|            | Example: bFindNext: BOOL;  
|            | TRUE: The search begins at the last search result location.  
|            | FALSE: The search begins at the beginning of the file. |

### Table 353: Element property “Control variables --> Cursor position”

| “Line” | Variable (integer data type). Contains the line of the cursor  
|        | Example: iRowCursor: INT; |
| “Column” | Variable (integer data type). Contains the column of the cursor  
|         | Example: iColumnCursor: INT; |
"Position"  Output variable (integer data type). Shows the **absolute** cursor position in the text.
Example: `iPosCursor: INT;`

"Set cursor"  Variable (BOOL). Controls the setting of the cursor at a specific location
Example: `iSetCursor: INT;`
TRUE: The cursor is moved. The new position is defined in the "Line" and "Column" properties.
FALSE: The "Line", "Column", and "Position" properties contain the actual values.
Note: The variable is used as the control variable for an input event triggered by a visualization user.

<table>
<thead>
<tr>
<th>Table 354: Element property “Control variables --&gt; Selection”</th>
</tr>
</thead>
</table>
| **Start position**  Output variable (integer data type). Shows the **absolute** position for starting the text selection
Example: `iPosSelection: INT;` |
| **End position**  Output variable (integer data type). Shows the **absolute** position for ending the text selection.
Example: `iPosEndSelection: INT;` |
| **Start line number**  Output variable (integer data type). Shows the line where the text selection begins
Example: `iRowSelection: INT;` |
| **Start column index**  Output variable (integer data type). Shows the column where the text selection begins
Example: `iColumnSelection: INT;` |
| **End line number**  Output variable (integer data type). Shows the line where the text selection ends
Example: `iRowEndSelection: INT;` |
| **End column index**  Output variable (integer data type). Shows the column where the text selection ends
Example: `iColumnEndSelection: INT;` |
| **Line to select**  Variable (integer data type). Contains the line number that is selected
Note: The selection is controlled by the variables in the "Trigger selection" property. |
| **Set selection**  Variable (BOOL). Controls the selection of a line.
Example: `bSetSelection: BOOL;`
TRUE: The line from the "Line to select" property is selected and highlighted in the text editor.
if the line is not in the current text segment of the text editor, then the text segment is moved to this line.
Note: The variable is used as the control variable for an input event triggered by a visualization user. The control variable is not reset automatically. You are responsible for this to occur in the visualization. |
Table 355: Element property "Control variables --> Error handling"

| Variable for error code | Variable (integer data type). Contains the error code when an error occurs. Example: iError: INT; The error codes are declared in GVL_ErrorCodes in the VisuElemTextEditor library. To display the error text, the VisuFctTextEditorGetErrorText() function of the library must be called. |

| Variable for content changed | Variable (BOOL). Shows whether the contents have changed. Example: bIsContentEdited: BOOL; TRUE: The contents of the text editor have changed. |

| Variable for access mode | Variable (BOOL). Controls the access privileges to the file. Example: bIsReadOnly: BOOL; TRUE: A visualization user has read-only permission. At runtime, the file contents are highlighted in gray in the text editor. FALSE: A visualization user has read/write permission. Note: The variable overwrites the setting in the “Editor mode” property. |

| Maximum line length | Maximum number of characters per line |

| Editor mode | ● "Read-only": A visualization user has read-only permissions to the file. At runtime, the file contents are highlighted in gray in the text editor. ● "Read/write": A visualization user has read-write permissions. |

Element property 'New files'

| Encoding | Character encoding of the new file: ● “ASCII” ● “Unicode (Little endian)” ● “Unicode (Big endian)” |

| New line character sequence | End of line character of the new file: ● “CR/LF”: Normal for Windows systems ● “LF”: Normal for UNIX systems Please note: When a visualization user opens an existing file, the end-of-line character of the file is detected and used automatically. |

These properties are available only when you select the “Preview: Support client animations and overlay of native elements” option in the Visualization Manager.

You can animate a movement, a rotation, and the visibility of a visualization element.
<table>
<thead>
<tr>
<th><strong>“Animation duration”</strong></th>
<th>Time that the element executes an animation (in milliseconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example: 500</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>“Move to the foreground”</strong></th>
<th>Property value (BOOL)</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRUE:</td>
<td>At runtime, the element is displayed in the foreground.</td>
</tr>
<tr>
<td>FALSE:</td>
<td>At runtime, the element is displayed in the layer where it was inserted in the visualization editor.</td>
</tr>
</tbody>
</table>

See also
- Chapter 1.3.5.21.5 “Using client animation” on page 1962

**Element property ‘Access rights’**

Requirement: User management is set up for the visualization.

<table>
<thead>
<tr>
<th><strong>“Access rights”</strong></th>
<th>Opens the “Access rights” dialog. There you can edit the access privileges for the element.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Status messages:</td>
<td></td>
</tr>
<tr>
<td>“Not set. Full rights.”: Access rights for all user groups: “operable”</td>
<td></td>
</tr>
<tr>
<td>“Rights are set: Limited rights”: Access is restricted for at least one group.</td>
<td></td>
</tr>
</tbody>
</table>

See also
- Chapter 1.3.5.19.3.1 “Dialog ‘Access Rights’” on page 1583

See also

**Visualization element ‘Path3D’**

Symbol:

![Path3D Symbol](image)

Tag: “Special controls”

The visualization element “Path3D” graphically displays the curves of two independent records as a 3D path. It is specially designed for use with CODESYS SoftMotion CNC in order to display the trajectory of a machine tool or a robot. The programmed path (path) and the path actually traveled (track) is visualized.

Although the visualization element is designed for use with CODESYS SoftMotion CNC, it can also be used to display any other record. In this case the application must provide the path data. The sample application 3D Path Generator, which is available in the CODESYS Store, shows how these data can be generated.

If the element is used together with SoftMotion CNC, then function blocks from the library SM3_CNC_Visu help to generate the data from the path and track. These function blocks are used by the sample project CNC_File_3DPath, which is stored in the installation directory of CODESYS.

- SMC_PathCopier
- SMC_PathCopierCompleteQueue
- SMC_PathCopierFile
- SMC_PositionTracker
A description of the function blocks can be found in the Library Manager in the SM3_CNC_Visu library.

The element does not work with the CODESYS HMI display variant.

See also
- CNC Example 6: Using Path3D with SoftMotion CNC

### Element properties

<table>
<thead>
<tr>
<th>“Element name”</th>
<th>Example: GenElemInst_1</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Type of element”</td>
<td>“Path3D”</td>
</tr>
</tbody>
</table>

### Element property 'Position'

The position defines the location and size of the element in the visualization window. These are based on the Cartesian coordinate system. The origin is located at the upper left corner of the window. The positive horizontal x-axis runs to the right. The positive vertical y-axis runs downwards.

| “X” | X coordinate of the upper left corner of the element  
Specified in pixels.  
Example: 10. |
| “Y” | Y coordinate of the upper left corner of the element  
Specified in pixels.  
Example: 10. |
| “Width” | Specified in pixels.  
Example: 150 |
| “Height” | Specified in pixels.  
Example: 30 |

You can also change the values by dragging the box symbols (_drag_ ) to other positions in the editor.

See also
- Chapter 1.3.5.3.2 “Positioning elements, adapting size and layer” on page 1113

### Path description
### Path data (VisuStruct3DTrack)

Variable of the type `VisuStruct3DTrack`, which is declared in the IEC code. Example: `PLC_PRG.pc.vs3dt`. A description of the structure can be found in the library manager in the library `VisuElem3DPath.library`.

The data structure describes a path or track through a certain number of points. The points are determined and buffered by the application. The track typically displays the last n positions, so that only a certain part of them is ever displayed at any one time. `VisuStruct3DTrack.pProjection` is a variable that is set by the visualization element and contains information about the path/track projection. It can be read (only) by the application. In addition, the methods `Projection.Apply` or `ApplyV` can be used in order to see whether the transformed position lies inside or outside the visualization display area, which is defined by `Projection.ElementRect`.

### Path color

Color of the path drawn

### Path line width

Path line width in pixels, e.g.: "2"

### Style of boundary points

Display of the points between two successive objects in the path
- End points are not displayed
- End points are marked with a circle
- End points are marked with a cross
- End points are marked with a plus

---

### Track description

The track data are structured in exactly the same way as the path data: `VisuStruct3DTrack`.

### Track data (VisuStruct3DTrack)

Variable of the type `VisuStruct3DTrack`, which is declared in the IEC code. Example: `PLC_PRG.pc.vs3dt`. A description of the structure can be found in the library manager in the library `VisuElem3DPath.library`.

### Track color

Color of the track drawn

### Track line width

Track line width in pixels, e.g.: "2"

---

### Camera control

The camera position for the 3D mode is controlled with a reference to the external data structure. This structure allows the following operations:

- Shifting to the left/to the right/upwards/downwards
- Rotation around the X/Y/Z axis
- Resetting of the view at X/Y, Y/Z or Z/X level, so that the path and the track are completely visible.

### Control data (VisuStruct3DControl)

Variable of the type `VisuStruct3DControl`, which is declared in the IEC code. Example: `PLC_PRG.pc.vs3dc`.

A description of the structure can be found in the library manager in the library `VisuElem3DPath`.

The values can be set via the application itself or via the visualization element "ControlPanel". The library `VisuElem3DPath` contains ready-to-use visualization frames that provide a possible user interface for these data.

### Additional aspects
### Highlighting

Individual parts of the path can be visually highlighted. This is typically used to mark the already processed part of a track with a different color. Each point in the path is given a unique ID, which in the case of a CNC editor is linked with the object ID on which the point lies. This ID ("highlight ID") can be specified via the application so that dynamic elements/parts of the track can be highlighted.

#### "Highlight mode"

Select one of the following highlight modes:
- Only the element whose ID corresponds to the value of the variable is highlighted.
- All elements whose ID (linked with the object ID in the case of a CNC editor) is smaller than or equal to the value in "Variable" are highlighted.

#### "Variable"

Project variable that specifies the ID of an element. Example: PLC_PRG.iVarElementID. This "highlight ID" is taken into account for the setting of the "highlight mode". The variable must be set in the IEC application.

#### "Highlight color"

These properties are available only when you select the "Preview: Support client animations and overlay of native elements" option in the Visualization Manager.

You can animate a movement, a rotation, and the visibility of a visualization element.

#### "Animation duration"

Time that the element executes an animation (in milliseconds)

Example: 500

#### "Move to the foreground"

Property value (BOOL)

**TRUE**: At runtime, the element is displayed in the foreground.

**FALSE**: At runtime, the element is displayed in the layer where it was inserted in the visualization editor.
Requirement: User management is set up for the visualization.

<table>
<thead>
<tr>
<th>“Access rights”</th>
<th>Opens the “Access rights” dialog. There you can edit the access privileges for the element.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Status messages:</td>
<td>• “Not set. Full rights.”: Access rights for all user groups: “operable”</td>
</tr>
<tr>
<td></td>
<td>• “Rights are set: Limited rights”: Access is restricted for at least one group.</td>
</tr>
</tbody>
</table>

See also

- Chapter 1.3.5.21.5 “Using client animation” on page 1962

Visualization element ‘Control Panel’

Symbol:

Tag: “Special controls”
This visualization element is used in connection with the “Path3D” visualization element. It is used for changing the position and orientation to the CNC path shown with “Path3D”.

See also

- Chapter 1.3.5.19.3.1 “Dialog ‘Access Rights’” on page 1583

Element properties

<table>
<thead>
<tr>
<th>“Element name”</th>
<th>Optional.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hint</td>
<td>Assign individual names for elements so that they are found faster in the element list.</td>
</tr>
<tr>
<td>Example</td>
<td>Camera_Path_1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>“Type of element”</th>
<th>“Frame”</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>“Clipping”</th>
<th>If you have set the “Scaling type” to “Fixed”, then only that part of the visualization is displayed that fits in the frame.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>“Show frame”</th>
<th>Displays the frame</th>
</tr>
</thead>
<tbody>
<tr>
<td>• “No frame”: The displayed area of the frame does not have borders.</td>
<td></td>
</tr>
<tr>
<td>• “Frame”: The displayed area of the frame has borders.</td>
<td></td>
</tr>
<tr>
<td>• “No frame with offset”: The displayed area of the frame does not have a border and the displayed area of the referenced visualization is reduced inwards by one pixel as compared to the frame area. The resulting gap prevents the referenced visualization from touching any adjacent elements.</td>
<td></td>
</tr>
</tbody>
</table>

See also

- Chapter 1.3.5.18.1.42 “Visualization element ‘Path3D’” on page 1499
**“Scaling type”**

Describes how the frame reacts when the visualization is resized:

- **“Isotropic”**: The frame retains its proportions. This allows the ratio of height to width to be preserved, even if the height and width of the visualization have been changed separately.
- **“Anisotropic”**: The frame depends on the size of the visualization, so that height and width of the referenced visualization can be changed separately.
- **“Fixed”**: The original size of the frame is retained, regardless of the visualization size. If you have also selected the “Clipping” option, then only the fitting part is displayed.
- **“Fixed and scrollable”**: The referenced visualization is displayed without scaling. If it is larger than the window area of the frame, then scrollbars are included on the frame. To set the position of the scrollbar with a variable, use the “Scroll position variable horizontal” or “Scroll position variable vertical” property.

**“Deactivation of the background drawing”**

- To optimize the performance of the visualization, the non-animated elements of the frame element are drawn as a background bitmap. This could result in the elements not being displayed in the expected order.
- Deactivation of the background drawing. This can prevent the behavior described above.

**Element property 'References'**

Contains the currently configured visualization references as a subnode

**“References”**

Clicking “Configure” opens the “Frame Configuration” dialog. This is used to manage the referenced visualizations.

Caution: Visualizations can be nested at any depth by means of frame elements. In order to use the “Switch to any visualization” frame selection type without any problems, a frame must not contain more than 21 referenced visualizations. For more information, see also the description for the “Input configuration” of an element: Action “Switch frame visualization”.

**List of the currently referenced visualizations**

Visualizations that have a button also have this displayed as a subnode. Each interface variable is listed with the currently assigned transfer parameters.

Example:

```
vis_FormA
```

- `iDataToDisplay_1:PLC_PRG.iVar1`
- `iDataToDisplay_2:PLC_PRG.iVar2`

Hint: You can change the assignment of the variables to an interface variable here and edit the value field. Or click the “Configure” button instead.

See also

- 🍃 Chapter 1.3.5.19.2.1 “Command 'Interface Editor'” on page 1556
- 🍃
- 🍃 Chapter 1.3.5.19.3.6 “Dialog 'Input Configuration'” on page 1586

**Element property 'Position'**

The position defines the location and size of the element in the visualization window. These are based on the Cartesian coordinate system. The origin is located at the upper left corner of the window. The positive horizontal x-axis runs to the right. The positive vertical y-axis runs downwards.
### Element property 'Center'

The properties contain fixed values for the coordinates of the point of rotation. This point of rotation is shown as the symbol. The point is used as the center for rotating and scaling.

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>X</code></td>
<td>X-coordinate of the point of rotation</td>
</tr>
<tr>
<td><code>Y</code></td>
<td>Y-coordinate of the point of rotation</td>
</tr>
</tbody>
</table>

You can also change the values by dragging the symbols () to other positions in the editor.

### Element property 'Colors'

The properties contain fixed values for setting colors.

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>Color</code></td>
<td>Color for the element in its normal state. Please note that the normal state is in effect if the expression in the &quot;Color variables ➔ Toggle color&quot; property is not defined or it has the value FALSE.</td>
</tr>
<tr>
<td><code>Alarm color</code></td>
<td>Color for the element in alarm state. Please note that the alarm state is in effect if the expression in the &quot;Color variables ➔ Toggle color&quot; property has the value TRUE.</td>
</tr>
<tr>
<td><code>Transparency</code></td>
<td>Value (0 to 255) for defining the transparency of the selected color. Example 255: The color is opaque. 0: The color is completely transparent.</td>
</tr>
<tr>
<td><code>Use gradient color</code></td>
<td>☑: The element is displayed with a color gradient.</td>
</tr>
<tr>
<td><code>Gradient setting</code></td>
<td>The “Color gradient editor” dialog box opens.</td>
</tr>
<tr>
<td><code>Frame color</code></td>
<td>Example: “Black”</td>
</tr>
<tr>
<td><code>Fill color</code></td>
<td>Example: “Light gray”</td>
</tr>
</tbody>
</table>
Element property 'Appearance'

The properties contain fixed values for setting the look of the element.

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>“Line width”</strong></td>
<td>Value in pixels</td>
</tr>
<tr>
<td></td>
<td>Example: 2</td>
</tr>
<tr>
<td></td>
<td>Note: The values 0 and 1 both result in a line weight of one pixel. If no line should be displayed, then the “Line style” property must be set to the option “Invisible”.</td>
</tr>
<tr>
<td><strong>“Line style”</strong></td>
<td>Type of line representation</td>
</tr>
<tr>
<td></td>
<td>“Solid”</td>
</tr>
<tr>
<td></td>
<td>“Dashes”</td>
</tr>
<tr>
<td></td>
<td>“Dots”</td>
</tr>
<tr>
<td></td>
<td>“Dash Dot”</td>
</tr>
<tr>
<td></td>
<td>“Dash Dot Dot”</td>
</tr>
<tr>
<td></td>
<td>“not visible”</td>
</tr>
</tbody>
</table>

You can assign variables in the “Appearance variables” property for controlling the appearance dynamically. The fixed values are defined here.

Element property 'Texts'

The properties contains character strings for labeling the element. The character string can also contain a placeholder with a format definition. In runtime mode, the placeholder is replaced by the current value in the specified format.

CODESYS accepts the specified texts automatically into the “GlobalTextList” text list. Therefore, these texts can be localized.

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>“Text”</strong></td>
<td>Character string (without single straight quotation marks) for the labeling the element. Add a line break by pressing the keyboard shortcut [Ctrl] + [Enter]. Example: Accesses: %i The variable that contains the current value for the placeholder is specified in the property “Text variable ➔ Text”.</td>
</tr>
<tr>
<td><strong>“Tooltip”</strong></td>
<td>Character string (without single straight quotation marks) that is displayed as the tooltip of an element. Example: Number of valid accesses. The variable that contains the current value for the placeholder is specified in the property “Text variable ➔ Tooltip”.</td>
</tr>
</tbody>
</table>

See also

- § “Element property 'Appearance variables’” on page 1285
- § “Element property 'Text variables’” on page 1229
- § Chapter 1.3.5.18.2 “Placeholders with format definition in the output text” on page 1546
The properties contain fixed values for the text properties.

<table>
<thead>
<tr>
<th>“Horizontal alignment”</th>
<th>Horizontal alignment of the text within the element.</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Vertical alignment”</td>
<td>Vertical alignment of the text within the element.</td>
</tr>
</tbody>
</table>

**“Text format”**
Definition for displaying texts that are too long
- **“Default”**: The long text is truncated.
- **“Line break”**: The text is split into parts.
- **“Ellipsis”**: The visible text ends with "..." indicating that it is not complete.

**“Font”**
Example: **“Default”**
- The “Font” dialog box opens.
- ▼: Drop-down list with style fonts.

**“Font color”**
Example: **“Black”**
- The “Color” dialog box opens.
- ▼: Drop-down list with style colors.

**“Transparency”**
Whole number (value range from 0 to 255). This determines the transparency of the respective color.
Example: **255**: The color is opaque.
**0**: The color is completely transparent.
Please note: If the color is a style color and already has a transparency value, then this property is write-protected.

The properties contain IEC variables for controlling the position of the element dynamically. The reference point is the upper left corner of the element. In runtime mode, the entire element is moved.

**“Movement”**

<table>
<thead>
<tr>
<th>“X”</th>
<th>Variable (numeric data type). Defines the X position (in pixels).</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example:</td>
<td>PLC_PRG.iPos_X.</td>
</tr>
<tr>
<td>Increasing</td>
<td>this value in runtime mode moves the element to the right.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>“Y”</th>
<th>Variable (numeric data type). Defines the Y position (in pixels).</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example:</td>
<td>PLC_PRG.iPos_Y.</td>
</tr>
<tr>
<td>Increasing</td>
<td>this value in runtime mode moves the element downwards.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>“Rotation”</th>
<th>Variable (numeric data type). Defines the angle of rotation (in degrees).</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example:</td>
<td>PLC_PRG.iAngle1.</td>
</tr>
<tr>
<td>The midpoint of the element rotates at the “Center” point. This rotation point is shown as the symbol.</td>
<td></td>
</tr>
<tr>
<td>In runtime mode, the alignment of the element remains the same with respect to the coordinate system of the visualization. Increasing the value rotates the element to the right.</td>
<td></td>
</tr>
</tbody>
</table>
### "Scaling"

Variable (integer data type). Causes centric stretching.

Example: PLC_PRG.iScaling.

The reference point is the "Center" property.

The value 1 shrinks the element by a factor of 0.001. The value 1000 returns the element to its original size.

### "Interior rotation"

Variable (numeric data type). Defines the angle of rotation (in degrees).

Example: PLC_PRG.iAngle2.

In runtime mode, the element rotates about the point of rotation specified in "Center" according to the value of the variable. In addition, the alignment of the element rotates according to the coordinate system of the visualization. Increasing the value in the code rotates clockwise.

The rotation point is shown as the symbol.

Note: If a static angle of rotation is specified in the property "Position → Angle", then the static angle of rotation is added to the variable angle of rotation (offset) when the visualization is executed.

---

You can link the variables to a unit conversion.

The properties “X”, “Y”, “Rotation”, and “Interior rotation” are supported by the "Client Animation" functionality.

See also
- Unit conversion

---

**Element property 'Relative movement'**

The properties contains variables for moving the element. The reference point is the position of the element ("Position" property). The shape of the element can change.

<table>
<thead>
<tr>
<th>&quot;Movement top-left&quot;</th>
<th>&quot;Movement bottom-right&quot;</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>&quot;X&quot;</strong> Variable (integer data type). It contains the number (in pixels) that the left edge is moved horizontally. Incrementing the value moves the element to the right.</td>
<td><strong>&quot;Y&quot;</strong> Variable (integer data type). It contains the number (in pixels) that the top edge is moved vertically. Incrementing the value moves the element to the down.</td>
</tr>
<tr>
<td>Example: PLC_PRG.iDeltaX</td>
<td>Example: PLC_PRG.iDeltaY</td>
</tr>
</tbody>
</table>
| "X" | Variable (integer data type). It contains the number (in pixels) that the right edge is moved horizontally. Incrementing the value moves the element to the right.  
Example: PLC_PRG.iDeltaWidth |
| "Y" | Variable (integer data type). It contains the number (in pixels) that the bottom edge is moved vertically. Incrementing the value moves the element to the down.  
Example: PLC_PRG.iDeltaHeight |

See also
- “Element property ‘Absolute movement’” on page 1227

**Element property ‘Text variables’**

These properties are variables with contents that replace a format definition.

| "Text variable" | Variable (data type compliant with the format definition). It contains what is printed instead of the format definition.  
Example: PLC_PRG.iAccesses  
Note: The format definition is part of the text in the property “Texts ➔ Text”.  
Note: If you specify a variable of type enumeration with text list support, then the name of the enumeration data type is added automatically in angle brackets after the variable name. Example: PLC_PRG.enVar <enumeration name>. Then the symbolic value of the enumeration component is printed instead of the numeric value when text is printed. Refer to the help page for the enumerations. |
| "Tooltip variable" | Variable (data type compliant with the format definition). It contains what is printed instead of the format definition.  
Example: PLC_PRG.iAccessesInTooltip  
Note: The format definition is part of the text in the property “Texts ➔ Tooltip”. |

See also
- Chapter 1.3.5.18.2 “Placeholders with format definition in the output text” on page 1546
- “Element property ‘Texts’” on page 1226
- *Enumerations*

**Element property ‘Dynamic texts’**

Dynamic texts are variably indexed texts of a text list. At runtime, the text is displayed that is currently indexed in the variable.
"Text list" | Variable (string) or name of the text list as a fixed string in single straight quotation marks.  
Example: 'Errorlist'  
▼: Drop-down list with the dialogs available in the text lists.

"Text index" | Text list ID. This refers to the desired output text.  
● As fixed string with the ID in single straight quotation marks.  
Example: '1'  
● As a variable (STRING) for dynamically controlling the text output.  
Example: strTextID  
Sample assignment: PLC_PRG.strTextID := '1';

"Tooltip index" | Text list ID. This refers to the desired output text.  
● As fixed string with the ID in single straight quotation marks.  
Example: '2'  
● As a variable (STRING) for dynamically controlling the text output.  
Example: strToolTipID  
Sample assignment: PLC_PRG.strToolTipID := '2';

See also  
● Text list

Element property 'Font variables'  
The variables allow for dynamic control of the text display.

"Font name" | Variable (STRING). Includes the font of the text.  
Example: PLC_PRG.stFontVar := 'Arial';  
The selection of fonts corresponds to the default "Font" dialog.

"Size" | Variable (numeric data type). Contains the font size (in pixels or points). The applied unit is specified in brackets after the variable name.  
● <pt>: Points (default)  
Example: PLC_PRG.iFontHeight <pt>  
Code: iFontHeight : INT := 12;  
● <px> : Pixels  
Example: PLC_PRG.iFontHeight <px>  
Code: iFontHeight : INT := 19;

| pt |  
| px |

If you click in the value field, a drop-down list opens on the right for setting the unit.  

Hint: The font size is specified in points (example: Arial 12). Use points when the variable font size should match a font, for example if a font is set in the property "Text property ➔ Font".
<table>
<thead>
<tr>
<th><strong>“Flags”</strong></th>
<th>Variable (DWORD). Contains the flags for displaying fonts. Flags:</th>
</tr>
</thead>
<tbody>
<tr>
<td>● 1: Italics</td>
<td>● 2: Bold</td>
</tr>
<tr>
<td>● 4: Underline</td>
<td>● 8: Strikethrough</td>
</tr>
<tr>
<td>Note: You can combine the font displays by adding the coding of the flags. For example, a bold and underlined text: PLC_PRG.dwFontType := 6;</td>
<td></td>
</tr>
</tbody>
</table>

| **“Character set”** | Variable (DWORD). Contains a character set number for the font. The selection of character set numbers corresponds to the “Script” setting of the standard “Font” dialog. |

| **“Color”** | Variable (DWORD). Includes the color of the text. Example: PLC_PRG.dwColorFont := 16#FF000000; |

<table>
<thead>
<tr>
<th><strong>“Flags for text alignment”</strong></th>
<th>Variable (integer data type). Contains the coding for text alignment. Example: PLC_PRG.dwTextAlignment. Coding:</th>
</tr>
</thead>
<tbody>
<tr>
<td>● 0: Top left</td>
<td>● 1: Horizontal center</td>
</tr>
<tr>
<td>● 2: Right</td>
<td>● 4: Vertical center</td>
</tr>
<tr>
<td>● 8: Bottom</td>
<td>Note: You can combine the text alignments by adding the coding of the flags. For example, a vertical and horizontal centered text: PLC_PRG.dwFontType := 5;</td>
</tr>
</tbody>
</table>

---

*Fixed values for displaying texts are set in “Text properties”.*

See also
- ¶ “Element property 'Text properties’” on page 1227

---

**Element property 'Colorvariables’**

The Element property is used as an interface for project variables to dynamically control colors at runtime.
“Toggle color”  The property controls the toggled color at runtime.
Value assignment:
- **FALSE**: The element is displayed with the color specified in the “Color” property.
- **TRUE**: The element is displayed with the color specified in the “Alarm color” property.
Assignment options:
- Placeholder for the user input variable
  - `<toggle/tap variable>`
  - `<NOT toggle/tap variable>`
The color change is not controlled by its own variable, but by a user input variable.
Note: Specify a variable for the mouse events “Tap” or “Toggle” in the input configuration of the element. Only then is the placeholder set. If you configure a variable in both “Toggle” and “Tap”, then the variable specified in “Tap” is used.
Hint: Click the symbol to insert the placeholder “<toggle/tap variable>”. When you activate the “Inputconfiguration” “Tap FALSE” property, then the “<NOT toggle/tap variable>” placeholder is displayed.
- Instance path of a project variable (BOOL)
  Example: PLC_PRG.xColorIsToggeled
Note: In the code, declare and implement the variable specified here. Its value assignment determines when the color changes.

“Normal state”

“Alarm state”  The properties listed below control the color depending on the state. The normal state is in effect if the variable in “Color variables”, “Toggle color” is not defined or it has the value FALSE. The alarm state is in effect if the variable in “Colorvariables”, “Toggle color” has the value TRUE.

“Frame color”  Assignment options:
- Variable (DWORD) for the frame color
  Example: PLC_PRG.dwBorderColor
- Color literal
  Example of green and opaque: 16#FF00FF00

“Filling color”  Assignment options:
- Variable (DWORD) for the fill color
  Example: PLC_PRG.dwFillColor
- Color literal
  Example of gray and opaque: 16#FF888888

The transparency part of the color value is evaluated only if the “Activate semi-transparent drawing” option of the visualization manager is selected.

Select the “Advanced” option in the toolbar of the properties view. Then all element properties are visible.

See also
- § Chapter 1.3.5.8.3 “Animating a color display” on page 1152
### Element property 'Appearance variables'

The properties contain variables for controlling the appearance of the element dynamically.

<table>
<thead>
<tr>
<th>“Line width”</th>
<th>Variable (integer data type). Contains the line weight (in pixels). Note: The values 0 and 1 both result in a line weight of one pixel. If no line should be displayed, then the “Line style” property must be set to the option “Invisible”.</th>
</tr>
</thead>
</table>
| “Line style” | Variable (DWORD). Controls the line style. Coding:  
  - 0: Solid line  
  - 1: Dashed line  
  - 2: Dotted line  
  - 3: Line type "Dash Dot"  
  - 3: Line type "Dash Dot Dot"  
  - 8: Invisible: The line is not drawn. |

**Fixed values can be set in the “Appearance” property. These values can be overwritten by dynamic variables at runtime.**

See also
- § “Element property ‘Appearance’” on page 1238

### Element property 'Switch frame variable'

The variable controls the switching of the referenced visualizations. This variable indexes one of the referenced frame visualizations and this is displayed in the frame. When the value of the variable changes, it switches to the recently indexed visualization.

| “Variable” | Variable (integer data type) that contains the index of the active visualization. Example: PLC_PRG.uiIndexVisu  
  
  Hint: The “Frame Configuration” dialog includes a list of referenced visualizations. The visualizations are automatically numerically indexed via the order in the list. Note: This variant of switching usually affects all connected display variants.  
  
  Array element (integer data type) for index access via CURRENTCLIENTID. Example: PLC_PRG.aIndexVisu[CURRENTCLIENTID]  
  
  Note: This variant of switching applies to the current client only, and therefore only on one display variant. That is the display variant where the value change was triggered (for example, by means of user input). |

See also
- § Chapter 1.3.5.19.2.9 “Command ‘Frame Selection’” on page 1564

### Element property 'State variables'

The variables control the element behavior dynamically.
"Invisible" Variable (BOOL). Toggles the visibility of the element. 
TRUE: The element is not visible at runtime.

"Deactivate inputs" Variable (BOOL). Toggles the operability of the element. 
TRUE: User inputs do not have any effect in runtime more. The element is shown as deactivated.

The "Invisible" property is supported by the "Client Animation" functionality.

See also

These properties are available only when you select the "Preview: Support client animations and overlay of native elements" option in the Visualization Manager.

You can animate a movement, a rotation, and the visibility of a visualization element.

"Animation duration" Time that the element executes an animation (in milliseconds)
Example: 500

"Move to the foreground" Property value (BOOL) 
TRUE: At runtime, the element is displayed in the foreground.  
FALSE: At runtime, the element is displayed in the layer where it was inserted in the visualization editor.

See also

● Chapter 1.3.5.21.5 “Using client animation” on page 1962

Element property 'Input configuration' The properties contain the configurations for the user input when using the mouse or keyboard. A user input defines an event and one or more actions that are executed when an event occurs.

The "Configure" button opens the "Input configuration" dialog box for creating or modifying a user input. Configured user inputs are briefly listed below the events. They each include the action that is triggered and the setting in short form.

Example: "Execute ST code": # PLC_PRG.i_x := 0;

"OnDialogClosed" Input event: The user closes the dialog box.

"OnMouseClick" Input event: The user clicks the mouse button completely. The mouse button is clicked and released.

"OnMouseDown" Input event: The user clicks down on the mouse button.

"OnMouseEnter" Input event: The user drags the mouse pointer to the element.

"OnMouseLeave" Input event: The user drags the mouse pointer away from the element.

"OnMouseMove" Input event: The user moves the mouse pointer over the element area.

"OnMouseUp" Input event: The user releases the mouse button over the element area.
### “Tap”
When a mouse click event occurs, the variable defined in “Variable” is described in the application. The coding depends on the options “Tap FALSE” and “Tap on enter if captured”.

### “Variable”
Variable (BOOL) that is set on mouse click.

**Example:** PLC_PRG.bIsTapped

- **TRUE:** A mouse click event exists. It lasts while the user presses the mouse button over the element. It ends when the button is released.
- **FALSE:** A mouse click event does not exist.

**Requirement:** The “Tap FALSE” option is not activated.

### “Tap FALSE”
- **TRUE:** The mouse click event leads to a complementary value in “Variable”.
- **FALSE:** A mouse click event does not exist.

### “Tap on enter if captured”
- **TRUE:** During user input, it is also taken into consideration whether the mouse pointer is dragged within the element area or not while the mouse button is pressed.
- **FALSE:** While the mouse click event exists.

- **TRUE:** While the mouse click event exists and the mouse pointer is moved over the element area.
- **FALSE:** A mouse click event does not exist. Or the user moves the mouse pointer outside of the element area while the mouse button is pressed.

The value is **TRUE** again as soon as the user moves the pointer back to the element area. The mouse is then captured.

### “Shift”
With the onset of a mouse click event, the variable is set; when the mouse click event is completed, the variable is reset.

### “Variable”
Variable (BOOL). Its value toggled when the mouse click event is ended. This is when the user releases the mouse button while the mouse pointer is over the element area.

If the user releases the mouse button while the mouse pointer is outside of the element area, then the mouse click event is not ended and the value is not toggled.

**Tip:** The user can cancel a started toggle input by dragging the mouse pointer out of the element area.

### “Toggle on up if captured”
- **TRUE:** The value toggles regardless of where the mouse pointer is when the mouse button is released. The mouse is then captured.

### “Keyboard shortcuts”
Keyboard shortcut on the element for triggering specific input actions.

When the keyboard shortcut event occurs, the input actions in the “Event(s)” property are triggered. In this way, it is not the input action itself that leads to this input action, but the mouse input action.

### “Key”
Key pressed for input action.

**Example:** [T]

**Note:** The following properties appear when a key is selected.
### Event(s)
- **None**
- **Mouse down**: Pressing the key triggers the input actions that are configured in the "OnMouseDown" property.
- **Mouse up**: Releasing the key triggers the input actions that are configured in the "OnMouseUp" property.
- **Mouse down/up**: Pressing and releasing the key triggers the input actions that are configured in the "OnMouseDown" property and the "OnMouseUp" property.

### Shift
- Combination with the Shift key
- Example: [Shift]+[T].

### Control
- Combination with the Ctrl key
- Example: [Ctrl]+[T].

### Alt
- Combination with the Alt key
- Example: [Alt]+[T].

---

> All keyboard shortcuts and their actions that are configured in the visualization are listed in the "Keyboard configuration" tab.

---

See also
- § Chapter 1.3.5.19.2.2 “Command ‘Keyboard Configuration’” on page 1558
- § Chapter 1.3.5.19.3.6 “Dialog ‘Input Configuration’” on page 1586

---

### Visualization element ‘Date range picker’

**Symbol:**

![Symbol](image)

**Tag:** “Date/time controls”

The element provides the capability of selecting the date and time range of a saved data set. The element is used with the “Trend” visualization element.

#### Element properties

<table>
<thead>
<tr>
<th>“Element name”</th>
<th>Example: DateTrend1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Optional</td>
<td>Hint: Assign individual names for elements so that they are found faster in the element list.</td>
</tr>
</tbody>
</table>

**Type of element**: “Date range picker”

---

### Element property 'Position'

The position defines the location and size of the element in the visualization window. These are based on the Cartesian coordinate system. The origin is located at the upper left corner of the window. The positive horizontal x-axis runs to the right. The positive vertical y-axis runs downwards.
<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>X coordinate of the upper left corner of the element. Specified in pixels. Example: 10.</td>
</tr>
<tr>
<td>Y</td>
<td>Y coordinate of the upper left corner of the element. Specified in pixels. Example: 10.</td>
</tr>
<tr>
<td>Width</td>
<td>Specified in pixels. Example: 150</td>
</tr>
<tr>
<td>Height</td>
<td>Specified in pixels. Example: 30</td>
</tr>
</tbody>
</table>

You can also change the values by dragging the box symbols to other positions in the editor.

See also

- Chapter 1.3.5.3.2 “Positioning elements, adapting size and layer” on page 1113

**Element property ‘Center’**

The properties contain fixed values for the coordinates of the point of rotation. This point of rotation is shown as the symbol. The point is used as the center for rotating and scaling.

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>X-coordinate of the point of rotation</td>
</tr>
<tr>
<td>Y</td>
<td>Y-coordinate of the point of rotation</td>
</tr>
</tbody>
</table>

You can also change the values by dragging the symbols to other positions in the editor.

**Element property 'Tick mark labels'**

- Show frame: The visualization element is displayed with a frame.
- Resolution: Resolution saved for the time stamp: “Millisecond” or “Microsecond”
- Attached element instance: The element can be assigned to a “Trend” visualization element. In this way, the time range of the trend element can be changed. The available visual elements are selected with the help of the input assistant.
"Two-line labelling"  
☑: The time stamps are displayed in two lines. The date is displayed in the first line and the time is displayed in the second line.  
☐: Time stamp is displayed in one line. The date and time can also be displayed in one line depending on the formatting.

"Omit irrelevant information in time stamp"  
☑: The time stamp has a shorter form. For example, the date is displayed only for the first tick mark, and only the time for the following tick marks. The settings in "Internationalization (format strings)" are ignored for this setting.  
☐: All information is displayed for all time stamps.

"Internationalization (format strings)"  
Only active when the parameter “Omit irrelevant information in timestamps” is deactivated.

"Date"  
Definition of the date format. The default setting is taken from the Windows control panel.

"Time"  
Definition of the time format. The default setting is taken from the Windows control panel.

Element property 'Text properties'  
The properties contain fixed values for the text properties.

"Horizontal alignment"  
Horizontal alignment of the text within the element.

"Vertical alignment"  
Vertical alignment of the text within the element.

"Font"  
Example: “Default”  
☑: The “Font” dialog box opens.  
▼: Drop-down list with style fonts.

"Font color"  
Example: “Black”  
☑: The “Color” dialog box opens.  
▼: Drop-down list with style colors.

"Transparency"  
Whole number (value range from 0 to 255). This determines the transparency of the respective color.  
Example: 255: The color is opaque.  
0: The color is completely transparent.  
Please note: If the color is a style color and already has a transparency value, then this property is write-protected.

Element property 'Additional buttons'  

"Jump to the largest possible time stamp"  
☑: An additional button ( ) is displayed for jumping to the last time stamp.

"Jump to the smallest possible time stamp"  
☑: An additional button ( ) is displayed for jumping to the first time stamp.

"Zoom out"  
☑: An additional button ( ) is displayed for setting the current min./max. range to the maximum range. The selected range is left.

Element property 'Absolute movement'  
The properties contain IEC variables for controlling the position of the element dynamically. The reference point is the upper left corner of the element. In runtime mode, the entire element is moved.
### Movement

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
<th>Example</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>X</strong></td>
<td>Variable (numeric data type). Defines the X position (in pixels).</td>
<td>Example: PLC_PRG.iPos_X.</td>
<td>Increasing this value in runtime mode moves the element to the right.</td>
</tr>
<tr>
<td><strong>Y</strong></td>
<td>Variable (numeric data type). Defines the Y position (in pixels).</td>
<td>Example: PLC_PRG.iPos_Y.</td>
<td>Increasing this value in runtime mode moves the element downwards.</td>
</tr>
<tr>
<td><strong>Rotation</strong></td>
<td>Variable (numeric data type). Defines the angle of rotation (in degrees).</td>
<td>Example: PLC_PRG.iAngle1.</td>
<td>The midpoint of the element rotates at the “Center” point. This rotation point is shown as the symbol. In runtime mode, the alignment of the element remains the same with respect to the coordinate system of the visualization. Increasing the value rotates the element to the right.</td>
</tr>
<tr>
<td><strong>Interior rotation</strong></td>
<td>Variable (numeric data type). Defines the angle of rotation (in degrees).</td>
<td>Example: PLC_PRG.iAngle2.</td>
<td>In runtime mode, the element rotates about the point of rotation specified in “Center” according to the value of the variable. In addition, the alignment of the element rotates according to the coordinate system of the visualization. Increasing the value in the code rotates clockwise. The rotation point is shown as the symbol. Note: If a static angle of rotation is specified in the “Position ➔ Angle” property, then the static angle of rotation is added to the variable angle of rotation (offset) when the visualization is executed.</td>
</tr>
</tbody>
</table>

You can link the variables to a unit conversion.

The “X”, “Y”, “Rotation”, and “Interior rotation” properties are supported by the “Client Animation” functionality.

See also
- Unit conversion
The variables control the element behavior dynamically.

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
</table>
| "Invisible"      | Variable (BOOL). Toggles the visibility of the element.  
|                  | TRUE: The element is not visible at runtime.         |

The "Invisible" property is supported by the "Client Animation" functionality.

See also
- 

These properties are available only when you select the "Preview: Support client animations and overlay of native elements" option in the Visualization Manager.

You can animate a movement, a rotation, and the visibility of a visualization element.

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
</table>
| "Animation duration" | Time that the element executes an animation (in milliseconds)  
|                  | Example: 500                                      |
| "Move to the foreground" | Property value (BOOL)  
|                  | TRUE: At runtime, the element is displayed in the foreground.  
|                  | FALSE: At runtime, the element is displayed in the layer where it was inserted in the visualization editor.         |

See also
- Chapter 1.3.5.21.5 “Using client animation” on page 1962

Element property 'Access rights'

Requirement: User management is set up for the visualization.

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
</table>
| "Access rights" | Opens the "Access rights" dialog. There you can edit the access privileges for the element.  
|                  | Status messages:  
|                  | • "Not set. Full rights.": Access rights for all user groups: "operable"  
|                  | • "Rights are set: Limited rights": Access is restricted for at least one group.         |

See also
- Chapter 1.3.5.19.3.1 “Dialog ‘Access Rights’” on page 1583

Visualization element 'Time range picker'

Symbol:

Tag: “Date/time controls”
The element provides configurable buttons for setting the time range of a trend display to a defined time. In the process the end time of the previous display is left unchanged and the start time is adapted.

**Element properties**

<table>
<thead>
<tr>
<th><strong>“Element name”</strong></th>
<th>Optional</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hint: Assign individual names for elements so that they are found faster in the element list.</td>
<td></td>
</tr>
<tr>
<td>Example: TimeRangeTemperature</td>
<td></td>
</tr>
</tbody>
</table>

| **“Type of element”** | “Time range picker” |

**Element property 'Position'**

The position defines the location and size of the element in the visualization window. These are based on the Cartesian coordinate system. The origin is located at the upper left corner of the window. The positive horizontal x-axis runs to the right. The positive vertical y-axis runs downwards.

<table>
<thead>
<tr>
<th><strong>“X”</strong></th>
<th>X coordinate of the upper left corner of the element</th>
</tr>
</thead>
<tbody>
<tr>
<td>Specified in pixels.</td>
<td></td>
</tr>
<tr>
<td>Example: 10</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>“Y”</strong></th>
<th>Y coordinate of the upper left corner of the element</th>
</tr>
</thead>
<tbody>
<tr>
<td>Specified in pixels.</td>
<td></td>
</tr>
<tr>
<td>Example: 10</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>“Width”</strong></th>
<th>Specified in pixels.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example: 150</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>“Height”</strong></th>
<th>Specified in pixels.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example: 30</td>
<td></td>
</tr>
</tbody>
</table>

You can also change the values by dragging the box symbols (_drag_ ) to other positions in the editor.

See also
- Chapter 1.3.5.3.2 “Positioning elements, adapting size and layer” on page 1113

<table>
<thead>
<tr>
<th><strong>“Orientation”</strong></th>
<th>Specifies whether the time picker element is aligned horizontally or vertically in the editor.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hint: Change the width to height ratio of the element in the editor.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>“Show frame”</strong></th>
<th>The visualization element is drawn with a frame.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th><strong>“Resolution”</strong></th>
<th>Resolution saved for the time stamp: “Millisecond” or “Microsecond”</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th><strong>“Attached element instance”</strong></th>
<th>Assignment to the element that processes the time picker</th>
</tr>
</thead>
<tbody>
<tr>
<td>The element can be assigned for example to a “Trend” visualization element. Then the time range of the trend element can be changed. The available visual elements are selected with the help of the input assistance (            ).</td>
<td></td>
</tr>
<tr>
<td>Example: GenElemInst_1</td>
<td></td>
</tr>
</tbody>
</table>
**Element property 'Texts'**

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Text&quot;</td>
<td>String label for the element. Example: Zoom</td>
</tr>
</tbody>
</table>

**Element property 'Text properties'**

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Font&quot;</td>
<td>Example: &quot;Default&quot;</td>
</tr>
<tr>
<td></td>
<td>☑️: The &quot;Font&quot; dialog box opens.</td>
</tr>
<tr>
<td></td>
<td>▼: Drop-down list with style fonts.</td>
</tr>
<tr>
<td>&quot;Font color&quot;</td>
<td>Example: &quot;Black&quot;</td>
</tr>
<tr>
<td></td>
<td>☑️: The &quot;Color&quot; dialog box opens.</td>
</tr>
<tr>
<td></td>
<td>▼: Drop-down list with style colors.</td>
</tr>
<tr>
<td>&quot;Transparency&quot;</td>
<td>Whole number (value range from 0 to 255). This determines the transparency of the respective color. Example: 255: The color is opaque. 0: The color is completely transparent.</td>
</tr>
<tr>
<td></td>
<td>Please note: If the color is a style color and already has a transparency value, then this property is write-protected.</td>
</tr>
</tbody>
</table>

**Property 'Times'** In "Times", the buttons that the element provides at runtime are defined and configured in an array.

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Provide &quot;All&quot; selection&quot;</td>
<td>☑️: Time selector bar extended by &quot;All&quot; button. The diagram represents a time interval that covers all time stamps.</td>
</tr>
<tr>
<td>&quot;Times&quot;</td>
<td>☑️: Adds another button to the time selection bar and increases the array by one entry. An additional index is present in the property &quot;Times ➔ Times ➔ Times ➔ [&lt;new&gt;]&quot;, &quot;Time&quot; is located under this index. The configuration of the button is to be entered there.</td>
</tr>
<tr>
<td></td>
<td>☑️: Array of all buttons in the time selection bar. Index corresponds to the number of buttons.</td>
</tr>
<tr>
<td></td>
<td>☑️: The associated button is removed from the time picker bar. The configuration entry is deleted from the &quot;Times&quot; property list.</td>
</tr>
<tr>
<td>&quot;[Index]&quot;</td>
<td>☑️: Time interval in standardized notation. Example: 3M for 3 months; 30m for 30 minutes. If a time interval is indicated in the field, then the button is labelled with it. If a user clicks on the button at runtime, the command is executed to switch the diagram to this time interval. The default is empty.</td>
</tr>
</tbody>
</table>

**Element property 'Control variables'**
<table>
<thead>
<tr>
<th><strong>Property</strong></th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>“Time”</strong></td>
<td>Displays which time is currently selected.</td>
</tr>
<tr>
<td>Variable</td>
<td>(STRING)</td>
</tr>
<tr>
<td>Example:</td>
<td>PLC_PRG.strSelectedTime</td>
</tr>
<tr>
<td><strong>“All” selected</strong></td>
<td>Displays the state of the &quot;All&quot; button</td>
</tr>
<tr>
<td>Variable</td>
<td>(BOOL)</td>
</tr>
<tr>
<td>Example:</td>
<td>PLC_PRG.AllTimesAreSelected</td>
</tr>
</tbody>
</table>

The properties contain IEC variables for controlling the position of the element dynamically. The reference point is the upper left corner of the element. In runtime mode, the entire element is moved.

### Movement

<table>
<thead>
<tr>
<th><strong>Variable</strong></th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>“X”</strong></td>
<td>Variable (numeric data type). Defines the X position (in pixels).</td>
</tr>
<tr>
<td>Example:</td>
<td>PLC_PRG.iPos_X</td>
</tr>
<tr>
<td>Increasing this value in runtime mode moves the element to the right.</td>
<td></td>
</tr>
<tr>
<td><strong>“Y”</strong></td>
<td>Variable (numeric data type). Defines the Y position (in pixels).</td>
</tr>
<tr>
<td>Example:</td>
<td>PLC_PRG.iPos_Y</td>
</tr>
<tr>
<td>Increasing this value in runtime mode moves the element downwards.</td>
<td></td>
</tr>
</tbody>
</table>

| **Rotation** | Variable (numeric data type). Defines the angle of rotation (in degrees). |
| Example: | PLC_PRG.iAngle1 |
| The midpoint of the element rotates at the “Center” point. This rotation point is shown as the symbol. |
| In runtime mode, the alignment of the element remains the same with respect to the coordinate system of the visualization. Increasing the value rotates the element to the right. |

| **“Interior rotation”** | Variable (numeric data type). Defines the angle of rotation (in degrees). |
| Example: | PLC_PRG.iAngle2 |
| In runtime mode, the element rotates about the point of rotation specified in “Center” according to the value of the variable. In addition, the alignment of the element rotates according to the coordinate system of the visualization. Increasing the value in the code rotates clockwise. |
| The rotation point is shown as the symbol. |
| Note: If a static angle of rotation is specified in the “Position ➔ Angle” property, then the static angle of rotation is added to the variable angle of rotation (offset) when the visualization is executed. |
You can link the variables to a unit conversion.

The “X”, “Y”, “Rotation”, and “Interior rotation” properties are supported by the "Client Animation" functionality.

See also
- Unit conversion

Element property 'State variables'

Invisible
Variable (BOOL). Toggles the visibility of the element.
TRUE: The element is not visible at runtime.

The “Invisible” property is supported by the "Client Animation" functionality.

See also
-

These properties are available only when you select the “Preview: Support client animations and overlay of native elements” option in the Visualization Manager.

You can animate a movement, a rotation, and the visibility of a visualization element.

Animation duration
Time that the element executes an animation (in milliseconds)
Example: 500

Move to the foreground
Property value (BOOL)
TRUE: At runtime, the element is displayed in the foreground.
FALSE: At runtime, the element is displayed in the layer where it was inserted in the visualization editor.

See also
- Chapter 1.3.5.21.5 “Using client animation” on page 1962

Element property 'Access rights'

Requirement: User management is set up for the visualization.
"Access rights" Opens the "Access rights" dialog. There you can edit the access privileges for the element. Status messages:
● "Not set. Full rights." : Access rights for all user groups : “operable”
● "Rights are set: Limited rights": Access is restricted for at least one group.

See also
●  "Chapter 1.3.5.19.3.1 "Dialog 'Access Rights” on page 1583"

Visualization element 'Date picker'

Symbol:

Tag: “Date/time controls”
The element is a calendar that displays the current date. A user can click a tag to select a date, which is saved to a variable. In addition, it can customize the time interval that the calendar displays. Clicking the calendar header changes the year. Clicking the arrows in the calendar header changes the month.

Language-dependent texts of the element

The element contains language-dependent texts that are managed in the System text list. This deals with the names of the month and the days of the week written out completely or abbreviated. When the date picker is added to a visualization, CODESYS generates the text list automatically below the POU view. The IDs correspond to the default text and therefore English terms. The text list makes it possible to translate these texts.

<table>
<thead>
<tr>
<th>ID</th>
<th>Standard</th>
</tr>
</thead>
<tbody>
<tr>
<td>Apr</td>
<td>Apr</td>
</tr>
<tr>
<td>April</td>
<td>April</td>
</tr>
</tbody>
</table>

See also
●  "Chapter 1.3.5.6 “Setting up multiple languages” on page 1143"

Element properties

<table>
<thead>
<tr>
<th>&quot;Element name&quot;</th>
<th>Example: DueDateCalendar</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Type of element&quot;</td>
<td>&quot;Date picker&quot;</td>
</tr>
</tbody>
</table>

Element property 'Position'
The position defines the location and size of the element in the visualization window. These are based on the Cartesian coordinate system. The origin is located at the upper left corner of the window. The positive horizontal x-axis runs to the right. The positive vertical y-axis runs downwards.
“X”  
X coordinate of the upper left corner of the element  
Specified in pixels.  
Example: 10.

“Y”  
Y coordinate of the upper left corner of the element  
Specified in pixels.  
Example: 10.

“Width”  
Specified in pixels.  
Example: 150

“Height”  
Specified in pixels.  
Example: 30

You can also change the values by dragging the box symbols (منذ) to other positions in the editor.

See also
● Chapter 1.3.5.3.2 “Positioning elements, adapting size and layer” on page 1113

Element property ‘Center’  
The properties contain fixed values for the coordinates of the point of rotation. This point of rotation is shown as the ☑ symbol. The point is used as the center for rotating and scaling.

“X”  
X-coordinate of the point of rotation

“Y”  
Y-coordinate of the point of rotation

You can also change the values by dragging the symbols ( منذ) to other positions in the editor.

“Variable”  
Input variable (DATE). Contains the date that a user selects in the calendar.  
Example: PLC_PRG.dtDueDate

“Design”  
● “From style”: All settings are preconfigured according to the style.  
● “Explicit”: The “Design settings” property appears. Here you can customize the calendar.

Design settings  
Requirement: This property is visible only if the “Design” property is set to “Explicit”.  
The values of the property can be predefined in the style. Then they are available in the drop-down list.

Table 356: “Header of date picker”

<table>
<thead>
<tr>
<th>Design of the header</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>“Font”</td>
<td>Style font or user-defined font</td>
</tr>
<tr>
<td>“Font color”</td>
<td>Style color or user-defined color</td>
</tr>
<tr>
<td>Arrows</td>
<td></td>
</tr>
<tr>
<td>---------------</td>
<td>---</td>
</tr>
<tr>
<td>Arrow color</td>
<td>Style color or user-defined color</td>
</tr>
<tr>
<td>Color of printed arrow</td>
<td></td>
</tr>
<tr>
<td>Background</td>
<td></td>
</tr>
<tr>
<td>Draw background</td>
<td>“From style”: The style defines whether and how a background is drawn. “Yes”: The background is filled with the color in the “Background color” property. “No”: The background is not filled with a color.</td>
</tr>
<tr>
<td>Fill color</td>
<td>Style color or user-defined color</td>
</tr>
</tbody>
</table>

Table 357: Design of the main display area

<table>
<thead>
<tr>
<th>Design of the main display area</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>“Today”</td>
<td>Design of today</td>
</tr>
<tr>
<td>“Font”</td>
<td>Style font or user-defined font</td>
</tr>
<tr>
<td>“Font color”</td>
<td>Style color or user-defined color</td>
</tr>
<tr>
<td>Draw background</td>
<td>“From style”: The style defines whether and which background is drawn. “Yes”: The background is filled with the color in the “Background color” property. “No”: The background is not filled with a color.</td>
</tr>
<tr>
<td>Background color</td>
<td>Style color or user-defined color. Used if “Yes” is selected in “Draw background”.</td>
</tr>
<tr>
<td>Show frame</td>
<td>“From style”: The style defines whether and how a frame is drawn. “Yes”: The frame is displayed with the following properties. “No”: A frame is not displayed.</td>
</tr>
<tr>
<td>Frame color</td>
<td>Used if “Yes” is selected in “Show frame”.</td>
</tr>
<tr>
<td>Rectangle type</td>
<td></td>
</tr>
<tr>
<td>Line width</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Selected day</th>
<th>Design of the selected day</th>
</tr>
</thead>
<tbody>
<tr>
<td>Font</td>
<td>Style font or user-defined font</td>
</tr>
<tr>
<td>Font color</td>
<td>Style color or user-defined color</td>
</tr>
<tr>
<td>Draw background</td>
<td>“From style”: The style defines whether and how a background is drawn. “Yes”: The background is filled with the color in the “Background color” property. “No”: The background is not filled with a color.</td>
</tr>
<tr>
<td>Background color</td>
<td>Style color or user-defined color</td>
</tr>
<tr>
<td>Show frame</td>
<td>“From style”: The style defines whether and how a background is drawn. “Yes”: The frame is displayed with the following properties. “No”: A frame is not displayed.</td>
</tr>
<tr>
<td>Frame color</td>
<td>Used if “Yes” is selected in “Show frame”.</td>
</tr>
<tr>
<td>Rectangle type</td>
<td></td>
</tr>
<tr>
<td>Line width</td>
<td></td>
</tr>
<tr>
<td><strong>Current month</strong></td>
<td>Design of the current month</td>
</tr>
<tr>
<td>------------------</td>
<td>-----------------------------</td>
</tr>
<tr>
<td><strong>Font</strong></td>
<td>Style font or user-defined font</td>
</tr>
<tr>
<td><strong>Font color</strong></td>
<td>Style color or user-defined color</td>
</tr>
</tbody>
</table>
| **Draw background** | “From style”: The style defines whether and how a background is drawn.  
  “Yes”: The background is filled with the color in the “Background color” property.  
  “No”: The background is not filled with a color. |
| **Background color** |
| **Show frame**   | “From style”: The style defines whether and how a frame is drawn.  
  “Yes”: The frame is displayed with the following properties.  
  “No”: A frame is not displayed. |
| **Frame color**  | Used if “Yes” is selected in “Show frame”. |
| **Rectangle type** |
| **Line width**   |

<table>
<thead>
<tr>
<th><strong>Other months</strong></th>
<th>Design of the previous and subsequent months</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Font</strong></td>
<td>Style font or user-defined font</td>
</tr>
<tr>
<td><strong>Font color</strong></td>
<td>Style color or user-defined color</td>
</tr>
<tr>
<td><strong>Display other month</strong></td>
<td>Design of the previous and subsequent months</td>
</tr>
</tbody>
</table>
| **Draw background** | “From style”: The style defines whether and how a background is drawn.  
  “Yes”: The background is filled with the color in the “Background color” property.  
  “No”: The background is not filled with a color. |
| **Background color** |
| **Show frame**   | “From style”: The style defines whether and how a frame is drawn.  
  “Yes”: The frame is displayed with the following properties.  
  “No”: A frame is not displayed. |
| **Frame color**  | Used if “Yes” is selected in “Show frame”. |
| **Rectangle type** |
| **Line width**   |

<table>
<thead>
<tr>
<th><strong>Day of week heading</strong></th>
<th>Design of the heading with the days of the week</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Font</strong></td>
<td>Style font or user-defined font</td>
</tr>
<tr>
<td><strong>Font color</strong></td>
<td>Style color or user-defined color</td>
</tr>
</tbody>
</table>
| **Draw background**     | “From style”: The style defines whether and how a background is drawn.  
  “Yes”: The background is filled with the color in the “Background color” property.  
  “No”: The background is not filled with a color. |
| **Background color**    |
| **Show frame**          | “From style”: The style defines whether and how a frame is drawn.  
  “Yes”: The frame is displayed with the following properties.  
  “No”: A frame is not displayed. |
<p>| <strong>Frame color</strong>         | Used if “Yes” is selected in “Show frame”. |</p>
<table>
<thead>
<tr>
<th><strong>Property</strong></th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Rectangle type</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Line width</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Display separator line</strong></td>
<td><em>From style</em>: The style defines whether and how a separator line is drawn.</td>
</tr>
<tr>
<td></td>
<td><em>Yes</em>: Display with the following properties.</td>
</tr>
<tr>
<td></td>
<td><em>No</em>: A separator line is not displayed.</td>
</tr>
</tbody>
</table>
| **Color of the separator line** | Used if “Yes” is selected in “Display separator line”.
| **Width of separator line** |                                                                              |
| **Background**             | Design of the calendar days                                                  |
| **Draw background**        | *From style*: The style defines whether and how a background is drawn.       |
|                            | *Yes*: The background is filled with the color in the “Fill color” property and framed in the “Frame color”. |
|                            | *No*: The background is not filled with a color.                             |
| **Fill color**             | Style color or user-defined color                                           |
| **Frame color**            |                                                                              |

**Element property 'Display type'**

<table>
<thead>
<tr>
<th><strong>Rows</strong></th>
<th>Number of month calendars per row (preset: 1)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Columns</strong></td>
<td>Number of month calendars per column (preset: 1)</td>
</tr>
</tbody>
</table>

**Element property 'Absolute movement'**

The properties contain IEC variables for controlling the position of the element dynamically. The reference point is the upper left corner of the element. In runtime mode, the entire element is moved.

<table>
<thead>
<tr>
<th><strong>Movement</strong></th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>X</strong></td>
<td>Variable (numeric data type). Defines the X position (in pixels).</td>
</tr>
<tr>
<td>Example: PLC_PRG.iPos_X.</td>
<td></td>
</tr>
<tr>
<td>Increasing this value in runtime mode moves the element to the right.</td>
<td></td>
</tr>
<tr>
<td><strong>Y</strong></td>
<td>Variable (numeric data type). Defines the Y position (in pixels).</td>
</tr>
<tr>
<td>Example: PLC_PRG.iPos_Y.</td>
<td></td>
</tr>
<tr>
<td>Increasing this value in runtime mode moves the element downwards.</td>
<td></td>
</tr>
</tbody>
</table>
**“Rotation”**
Variable (numeric data type). Defines the angle of rotation (in degrees).

**Example:** PLC_PRG.iAngle1.

The midpoint of the element rotates at the “Center” point. This rotation point is shown as the symbol.

In runtime mode, the alignment of the element remains the same with respect to the coordinate system of the visualization. Increasing the value rotates the element to the right.

---

**“Interior rotation”**
Variable (numeric data type). Defines the angle of rotation (in degrees).

**Example:** PLC_PRG.iAngle2.

In runtime mode, the element rotates about the point of rotation specified in “Center” according to the value of the variable. In addition, the alignment of the element rotates according to the coordinate system of the visualization. Increasing the value in the code rotates clockwise.

The rotation point is shown as the symbol.

Note: If a static angle of rotation is specified in the “Position Angle” property, then the static angle of rotation is added to the variable angle of rotation (offset) when the visualization is executed.

---

You can link the variables to a unit conversion.

---

The “X”, “Y”, “Rotation”, and “Interior rotation” properties are supported by the "Client Animation" functionality.

---

See also
- **Unit conversion**
- **

Element property ‘State variables’**
The variables control the element behavior dynamically.

<table>
<thead>
<tr>
<th><strong>“Invisible”</strong></th>
<th>Variable (BOOL). Toggles the visibility of the element.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>TRUE</strong></td>
<td>The element is not visible at runtime.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>“Deactivate inputs”</strong></th>
<th>Variable (BOOL). Toggles the operability of the element.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>TRUE</strong></td>
<td>User inputs do not have any effect in runtime more. The element is shown as deactivated.</td>
</tr>
</tbody>
</table>
The “Invisible” property is supported by the “Client Animation” functionality.

See also

These properties are available only when you select the “Preview: Support client animations and overlay of native elements” option in the Visualization Manager.

You can animate a movement, a rotation, and the visibility of a visualization element.

<table>
<thead>
<tr>
<th>“Animation duration”</th>
<th>Time that the element executes an animation (in milliseconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Example: 500</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>“Move to the foreground”</th>
<th>Property value (BOOL)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>TRUE: At runtime, the element is displayed in the foreground.</td>
</tr>
<tr>
<td></td>
<td>FALSE: At runtime, the element is displayed in the layer where it was inserted in the visualization editor.</td>
</tr>
</tbody>
</table>

See also

- Chapter 1.3.5.21.5 “Using client animation” on page 1962

**Element property 'Access rights'**

Requirement: User management is set up for the visualization.

<table>
<thead>
<tr>
<th>“Access rights”</th>
<th>Opens the “Access rights” dialog. There you can edit the access privileges for the element.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Status messages:</td>
</tr>
<tr>
<td></td>
<td>- “Not set. Full rights.”: Access rights for all user groups: “operable”</td>
</tr>
<tr>
<td></td>
<td>- “Rights are set: Limited rights”: Access is restricted for at least one group.</td>
</tr>
</tbody>
</table>

See also

- Chapter 1.3.5.19.3.1 “Dialog ‘Access Rights’” on page 1583

See also

- 

**Visualization Element ‘Analog clock’**

Symbol:

Tag: “Date/time controls”

The element is a clock that displays the current time of day. The clock can also display any time.
### Element properties

<table>
<thead>
<tr>
<th>“Element name”</th>
<th>Example: GenElemInst_1</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Type of element”</td>
<td>“Analog clock”</td>
</tr>
</tbody>
</table>

#### Element property 'Position'

The position defines the location and size of the element in the visualization window. These are based on the Cartesian coordinate system. The origin is located at the upper left corner of the window. The positive horizontal x-axis runs to the right. The positive vertical y-axis runs downwards.

<table>
<thead>
<tr>
<th>“X”</th>
<th>X coordinate of the upper left corner of the element</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Specified in pixels.</td>
</tr>
<tr>
<td></td>
<td>Example: 10.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>“Y”</th>
<th>Y coordinate of the upper left corner of the element</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Specified in pixels.</td>
</tr>
<tr>
<td></td>
<td>Example: 10.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>“Width”</th>
<th>Specified in pixels.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Example: 150</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>“Height”</th>
<th>Specified in pixels.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Example: 30</td>
</tr>
</tbody>
</table>

You can also change the values by dragging the box symbols (_drag_ ) to other positions in the editor.

See also
- [Chapter 1.3.5.3.2 “Positioning elements, adapting size and layer” on page 1113](#)

#### Element property 'Center'

The properties contain fixed values for the coordinates of the point of rotation. This point of rotation is shown as the _rotate_ symbol. The point is used as the center for rotating and scaling.

<table>
<thead>
<tr>
<th>“X”</th>
<th>X-coordinate of the point of rotation</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>“Y”</th>
<th>Y-coordinate of the point of rotation</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

You can also change the values by dragging the symbols (_rotate_ ) to other positions in the editor.
### “Use system time”
- The system time of the PLC is displayed.

### “Variable”
- Variable (time data type TOD, TIME_OF_DAY). This receives the time of day that is not the system time.
  - **Example:** PLC_PRG.todTimeTokio
- Requirement: The “Use system time” property is not activated.

**See also**
- *Time/date type TOD*

### “Design”
- “From style”: All settings are preconfigured according to the style.
- “Explicit”: The “Settings” property appears. Here you can customize the analog clock.

---

**Element property 'Absolute movement'**

The properties contain IEC variables for controlling the position of the element dynamically. The reference point is the upper left corner of the element. In runtime mode, the entire element is moved.

### “Movement”

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>“X”</strong></td>
<td>Variable (numeric data type). Defines the X position (in pixels).</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td>PLC_PRG.iPos_X.</td>
</tr>
<tr>
<td></td>
<td>Increasing this value in runtime mode moves the element to the right.</td>
</tr>
<tr>
<td><strong>“Y”</strong></td>
<td>Variable (numeric data type). Defines the Y position (in pixels).</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td>PLC_PRG.iPos_Y.</td>
</tr>
<tr>
<td></td>
<td>Increasing this value in runtime mode moves the element downwards.</td>
</tr>
<tr>
<td><strong>“Rotation”</strong></td>
<td>Variable (numeric data type). Defines the angle of rotation (in degrees).</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td>PLC_PRG.iAngle1.</td>
</tr>
<tr>
<td></td>
<td>The midpoint of the element rotates at the “Center” point. This rotation point is shown as the symbol.</td>
</tr>
<tr>
<td></td>
<td>In runtime mode, the alignment of the element remains the same with respect to the coordinate system of the visualization. Increasing the value rotates the element to the right.</td>
</tr>
<tr>
<td><strong>“Scaling”</strong></td>
<td>Variable (integer data type). Causes centric stretching.</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td>PLC_PRG.iScaling.</td>
</tr>
<tr>
<td></td>
<td>The reference point is the “Center” property.</td>
</tr>
<tr>
<td></td>
<td>The value 1 shrinks the element by a factor of 0.001. The value 1000 returns the element to its original size.</td>
</tr>
</tbody>
</table>
"Interior rotation"  
Variable (numeric data type). Defines the angle of rotation (in degrees).  
**Example:** PLC_PRG.iAngle2.

In runtime mode, the element rotates about the point of rotation specified in "Center" according to the value of the variable. In addition, the alignment of the element rotates according to the coordinate system of the visualization. Increasing the value in the code rotates clockwise.

The rotation point is shown as the symbol.

Note: If a static angle of rotation is specified in the property "Position → Angle", then the static angle of rotation is added to the variable angle of rotation (offset) when the visualization is executed.

"Use REAL values"  
Note: Only available if the device supports the use of REAL coordinates.

- The properties of the absolute movement are interpreted as REAL values. The values are not rounded.
- The option allows for the individual fine-tuning of drawing the element, for example for the visualization of a smoother rotation.
- **Hint:** If a horizontal or vertical line is drawn blurry on a specific visualization platform, then this can be corrected by an offset of 0.5px in the direction of the line thickness.

You can link the variables to a unit conversion.

The properties “X”, “Y”, “Rotation”, and “Interior rotation” are supported by the "Client Animation" functionality.

See also
- **Unit conversion**
- 

**Element property 'Settings'**

Requirement: The “Property” is “Explicit”. Only then is the “Clock Settings” category visible.

**Table 358: "Background"**

<table>
<thead>
<tr>
<th>&quot;Background color&quot;</th>
<th>Color variants of the default background image</th>
</tr>
</thead>
<tbody>
<tr>
<td>• &quot;Yellow&quot;</td>
<td>• &quot;Red&quot;</td>
</tr>
<tr>
<td>• &quot;Blue&quot;</td>
<td>• &quot;Green&quot;</td>
</tr>
<tr>
<td>• &quot;Black&quot;</td>
<td></td>
</tr>
</tbody>
</table>

| "Own background" | Background display with the specific "Image". Replaces the default background image. |
“Image” | Image from an image pool or library  
Example: myImagepool.myImage

“Transparency color” | The transparent color in the image representation.  
Example: “White”. The white parts of the image are transparent.

“Use background color” | The image background is displayed using the color defined in the “Background color” property.  
Requirement: No image reference is given in the “Image” property.

“Background color” | Style color or color  
Requirement: “Use background color” is activated.

Table 359: “Hands”

| “Hand style” | Example: “Thin arrow” |
| “Color hour hand” | Style color or color for the hands |
| “Color minute hand” |
| “Color second hand” |

Table 360: “Lines”

| “Lines style” | Clock face graduation  
- “None”  
- “Line”: Graduation lines by hour  
- “Hours and minutes”: Graduation lines by hours and minutes  
- “Dots”: Graduation dots by hour |
| “Color” | Color of the clock face graduation |
| “Line width” | Line weight of the clock face graduation |
| “Scale in 3D” | ✅ Representation of the clock face with 3D effect |

Table 361: “Numerics”

| “Style of numerics” | Digits on the clock face  
- “None”  
- “Quarter”  
- “All” |
| “Font” | Font for displaying the digits |
| “Font color” | Font for displaying the digits |

Table 362: “Center point”

| “Color” | Color of the center of the clock |

Table 363: “Positioning”

| “Usage of” |  
- “Default style values”: Presetting of the style values  
| “Positioning” | Requirement: Visible when the “Usage or” property is set to “Individual settings”.

**“Numerics movement”**
Value (in pixels) for shifting the digits.
Example: 80

**“Line movement”**
Value (in pixels) for shifting the hour lines.
Example: 100

**“Hands scaling”**
Factor for scaling the length of the hour hand. You can customize the exact position of the hour hand relative to the background image.
Example: 100

**“Scaling type”**
Defines the scaling of the height and width of the element.
- **“Anisotropic”**: The background image is scaled to the size of the element. The height and width are scaled independently of each other.
- **“Isotropic”**: The background image is scaled to the size of the element, retaining its proportion. The proportion of height and width is fixed.

**“Optimized drawing”**
- : The background image is drawn one time. When the hour hand moves, only the affected part of the image is redrawn.
- : The background image is redrawn in cycles.

**Hint:** Deactivate this option only for extreme exceptions.

**Element property 'Absolute movement'**
The properties contain IEC variables for controlling the position of the element dynamically. The reference point is the upper left corner of the element. In runtime mode, the entire element is moved.

**“Movement”**

**“X”**
Variable (numeric data type). Defines the X position (in pixels).
Example: PLC_PRG.iPos_X.
Increasing this value in runtime mode moves the element to the right.

**“Y”**
Variable (numeric data type). Defines the Y position (in pixels).
Example: PLC_PRG.iPos_Y.
Increasing this value in runtime mode moves the element downwards.

**“Rotation”**
Variable (numeric data type). Defines the angle of rotation (in degrees).
Example: PLC_PRG.iAngle1.
The midpoint of the element rotates at the “Center” point. This rotation point is shown as the symbol.
In runtime mode, the alignment of the element remains the same with respect to the coordinate system of the visualization. Increasing the value rotates the element to the right.
**“Interior rotation”**

Variable (numeric data type). Defines the angle of rotation (in degrees).

**Example:** PLC_PRG.iAngle2.

In runtime mode, the element rotates about the point of rotation specified in “Center” according to the value of the variable. In addition, the alignment of the element rotates according to the coordinate system of the visualization. Increasing the value in the code rotates clockwise.

The rotation point is shown as the symbol.

Note: If a static angle of rotation is specified in the “Position ➔ Angle” property, then the static angle of rotation is added to the variable angle of rotation (offset) when the visualization is executed.

---

**“Invisible”**

Variable (BOOL). Toggles the visibility of the element.

**TRUE:** The element is not visible at runtime.

**See also**
- Unit conversion
- 

**Element property ‘State variables’**

The variables control the element behavior dynamically.

**The “Invisible” property is supported by the "Client Animation" functionality.**

---

**See also**
- 

These properties are available only when you select the “Preview: Support client animations and overlay of native elements” option in the Visualization Manager.

You can animate a movement, a rotation, and the visibility of a visualization element.
“Animation duration”

Time that the element executes an animation (in milliseconds)

Example: 500

“Move to the foreground”

Property value (BOOL)

TRUE: At runtime, the element is displayed in the foreground.

FALSE: At runtime, the element is displayed in the layer where it was inserted in the visualization editor.

See also

● "Chapter 1.3.5.21.5 “Using client animation” on page 1962"

See also

● "Visualization element “Date and time picker”"

Symbol:

Tag: “Date/time controls”

The element provides the capability of selecting the date or time. The value can be changed by means of the keyboard or arrow keys. The date can be selected from a calendar.

Element properties

“Element name”

Optional

Hint: Assign individual names for elements so that they are found faster in the element list.

Example: StartDateAndTime

“Type of element”

“Date and time picker”

The position defines the location and size of the element in the visualization window. These are based on the Cartesian coordinate system. The origin is located at the upper left corner of the window. The positive horizontal x-axis runs to the right. The positive vertical y-axis runs downwards.

“X”

X coordinate of the upper left corner of the element

Specified in pixels.

Example: 10.

“Y”

Y coordinate of the upper left corner of the element

Specified in pixels.

Example: 10.
**“Width”**
Specified in pixels.
Example: 150

<table>
<thead>
<tr>
<th><strong>“Height”</strong></th>
<th>Specified in pixels.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Example: 30</td>
</tr>
</tbody>
</table>

You can also change the values by dragging the box symbols (estruction) to other positions in the editor.

See also
- § Chapter 1.3.5.3.2 “Positioning elements, adapting size and layer” on page 1113

**Element property 'Center’**
The properties contain fixed values for the coordinates of the point of rotation. This point of rotation is shown as the symbol. The point is used as the center for rotating and scaling.

<table>
<thead>
<tr>
<th><strong>“X”</strong></th>
<th>X-coordinate of the point of rotation</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>“Y”</strong></td>
<td>Y-coordinate of the point of rotation</td>
</tr>
</tbody>
</table>

You can also change the values by dragging the symbols (estruction) to other positions in the editor.

**“Variable”**
Variable (DATE, DT, TIME, LTIME, TOD)
The value of the value of the variable is shown by the element and can be modified by the element.
The data type automatically determines the displayed value units:
- **TIME**: Day, hour, minute, second (by default, milliseconds are not shown)
- **DATE**: Year, month, day
- **DT**: Year, month, day, hour, minute, second
- **TOD**: Hour, minute, second (by default, milliseconds are not shown)
- **LTIME**: Day, hour, minute, second (by default, milliseconds, microseconds, and nanoseconds are not shown)

**“Format string”**
The format can restrict the output to individual values.
Example for LTIME: Format: HH:mm:ss.ms.us.ns --> displayed: 08:15:12.780.150.360
LTIME limited: format: HH:mm --displayed: 08:15
Basically, all usual formats available for %t are also supported.

**“Design date time picker”**
- **“From style”**: All settings are preconfigured according to the style.
- **“Explicit”**: The property “Designsettings” appears. Here you can customize the calendar.
"Design date picker"
- "From style": All settings are preconfigured according to the style.
- "Explicit": The property "Design settings" appears. Here you can customize the calendar.

"Positioning date picker"
- "Dynamic": The calendar is adapted and positioned automatically.
- "Manual": The "Position settings" property appears. Here you can customize the calendar.

See also
- Chapter 1.3.5.18.2 “Placeholders with format definition in the output text” on page 1546

Element property 'Absolute movement'
The properties contain IEC variables for controlling the position of the element dynamically. The reference point is the upper left corner of the element. In runtime mode, the entire element is moved.

"Movement"

<table>
<thead>
<tr>
<th>&quot;X&quot;</th>
<th>Variable (numeric data type). Defines the X position (in pixels).</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Example: PLC_PRG.iPos_X.</td>
</tr>
<tr>
<td></td>
<td>Increasing this value in runtime mode moves the element to the right.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>&quot;Y&quot;</th>
<th>Variable (numeric data type). Defines the Y position (in pixels).</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Example: PLC_PRG.iPos_Y.</td>
</tr>
<tr>
<td></td>
<td>Increasing this value in runtime mode moves the element downwards.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>&quot;Rotation&quot;</th>
<th>Variable (numeric data type). Defines the angle of rotation (in degrees).</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Example: PLC_PRG.iAngle1.</td>
</tr>
<tr>
<td></td>
<td>The midpoint of the element rotates at the &quot;Center&quot; point. This rotation point is shown as the + symbol.</td>
</tr>
<tr>
<td></td>
<td>In runtime mode, the alignment of the element remains the same with respect to the coordinate system of the visualization. Increasing the value rotates the element to the right.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>&quot;Scaling&quot;</th>
<th>Variable (integer data type). Causes centric stretching.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Example: PLC_PRG.iScaling.</td>
</tr>
<tr>
<td></td>
<td>The reference point is the &quot;Center&quot; property.</td>
</tr>
<tr>
<td></td>
<td>The value 1 shrinks the element by a factor of 0.001. The value 1000 returns the element to its original size.</td>
</tr>
</tbody>
</table>
The “Interior rotation” variable (numeric data type) defines the angle of rotation (in degrees).

**Example:** PLC_PRG.iAngle2.

In runtime mode, the element rotates about the point of rotation specified in “Center” according to the value of the variable. In addition, the alignment of the element rotates according to the coordinate system of the visualization. Increasing the value in the code rotates clockwise.

The rotation point is shown as the symbol.

Note: If a static angle of rotation is specified in the property “Position ➔ Angle”, then the static angle of rotation is added to the variable angle of rotation (offset) when the visualization is executed.

---

**“Invisible”**

Variable (BOOL). Toggles the visibility of the element.

**TRUE:** The element is not visible at runtime.

**“Deactivate inputs”**

Variable (BOOL). Toggles the operability of the element.

**TRUE:** User inputs do not have any effect in runtime more. The element is shown as deactivated.

---

The “Invisible” property is supported by the "Client Animation" functionality.

---

See also

- Unit conversion

---

Element property 'State variables'

The variables control the element behavior dynamically.

---

These properties are available only when you select the “Preview: Support client animations and overlay of native elements” option in the Visualization Manager.

You can animate a movement, a rotation, and the visibility of a visualization element.
<table>
<thead>
<tr>
<th><strong>“Animation duration”</strong></th>
<th>Time that the element executes an animation (in milliseconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Example:</strong> 500</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>“Move to the foreground”</strong></th>
<th>Property value (BOOL)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>TRUE:</strong> At runtime, the element is displayed in the foreground.</td>
<td></td>
</tr>
<tr>
<td><strong>FALSE:</strong> At runtime, the element is displayed in the layer where it was inserted in the visualization editor.</td>
<td></td>
</tr>
</tbody>
</table>

**See also**
- ¶ Chapter 1.3.5.21.5 “Using client animation” on page 1962

**Element property 'Access rights'**

**Requirement:** User management is set up for the visualization.

<table>
<thead>
<tr>
<th><strong>“Access rights”</strong></th>
<th>Opens the “Access rights” dialog. There you can edit the access privileges for the element.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Status messages:</strong></td>
<td></td>
</tr>
<tr>
<td><strong>“Not set. Full rights.”:</strong> Access rights for all user groups: “operable”</td>
<td></td>
</tr>
<tr>
<td><strong>“Rights are set: Limited rights.”:</strong> Access is restricted for at least one group.</td>
<td></td>
</tr>
</tbody>
</table>

**See also**
- ¶ Chapter 1.3.5.19.3.1 “Dialog ‘Access Rights’” on page 1583

**1.3.5.20 Reference, visualization style editor**

<table>
<thead>
<tr>
<th>1.3.5.20.1</th>
<th>Dialog ‘Create a New Visualization Style’</th>
<th>1946</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.3.5.20.2</td>
<td>Dialog ‘Open Existing Style as a Copy’</td>
<td>1947</td>
</tr>
<tr>
<td>1.3.5.20.3</td>
<td>Editor ‘Visualization Style Editor’</td>
<td>1947</td>
</tr>
</tbody>
</table>

**1.3.5.20.1 Dialog ‘Create a New Visualization Style’**

**Symbol:** 🎨

**Function:** The dialog prompts you to specify data for a new created style file.

**Call:**
- In CODESYS:
  - In the “Visualization Manager” object (tab “Settings”, group “Style Settings”): 
    - Clicking 🎨 opens a drop-down list. Click “Create and Edit Derived Style”.
- In the visualization style editor:
  - Menu bar: “File ➤ New Style”

<table>
<thead>
<tr>
<th><strong>“Name”</strong></th>
<th>Name of the new style.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Example:</strong> Style_CI</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>“Storage location”</strong></th>
<th>Working directory for style editing</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th><strong>“Base style”</strong></th>
<th>Style to base the new style on. The drop-down list includes all styles that are installed in the repository.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>“&lt;none&gt;”:</strong> The new style does not derive itself from an existing style.</td>
<td></td>
</tr>
</tbody>
</table>
The profile is intended for informational purposes. For example, you find elements that are not preconfigured with special style entries, and information from the profile. In addition, CODESYS checks in the profile whether a required style is missing.

**Example:** CODESYS V3.5 SP9

Click “OK” The new style is created and opened for editing in the visualization style editor. It already includes all required style entries and the localization in German (language column de).

**See also**
- Chapter 1.3.5.20.3 “Editor ‘Visualization Style Editor’” on page 1947
- Chapter 1.3.5.17 “Applying visualization styles” on page 1216

### 1.3.5.20.2 Dialog ‘Open Existing Style as a Copy’

**Function:** This dialog prompts you to specify data for copying a style file.

**Call:**
- In CODESYS:
  - In the editor of the “Visualization Manager” object (tab “Settings”, group “Style Settings”, click for a drop-down list). Click “Copy and Edit Style”.
- In the visualization style editor:
  - Menu bar: “File ➔ Open as Copy”

<table>
<thead>
<tr>
<th>“Style”</th>
<th>Style to be copied.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Example:</strong> Basic Style, 3.5.9.0</td>
<td></td>
</tr>
<tr>
<td><strong>Note:</strong> You can also select a style from the repository.</td>
<td></td>
</tr>
</tbody>
</table>

| “Destination” | Working directory for style editing |
| “OK” | A copy of the style is created and opened for editing in the visualization style editor. |

**See also**
- Chapter 1.3.5.17 “Applying visualization styles” on page 1216
- Chapter 1.3.5.20.3 “Editor ‘Visualization Style Editor’” on page 1947

### 1.3.5.20.3 Editor ‘Visualization Style Editor’

**Symbol:** 📏

**Function:** The editor is used for creating, deriving, editing, and localizing visualization styles. In addition, it makes it possible to check and install a style or a hierarchy of styles.

**Call:**
- In CODESYS:
  - In the “Visualization Manager” object (tab “Settings”, group “Style Settings”, click for a drop-down list). Click “Open Style Editor”.
- Start menu > CODESYS installation folder > 'CODESYS' > 'Visualization Style Editor'
Menu 'File'

- **“New style”**
The “Create a New Visualization Style” dialog box opens.

- **“Open”**
The “Open Dialog” dialog box opens. This dialog prompts you to select a style file (format .visustyle.xml) to be opened and edited.

- **“Open as copy”**
The “Open Existing Style as Dialog” dialog box opens. This dialog prompts you to select a style that is copied, saved to the target location, and opened for editing.

- **“Close”**
Closes the style open in the editor.

- **“Save”**
Saves the changes of the open style.

- **“Save As”**
The “Select Visualization Style(s)” dialog box opens. This dialog prompts you to select a file to save the current settings.

- **“Save and Install”**
Saves the open visualization style and installs it to the visualization style repository.

- **“Recently opened files”**
Lists the files for selection that were last opened.

- **“Abort”**
Closes the visualization style editor.

See also
- Chapter 1.3.5.20.1 “Dialog ’Create a New Visualization Style’” on page 1946

Menu 'Styles'

- **“New Entry (as Child)”**
Creates another style entry as a child of the selected style property.

- **“New Entry (Afterwards)”**
Creates a new style entry in the list after the selected style property.

- **“Move Down”**
Moves the selected style entry down.
  Requirement: Sort order is flat.

- **“Move Up”**
Moves the selected style entry up.
  Requirement: Sort order is flat.

- **“Sort Order”**
Toggles between three sort orders:
  - Flat structure and alphabetical order
  - Flat structure and order according to the position of the entry in the XML style file
    This position also determines the position of the property in CODESYS. The property appears, for example, in the “Properties” view below the “Values” column in the drop-down list for style properties.
  - Hierarchical structure of entries
    Requirement: The names of the style properties contain at least one dash.

- **“Check”**
The settings of the style properties are checked for consistency errors. This check is also performed when saving the style.

Menu 'Localization'

- **“New Entry (as Child)”**
Creates another style entry as a child of the selected style property.

- **“New Entry (Afterwards)”**
Creates a new style entry in the list after the selected style property.

- **“Move Down”**
Moves the selected style entry down.
  Requirement: Sort order is flat.

- **“Move Up”**
Moves the selected style entry up.
  Requirement: Sort order is flat.

- **“Sort Order”**
Toggles between three sort orders:
  - Flat structure and alphabetical order
  - Flat structure and order according to the position of the entry in the XML style file
    This position also determines the position of the property in CODESYS. The property appears, for example, in the “Properties” view below the “Values” column in the drop-down list for style properties.
  - Hierarchical structure of entries
    Requirement: The names of the style properties contain at least one dash.

- **“Check”**
The settings of the style properties are checked for consistency errors. This check is also performed when saving the style.
## “Add Language”
The dialog box “Add New Language” opens. The dialog prompts you to specify data for creating a new language column.

## “Remove Selected Language”
Removes the columns of the selected cell.

## “Rename Selected Language”
The “Rename Language” dialog box opens. The dialog is used for renaming the column that defines the selected cell and removes all previous translations.

### Table 364: Dialog box “Add New Language”

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Name of the new language as a language code according to ISO 639-1.</td>
</tr>
<tr>
<td>Example:</td>
<td>Example: de, en, es, it, fr, ja</td>
</tr>
<tr>
<td>Copy from existing</td>
<td>All existing language columns are available for selection. The selected language is copied with all entered translations.</td>
</tr>
<tr>
<td>&lt;do not copy text&gt;</td>
<td>The new language receives a blank translation column.</td>
</tr>
</tbody>
</table>

### Tab ‘General’
This tab contains the general metadata of the open style file and allows it to be edited.

### Table 365: “Identification”

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Company</td>
<td>Example: Xy-z GmbH</td>
</tr>
<tr>
<td>Tip</td>
<td>In the installed styles, CODESYS can filter by the company names specified here.</td>
</tr>
<tr>
<td>Name</td>
<td>Example: Style_A</td>
</tr>
<tr>
<td>Version</td>
<td>User-defined version number</td>
</tr>
<tr>
<td>Example:</td>
<td>1.1.1.1</td>
</tr>
</tbody>
</table>

### Table 366: “General Settings”

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Base style</td>
<td>Name and version of the style that the open style is based on.</td>
</tr>
<tr>
<td>Tip</td>
<td>The derived style properties from the base style are highlighted in yellow in the “Style Properties” tab.</td>
</tr>
<tr>
<td>Partial style (usable only as base for other visualization styles)</td>
<td>The style is identified as incomplete. Therefore, it can be used for other styles as a base style only.</td>
</tr>
<tr>
<td>Example:</td>
<td>Style only with color definitions that derive this to many other styles.</td>
</tr>
<tr>
<td>Note: CODESYS does not check for consistency errors of an incomplete style for itself.</td>
<td></td>
</tr>
<tr>
<td>From</td>
<td>The “Select Base Style” dialog box opens. This dialog prompts you to select a style file that is saved to the file system (and not does not have to be installed). The file is used as a base style.</td>
</tr>
</tbody>
</table>

### Table 367: “Informational”

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Visualization profile</td>
<td>The profile is intended for informational purposes. For example, you find elements that are not preconfigured with special style entries, and information from the profile. In addition, CODESYS checks in the profile whether a required style is missing.</td>
</tr>
</tbody>
</table>
Tab 'Style Properties'  
This tab lists the names of the style properties with the associated values and makes it possible to edit it, even by means of the commands in the “Styles” menu.

The style properties can be defined for colors, fonts, images, and any values.

The style properties defined in a base style are derived and highlighted in yellow.

| "Name" | Name of the style property.  
If the name contains a dash, then the Visualization Style Editor can sort the style properties by the prefixed terms before the dash and display them in a hierarchy. A name can contain more than one dash. |
| "Value" | Value that is assigned to the style property. |
| "Type" | Data type of the style property; selected from a drop-down list. Note: This is possible and necessary only for specific style properties with a data type that is not implicitly defined. |
| "Attribute" | "hide": The associated style property is not listed in the drop-down lists in CODESYS. |
| "Used by" | Visualization element that can be configured with this style property. Can be edited. |
| Comment | Example: Special setting for Bar Display. Optional. |
| Double-click a cell. | An input field opens for editing. |
| [Del] | Removes the selected row. |

Tab 'Localization'  
This tab makes it possible to translate the names of the style properties into other languages.

| "Name" | Lists the name of the style properties as they are defined in the “Style Properties” tab. |
| "<language>" | Identification of the language name (as language code according to ISO 639-1) in which the style property name should be translated. |
| Double-click a cell. | An input field opens for editing. |

1.3.5.21 Tutorial

| 1.3.5.21.1 | Visualizing a refrigerator controller.......................................................... 1951 |
| 1.3.5.21.2 | Displaying array data in histograms.............................................................. 1958 |
| 1.3.5.21.3 | Displaying structured variable values in tables.......................................... 1960 |
| 1.3.5.21.4 | Displaying web content.................................................................................. 1961 |
| 1.3.5.21.5 | Using client animation.................................................................................... 1962 |

Here you find instructions specific to different use cases.

This collection of instructions is expanded regularly.
1.3.5.21.1 Visualizing a refrigerator controller

This tutorial demonstrates how to add visualizations to the project and link the elements of the visualization to the variables of the control program.

Preparation

This tutorial is based on the sample program RefrigeratorControl, which was created in the section "Your First Program in CODESYS". The finished project is located in the found in the "Projects" subfolder of the CODESYS installation directory.

See also

- Your First CODESYS Program

Creating visualizations

The visualization consists of the following three visualization screens:

- **Visualization**: Control elements and display of the refrigerator
- **Diagnosis**: History of the set and actual temperature, parameter settings
- **Live Visu**: Animation with refrigerator

1. Select the “Application” object in the device tree.
2. Click “Project ➔ Add Object ➔ Visualization”.
3. Specify Live_Visu as the name.
4. Create two more visualizations with the names Diagnosis and Visualization.

Structure of the Visualization visualization

This screen consists of control and display elements that control the refrigerator.

- (1) Numeric display of the actual temperature
- (2) Pointer to display of the actual temperature
- (3) Numeric display of the set temperature
- (4) Potentiometer for setting the set temperature
- (5) Label for compressor lamp
1. Open the Visualization visualization in the editor.
2. Drag a “Rectangle” visualization element to the editor.
   Change the following properties:
   - “Texts ➔ Text”: Actual temperature: %2.1f °C
   - “Text variables ➔ Text variable”: Glob_Var.rTempActual
3. Drag a “Meter 180°” visualization element to the editor.
   Change the following properties:
   - “Value”: Glob_Var.rTempActual
   - “Scale ➔ Scale end”: 20
   - “Scale ➔ Main scale”: 5
   - “Scale ➔ Subscale”: 1
4. Drag a “Rectangle” visualization element to the editor.
   Change the following properties:
   - “Texts ➔ Text”: Temperature presetting: %.1f °C
   - “Text variables ➔ Text variable”: Glob_Var.rTempSet
5. Drag a “Potentiometer” visualization element to the editor.
   Change the following properties:
   - “Variable”: Glob_Var.rTempSet
   - “Background ➔ Background color”: “yellow”
   - “Pointer ➔ Color”: “red”
   - “Scale ➔ Subscale position”: “Outward”
   - “Scale ➔ Scale start”: 3
   - “Scale ➔ Scale end”: 13
   - “Scale ➔ Subscale”: 1
   - “Scale ➔ Main scale”: 1
   - “Label ➔ Unit”: “°C”
   - “Label ➔ Scale format (C syntax)”: %.0f
   - “Label ➔ Max. text width of labels”: 21
   - “Label ➔ Height of labels”: 15
6. Drag a “Label” visualization element to the editor.
   Change the following properties:
   - “Texts ➔ Text”: Cooling compressor
7. Drag a “Lamp” visualization element to the editor. Position it behind the text Cooling compressor.
   Change the following properties:
   - “Variable”: Glob_Var.bCompressor
8. Drag a “Label” visualization element to the editor.
   Change the following properties:
   - “Texts ➔ Text”: Signal (beep)
9. Drag a “Lamp” visualization element to the editor. Position it behind the text "Signal (beep)".
   Change the following properties:
   ● “Variable”: Glob_Var.bSignal
   ● “Background Image”: Red

10. Drag a “Rectangle” visualization element to the editor.
    Change the following properties:
    ● “Texts Text”: Door open

11. Drag a “Rocker switch” visualization element to the editor.
    Change the following properties:
    ● “Variable”: Glob_Var.rDoorOpen

In this screen, you can monitor the temperature curve and optimize the parameters.

Structure of Diagnosis visualization

- (1) “Label” elements for the heading
- (2) “Trace” element for displaying the temperature curve
- (3) “Rectangle” elements for displaying the values

1. Open the Diagnosis visualization in the editor.
2. Drag a “Label” visualization element to the editor.
   Change the following properties:
   ● “Texts Text”: Refrigerator Diagnosis & Service Menu
   ● “Text properties Font”: Arial, Standard, 18
3. Drag a “Trace” visualization element to the editor.
4. Click the Diagnosis_Trace1 value of the “Trace” property.
   The “Trace Configuration” dialog opens.
5. Select the “MainTask” in “Task”.
6. Click the “Add Variable” link.
   A variable is added to the trace. The variable settings are displayed in the dialog.
7. Select Glob_Var.bCompressor for the variable.
8. Add the Glob_Var.rTempSet and Glob_Var.rTempActual variables to the trace. For the other settings, you can use the default values.
9. Click “OK” to close the dialog.

10. Drag a “Rectangle” visualization element to the editor. Position it on the right next to the trace element.
    Change the following properties:
    ● “Texts ➔ Text”: %s
    ● “Text variable ➔ Text variable”: PLC_PRG.rHysteresis

11. Configure the “OnMouseDown” input configuration of the element. Click “Input configuration ➔ OnMouseDown ➔ Configure”.
    The “Input Configuration” dialog opens.

12. Assign the “Write Variable” command to the action. Accept the default values and click “OK”.

13. Drag a “Label” visualization element to the editor. Position it over the first rectangle.
    Change the following properties:
    ● “Texts ➔ Text”: Hysteresis Regulator

14. Adjust the size and position of both elements.

15. Select both of the “Rectangle” and “Label” elements and duplicate them by means of copy and paste.

16. Adjust the labels and variables of the copied elements.
    ● “Text”: Compressor Efficiency, “Text variable”: Simulation.P_Cooling
    ● “Text”: Time until Beep for DoorOpen, “Text variable”: Glob_Var.timDoorOpenThreshold
    ● “Text”: Time until Beep for Compressor On, “Text variable”: Glob_Var.timAlarmThreshold

See also
● § 1.3.5.19.3.6 “Dialog ‘Input Configuration’” on page 1586

Structure of 'Live Visu' visualization
This screen includes the representation of a refrigerator. The refrigerator consists of several polygon type visualization elements. The doors of the refrigerator are drawn in both the closed and open states. Both doors consist of a group of single elements.
1. Open the Live Visu visualization in the editor.
2. Select the "Polygon" visualization element in the "Toolbox" view.
3. Click several times in the editor to create a surface. Right-click to stop adding corner marks.
4. Move the corner marks to the required position so that the element (1) is formed.
5. Select the element. Change the following properties:
   - "Colors ➔ Use gradient color": ✔
   - "Appearance ➔ Line style": "Invisible"
6. Click in the property "Colors ➔ Use gradient color".
7. Select the color “Gray” for “Color 1” in the “Color gradient editor” dialog.

8. Create all other elements with the “Polygon” visualization element.

9. Group the elements of the closed doors (2+3+4) and the open doors (5+6+7+8). Select the elements by pressing the [Shift] key and clicking “Visualization ➔ Group”.

10. Move the elements together so that the completed refrigerator is formed. Position the open doors precisely on the closed doors.

11. Select the "Open doors" group.

12. In the properties, double-click the input field “State variable ➔ Invisible”.

13. Open the input assistance by pressing [F2].

14. Select the rDoorOpen variable in the “Variables” category (below “Application ➔ Glob_Var”).

15. Negate the variable with NOT (--> NOT Glob_Var.rDoorOpen).

   ⇒ If the rDoorOpen variable is FALSE (door is closed), then the element is invisible. Then the underlying doors are visible.

16. Copy the following elements from the Visualization screen:
   - Potentiometer for setting the temperature
   - Rectangle for displaying the set temperature
   - Door open switch
   - Cooling compressor lamp
   - Signal (beep) lamp

17. Insert the elements from the clipboard to the Live_Visu visualization screen.
18. Reduce the elements and position them on the refrigerator.

When the visualization is complete, test it in simulation mode.

1. Click “Online ➔ Simulation”.
2. Click “Online ➔ Login”.
   ➔ A dialog opens and prompts you to create and download the application.
3. Click “Yes” to confirm the dialog.
4. Click “Debug ➔ Start”.
5. Open the Live_Visu visualization in the editor.
   ➔ The refrigerator in online mode.
6. Open the doors with the switch and monitor the temperature and the alarms. Change the parameters in the Diagnosis screen and watch the reaction in the temperature curve.
1.3.5.21.2 Displaying array data in histograms

Setting element properties for histograms

Requirements

- A project contains a visualization object and a program.
- A one-dimensional array is declared in the program (example: `histogram : ARRAY[1..10] OF INT;`).
- In the program, `histogram` data is assigned to the array (example: within the range from 0 to 50).

1. Double-click the “Visualization” object in the device tree.
2. In the “Toolbox” view, click “Measurement Controls” and drag the “Histogram” element to the visualization editor.
3. In the visualization editor, click the inserted “Histogram”.
   ⇒ The “Properties” view opens.
4. In the “Properties” view, double-click the “Value” input field in the “Data array” element property. Then click .
5. In the “Input Assistant” dialog box in the “Variables” category of the “PLC (PRG)” program, select the array (example here: `histogram : ARRAY[1..10] OF INT;`) and click “OK”.
6. To display only part of the array as a histogram, activate the “Use subrange” option and specify the index values of the array in “Start index” and “End index” to define the subrange.
7. Select the “Display type” (example: “Bar”).
8. Specify a value between 1 and 100 (example: 30) for the “Relative bar width”.
9. Click the histogram in the visualization editor and change the size and position as desired.
   ⇒ The “Position” property changes its values accordingly.
10. Specify the values for the “Scale” element property. Select the values for “Scale start” and “Scale end” so that the array is displayed completely. For the example: “Scale start” 0, “Scale end” 50.
    For the distance between values on the rough scale, specify the value 10, for example, in “Main scale”.
11. In the “Label” element property, specify the “Unit” for the display values.
12. Click “Build ➔ Build”.
13. If the project has been compiled without errors, then click “Online ➤ Login” and start the application by clicking “Debug ➤ Start”.

   The histogram is displayed in the visualization as follows:

   ![Histogram Image]

   The visualization shows a histogram with bars all the same color (example: green). Now you want the bars with values less than 30, for example, to be displayed in another color (example: red).

   1. Click the element property “Colors ➤ Alarm color”.
   2. Specify the limiting value in “Alarm value” above or below which the bars should be displayed in another color.
   3. Select “More” from the drop-down list in “Alarm condition” if all values greater than the “Alarm value” should be displayed in another color. Otherwise, select “Less”.
   4. Select an “Alarm color” (example: “Red”).
   5. Click “Build ➤ Build”.

Defining alarm colors for the histogram
6. If the project has been compiled without errors, then click “Online ➤ Login” and start the application by clicking “Debug ➤ Start”.

⇒ In the example histogram, all bars with values greater than \(30\) are displayed in red.

See also
- § Chapter 1.3.5.18.1.29 “Visualization element 'Histogram'” on page 1443

### 1.3.5.21.3 Displaying structured variable values in tables

Subsequent instructions describe an example of how an array of a structure is displayed in a table. As a preparation, create the MyStruct DUT and the declarations in the PLC_PRG program.

```
TYPE MyStruct :
  STRUCT
    iNo  : INT;
    bOnStock : BOOL;
    strPartNumber : STRING;
  END_STRUCT
END_TYPE

PROGRAM PLC_PRG
VAR
  arrStruct : ARRAY[0..6] OF MyStruct;
  iSelectedColumn : INT;
END_VAR
```

1. Drag the “Table” visualization element to the visualization editor.
2. Assign the array variable `arrStruct` to the “Data array” property.
   ⇒ The structure members are displayed as column headings and the array index as row headings.
3. Change the “Columns ➤ Column ➤ [0] ➤ Column header” property to an informative heading (example: Number).

5. Assign a color to the “Selection ⇒ Selection color” property.

6. Define the “Selection ⇒ Selection type” property as Row selection.

7. In the “Selection ⇒ Variable for selected row” property, define the PLC_PRG.iSelectedColumn variable.

- The following display results in online mode:

<table>
<thead>
<tr>
<th>Number</th>
<th>in Stock</th>
<th>Part Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>FALSE</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>FALSE</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>FALSE</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>FALSE</td>
</tr>
</tbody>
</table>

See also
- Chapter 1.3.5.18.1.13 “Visualization element ‘Table’” on page 1337

1.3.5.21.4 Displaying web content

Displaying websites in a visualization

☐ Requirement: A visualization open in a CODESYS project. The “Toolbox” and “Properties” views of the visualization are also open.

1. Drag the “Web Browser” element from “Special Controls” to the visualization editor.
2. Select the element in the editor.

- In the “Properties” view, the element properties are listed for the “Web Browser” element.

3. In the “Position” property, specify the size (in pixels) for the “Width” and the “Height” (example: 600).

4. In “Control variables ⇒ URL”, specify the URL for the website (example: 'http://de.wikipedia.org'). You can also specify a variable here (STRING or WSTRING) where the URL is assigned in the project.

5. In “Control variables ⇒ Display”, specify a Boolean variable (example: bSetURL).

- If the variable bSetURL has the value TRUE, then the website 'http://de.wikipedia.org' is displayed at runtime.

Configuring the buttons for forward and back navigation of the website

☐ Requirement: The “Web Browser” element of your visualization is configured as described above.

1. In a POU, declare both Boolean variables bGoBack and bGoForward.
2. In the visualization editor, click the “Web Browser”.
3. For the property "Control variables ➔ Back", select the variable bGoBack from the input assistant. For the property "Control variables ➔ Forward", select the variable bGoForward.

4. In “General Controls”, add the “Button” element to your visualization two times.

5. Click a “Button” in the visualization editor and drag the “Button” to the required position (for example above the “Web Browser” element).

6. In the property “Texts ➔ Text”, specify the character >. In “Text properties ➔ Font”, select a font from the input assistant (example: Arial, Fett, 14).

7. Configure the property “Input configuration ➔ OnMouseClick” so that the variable bGoForward switches.

8. Configure the second button for back navigation in the same way as described in steps 5 to 7. If the variable bSetURL has the value TRUE, then the website 'http://de.wikipedia.org' is displayed with the forward and back buttons. When you click the buttons, navigation to the previous and next websites is successful.

See also

● ☞ Chapter 1.3.5.18.1.38 “Visualization element 'Webbrowser'” on page 1483

● ☞ Chapter 1.3.5.19.3.6 “Dialog 'Input Configuration'” on page 1586

1.3.5.21.5 Using client animation

The example shows a visualization with 3 screens. A menu controls the navigation of the screens. The menu is hidden until it moves into view by means of a hamburger button. During the movement, the transparency of the menu is changed. After the screen is selected, the menu moves back out of view. The animation is computed entirely on the target system. The CODESYS visualization only defines the target values (positions, transparency).

1. Preparation

1. Create a new standard project with the CODESYS Control Win V3 controller.

2. Add a “Visualization” object below the “Application”. Choose the name Visu_Main.

3. Remove the “TargetVisu” object below the Visualization Manager.

4. Open the Visualization Manager in the editor and select the “Preview: Support client animations and overlay of native elements” option.

See also

● ☞

2. Creating the PLC_PRG program

The program checks whether the menu button has been pressed. If the menu bar is not visible (position ~300), then the position is moved to the visible area (0). If the menu bar is already visible (position 0), then the position is moved to the hidden area.
1. Open the “PLC_PRG” program in the editor.

2. Input the following code into the declaration editor:

```plaintext
PROGRAM PLC_PRG
VAR
  iSelection : INT;                  // to switch the
  xVisible : BOOL;                    // auxiliary variable to
toggle the menu bar
  iMenuPos : INT := -300;            // position of the menu bar
  xToggle : BOOL;                     // button variable to
toggle the menu bar
END_VAR
```

3. Input the following code into the implementation:

```plaintext
IF xToggle THEN
  xToggle := FALSE;
IF xVisible THEN
  xVisible := FALSE;
iMenuPos := -300;
ELSE
  xVisible := TRUE;
iMenuPos := 0;
END_IF
END_IF
```

3. Creating the menu bar

The menu bar has 3 menu items. A visualization screen is displayed by clicking the corresponding menu item.

1. Insert a “Visu_Menu” visualization below the application.
2. Open the object properties. In the “Visualization” tab, set the “Visualization size” to a “Width” of 300 and a “Height” of 180.
3. Open the visualization in the editor.
4. Select the “Advanced” option in the “Properties” view.
5. In the upper left corner, add a button with a “Width” of 300 and a “Height” of 60.
6. Label the button as “Visu 1”. Set the font size to 24.
7. Open the “Inputconfiguration ➔ OnMouseClick” property.
9. Input the following ST code:

```plaintext
PLC_PRG.iSelection := 0;
PLC_PRG.xToggle := TRUE;
```
10. Set the “Button state variable ➔ Digital variable” property to PLC_PRG.iSelection=0
11. Add two more buttons named “Visu 2” and “Visu 3”.

---

2020/12/10
12. Edit the button properties of "Visu2" (PLC_PRG.iSelection = 1) and "Visu3" (PLC_PRG.iSelection = 2).

Result:

4. Creating more visualization screens

1. Insert the "Visu1" visualization below the application.
2. Open the object properties. In the “Visualization” tab, set the “Visualization size” to a “Width” of 800 and a “Height” of 600.
3. Change the background color of the screen (for example, light gray).
4. Insert a “Label” object into the visualization screen and name the element (example: "Visu 1").
5. Insert two more visualizations "Visu2" and "Visu3" below the application. Edit the properties in the same way as for "Visu1".

5. Creating the main visualization screen

On this screen, you can see the menu bar and a button to show or hide the menu bar. The different visualization screens are navigated in a “Frame” visualization element.

1. Open the properties of the “Visu_Main” visualization. In the “Visualization” tab, set the “Visualization size” to a “Width” of 800 and a “Height” of 600.
2. Open the visualization in the editor.
3. Insert a “Frame” element into the visualization.
   ⇒ The “Frame Configuration” dialog opens.
4. Add the “Visu1” (Index 0), “Visu2” (Index 1), and “Visu3” (Index 2) visualizations.
5. Set the property values of “Position” as follows: “X” = 0, “Y” = 0, “Width” = 800, and “Height” = 600.
6. Set the property value of “Toggle variable ⇒ Variable” to PLC_PRG.iSelection.
7. Insert a “Button” element into the visualization.
8. Set the property values of “Position” as follows: “X” = 0, “Y” = 0, “Width” = 60, and “Height” = 60.
9. Set the property value of “Texts ⇒ Text” to =.
10. Set the property value of “Text properties ⇒ Font” to Arial; 36.
11. Open the “Inputconfiguration ⇒ OnMouseClick” property.
13. Input the following ST code:

   PLC_PRG.xToggle := TRUE;

14. Set the property value of “Button state variable ➔ Digital variable” to PLC_PRG.xVisible.
15. Insert the “Visu_Menu” visualization element from the “Current Project” category into the visualization.
16. Set the property values of “Position” as follows: “X” = 0, “Y” = 0, “Width” = 300, and “Height” = 180.
17. Set the property value “Absolute movement ➔ Movement ➔ X” to PLC_PRG.iMenuPos.
18. Set the property value of “State variables ➔ Invisible” to not(PLC_PRG.xVisible).
19. Set the property value of “Animation duration” to 2000.

Result:

![Visualization screenshot]

See also
- Chapter 1.3.5.18.1.6 “Visualization element ‘Frame’” on page 1287

6. Downloading the project to the controller and starting the WebVisu

1. Build the project and download it to the PLC.
2. Start the project.
3. In the browser, connect to the visualization (http://localhost:8080).
   ⇒ The WebVisu connects to the controller and the visualization opens.
4. In the visualization, click the menu button.
   ⇒ The menu moves into view.
5. Select a menu item.
   ⇒ The visualization screen is selected and the menu moves back out of view.

See also

1.4 Libraries and solutions

1.4.1 Information on customer libraries

Target change from AC500 V2 to AC500 V3

After a target change from AC500 V2 to AC500 V3 the customer libraries have to be converted manually using the Library Converter Target Change § Chapter 1.5.6.1.3 “Later change-over of target system” on page 3245.

Some Standard CODESYS libraries are automatically converted during the target change.

1.4.2 AC500 V3 libraries

Function block descriptions of all V3 libraries are available in the library manager.
1. Under “Application” open “Library Manager”.
2. Select “Add Library”.

A list of all available libraries is displayed.

Libraries in folder “ABB - AC500” are created by ABB and tested in combination with Automation Builder.

We recommend to use libraries of subfolder “Use Cases” for your project.

Libraries in subfolder “Intern” are necessary for internal procedures.

All 3S libraries distributed with Automation Builder are required by ABB libraries and have been tested in combination with AC500 and Automation Builder. Additional 3S libraries that are not distributed with Automation Builder can easily be added. There are no known major issues with using them, however, be aware that they are not tested by ABB.

3. Add a library.
4. Choose the added library in Library Manager to access the documentation.

1.4.2.1 Reference to CODESYS (V3)

Note that CODESYS V3 libraries are used. Documentation is integrated in the Library Manager.

For information on programming, see CODESYS programming Chapter 1.3.1.19.1 “Information for library developers” on page 372.
1.4.2.2 ACS/DCS drives libraries - System technology

1.4.2.2.1 Introduction

Scope of the document

The purpose of system technology document is to give an overview and explain the overall concepts of the Drives library in V3. The library contains function blocks to establish communication, to control the ABB ACS / DCS drives from AC500 V3 PLCs.

Safety instructions and preconditions to use drives library

The user has to read the following instructions and documents before using the libraries:

- All pertinent state, regional, and local safety regulations must be observed when installing and using this product. When functions or devices are used for applications with technical safety requirements, the relevant instructions must be followed.
- Read the complete safety instructions of the user's manuals for the devices you are using, before installation and commissioning.
- Read all safety instructions of the AC500 PLC. See System description AC500 in the online help in Automation Builder.
- Read the user information of the devices and functions you are using, see online help in Automation Builder.
- Installation and commissioning of the drive(s) is not part of this document nor the online help of Automation Builder. Installation and commissioning of the drive(s) must be done according to the related drives manuals and safety instructions.

The library package has been released for the software and firmware versions listed in the Readme file of the package only.

In no event will ABB or its representatives be liable for loss of data, profits, revenue or consequential, incidental or other damage that may result from the use of other versions of product, software or firmware versions. The error-free operation of the Drives V3 Library with other devices, software or firmware versions should be possible but cannot be guaranteed and may need adaptations e.g. of example programs.

The user must follow all applicable safety instructions and the guidelines mentioned in the user documents of the ABB products.

Read the complete safety instructions for the AC500 before installation and commissioning.

CAUTION!
Generally, the user in all applications is fully and alone responsible for checking all functions carefully, especially for safe and reliable operation.

The function blocks contained in the library can only be executed in RUN mode of the PLC, but not in simulation mode.

Comparison of V2 and V3 drives library

The below table compares the FBs in the V2 library package and corresponding adapted FBs in the V3 library package.

The V2 package (PS553-Drives) has different library files for each protocol and the same is replaced with a single library in V3 (ABB_Drives_AC500).
<table>
<thead>
<tr>
<th>Library Name</th>
<th>Function Block</th>
<th>Library Name</th>
<th>Function Block</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACSDrives-Base_AC500_V2</td>
<td>ACS3XX_DRIVES_CTRL_BASIC</td>
<td>Not supported – use DrvControlModbusACS</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ACS_DRIVES_CTRL_ENG</td>
<td>ABB_Drives_AC500</td>
<td>DrvControlModbusEng</td>
</tr>
<tr>
<td></td>
<td>ACS_DRIVES_CTRL_STANDARD</td>
<td></td>
<td>DrvControlModbusACS</td>
</tr>
<tr>
<td></td>
<td>ACS_DRIVES_CTRL_STANDARD_GEN</td>
<td></td>
<td>DrvControlACS</td>
</tr>
<tr>
<td></td>
<td>ACS.MOD_READ_N_PRM</td>
<td></td>
<td>DrvModbusRead</td>
</tr>
<tr>
<td></td>
<td>ACS.MOD_WRITE_N_PRM</td>
<td></td>
<td>DrvModbusWrite</td>
</tr>
<tr>
<td></td>
<td>ACS_REF_SCALING</td>
<td></td>
<td>DrvScaling</td>
</tr>
<tr>
<td>ACSDrivesCom-ModRTU_AC500_V20</td>
<td>ACS3XX_COM_MOD_RTU</td>
<td>Not supported</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ACS.COM_MOD_RTU</td>
<td>ABB_Drives_AC500</td>
<td>DrvModbusRtu</td>
</tr>
<tr>
<td></td>
<td>ACS.COM_MOD_RTU_ENHANCED</td>
<td></td>
<td>DrvModbusRtu</td>
</tr>
<tr>
<td></td>
<td>ACS.COM_MOD_RTU_EN</td>
<td>ABB.ModbusRtu.AC500</td>
<td>ModRtuToken</td>
</tr>
<tr>
<td></td>
<td>ACS.COM_MOD_RTU_GEN</td>
<td></td>
<td>ModRtuRead</td>
</tr>
<tr>
<td></td>
<td>ACS.COM_MOD_RTU_GEN_READ_N_PRM</td>
<td></td>
<td>ModRtuWrite</td>
</tr>
<tr>
<td></td>
<td>ACS.COM_MOD_RTU_GEN_WRITE_N_PRM</td>
<td></td>
<td>ModRtuReadWrite23</td>
</tr>
<tr>
<td>ACSDrivesCom-ModTCP_AC500_V22</td>
<td>ACS.COM_MOD_TCP</td>
<td>ABB_Drives_AC500</td>
<td>DrvModbusTcp</td>
</tr>
<tr>
<td></td>
<td>ACS.COM_MOD_TCP_ENHANCED</td>
<td></td>
<td>DrvModbusTcp</td>
</tr>
<tr>
<td>ACSDrivesCom-ModTCP_Ext_AC500_V24</td>
<td>ACS.COM_MOD_TCPx</td>
<td>ABB_Drives_AC500</td>
<td>DrvModbusTcp</td>
</tr>
<tr>
<td></td>
<td>ACS.COM_MOD_TCPx_ENHANCED</td>
<td></td>
<td>DrvModbusTcp</td>
</tr>
<tr>
<td>DCSDrives_AC500_V24</td>
<td>DCS_DRIVES_CTRL</td>
<td>ABB_Drives_AC500</td>
<td>DrvControlModbusDCS</td>
</tr>
<tr>
<td></td>
<td>DCS_DRIVES_CTRL_GEN</td>
<td></td>
<td>DrvControlDCS</td>
</tr>
<tr>
<td>ACSDrives-ComPN_AC500_V24</td>
<td>ACS_PN_WRITE_N_PRM_DPV1</td>
<td>Will be supported in next Release</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ACS_PN_READ_N_PRM_DPV1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ACSDrives-ComPB_AC500_V24</td>
<td>ACS_PB_READ_N_PRM_DPV1</td>
<td>Will be supported in next Release</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ACS_PB_WRITE_N_PRM_DPV1</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>ACS_COM_PB</td>
<td>Not supported</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ACS_COM_PB_PZD</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>ACS_PB_READ_PRM_DP_DPV0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Overview of the drives library for V3 PLC

This document will briefly explain about communication settings between PLCs with drives, how to control the drives from PLC using the control function blocks.

Each input and output of the function blocks are explained in the integrated documentation in the library.

This library is released for the following products:

- AC500 V3 CPU
- ABB Drives:
  - ACS380, ACS480, ACS580, ACH580, ACQ580, ACSM1, ACS880, DCS550, DCS800, DCS880 other drives may still work, but are not tested.
  - To use the control blocks the Communication Profile must be “ABB Drives Profile” or “ABB Drives Profile enhanced”
- Fieldbus Adapters: FENA-01, FENA-11, FENA-21, FSCA-01, FCAN-01, FECA-01, RETA-01, RETA-02, RCAN-01.
  Fieldbus adapter support is dependent on the drive and for more details refer the corresponding drive manual.

Drives Library in V3 will support following protocols for the communication:

- Modbus TCP (onboard ETH1 and ETH2 ports)
- Modbus RTU (onboard COM1 port)
- PROFINET (using communication module CM579-PNIO)
- EtherCAT (using communication module CM579-ETHCAT)
- CANopen (onboard CAN port)

Modbus TCP

The following hardware components must be available:

- AC500 V3 PLC with ETH option. Configure onboard ETH1 or ETH2 for Modbus TCP.
- Drive with FieldBusPlug
  - ACS Drives and DCS880: FENA-01 or FENA-11 or FENA-21
  - DCS550 and DCS800: RETA-01
- RJ45 Ethernet cable
ACS drives

Fig. 13: FB - Overview of Modbus TCP connection with ACS drives

To exchange only status word, actual speed, control word and speed reference:

- Communication profile in drive parameters: ABB Drives classic
- Communication function block in AC500 program:
  Use function block 'DrvModbusTcp'. For more details see Chapter 1.4.2.2.2.3.1.8 “DrvModbusTcp” on page 1995.
- DrvModbusTcp
- Control function block in AC500 program:
  Use function block 'DrvControlModbusACS'. For more details see Chapter 1.4.2.2.2.3.1.4 “DrvControlModbusACS” on page 1991.
- Scaling of the speed or torque (optional):
  Use function block 'DrvScaling'. For more details see Chapter 1.4.2.2.2.3.1.1 “DrvScaling” on page 1986.

To exchange status word, actual value1 (speed), actual value2 (torque), control word, reference1 (speed), reference value2 (torque) and up to 12 more values read from drive and up to 12 more values write to the drive:

- Communication profile in drive parameters: ABB Drives enhanced
- Communication function block in AC500 program:
  Use the function block ‘DrvModbusTcp’ with input EnhancedProfile = TRUE. For more details see Chapter 1.4.2.2.2.3.1.8 “DrvModbusTcp” on page 1995.
- DrvModbusTcp
- Control function block in AC500 program:
  Use function block ‘DrvControlModbusACS’. For more details see Chapter 1.4.2.2.2.3.1.4 “DrvControlModbusACS” on page 1991.
- Scaling of the speed or torque (optional): Use function block ‘DrvScaling’. For more details see Chapter 1.4.2.2.2.3.1.1 “DrvScaling” on page 1986.
To exchange more than above mentioned values use additionally the following blocks:

- **Read the values:**
  Use the function block ‘DrvModbusRead’. For more details see Chapter 1.4.2.2.2.3.1.6 “DrvModbusRead” on page 1994.

- **Write the values:**
  Use the function block ‘DrvModbusWrite’. For more details see Chapter 1.4.2.2.2.3.1.7 “DrvModbusWrite” on page 1995.

- **Read Write the values:**
  Use the function block ‘DrvModbusReadWrite23’. For more details see Chapter 1.4.2.2.2.3.1.11 “DrvModbusReadWrite23” on page 2015.

### DCS drives

![Diagram of Modbus TCP connection with DCS drives](image)

*Fig. 14: FB - Overview of Modbus TCP connection with DCS drives*

**DCS550 and DCS800 drives**

To exchange only status word, actual speed, control word and speed reference:

- **Communication function block in AC500 program:**
  Use function block ‘DrvModbusTcp’. For more details see Chapter 1.4.2.2.2.3.1.8 “DrvModbusTcp” on page 1995.

- **DrvModbusTcp**

- **Control function block in AC500 program:**
  Use function block ‘DrvControlModbusDCS’. For more details see Chapter 1.4.2.2.2.3.1.5 “DrvControlModbusDCS” on page 1993.

- **Scaling of the speed or torque (optional):**
  Use function block ‘DrvScaling’. For more details see Chapter 1.4.2.2.2.3.1.1 “DrvScaling” on page 1986.
To exchange more than above mentioned values use additionally the following blocks:

- **Read the values:**
  Use the function block ‘DrvModbusRead’. For more details see Chapter 1.4.2.2.3.1.6 “DrvModbusRead” on page 1994.

- **Write the values:**
  Use the function block ‘DrvModbusWrite’. For more details see Chapter 1.4.2.2.3.1.7 “DrvModbusWrite” on page 1995.

- **Read Write the values:**
  Use the function block ‘DrvModbusReadWrite23’. For more details see Chapter 1.4.2.2.3.1.11 “DrvModbusReadWrite23” on page 2015.

**DCS80 drives**

To exchange only status word, actual speed, control word and speed reference:

- **Communication profile in drive parameters:** ABB Drives classic
- **Communication function block in AC500 program:**
  Use function block ‘DrvModbusTcp’. For more details see Chapter 1.4.2.2.3.1.8 “DrvModbusTcp” on page 1995.

- **DrvModbusTcp**
- **Control function block in AC500 program:**
  Use function block ‘DrvControlModbusDCS’. For more details see Chapter 1.4.2.2.3.1.5 “DrvControlModbusDCS” on page 1993.
- **Scaling of the speed or torque (optional):**
  Use function block ‘DrvScaling’. For more details see Chapter 1.4.2.2.3.1.1 “DrvScaling” on page 1986.

To exchange status word, actual value1 (speed), actual value2 (torque), control word, reference1 (speed), reference value2 (torque) and up to 12 more values read from drive and up to 12 more values write to the drive:

- **Communication profile in drive parameters:** ABB Drives enhanced
- **Communication function block in AC500 program:**
  Use the function block ‘DrvModbusTcp’ with input EnhancedProfile = TRUE. For more details see Chapter 1.4.2.2.3.1.8 “DrvModbusTcp” on page 1995.

- **DrvModbusTcp**
- **Control function block in AC500 program:**
  Use function block ‘DrvControlModbusDCS’. For more details see Chapter 1.4.2.2.3.1.5 “DrvControlModbusDCS” on page 1993.
- **Scaling of the speed or torque (optional):** Use function block ‘DrvScaling’. For more details see Chapter 1.4.2.2.3.1.1 “DrvScaling” on page 1986.

To exchange more than above mentioned values use additionally the following blocks:

- **Read the values:**
  Use the function block ‘DrvModbusRead’. For more details see Chapter 1.4.2.2.3.1.6 “DrvModbusRead” on page 1994.

- **Write the values:**
  Use the function block ‘DrvModbusWrite’. For more details see Chapter 1.4.2.2.3.1.7 “DrvModbusWrite” on page 1995.

- **Read Write the values:**
  Use the function block ‘DrvModbusReadWrite23’. For more details see Chapter 1.4.2.2.3.1.11 “DrvModbusReadWrite23” on page 2015.
Modbus RTU

The following hardware components must be available:

- AC500 V3 PLC. Configure onboard COM1 for the Modbus RTU communication.
- Drive with FieldBusPlug
  - ACS Drives and DCS880: Embedded fieldbus or FSCA-01
  - DCS550 and DCS800: Embedded fieldbus or RMBA-01
- Twisted pair serial cable

ACS drives

![ACS drives diagram](image)

*Fig. 15: FB - Overview of Modbus RTU connection with ACS drives*

To exchange only status word, actual speed, control word and speed reference:

- Communication profile in drive parameters: ABB Drives classic
- Communication function block in AC500 program:
  Use function block 'DrvModbusRtu'. For more details see *Chapter 1.4.2.2.2.3.1.9 “DrvModbusRtu” on page 2002.*
- DrvModbusRtu
- Control function block in AC500 program:
  Use function block 'DrvControlModbusACS'. For more details see *Chapter 1.4.2.2.2.3.1.4 “DrvControlModbusACS” on page 1991.*
- Scaling of the speed or torque (optional):
  Use function block 'DrvScaling'. For more details see *Chapter 1.4.2.2.3.1.1 “DrvScaling” on page 1986.*
To exchange status word, actual value1 (speed), actual value2 (torque), control word, reference1 (speed), reference value2 (torque) and up to 12 more values read from drive and up to 12 more values write to the drive:

- Communication profile in drive parameters: ABB Drives enhanced
- Communication function block in AC500 program:
  Use the function block ‘DrvModbusRtu’. For more details see \(\text{Chapter 1.4.2.2.3.1.9 “DrvModbusRtu” on page 2002}.\)

- DrvModbusRtu
- Control function block in AC500 program:
  Use function block ‘DrvControlModbusACS’. For more details see \(\text{Chapter 1.4.2.2.3.1.4 “DrvControlModbusACS” on page 1991}.\)
- Scaling of the speed or torque (optional): Use function block ‘DrvScaling’. For more details see \(\text{Chapter 1.4.2.2.3.1.1 “DrvScaling” on page 1986}.\)

To exchange more than above mentioned values use additionally the following blocks:

- Read the values:
  Use the function block ‘DrvModbusRead’. For more details see \(\text{Chapter 1.4.2.2.3.1.6 “DrvModbusRead” on page 1994}.\)

- Write the values:
  Use the function block ‘DrvModbusWrite’. For more details see \(\text{Chapter 1.4.2.2.3.1.7 “DrvModbusWrite” on page 1995}.\)

- Read Write the values:
  Use the function block ‘DrvModbusReadWrite23’. For more details see \(\text{Chapter 1.4.2.2.3.1.11 “DrvModbusReadWrite23” on page 2015}.\)

**DCS drives**

![Diagram of Modbus RTU connection with DCS drives](image-url)

*Fig. 16: FB - Overview of Modbus RTU connection with DCS drives*
To exchange only status word, actual speed, control word and speed reference:

- **Communication function block in AC500 program:**
  Use function block 'DrvModbusRtu'. For more details see Chapter 1.4.2.2.3.1.9 “DrvModbusRtu” on page 2002.
- **DrvModbusRtu**
- **Control function block in AC500 program:**
  Use function block 'DrvControlModbusDCS'. For more details see Chapter 1.4.2.2.3.1.5 “DrvControlModbusDCS” on page 1993.
- **Scaling of the speed or torque (optional):**
  Use function block 'DrvScaling'. For more details see Chapter 1.4.2.2.3.1.1 “DrvScaling” on page 1986.

To exchange more than above mentioned values use additionally the following blocks:

- **Read the values:**
  Use the function block ‘DrvModbusRead’. For more details see Chapter 1.4.2.2.3.1.6 “DrvModbusRead” on page 1994.
- **Write the values:**
  Use the function block ‘DrvModbusWrite’. For more details see Chapter 1.4.2.2.3.1.7 “DrvModbusWrite” on page 1995.
- **Read Write the values:**
  Use the function block ‘DrvModbusReadWrite23’. For more details see Chapter 1.4.2.2.3.1.11 “DrvModbusReadWrite23” on page 2015.

**DCS800 drives**

To exchange only status word, actual speed, control word and speed reference:

- **Communication profile in drive parameters:** ABB Drives classic
- **Communication function block in AC500 program:**
  Use function block ‘DrvModbusRtu’. For more details see Chapter 1.4.2.2.3.1.9 “DrvModbusRtu” on page 2002.
- **DrvModbusRtu**
- **Control function block in AC500 program:**
  Use function block ‘DrvControlModbusDCS’. For more details see Chapter 1.4.2.2.3.1.5 “DrvControlModbusDCS” on page 1993.
- **Scaling of the speed or torque (optional):**
  Use function block ‘DrvScaling’. For more details see Chapter 1.4.2.2.3.1.1 “DrvScaling” on page 1986.

To exchange status word, actual value1 (speed), actual value2 (torque), control word, reference1 (speed), reference value2 (torque) and up to 12 more values read from drive and up to 12 more values write to the drive:

- **Communication profile in drive parameters:** ABB Drives enhanced
- **Communication function block in AC500 program:**
  Use the function block ‘DrvModbusRtu’. For more details see Chapter 1.4.2.2.3.1.9 “DrvModbusRtu” on page 2002.
- **DrvModbusRtu**
- **Control function block in AC500 program:**
  Use function block ‘DrvControlModbusDCS’. For more details see Chapter 1.4.2.2.3.1.5 “DrvControlModbusDCS” on page 1993.
- **Scaling of the speed or torque (optional):** Use function block ‘DrvScaling’. For more details see Chapter 1.4.2.2.3.1.1 “DrvScaling” on page 1986.
To exchange more than above mentioned values use additionally the following blocks:

- **Read the values:**
  Use the function block ‘DrvModbusRead’. For more details see Chapter 1.4.2.2.2.3.1.6 “DrvModbusRead” on page 1994.

- **Write the values:**
  Use the function block ‘DrvModbusWrite’. For more details see Chapter 1.4.2.2.2.3.1.7 “DrvModbusWrite” on page 1995.

- **Read Write the values:**
  Use the function block ‘DrvModbusReadWrite23’. For more details see Chapter 1.4.2.2.2.3.1.11 “DrvModbusReadWrite23” on page 2015.

### PROFINET

The following hardware components must be available:

- AC500 V3 PLC with CM579-PNIO (PROFINET Master communication module)
- Drive with FieldBusPlug
  - ACS Drives and DCS880: FENA-01 or FENA-11 or FENA-21
  - DCS550 and DCS800: RETA-02
- RJ45 Ethernet cable

The following values should be mapped in the fieldbus configuration of the drive and the configuration of AC500. These settings must be done in the Automation Builder hardware configuration.

- **Drive → AC500:** Status word and actual value 1 (speed) and optional actual value 2 (torque).
- **AC500 → Drive:** Control word and reference value 1 (speed) and optional reference value 2 (torque).

The following function blocks can be configured in the AC500 program.

- **Communication profile:** ABB Drives Profile
- **Control block:**
  - ACS Drives: Use function block ‘DrvControlACS’. For more details see Chapter 1.4.2.2.2.3.1.2 “DrvControlACS” on page 1987.
  - DCS Drives: Use function block ‘DrvControlDCS’. For more details see Chapter 1.4.2.2.2.3.1.3 “DrvControlDCS” on page 1989.
- **Scaling of the speed or torque (optional):** Use function block ‘DrvScaling’. For more details see Chapter 1.4.2.2.2.3.1.1 “DrvScaling” on page 1986.
ACS drives

Fig. 17: FB - Overview of PROFINET connection with ACS drives

DCS drives

Fig. 18: FB - Overview of PROFINET connection with DCS drives

EtherCAT

The following hardware components must be available:

- AC500 V3 PLC with CM579-ETHCAT (EtherCAT Master communication module)
- Drive with FieldBusPlug
  - ACS Drives and DCS880: FECA-01
  - DCS550 and DCS800: RECA-01
- RJ45 Ethernet cable
The following values should be mapped in the fieldbus configuration of the drive and the configuration of AC500. These settings must be done in the Automation Builder hardware configuration.

- Drive → AC500: Status word and actual value 1 (speed) and optional actual value 2 (torque).
- AC500 → Drive: Control word and reference value 1 (speed) and optional reference value 2 (torque).

A direct Ethernet cable from CM579-ETHCAT to FECA-01 module is recommended, connection through switch is not recommended since it will slow down the connectivity. Also, the drives need to be connected in the same sequence as they are added in the Automation Builder when multiple drives are connected.

The following function blocks can be configured in the AC500 program.

- Communication profile: ABB Drives Profile
- Control block:
  - ACS Drives: Use function block ‘DrvControlACS’. For more details see Chapter 1.4.2.2.3.1.2 "DrvControlACS" on page 1987.
  - DCS Drives: Use function block ‘DrvControlDCS’. For more details see Chapter 1.4.2.2.3.1.3 "DrvControlDCS" on page 1989.
- Scaling of the speed or torque (optional): Use function block ‘DrvScaling’. For more details see Chapter 1.4.2.2.3.1.1 “DrvScaling” on page 1986.

ACS drives

![Fig. 19: FB - Overview of EtherCAT connection with ACS drives](image-url)
**DCS drives**

![Diagram of EtherCAT connection with DCS drives]

*Fig. 20: FB - Overview of EtherCAT connection with DCS drives*

**CANopen**

The following hardware components must be available:

- AC500 V3 PLC. Configure onboard CAN port for CANopen communication.
- Drive with FieldBusPlug
  - ACS Drives and DCS880: FCAN-01
  - DCS550 and DCS800: RCAN-01
- CANopen communication cable with 120 Ω resistor.

The following values should be mapped in the fieldbus configuration of the drive and the configuration of AC500. These settings must be done in the Automation Builder hardware configuration.

- Drive → AC500: Status word and actual value 1 (speed) and optional actual value 2 (torque).
- AC500 → Drive: Control word and reference value 1 (speed) and optional reference value 2 (torque).

The following function blocks can be configured in the AC500 program.

- Communication profile: ABB Drives Profile
- Control block:
  - ACS Drives: Use function block ‘DrvControlACS’. For more details see Chapter 1.4.2.2.3.1.2 “DrvControlACS” on page 1987.
  - DCS Drives: Use function block ‘DrvControlDCS’. For more details see Chapter 1.4.2.2.3.1.3 “DrvControlDCS” on page 1989.
- Scaling of the speed or torque (optional): Use function block ‘DrvScaling’. For more details see Chapter 1.4.2.2.3.1.1 “DrvScaling” on page 1986.
ACS drives

Fig. 21: FB - Overview of CANopen connection with ACS drives

DCS drives

Fig. 22: FB - Overview of CANopen connection with DCS drives

CANopen with CAN CiA402 Profile for generic Drives

The following hardware components must be available:

- AC500 V3 PLC. Configure onboard CAN port for CANopen communication.
- Any drive with CAN FieldBusPlug and CAN CiA402 profile.
- CANopen communication cable with 120 Ω resistor.
The following values should be mapped in the fieldbus configuration of the drive and the configuration of AC500. These settings must be done in the Automation Builder hardware configuration.

- Drive → AC500: Status word and actual speed.
- AC500 → Drive: Control word and reference speed.

The following function blocks can be configured in the AC500 program.

- Communication profile: CANopen device profile CiA402
- Control block: Use function block ‘DrvControlCANCiA402’. For more details see Chapter 1.4.2.2.3.1.13 "DrvControlCANCiA402" on page 2018.

General drives with CAN CiA402 interface

![Fig. 23: FB - Overview of CANopen CiA402 with any drives](image)
Compatibility

To check the compatibility of the drives and their communication modules please refer to the following table, it shows the tested combinations.

<table>
<thead>
<tr>
<th>Communication Modules</th>
<th>PLC Communication Modules</th>
<th>Drive FieldBusPlug Plug</th>
<th>Drive</th>
</tr>
</thead>
<tbody>
<tr>
<td>PLC coupler</td>
<td>Firmware version</td>
<td>FBP comm sw ver</td>
<td>FBP appl sw ver</td>
</tr>
<tr>
<td>Modbus RTU - Classic</td>
<td>Onboard</td>
<td>FSCA-01</td>
<td>1.63</td>
</tr>
<tr>
<td></td>
<td>Embedded</td>
<td>ACS380</td>
<td>2.04.0.3</td>
</tr>
<tr>
<td>Modbus RTU – Enhanced</td>
<td>Onboard</td>
<td>FSCA-01</td>
<td>1.63</td>
</tr>
<tr>
<td>Modbus TCP</td>
<td>Onboard ETH1 / ETH2</td>
<td>RETA-01</td>
<td>1.30</td>
</tr>
<tr>
<td>Modbus TCP - Enhanced</td>
<td>Onboard ETH1 / ETH2</td>
<td>FENA-21</td>
<td>3.20</td>
</tr>
<tr>
<td>PROFINE T</td>
<td>CM579-PNIO</td>
<td>2.8.6.21</td>
<td>FENA-21</td>
</tr>
<tr>
<td>EtherCAT</td>
<td>CM579-ETHCAT</td>
<td>4.4.3.21</td>
<td>FECA-01</td>
</tr>
<tr>
<td>CANopen (ABB Profile)</td>
<td>Onboard</td>
<td>FCAN-01</td>
<td>1.16</td>
</tr>
<tr>
<td>CANopen (CiA402)</td>
<td>Onboard</td>
<td>FCAN-01</td>
<td>1.16</td>
</tr>
</tbody>
</table>

1.4.2.2.2 Overview of the library

Installation

The library is part of the Automation Builder 2.2. or higher. Use the Library manager to add the library into project.

For more details on the package, refer to the release notes of the latest Automation Builder.
## Hardware and software requirement

<table>
<thead>
<tr>
<th>Hardware</th>
<th>Software</th>
</tr>
</thead>
<tbody>
<tr>
<td>AC500 V3 PLCs: PM5630-2ETH, PM5650-2ETH, PM5670-2ETH and PM5675-2ETH</td>
<td>Automation Builder 2.2. or higher</td>
</tr>
<tr>
<td>ABB Drive: ACS380, ACS480, ACS580, ACH580, ACQ580, ACS880, ACM1, DCS550, DCS800, DCS880. (other drives may work, but are not tested)</td>
<td>Drive Composer Pro, Drive Studio, Drive Window or Drive Window Light</td>
</tr>
<tr>
<td>FieldBusPlug: FSCA-01, RMBA-01, FENA-01 / FENA-11 / FENA-21, RETA-01, FECA-01, RETA-02, FCAN-01, RCAN-01. (other FieldBusPlugs may work, but are not tested)</td>
<td></td>
</tr>
</tbody>
</table>

*Drive configuration tool and FieldBusPlug support is dependent on the drive used, for the compatible tool details refer to the drive manual.*

## Description of the library

This chapter briefly explains the functions, function blocks, structures, enumerations and visualization present in the library.
Fig. 24: FB - Overview of the Drives Library
Function Blocks

DrvScaling

DrvScaling function block is used to scale the speed or torque reference to the drive based on the maximum values defined.

Function block DrvScaling can be used to scale the variables from fieldbus equivalent values to values used in the program. Fieldbus variables are given in fieldbus equivalent values as INT values. With the scaling a conversion from INT (fieldbus) to REAL (program) and vice versa is performed. Reference1 and Actual Value1 (speed) are mostly given in the range of -20000 ... +20000. Reference2 and Actual Value2 (torque) are mostly given in the range of 0 ... +10000.
This function block can be used to control ACS drives with ABB drives profile using direct input of status word (SW) from drive via any supported fieldbus communication like PROFINET, EtherCAT, CANopen.

Control word (CW) will be built by the function block according to the ABB drives profile state machine. Output CW has to be send to the drive via any fieldbus communication supported. Function block provides standard start/stop signals to control the drive and standard diagnosis signals are read from the drive.
<table>
<thead>
<tr>
<th>Drive Parameter</th>
<th>ACS380/ ACS480/ ACS580/ ACH580/ ACQ580/ ACS880</th>
<th>ACSM1</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXT1 COMMANDS</td>
<td>20.01 = Fieldbus A</td>
<td>10.01 = FBA</td>
<td>Fieldbus interface as source for start and stop</td>
</tr>
<tr>
<td>EXT1 / EXT2 SEL</td>
<td>19.11 = MCW Bit11 (06.01)</td>
<td>34.01 = P02.12 bit 15</td>
<td>Fieldbus interface as source to switch to EXT2 control place</td>
</tr>
<tr>
<td>REF1 SELECT</td>
<td>22.11 = FBA ref1</td>
<td>24.01 = FBA ref1</td>
<td>Fieldbus interface as source to speed reference</td>
</tr>
<tr>
<td>FAULT RESET SELECT</td>
<td>31.11 = P06.01 bit 7</td>
<td>10.08 = P02.12 bit 8</td>
<td>Fieldbus interface as source for fault reset</td>
</tr>
<tr>
<td>PROFILE</td>
<td>51.02 = Drives Classic / Enhanced</td>
<td>51.02 = Drives Classic / Enhanced</td>
<td>Control profile to ABB Drives profile classic or enhanced</td>
</tr>
</tbody>
</table>
This function block can be used to control DCS drives with ABB drives profile using direct input of status word (SW) from drive via any supported fieldbus communication like PROFINET, EtherCAT, CANopen.

Control word (CW) will be built by the function block according to the ABB drives profile state machine. Output CW must be sent to the drive via any fieldbus communication supported. Function block provides standard start/stop signals to control the drive and standard diagnosis signals are read from the drive.
<table>
<thead>
<tr>
<th>Drive Parameter</th>
<th>DCS550</th>
<th>DCS800</th>
<th>DCS880</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXT1 COMMANDS</td>
<td>10.01 = Main Ctrl Word</td>
<td>10.01 = Main Ctrl Word</td>
<td>20.01 = Main Ctrl Word</td>
<td>Fieldbus interface as source for start and stop</td>
</tr>
<tr>
<td>EXT1 / EXT2 SEL</td>
<td>10.07 (HandAuto)</td>
<td>10.07 (HandAuto)</td>
<td>19.11 = MCW Bit11 (06.01)</td>
<td>Fieldbus interface as source to switch to EXT2 control place</td>
</tr>
<tr>
<td></td>
<td>MCW: Bit11</td>
<td>MCW: Bit11</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>11.02 (Ref1Mux)</td>
<td>11.02 (Ref1Mux)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>MCW: Bit11</td>
<td>MCW: Bit11</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>11.12 (Ref2Mux)</td>
<td>11.12 (Ref2Mux)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Invert 11.02</td>
<td>Invert 11.02</td>
<td></td>
<td></td>
</tr>
<tr>
<td>REF1 SELECT</td>
<td>11.03 = SpeedRef2301</td>
<td>11.03 = SpeedRef2301</td>
<td>22.11 = FBA ref1</td>
<td>Fieldbus interface as source to speed reference</td>
</tr>
<tr>
<td>FAULT RESET SELECT</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>Fieldbus interface as source for fault reset</td>
</tr>
<tr>
<td>PROFILE</td>
<td>NA</td>
<td>NA</td>
<td>51.02 = Drives Classic / Enhanced</td>
<td>Control profile to ABB Drives profile classic or enhanced</td>
</tr>
</tbody>
</table>
This function block can be used to control ACS drives with ABB Drives profile or ABB Drives enhanced profile using Modbus communication block like DrvModbusTcp or DrvModbusRtu.

Status Word (SW) is read from drive through Modbus communication block using DriveData interface (For more details on the structure refer to Chapter 1.4.2.2.3.3 “Structure: DrvDataType” on page 2019).
Control Word (CW) will be built by the function block according to the ABB drives profile state machine. CW will be sent via DriveData and the used communication block to the drive. Function block provides standard start/stop signals to control the drive and standard diagnosis signals are read from the drive.

The function block should be used for ACS drives using ABB drive (Classic/Enhanced) profile for Modbus protocol only. The data transfer to the ACS drive is realized via the IN_OUT variable DriveData, which must be connected to DrvModbusTcp or DrvModbusRtu function block.
This function block can be used to control DCS drives with ABB Drives profile using Modbus communication block like DrvModbusTcp or DrvModbusRtu.

Status Word (SW) is read from drive through Modbus communication block using DriveData interface (For more details on the structure refer to Chapter 1.4.2.2.3.3 “Structure: DrvData-Type” on page 2019).

Control Word (CW) will be built by the function block according to the ABB drives profile state machine. CW will be sent via DriveData and the used communication block to the drive. Function block provides standard start/stop signals to control the drive and standard diagnosis signals are read from the drive.
The function block should be used for DCS drives using ABB drive profile for Modbus protocol only. The data transfer to the DCS drive is realized via the IN_OUT variable DriveData, which must be connected to DrvModbusTcp or DrvModbusRtu function block.

**DrvModbusRead**

The function block 'DrvModbusRead' reads one or more parameters / values of the drive. The number of data to be read is specified at the input 'Nvar'. The first parameter number is specified at the input 'PrmNum'. All parameters must be accessible from consecutive Modbus registers in the drive. The values of the parameters are stored in the PLC memory area, defined at the input 'Data'.

The values in the PLC memory area are updated when the read job was performed without error. This is indicated by JobDone = TRUE and ModMastErrorAct = FALSE.

If the Modbus job was finished with an error, the output ModMastErrorAct is set for one cycle. The Error ID returned by the Modbus job is shown at the output ModMastErrorIDLast. The output ModMastErrorIDLast will show that last Error ID until the input Enable is set from TRUE to FALSE.

As long as the Enable = TRUE a new read job is requested each time the further read job is terminated. The Modbus job is started from the Communication Block which is connected to the same 'DriveData' variable. It uses the Modbus function code 03 (read n words). The drive (Modbus device) from which the parameter is read is specified at this Communication Block. The Communication Blocks are available from the library e.g. DrvModbusTcp or DrvModbusRtu.

The function block is activated (Enable = TRUE) or deactivated (Enable = FALSE) via input Enable. If the block is active, the current values are available at the outputs. To start a new read job the input Enable must be set to TRUE. If the input values are valid, a request to perform a Modbus job is sent to the Communication Block via the 'DriveData' variable. If at least 1 input is invalid, no job is generated, and the error is displayed at the outputs Error and ErrorID instead.
**DrvModbusWrite**

Function block 'DrvModbusWrite' writes 'n' parameters to the drive. The number of parameters to be written must be available in the PLC memory area, defined at the input Data. The write job has been performed without error if JobDone = TRUE and ModMastErrorAct = FALSE.

If the Modbus job was finished with an error, the output ModMastErrorAct is set for one cycle. The Error ID returned by the Modbus job is shown at the output ModMastErrorIDLast. The output ModMastErrorIDLast will show that last Error ID until the input Execute is set from TRUE to FALSE.

To start a new write job the input Execute must be set from FALSE to TRUE (edge sensitive). The Modbus job is started from the Communication Block which is connected to the same DriveData variable. It uses the Modbus function code 16 (write n words). The drive (Modbus device) to which the parameter is written is specified at the Communication Block.

Drive parameters are only saved temporarily, if changed via fieldbus. To make these changes permanent in the drive the special parameter "PARAMETER SAVE" must be set.

Please see drive manuals for the parameter details.

**DrvModbusTcp**

Function block DrvModbusTcp controls the Modbus TCP communication to ACS/DCS drives and provides the basic values (CW, Ref1, Ref2, SW, Act1, Act2) which are used for the basic control of drives with ABB Drives Profile or ABB Drives Enhanced Profile.

**ABB drives classic profile**

With input parameter EnhancedProfile = FALSE, the function block works for ABB Drives Classic Profile.
The function block continuously reads data from the drive starting at Modbus register 400004. So at least the Status Word (SW), Actual Value 1 (Speed Reference), Actual Value 2 (Actual Value 2) are continuously read from the drive and written to the DriveData variable. These values are stored in DriveData.StatusWord, DriveData.ActValue1 and DriveData.ActValue2.

The following table shows the performed Modbus read job and the needed mapping in the drive as well as the area where the data is stored in the AC500.

<table>
<thead>
<tr>
<th>Modbus register address in drive</th>
<th>Mapping configuration in drive</th>
<th>Written in AC500</th>
<th>Condition at function block</th>
</tr>
</thead>
<tbody>
<tr>
<td>[ACS380, ACS480, ACS580, ACH580, ACQ580, ACS880, ACSM1]</td>
<td>[DCS550, DCS800]</td>
<td>DCS880</td>
<td>Enable = TRUE</td>
</tr>
<tr>
<td>Communication module</td>
<td>FENA-01/11/21</td>
<td>RETA-01</td>
<td>FENA-01/11/21</td>
</tr>
<tr>
<td>400004</td>
<td>Status Word (SW)</td>
<td>Status Word (SW)</td>
<td>Status Word (SW)</td>
</tr>
<tr>
<td>400005</td>
<td>Actual Value 1</td>
<td>Actual Value 1</td>
<td>Actual Value 1</td>
</tr>
<tr>
<td>400006</td>
<td>Actual Value 2</td>
<td>Actual Value 2</td>
<td>Actual Value 2</td>
</tr>
</tbody>
</table>

To write the Control Word (CW), Reference Value 1 (Speed Reference) or Reference Value 2 (Reference Value 2) from the DriveData variable (DriveData.ControlWord, DriveData.Reference1, DriveData.Reference2) to the drive, the input EnableWrite has to be TRUE (default).

If the input SteadyWrite = TRUE (default = FALSE) these values are written steadily.
If the input SteadyWrite = FALSE (default) these values are only written if there was a change on any of those values.

These 3 values are written to the ACS drive starting at Modbus register 400001.

The function block checks if there are changes of the Control Word (wControlWord), Reference Value 1 (iRefValue1) or Reference Value 2 (iRefValue2) on the DriveData variable. If there is a change a write job is requested to send these 3 values to the ACS/DCS drive starting at Modbus register 400001.

The following table shows the performed Modbus write job and the needed mapping in the drive as well as the area where the data is stored in the AC500.
<table>
<thead>
<tr>
<th>Modbus register address in drive</th>
<th>Mapping configuration in drive</th>
<th>Written from AC500</th>
<th>Condition at function block</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACS380, ACS480, ACS580, ACH580, ACQ580, ACS880, ACSM1</td>
<td>DCS550, DCS800</td>
<td>DCS880</td>
<td></td>
</tr>
<tr>
<td>Communication module</td>
<td>FENA-01/ 11/21</td>
<td>RETA-01</td>
<td>FENA-01/11/21</td>
</tr>
<tr>
<td>400001</td>
<td>Control Word (CW)</td>
<td>Control Word (CW)</td>
<td>Control Word (CW)</td>
</tr>
<tr>
<td>400002</td>
<td>Reference Value 1</td>
<td>Reference Value 1</td>
<td>Reference Value 1</td>
</tr>
<tr>
<td>400003</td>
<td>Reference Value 2</td>
<td>Reference Value 2</td>
<td>Reference Value 2</td>
</tr>
</tbody>
</table>

**ABB drives enhanced profile**

With input parameter EnhancedProfile = TRUE, the function block works for ABB Drives Enhanced Profile.

**Reading status information from drives**

The function block continuously reads data from the drive starting at Modbus register 400051. So at least the Status Word (SW), Actual Value 1 (Speed Reference), Actual Value 2 (Actual Value 2) are continuously read from the drive and written to the DriveData variable.

These values are stored in DriveData.StatusWord, DriveData.ActValue1 and DriveData.ActValue2.

Apart from these three parameters there is also an option to read 12 additional drive parameters.

Using the input NvarRead the function block can be configured to read between 0 and 12 parameters from the drive. All read data is then written to the array at the ReadValue output array. Configuration in ACS drive is depending on configured parameters in group FBA DATA IN.
### Modbus register address in drive

<table>
<thead>
<tr>
<th>Modbus register address in drive</th>
<th>Mapping configuration in drive</th>
<th>Written in AC500</th>
<th>Condition at function block</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACS380, ACS480, ACS580, ACH580, ACQ580, ACS880, ACSM1</td>
<td>DCS550, DCS800</td>
<td>DCS880</td>
<td></td>
</tr>
</tbody>
</table>

### Communication module

<table>
<thead>
<tr>
<th>Communication module</th>
<th>Module Code</th>
<th>Status Word (SW)</th>
<th>Actual Value 1</th>
<th>Actual Value 2</th>
<th>FBA Data IN 1</th>
<th>FBA Data IN 2</th>
<th>...</th>
<th>FBA Data IN 11</th>
<th>FBA Data IN 12</th>
<th>Enable</th>
<th>NVarRead</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>FENA-01/11/21</td>
<td>Status Word (SW)</td>
<td>Actual Value 1</td>
<td>Actual Value 2</td>
<td>FBA Data IN 1</td>
<td>FBA Data IN 2</td>
<td>...</td>
<td>FBA Data IN 11</td>
<td>FBA Data IN 12</td>
<td>TRUE</td>
<td>&gt;= 1</td>
</tr>
<tr>
<td></td>
<td>RETA-01</td>
<td>Status Word (SW)</td>
<td>Actual Value 1</td>
<td>Actual Value 2</td>
<td>FBA Data IN 1</td>
<td>FBA Data IN 2</td>
<td>...</td>
<td>FBA Data IN 11</td>
<td>FBA Data IN 12</td>
<td>TRUE</td>
<td>&gt;= 2</td>
</tr>
</tbody>
</table>

If 32-bit parameters are mapped to DATA IN,

- The following field in DATA IN must be left open (= 0)
- The word order of the High-Word (HW) and Low-Word (LW) can be configured in the drive.
  (using FENA-X1: Par. 51.22)
- To retrieve the original 32-bit value from the drive in AC500 the HW and LW from ReadValues fields must be recombined in the program.

Function block DATA IN has to be configured in drive in the following groups see also FENA-x1 manual.

<table>
<thead>
<tr>
<th>Drive</th>
<th>Parameter Group</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACS355</td>
<td>54.01 ... 54.10</td>
</tr>
<tr>
<td>ACS380, ACS480, ACS580, ACH580, ACQ580, ACS880, ACSM1</td>
<td>52.01 ... 52.12 52.01 ... 52.12 if installed as adapter A</td>
</tr>
</tbody>
</table>
To write the Control Word (CW), Reference Value 1 (Speed Reference) or Reference Value 2 (Reference Value 2) from the DriveData variable (DriveData.ControlWord, DriveData.Reference1, DriveData.Reference2) to the drive, the input EnableWrite has to be TRUE (default).

If the input SteadyWrite = TRUE (default = FALSE) these values are written steadily.

If the input SteadyWrite = FALSE (default) these values are only written if there was a change on any of those values.

These 3 values are written to the ACS drive starting at Modbus register 400001.

Apart from these three there parameters there is also an option to write 12 additional drive parameters.

Using the input NvarWrite the function block can be configured to write between 0 and 12 parameters to the drive. The necessary values must be present in the array connected to WriteValues input. Configuration in ACS drive is depending on configured parameters in group FBA DATA OUT.

<table>
<thead>
<tr>
<th>Modbus register address in drive</th>
<th>Mapping configuration in drive</th>
<th>Written from AC500</th>
<th>Condition at function block</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACS380, ACS480, ACS580, ACH580, ACQ580, ACS880, ACSM1</td>
<td>DCS550, DCS800</td>
<td>DCS880</td>
<td></td>
</tr>
<tr>
<td>Communication module</td>
<td>FENA-01/11/21</td>
<td>RETA-01</td>
<td>FENA-01/11/21</td>
</tr>
<tr>
<td>400001</td>
<td>Control Word (CW)</td>
<td>Control Word (CW)</td>
<td>Control Word (CW)</td>
</tr>
<tr>
<td>400002</td>
<td>Reference Value 1</td>
<td>Reference Value 1</td>
<td>Reference Value 1</td>
</tr>
<tr>
<td>400003</td>
<td>Reference Value 2</td>
<td>Reference Value 2</td>
<td>Reference Value 2</td>
</tr>
<tr>
<td>400004</td>
<td>FBA Data OUT 1</td>
<td>FBA Data OUT 1</td>
<td>FBA Data OUT 1</td>
</tr>
<tr>
<td>400005</td>
<td>FBA Data OUT 2</td>
<td>FBA Data OUT 2</td>
<td>FBA Data OUT 2</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>400015</td>
<td>FBA Data OUT 12</td>
<td>FBA Data OUT 12</td>
<td>FBA Data OUT 12</td>
</tr>
</tbody>
</table>

If a Modbus TCP job tries to access a register in the drive which has no valid mapping information then job is aborted with an error.

Therefore, the drive parameters in FBA DATA IN group and FBA DATA OUT must be configured according to the used 'NvarRead' and 'NvarWrite' input number respectively.
If 32-bit parameters are mapped to DATA OUT,
- The next/following field in DATA OUT must be left open (= 0)
- The word order of the High-Word (HW) and Low-Word (LW) can be configured in the drive.
  (using FENA-X1: Par. 51.22)
- To retrieve the original 32-bit value from the drive in AC500 the HW and LW from WriteValues fields must be recombined in the program.

ACS drive parameters are only saved temporarily, if changed via fieldbus. To make these changes permanent in the drive the special parameter "PARAMETER SAVE" must be set.
Please see also drive manuals which parameter must be set.
For ACS380, ACS480, ACS580, ACH580, ACQ580, ACS880 and DCS880 – Par 96.07 = 1
For ACSM1, DCS800 and DCS550 – Par 16.06 = 1

Diagnosis

If a Modbus TCP job tries to access a register in the drive which has no valid mapping information the job is aborted with an error.

The output ModMastErrorAct reflects that an actual error occurred. This output is only TRUE for one cycle. At that cycle the output ModMastErrorIDLast reflects the actual ErrorID from the ModTcpMast job. The ModMastErrorIDLast will keep this Error ID until a new rising edge of the Enable is given.

However, there are internal diagnosis variables available, which are not shown at any output, but can be accessed from the function block instance.

These additional diagnosis variables can be accessed by opening the function block instance or through the block visualization "VisuDrvModbusTcp".
- iWriteErrCnt: number of errors in write jobs since Enable = TRUE.
- wLastWriteErno: holds the error number of the last executed write job.
- iReadErrCnt: number of errors in read jobs since Enable = TRUE.
- wLastReadErno: holds the error number of the last executed read job.
- iReadWriteErrCnt: number of errors in read write jobs since Enable = TRUE.
- wLastReadWriteErno: holds the error number of the last executed read write job.

If the user changes drive profile while drive is online with PLC, function block outputs may give wrong indication.
### Drive parameter settings

<table>
<thead>
<tr>
<th>Settings in the drive according to AC500 configuration</th>
<th>ACS380, ACS480, ACS580, ACH580, ACQ580, ACS880, ACSM1</th>
<th>DCS550, DCS800</th>
<th>DCS880</th>
</tr>
</thead>
<tbody>
<tr>
<td>Communication module</td>
<td>FENA-01 /11/21</td>
<td>RETA-01</td>
<td>FENA-01/11/21</td>
</tr>
<tr>
<td>Fieldbus activation = EXT FBA / ENABLE</td>
<td>50.01</td>
<td>98.02</td>
<td>50.01</td>
</tr>
<tr>
<td>FBA A Comm loss func</td>
<td>50.02</td>
<td>30.35</td>
<td>50.02</td>
</tr>
<tr>
<td>Comm Rate = Auto (0)</td>
<td>51.03</td>
<td>51.02</td>
<td>51.03</td>
</tr>
<tr>
<td>IP configuration = Static IP</td>
<td>51.04</td>
<td>51.03</td>
<td>51.04</td>
</tr>
<tr>
<td>IP address1 … IP address2</td>
<td>51.05 … 51.08</td>
<td>51.04 … 51.07</td>
<td>51.05 … 51.08</td>
</tr>
<tr>
<td>Subnet CIDR = 24 (eg: 255.255.255.0)</td>
<td>51.09</td>
<td>51.08 … 51.11</td>
<td>51.09</td>
</tr>
<tr>
<td>Gateway Address (normally = 0.0.0.0)</td>
<td>51.10 … 51.13</td>
<td>51.12 … 51.15</td>
<td>51.10 … 51.13</td>
</tr>
<tr>
<td>Protocol / Profile = MB/TCP ABB E or MB/TCP ABB C</td>
<td>51.02</td>
<td>51.16</td>
<td>51.02</td>
</tr>
<tr>
<td>Word order for 32-bit parameter</td>
<td>51.22</td>
<td>No 32-bit access</td>
<td>51.22</td>
</tr>
<tr>
<td>Modbus timeout. Depending on timeout mode. Value in 100 ms</td>
<td>51.20</td>
<td>51.17</td>
<td>51.20</td>
</tr>
<tr>
<td>Modbus timeout mode: If input “SteadyWrite” is false set to “Any message“ If input “SteadyWrite” is true can also be set to “Control RW“</td>
<td>51.21</td>
<td></td>
<td>51.21</td>
</tr>
<tr>
<td>Refresh settings in drive</td>
<td>51.27</td>
<td>51.27</td>
<td>51.27</td>
</tr>
</tbody>
</table>

- Please refer the respective drive / fieldbus module manual for the parameter settings if the drive setting is not mentioned in above table.
- For RETA-01/-02 IP address could also be set via hardware Dip-Switches. If any switch is set (192.168.0.xxx) with xxx = Dip-Switches setting
- ACS drive parameters are only saved temporarily, if changed via fieldbus. To make these changes permanent in the drive the special parameter “PARAMETER SAVE” must be set.
- Please see also drive manuals which parameter must be set.
  - For ACS380, ACS480, ACS580, ACH580, ACQ580, ACS880 and DCS880 – Par 96.07 = 1
  - For ACSM1, DCS800 and DCS550 – Par 16.06 = 1
**DrvModbusRtu**

Function block DrvModbusRtu controls the Modbus RTU communication to ACS/DCS drives and is used for the basic control of drives with ABB Drives Profile or ABB Drives Enhanced Profile.

**ABB drives classic profile**

With input parameter EnhancedProfile = FALSE, the function block works for ABB Drives Classic Profile.

**Reading status information from drives**

The function block continuously reads data from the drive starting at Modbus register 400004. So at least the Status Word (wStatusWord), Actual Value 1 (iActValue1), Actual Value 2 (iActValue2) are continuously read from the drive and written to the DriveData variable.

These values are stored in DriveData.wStatusWord, DriveData.iActValue1 and DriveData.iActValue2.

With input NvarRead the function block can be configured to read in the same job between 0 ... 24 data more from the drive. These additional data are written to the array at the 'ReadValues' output. These data must be configured in the drive and are only accessible if the embedded Modbus is used.

The following table shows the performed Modbus read job and the needed mapping in the drive as well as the area where the data is stored in the AC500.
<table>
<thead>
<tr>
<th>Modbus register address in drive</th>
<th>Mapping configuration in drive</th>
<th>Written in AC500</th>
<th>Condition at function block</th>
</tr>
</thead>
<tbody>
<tr>
<td>Communication module</td>
<td>FSCA-01</td>
<td>RMBA-01</td>
<td>Embedded fieldbus</td>
</tr>
</tbody>
</table>

| 400004 | Status Word (SW) | Status Word (SW) | Status Word (SW) | 58.104 = 4 | DriveData.wStatusWord | Enable = TRUE |
| 400005 | Actual Value 1 | Actual Value 1 | Actual Value 1 | 58.105 = 5 | DriveData.iActValue1 | Enable = TRUE |
| 400006 | Actual Value 2 | Actual Value 2 | Actual Value 2 | 58.106 = 6 | DriveData.iActValue2 | Enable = TRUE |
| 400007 | - | - | 58.107 DATA I/O 7 | ReadValues[1] | Enable = TRUE |
| ... | ... | ... | ... | ... | ... | ... |
| 400014 | - | - | 58.114 DATA I/O 14 | ReadValues[8] | Enable = TRUE |
| ... | ... | ... | ... | ... | ... | ... |
| 400024 | - | - | 58.124 DATA I/O 24 | ReadValues[18] | Enable = TRUE |

More details on the limits for the data read and write is explained in Chapter 1.4.2.2.2.4 “Limits for the data read and write between AC500 and drives” on page 2020. The value is dependent on the Drive used.

Writing control word and reference value to drives

The function block checks if there are changes of the Control Word (wControlWord), Reference Value 1 (iRefValue1) or Reference Value 2 (iRefValue2) on the DriveData variable. If there is a change a write job is requested to send these 3 values to the ACS/DCS drive starting at Modbus register 400001.

The following table shows the performed Modbus write job and the needed mapping in the drive as well as the area where the data is stored in the AC500.
### Modbus register address in drive

<table>
<thead>
<tr>
<th>Modbus register address in drive</th>
<th>Mapping configuration in drive</th>
<th>Written from AC500</th>
<th>Condition at function block</th>
</tr>
</thead>
</table>

#### Communication module

<table>
<thead>
<tr>
<th>Communication module</th>
<th>Module 1</th>
<th>Module 2</th>
<th>Embedded fieldbus</th>
</tr>
</thead>
<tbody>
<tr>
<td>FSCA-01</td>
<td>RMBA-01</td>
<td></td>
<td>Embedded fieldbus</td>
</tr>
</tbody>
</table>

#### Reading status information from drives

- The function block continuously reads data from the drive starting at Modbus register 400051. So at least the Status Word (wStatusWord), Actual Value 1 (iActValue1), Actual Value 2 (iActValue2) are continuously read from the drive and written to the DriveData variable.

- These values are stored in DriveData.wStatusWord, DriveData.iActValue1 and DriveData.iActValue2.

- Apart from these three parameters there is also an option to read 12 additional drive parameters in the same job.

- Using the input NvarRead the function block can be configured to read between 1 and 12 more parameters from the drive. All read data is then written to the array at the ReadValues output.

- Configuration in ACS drive is depending on configured parameters in group FBA DATA IN.

- The following table shows the performed Modbus read job and the needed mapping in the drive as well as the area where the data is stored in the AC500.

---

More details on the limits for the data read and write is explained in Chapter 1.4.2.2.2.4 “Limits for the data read and write between AC500 and drives” on page 2020. The value is dependent on the Drive used.

**ABB drives enhanced profile**

With input parameter EnhancedProfile = TRUE, the function block works for ABB Drives Enhanced Profile.

The ABB Drives Profile Enhanced communication profile provides register mapped access to the Control, Status, Reference and Actual Values of the ABB Drives Profile Enhanced. The mapping of the registers has been enhanced to allow additional writing of up to 12 control and reading of up to 12 additional status parameters in a single Modbus job.
The function block checks if there are changes in any of the following values since last write job:

- Control Word (wControlWord),
- Reference Value 1 (iRefValue1),
- Reference Value 2 (iRefValue2) on the DriveData variable,
- values in the input array WriteValues – WriteValues[1..NvarWrite].

If there is a change a write job is requested to send the 3 control values and the values in WriteValues array (WriteValues[1..NvarWrite]) to the ACS/DCS drive starting at Modbus register 400001. Configuration in ACS drive is depending on configured parameters in group FBA DATA OUT.

The following table shows the performed Modbus write job and the needed mapping in the drive as well as the area where the data is stored in the AC500.
<table>
<thead>
<tr>
<th>Modbus register address in drive</th>
<th>Mapping configuration in drive</th>
<th>Written from AC500</th>
<th>Condition at function block</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACS380, ACS480, ACS580, ACH580, ACS, ACS880, ACS580, ACH580, ACH580, AC800, ACH580, AC880</td>
<td>DCS50, DCS800</td>
<td>DCS880</td>
<td></td>
</tr>
<tr>
<td>Communication module</td>
<td>FSCA-01</td>
<td>RMBA-01</td>
<td>FSCA-01</td>
</tr>
<tr>
<td>400001</td>
<td>Control Word (CW)</td>
<td>Control Word (CW)</td>
<td>Control Word (CW)</td>
</tr>
<tr>
<td>400002</td>
<td>Reference Value 1</td>
<td>Reference Value 1</td>
<td>Reference Value 1</td>
</tr>
<tr>
<td>400003</td>
<td>Reference Value 2</td>
<td>Reference Value 2</td>
<td>Reference Value 2</td>
</tr>
<tr>
<td>400004</td>
<td>FBA Data OUT 1</td>
<td>FBA Data OUT 1</td>
<td>FBA Data OUT 1</td>
</tr>
<tr>
<td>400005</td>
<td>FBA Data OUT 2</td>
<td>FBA Data OUT 2</td>
<td>FBA Data OUT 2</td>
</tr>
</tbody>
</table>
| ... | ... | ... | ... | ... | ...
| 400015 | FBA Data OUT 12 | FBA Data OUT 12 | FBA Data OUT 12 | WriteValues[12] | Enable = TRUE NVarWrite >= 12 |

**Reconnection pause**

When one or more drives in the Modbus RTU lines are offline, all the other drives have to wait for the TimeOut to elapse until a line token is assigned to next drive. Reconnection pause input helps in skipping the drives which are offline from the next Modbus job and execute Modbus job operations only for the drives which are online.

“ReconnectPause” is time in seconds before next retry to connect after a timeout was detected. Timeout is detected with ModMastErrorIDLast = 16#120 (ERR_TIMEOUT).

This feature can be used with the DrvModbusRtu function block in both ABB Drives Profile and ABB Drives Enhanced Profile. User must configure the reconnect pause input value using the input variable “ReconnectPause”.

For the generic RTU block ModRtuToken (part of AC500_ModbusRtu library), also the value for the reconnect pause must be configured at input variable “ReconnectPause”.

**Diagnosis**

The output ErrorID which reflects an actual error number is only valid for one cycle if output Error is set to TRUE. To capture this error number an external function must be programmed.

The output ModMastErrorAct reflects that an actual error occurred. This output is only TRUE for one cycle. At that cycle the output ModMastErrorIDLast reflects the actual ErrorID from the ModRtuMast job. The ModMastErrorIDLast will keep this error ID until a new rising edge of the Enable input is given.
However, there are internal diagnosis variables available, which are not shown at any output, but can be accessed from the function block instance.

These additional diagnosis variables can be accessed by opening the function block instance or through the block visualization “VisuDrvModbusRTU”.

- iWriteErrCnt: number of errors in write jobs since Enable = TRUE.
- wLastWriteErno: holds the error number of the last executed write job.
- iReadErrCnt: number of errors in read jobs since Enable = TRUE.
- wLastReadErno: holds the error number of the last executed read job.
- iReadWriteErrCnt: number of errors in read write jobs since Enable = TRUE.
- wLastReadWriteErno: holds the error number of the last executed read write job.

If several drives are used, for each drive a communication function block such as DrvModbusRtu must be programmed. Also, every other generic Modbus server device on the same Modbus RTU line must be programmed with its own ModRtuToken function block. All those communication function blocks of one Modbus RTU line must be linked together via one variable of type ModRtuTokenType, connected to the InOut LineToken. Via this variable the Modbus token is passed to the next drive/device, so only one drive/device at a time is communicating with the PLC.

ModRtuToken function block and ModRtuTokenType structure are part of AC500_ModbusRtu library. Kindly refer the same.

If the user changes drive profile while drive is online with PLC, function block outputs may give wrong indication.

If a Modbus RTU job tries to access a register in the drive which has no valid mapping information then the job is aborted with an error.

Therefore, the drive parameters in FBA DATA IN group and FBA DATA OUT must be configured according to the used ‘NvarRead’ and ‘NvarWrite’ input number respectively.

Modbus RTU using Embedded Fieldbus:
When embedded fieldbus is used for the Modbus RTU communication, user can read maximum of 24 parameters (based on the limitation in drive) from the DATA I/O parameters in the embedded fieldbus parameter group. These parameters can only be used for reading operation and cannot be configured to write data.
## Drive parameter settings

<table>
<thead>
<tr>
<th>Settings in the drive according to AC500 configuration</th>
<th>ACS380, ACS480, ACS580, ACH580, ACQ580, ACS880, ACSM1, DCS880</th>
<th>DCS550, DCS800</th>
<th>ACS380, ACS480, ACS580, ACH580, ACQ580, ACS880, DCS880</th>
</tr>
</thead>
<tbody>
<tr>
<td>Communication module</td>
<td>FSCA-01</td>
<td>RMB-01</td>
<td>Embedded fieldbus</td>
</tr>
<tr>
<td>Fieldbus activation = EXT FBA / ENABLE</td>
<td>50.01</td>
<td>98.02</td>
<td>58.01</td>
</tr>
<tr>
<td>FBA A Comm loss func</td>
<td>50.02</td>
<td>30.35</td>
<td>50.02</td>
</tr>
<tr>
<td>Slave number</td>
<td>51.03</td>
<td>51.02</td>
<td>58.03</td>
</tr>
<tr>
<td>Transmission rate</td>
<td>51.04</td>
<td>51.03</td>
<td>58.04</td>
</tr>
<tr>
<td>Parity</td>
<td>51.05</td>
<td>51.05</td>
<td>58.05</td>
</tr>
<tr>
<td>Protocol / Profile = ABB Classic / ABB Enhanced</td>
<td>51.02</td>
<td>51.16</td>
<td>51.02</td>
</tr>
<tr>
<td>Word order for 32-bit parameter</td>
<td>51.22</td>
<td>No 32-bit access</td>
<td>51.22</td>
</tr>
<tr>
<td>Mapping of control word, Mod-bus reg 400001</td>
<td>Fix</td>
<td>Fix</td>
<td>58.101</td>
</tr>
<tr>
<td>Mapping of reference value 1, Modbus reg 400002</td>
<td>Fix</td>
<td>Fix</td>
<td>58.102</td>
</tr>
<tr>
<td>Mapping of reference value 2, Modbus reg 400003</td>
<td>Fix</td>
<td>Fix</td>
<td>58.103</td>
</tr>
<tr>
<td>Mapping of status word, Modbus reg 400004</td>
<td>Fix</td>
<td>Fix</td>
<td>58.104</td>
</tr>
<tr>
<td>Mapping of actual value 1, Modbus reg 400005</td>
<td>Fix</td>
<td>Fix</td>
<td>58.105</td>
</tr>
<tr>
<td>Mapping of actual value 2, Modbus reg 400006</td>
<td>Fix</td>
<td>Fix</td>
<td>58.106</td>
</tr>
<tr>
<td>Timeout mode = None (0) or Any Message (1), but not Ctrl write (2) as these values are only written after changes</td>
<td>51.07</td>
<td></td>
<td>58.15</td>
</tr>
<tr>
<td>Modbus timeout. Depending on timeout mode. Value in 100 ms</td>
<td></td>
<td></td>
<td>58.17</td>
</tr>
<tr>
<td>Refresh settings in drive</td>
<td>51.27</td>
<td>51.27</td>
<td>58.06</td>
</tr>
</tbody>
</table>
– Please refer the respective drive / fieldbus module manual for the parameter settings if the drive setting is not mentioned in above table.

– ACS drive parameters are only saved temporarily, if changed via fieldbus. To make these changes permanent in the drive the special parameter “PARAMETER SAVE” must be set.

Please see also drive manuals which parameter must be set.

For ACS380, ACS480, ACS580, ACH580, ACQ580, ACS880 and DCS880 – Par 96.07 = 1

For ACSM1, DCS800 and DCS550 – Par 16.06 = 1
**DrvModbusRtuBroadcast**

Function block DrvModbusRtuBroadcast is a communication block which sends the broadcast messages via the Modbus RTU communication to all ACS/DCS drives and other Modbus devices connected to the same Modbus RTU line (physical line). The function block can be used with all drives with either ABB Drives Profile or ABB Drives Enhanced Profile but not a mix of both profiles.

As the broadcast job will be received by all devices on the same physical Modbus line it’s highly recommended to use this block only in case there are no other Modbus devices connected to this line and all drives use the same profile.

This function block does not perform any Modbus read operation, hence it does not read any values such as status word, actual value 1 and actual value 2 etc., from any of the drive.

This function block should not be used along with 'DrvModbusRead' and 'DrvModbusReadWrite23' function blocks. They will be ignored showing an error. This function block should be only used independently or in combination with 'DrvModbusWrite' function block for broadcasting write operation.

A successful broadcast message for writing control word, reference values and additional mapped parameters (only in case of Enhanced Profile) is indicated by JobDone = TRUE and ModMastErrorAct = FALSE. A next broadcast job for writing these values can once again started with a fresh rising edge at ‘SendCtrlValues’ input.

Apart from sending control values and up to 12 additional values from WriteValues array (only in case of ABB Drives Enhanced Profile) a normal Modbus write function block "DrvModbusWrite" can be used to send broadcast write messages to specific address on all drives connected to the Modbus RTU line. The requests to process broadcast write Modbus jobs is transferred via the DriveData structure at the InOut variable DriveData which can be connected to multiple instances of write function block 'DrvModbusWrite'.

After each successful broadcast write job a fixed pause of 250 ms is implemented before any other Modbus job within the same line will be started.

**ABB drives classic profile**

With input parameter EnhancedProfile = FALSE, the function block works for ABB Drives Classic Profile.

**Writing control word and reference value to drives**

A rising edge from FALSE to TRUE at input ‘SendCtrlValues' starts sending broadcast message with Control Word and Reference Values to all the drives starting at Modbus register 400001.

Following control values: Control Word (wControlWord), Reference Value 1 (iRefValue1) or Reference Value 2 (iRefValue2) are taken from DriveData variable for sending broadcast message.

The following table shows the performed Modbus write job and the needed mapping in the drive as well as the area where the data is stored in the AC500.
ABB drives enhanced profile

With input parameter EnhancedProfile = TRUE, the function block works for ABB Drives Enhanced Profile.

With the ABB Drives Profile Enhanced profile, along with 3 control values Control Word, Reference Value 1, Reference Value 2, up to 12 additional values can be sent as broadcast message in a single Modbus job.

Writing control word and reference values to drives

A rising edge from FALSE to TRUE at input 'SendCtrlValues' starts sending broadcast message with Control Word and reference values to all the drives starting at Modbus register 400001.

Following control values: Control Word (wControlWord), Reference Value 1 (iRefValue1) or Reference Value 2 (iRefValue2) from DriveData along with values in the input array WriteValues – WriteValues[1..NvarWrite] are taken for sending broadcast message.

For the additional 12 values the configuration in ACS drive is depending on configured parameters in group FBA DATA OUT.

The following table shows the performed Modbus broadcast write job and the needed mapping in the drive as well as the area where the data is taken from the AC500.

<table>
<thead>
<tr>
<th>Modbus register address in drive</th>
<th>Mapping configuration in drive</th>
<th>Written from AC500</th>
<th>Condition at function block and input SendCtrlValues</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACS380, ACS480, ACS580, ACH580, ACQ580, ACS880, ACSM1DCS880 DCS800</td>
<td>DCS50, DCS800</td>
<td>ACS380, ACS480, ACS580, ACH580, ACQ580, ACS880DCS880 DCS880</td>
<td>DriveData.wControlWord, Enable = TRUE and Rising Edge at SendCtrlValues</td>
</tr>
</tbody>
</table>

Communication module

<table>
<thead>
<tr>
<th>400001</th>
<th>Control Word (CW)</th>
<th>Control Word (CW)</th>
<th>Control Word (CW) 58.101 = 1</th>
<th>DriveData.wControlWord, Enable = TRUE and Rising Edge at SendCtrlValues</th>
</tr>
</thead>
<tbody>
<tr>
<td>400002</td>
<td>Reference Value 1</td>
<td>Reference Value 1</td>
<td>Reference Value 1 58.102 = 2</td>
<td>DriveData.iRefValue1, Enable = TRUE and Rising Edge at SendCtrlValues</td>
</tr>
<tr>
<td>400003</td>
<td>Reference Value 2</td>
<td>Reference Value 2</td>
<td>Reference Value 2 58.103 = 3</td>
<td>DriveData.iRefValue2, Enable = TRUE and Rising Edge at SendCtrlValues</td>
</tr>
</tbody>
</table>
### Diagnosis

The output ErrorID which reflects an actual error number is only valid for one cycle if output Error is set to TRUE. To capture this error number an external function must be programmed.

The output ModMastErrorAct reflects an actual error occurred in Modbus job. This output is only TRUE for one cycle. At that cycle the output ModMastErrorIDLast reflects the actual ErrorID from the ModRtuMast job. The ModMastErrorIDLast will keep this error ID until a new rising edge of the Enable input is given.

However, there are internal diagnosis variables available, which are not shown at any output, but can be accessed from the function block instance.

These additional diagnosis variables can be accessed by opening the function block instance or through the block visualization “VisuDrvModbusRTUBroadcast”.

- **iWriteErrCnt**: number of errors in write jobs since Enable = TRUE.
- **wLastWriteErno**: holds the error number of the last executed write job.

---

**For all drives, which are connected to same Modbus RTU line, one instance of broadcast block DrvModbusRtuBroadcast is enough and it must be connected to same LineToken of DrvModbusRtu function blocks which are used for communication between PLC and each drive on Modbus RTU line. All those communication function blocks of one Modbus RTU line must be linked together via one variable of type ModRtuTokenType, connected to the InOut LineToken. Via this variable the Modbus token is passed to the next drive / device, so only one drive / device at a time is communicating with the PLC.**
All the drives should be configured either in Classic Profile or Enhanced Profile and accordingly the function block DrvModbusRtuBroadcast should be parameterized. Mix of profile with few drives in Classic and few drives in Enhanced should not be used when using DrvModbusRtuBroadcast block, if using such configuration along with DrvModbusRtuBroadcast may lead to incorrect operation.

If the user changes drive profile while drive is online with PLC, function block outputs may give wrong indication.

The Modbus RTU broadcast job is sent to all devices on the same physical Modbus RTU line.

Therefore, if other Modbus devices than ACS / DCS drives are connected to the same line using the ModRtuToken communication block it’s highly recommended not to use the DrvModbusRtuBroadcast function block.

This might only be used, if the user is aware about the behavior of the connected devices if they receive the Modbus broadcast job.

If a Modbus RTU broadcast job is sent to access a register in the drive which has no valid mapping information then Modbus broadcast job is not aborted but will just send out the broadcast message without any error in the function block. This broadcast message is ignored by drives which have no valid mapping information.

Therefore, the drive parameters in FBA DATA OUT have to be configured according to the used ‘NvarWrite’ input number respectively.
## Drive parameter settings

<table>
<thead>
<tr>
<th>Settings in the drive according to AC500 configuration</th>
<th>ACS380, ACS480, ACS580, ACH580, ACQ580, ACS880, ACSM1, DCS880</th>
<th>DCS550, DCS800</th>
<th>ACS380, ACS480, ACS580, ACH580, ACQ580, ACS880, DCS880</th>
</tr>
</thead>
<tbody>
<tr>
<td>Communication module</td>
<td>FSCA-01</td>
<td>RMB-01</td>
<td>Embedded fieldbus</td>
</tr>
<tr>
<td>Fieldbus activation = Modbus / RS-485 comm</td>
<td>50.01</td>
<td>98.02</td>
<td>58.01</td>
</tr>
<tr>
<td>FBA A Comm loss func</td>
<td>50.02</td>
<td>30.35</td>
<td>50.02</td>
</tr>
<tr>
<td>Slave number</td>
<td>51.03</td>
<td>51.02</td>
<td>58.03</td>
</tr>
<tr>
<td>Transmission rate</td>
<td>51.04</td>
<td>51.03</td>
<td>58.04</td>
</tr>
<tr>
<td>Parity</td>
<td>51.05</td>
<td>51.05</td>
<td>58.05</td>
</tr>
<tr>
<td>Protocol / Profile = ABB Classic / ABB Enhanced</td>
<td>51.02</td>
<td>51.16</td>
<td></td>
</tr>
<tr>
<td>Mapping of control word, Modbus reg 400001</td>
<td>Fix</td>
<td>Fix</td>
<td>58.101</td>
</tr>
<tr>
<td>Mapping of reference value 1, Modbus reg 400002</td>
<td>Fix</td>
<td>Fix</td>
<td>58.102</td>
</tr>
<tr>
<td>Mapping of reference value 2, Modbus reg 400003</td>
<td>Fix</td>
<td>Fix</td>
<td>58.103</td>
</tr>
<tr>
<td>Mapping of status word, Modbus reg 400004</td>
<td>Fix</td>
<td>Fix</td>
<td>58.104</td>
</tr>
<tr>
<td>Mapping of actual value 1, Modbus reg 400005</td>
<td>Fix</td>
<td>Fix</td>
<td>58.105</td>
</tr>
<tr>
<td>Mapping of actual value 2, Modbus reg 400006</td>
<td>Fix</td>
<td>Fix</td>
<td>58.106</td>
</tr>
<tr>
<td>Timeout mode = None (0) or Any Message (1), but not Ctrl write (2) as these values are only written after changes</td>
<td>51.07</td>
<td></td>
<td>58.15</td>
</tr>
<tr>
<td>Modbus timeout. Depending on timeout mode. Value in 100 ms</td>
<td>58.17</td>
<td></td>
<td>58.16</td>
</tr>
<tr>
<td>Refresh settings in drive</td>
<td>51.27</td>
<td>51.27</td>
<td>58.06</td>
</tr>
</tbody>
</table>

---

- Please refer the respective drive / fieldbus module manual for the parameter settings if the drive setting is not mentioned in above table.
- ACS drive parameters are only saved temporarily, if changed via fieldbus. To make these changes permanent in the drive the special parameter "PARAMETER SAVE" must be set.

Please see also drive manuals which parameter must be set.

For ACS380, ACS480, ACS580, ACH580, ACQ580, ACS880 and DCS880
- Par 96.07 = 1

For ACSM1, DCS800 and DCS550 – Par 16.06 = 1
The function block 'DrvModbusReadWrite23' reads and writes one or more parameters of the drive via DriveData connected to Modbus TCP / Modbus RTU communication blocks with Modbus function code FCT = 23. This function block internally calls DrvModbusWrite to execute ReadWrite job with FCT = 23, used along with the internal structure for Fct23, DrvModFct23Type.

The number of parameters to be read is specified at the input 'NvarRead'. The first address for read operation is specified at the input 'PrmNumRead'. The values of the data are stored in the PLC memory area, defined at the input 'DataRead'.

The number of parameters to be written is specified at the input 'NvarWrite'. The first address for write operation is specified at the input 'PrmNumWrite'. The values of the data that should be written must be stored in the PLC memory area, defined at the input 'DataWrite'.

To start a new ReadWrite job the input Execute must be set from FALSE to TRUE (edge sensitive). The Modbus job is started from the communication block DrvModbusTcp or DrvModbusRtu which is connected to the same DriveData variable. It uses the Modbus function code 23 (Read and write n words). The drive (Modbus device) to which the parameter is written is specified at the Communication Block.

The values in the PLC memory area are updated when the ReadWrite job was performed without error. The ReadWrite job has been performed without error if JobDone = TRUE and ModMastErrorAct = FALSE.

If the Modbus job was finished with an error, the output ModMastErrorAct is set for one cycle. The Error ID returned by the Modbus job is shown at the output ModMastErrorIDLast. The output ModMastErrorIDLast will show that last Error ID until the input Execute is set from TRUE to FALSE.

After termination of this job, even if it was not successful, a next ReadWrite job can once again only be started with a rising edge at 'Execute' input.

Drive parameters are only saved temporarily, if changed via fieldbus. To make these changes permanent in the drive the special parameter "PARAMETER SAVE" must be set.

Please see drive manuals for the parameter details.

For ACS380, ACS480, ACS580, ACH580, ACQ580, ACS880 and DCS880 – Par 96.07 = 1

For ACM1, DCS800 and DCS550 – Par 16.06 = 1
### DrvControlModbusEng

<table>
<thead>
<tr>
<th>Function</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enable</td>
<td>BOOL</td>
</tr>
<tr>
<td>Off1</td>
<td>BOOL</td>
</tr>
<tr>
<td>Off2</td>
<td>BOOL</td>
</tr>
<tr>
<td>Off3</td>
<td>BOOL</td>
</tr>
<tr>
<td>InhibitOp</td>
<td>BOOL</td>
</tr>
<tr>
<td>RampOutZero</td>
<td>BOOL</td>
</tr>
<tr>
<td>RampHold</td>
<td>BOOL</td>
</tr>
<tr>
<td>RampInZero</td>
<td>BOOL</td>
</tr>
<tr>
<td>Reset</td>
<td>BOOL</td>
</tr>
<tr>
<td>ControlWordB8</td>
<td>BOOL</td>
</tr>
<tr>
<td>ControlWordB9</td>
<td>BOOL</td>
</tr>
<tr>
<td>RemoteCmd</td>
<td>BOOL</td>
</tr>
<tr>
<td>ExtCtrlLoc</td>
<td>BOOL</td>
</tr>
<tr>
<td>ControlWordB12</td>
<td>BOOL</td>
</tr>
<tr>
<td>ControlWordB13</td>
<td>BOOL</td>
</tr>
<tr>
<td>ControlWordB14</td>
<td>BOOL</td>
</tr>
<tr>
<td>ControlWordB15</td>
<td>BOOL</td>
</tr>
<tr>
<td>UseControlWord</td>
<td>BOOL</td>
</tr>
<tr>
<td>ControlWord</td>
<td>WORD</td>
</tr>
<tr>
<td>RefValue1</td>
<td>INT</td>
</tr>
<tr>
<td>RefValue2</td>
<td>INT</td>
</tr>
<tr>
<td>DriveData</td>
<td>DrvDataType</td>
</tr>
</tbody>
</table>

### DrvControlModbusEng Diagram


- **DriveData Variable**
The function block DrvControlModbusEng is designed for user specific control of the drive by setting the Control Word (CW) by the user itself in the program. Therefore, the user should have a detailed knowledge of the ABB Drives Profile handling. The reference and actual values must be given in fieldbus equivalent, e.g. range -20000 ... +20000.

Inputs RefValue1, RefValue2 and the generated Control Word are written to the DriveData variable which transfers these values to a communication function block, e.g. DrvModbusRtu, DrvModbusTcp or DrvModbusRtuBroadcast communication function block writes to the drive. In the same way ActValue1, ActValue2 and the Status Word are transferred from the communication function block to the DrvControlModbusEng block, where they are written to the outputs.

The Control Word can be generated in 2 ways.

First way is to set the single bits of the Control Word separately at the inputs Off1, Off2 ... ControlWordB15 while the input UseControlWord = FALSE.

Second way is to set the input UseControlWord = TRUE and write the Control Word as a whole word directly to the input Control Word. The generated Control Word is written to the DriveData variable and for diagnosis purpose also available at output UsedControlWord.

The input and output names of the bits in Control Word and Status Word reflect the functions used with ABB Drive Profile. So the block should be used with ABB Drives Profile setting in the drive.

---

The function block does not execute any functionality expect data transfer to and from the DriveData variable. There is no special drive parameter setting necessary to use this block.

The programmer using this block should have a detailed understanding of how to set the Control Word according to the Status Word and the description of the used drive.

For standard speed and torque control application it is recommended to use the DrvControlModbusACS instead.
The function block DrvControlCANCiA402 is used for the control of ABB ACS Drive or non-ABB drives from AC500 using CANCiA402 profile. The CANopen CiA402 function block and visualization from the library can also be used for the 3rd party drives which comply to the CANopen CiA402 profile.
**Function: DrvModPara32Bit**

![Diagram of DrvModPara32Bit](image)

Creates the Modbus address for 32-bit parameters of the ACSxxx drives.

To access 32-bit parameters in ACSxxx drives using Modbus a special address calculation must be performed.

This block calculates the 6-digit address out of the 5-digit address used for 16-bit parameters.

Input is the 5-digit address: GGii, where GG = parameter group and ii = the index.

E.g. Par 12.02 ➔ address = 1202.

Output is the calculated address for 32-bit parameters according to the following rule:

\[
\text{DrvModPara32Bit} = 20000 + (200 \times GG) + (2 \times ii) \text{ e.g. Par. 14.54 ➔ output} = 22908
\]

This output can be connected directly to the input "PrmNum" of one of the blocks DrvModbusRead or DrvModbusWrite or inputs “PrmNumRead” and “PrmNumWrite” of the block DrvModbusReadWrite23.

**Structure: DrvDataType**

Structure DrvDataType is used for the DriveData variable to exchange the data for one drive.

Structure DrvDataType is used for the DriveData variable which must be connected to all function blocks related to the same drive.

Besides the element “sName” all variables should not be written by the user directly. They are read and written within the function blocks. The DrvDataType contains some more internal, invisible variables which are used for internal functionality and not meant for user access.

The following table shows the visible variables of DrvDataType.

<table>
<thead>
<tr>
<th>Variable</th>
<th>Data Type</th>
<th>Default value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>wStatusWord</td>
<td>WORD</td>
<td>0</td>
<td>Actual status word from drive</td>
</tr>
<tr>
<td>iActValue1</td>
<td>INT</td>
<td>0</td>
<td>Actual value1 from drive – mostly equal speed</td>
</tr>
<tr>
<td>iActValue2</td>
<td>INT</td>
<td>0</td>
<td>Actual value2 from drive – mapping is made in drive configuration</td>
</tr>
<tr>
<td>wControlWord</td>
<td>WORD</td>
<td>0</td>
<td>Control word to drive</td>
</tr>
<tr>
<td>iRefValue1</td>
<td>INT</td>
<td>0</td>
<td>Reference value1 to drive – mostly speed reference</td>
</tr>
<tr>
<td>iRefValue2</td>
<td>INT</td>
<td>0</td>
<td>Reference value2 to drive – mapping is made in drive configuration</td>
</tr>
<tr>
<td>xOnline</td>
<td>BOOL</td>
<td>FALSE</td>
<td>Connection established – set in Modbus communication function block after successful reading and writing one Modbus job</td>
</tr>
<tr>
<td>xCtrlBlockUsedf</td>
<td>BOOL</td>
<td>FALSE</td>
<td>A control block is used to generate the control word, ref1 and ref2 values</td>
</tr>
<tr>
<td>sName</td>
<td>STRING</td>
<td>’Default Drive Name’</td>
<td>Name for drive, which can be set by user directly to DriveData variable</td>
</tr>
</tbody>
</table>
Limits for the data read and write between AC500 and drives

The below table defines the limits for the reading of data from the drive and limits for writing data to drives from AC500.

If fieldbus adapter Plug (FBA) is used, then parameter group FBA DATA IN (e.g. 52) and group FBA DATA OUT (e.g. 53) is accessed in the drive. For the embedded fieldbus (EFB) parameters are used in EFB group (e.g. 58).

According to the table below, limits are defined for the variables ‘NVarRead’, ‘NVarWrite’ in DrvModbusRtu and DrvModbusTcp blocks.

<table>
<thead>
<tr>
<th>Drive</th>
<th>Fieldbus Adapter (FBA)</th>
<th>Embedded Fieldbus (EFB)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Data In (Group 52)</td>
<td>Data Out (Group 53)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ACS380</td>
<td>12</td>
<td>12</td>
</tr>
<tr>
<td>ACS480</td>
<td>12</td>
<td>12</td>
</tr>
<tr>
<td>ACS580</td>
<td>12</td>
<td>12</td>
</tr>
<tr>
<td>ACQ580</td>
<td>12</td>
<td>12</td>
</tr>
<tr>
<td>ACH580</td>
<td>12</td>
<td>12</td>
</tr>
<tr>
<td>ACS880</td>
<td>12</td>
<td>12</td>
</tr>
<tr>
<td>ACSM1</td>
<td>12</td>
<td>12</td>
</tr>
<tr>
<td>DCS550</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DCS800</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DCS880</td>
<td>12</td>
<td>12</td>
</tr>
</tbody>
</table>

1.4.2.3 PLCopen libraries

1.4.2.3.1 Common function block state machine

Most of the V3 function blocks follow the behavior model and style as recommended by PLC Open.

- Clear separation between “Edge triggered FBs” (“Execute”) or “Level triggered FBs” (“Enable”)
- Binary status outputs: “Done”, “Busy”, “Error” (exclusive)
- Standardized state machine
- CamelCase naming for function block and all inputs and outputs

Example: Edge_Triggered_Function_Block_EthOwnIP according to PLCopen
Currently the following “function block state machines” are used:

- “Edge Triggered” (Input “Execute”), for example EthOwnIP
- “Level Controlled” (Input “Enable”)
- “Level Controlled Continuous” (Input “Enable”, no “Done” output, for example PID)

In contrast to AC500 V2 POUs, either “Done” or “Error” is set, not both outputs at the same time in case of an error.

The state machines are explained in the following chapters.

Edge triggered (AbbETrig)

After a rising edge at the input “Execute” the state goes from “Dormant” to “Busy”. In the first cycle all inputs are sampled and stored. When the task is completed successfully the state goes from “Busy” to “Done”.

In case of an error the state goes to “Error”.

The states “Done” or “Error” are stable for minimum one cycle and as long as “Execute” is “TRUE”. With a falling edge of “Execute”, the state goes via Reset to “Dormant”.

Description of standard inputs and outputs:

- “Execute”
  A rising edge starts the operation, the output “Busy” goes to “TRUE”. In the first cycle all other inputs are read and stored, afterwards they are ignored.
  A falling edge does not stop the operation.
  After “Done = TRUE” or “Error = TRUE” and “Execute = FALSE” all outputs will be reset.
- “Busy”
  Operation is running (while outputs “Done” and “Error” are “FALSE”)
- “Done”
  Operation is completed without error (while outputs “Busy” and “Error” are “FALSE”). This output is “TRUE” for at least one cycle or until “Execute” is set to “FALSE”
- “Error”
  Operation is stopped with error (while outputs “Busy” and “Done” are “FALSE”). This output is “TRUE” for at least one cycle or until “Execute” is set to “FALSE”.
  The output “ErrorID” gives more details about the error.
Level controlled (AbbLCon)

After a rising edge at the input “Enable” the state goes “Dormant” to “Busy”. All inputs are sampled and considered continuously.

When the task is completed successfully the state goes from “Busy” to “Done”.

In case of an error the state goes to “Error”.

The states “Done” or “Error” are stable for minimum one cycle and as long as “Enable is TRUE”. With a falling edge of “Enable”, the state goes via Reset to “Dormant”.

The Busy state can be aborted from outside by setting the “Enable” input to “FALSE”.

After Aborting is done the state goes back to “Dormant”.

Description of standard inputs and outputs:

- **“Enable”**
  A rising edge (“Enable = TRUE”) starts the operation, the output “Busy” goes to “TRUE”. All other inputs are read and considered continuously. A falling edge (“Enable = FALSE”) aborts the operation.
  During Aborting the Busy is still “TRUE”. Afterward all outputs are reset.

- **“Busy”**
  Operation is running (while outputs “Done” and “Error” are “FALSE”)

- **“Done”**
  Operation is completed without error (while outputs “Busy” and “Error” are “FALSE”).
  This output is “TRUE” for at least one cycle or until “Enable” is set to “FALSE”

- **“Error”**
  Operation is stopped with error (while outputs “Busy” and “Done” are “FALSE”).
  This output is “TRUE” for at least one cycle or until “Enable” is set to “FALSE”.
  The output “ErrorID” gives more details about the error.

Level controlled continuous (AbbLConC)

This state machine is a special case of “Level Controlled”. Only difference is that this function block type is never done, for example a PID which never stops.

Therefore these function blocks have no “Done” output.
Description of standard inputs and outputs:

- "Enable"
  A rising edge ("Enable = TRUE") starts the operation, the output “Busy” goes to “TRUE”. All other inputs are read and considered continuously.
  A falling edge ("Enable = FALSE") aborts the operation.
  During Aborting the “Busy” is still “TRUE”. Afterward all outputs are reset.
- "Busy"
  Operation is running (while output “Error is FALSE”)
- "Error"
  Operation is stopped with error (while output “Busy is FALSE”).
  This output is “TRUE” for at least one cycle or until “Enable” is set to “FALSE”. The output “ErrorID” gives more details about the error.

**Error_ID**

Each library contains an enumeration “ERROR_ID”, which is valid for this library but not across all libraries

Only the following errors are unique:

- 16#5FFF FATAL_ERROR from state machine
- 16#4000 errors are used for input errors, same scheme like in V2:

**4000hex...4FFFhex - Block Input Error**

The error 4xoxhex is used in case of detected Function Block input parameter errors. The error is structured as follows:

![Error Structure Diagram]

- 0 = Invalid value
- 1 = Value too low
- 2 = Value too high
- 3 = Wrong combination of the parameters
- 1..FF = Number of the input
Compatibility with V2 function blocks

In order to ensure compatibility with V2 applications a lot of ABB function blocks are delivered together with a compatible version in classic style:

Names in CAPITAL letters, input “EN” and outputs “DONE”, “ERR” and “ERNO”:

The classic blocks internally use the PLCopen style function blocks. The inputs and outputs are mapped in the following way:

1.4.2.4 Motion control library

1.4.2.4.1 Preconditions for the use of the libraries

The user has to read the following instructions and documents before using the libraries:

- All pertinent state, regional, and local safety regulations must be observed when installing and using this product. When functions or devices are used for applications with technical safety requirements, the relevant instructions must be followed.
- Read the complete safety instructions of the user's manuals for the drives you are using, before installation and commissioning.
- Read all safety instructions of the AC500 PLC. See System description AC500 or chapter Regulations "Regulations" on page 2238 in the online help.
- Read the Important user Information. See chapter Safety Instructions in the online help "Safety instructions" on page 2230.

The version 1.0.0.x of the Motion Control Library Package PS5611-Motion has been released for:

- AC500 V3 PLCs, FW version 3.3.1x or higher
- Automation Builder version 2.3.x or higher

The PS5611-Motion libraries have been tested with the following product / firmware / software versions:
CM579-ETHCAT EtherCAT Communication Module firmware 4.4.3.21 or higher
ABB e190 Drive
CD522 module

In no event will ABB or its representatives be liable for loss of data, profits, revenue or consequential, incidental or other damage that may result from the use of other versions of product / software / firmware versions. The error-free operation of the PS5611 - Motion with other devices / software / firmware versions should be possible but can not be guaranteed and may need adaptations e.g. of example programs.

CAUTION!
Generally, the user in all applications is fully and alone responsible for checking all functions carefully, especially for safe and reliable operation.

The Function Blocks contained in the library can only be executed in RUN mode of the PLC, but not in simulation mode.

1.4.2.4.2 Overview
The PS5611-Motion is a Motion Control library for AC500 V3 CPUs, to create Motion Control applications based on Function Blocks according to the standard of PLCopen Motion Control Chapter 1.4.2.4.3 “PLCopen” on page 2033. These Function Blocks can be used for PLC-based Motion Control and cover a wide range of possible Motion Control functionalities. Starting from single axis movements to master-follower axes to perform electronic gearing and CAM functions.

This documentation contains the following chapters:
● Overview
In the subsequent chapters general informations are provided for a better understanding of Motion Control with AC500 PLC and PS5611-Motion. There is also a tabular overview of the available PLCopen Function Blocks and their compatibility with PLC-based Motion Control and the provided drive-based Motion Control axis implementations.
● PLCopen
The principle of the PLCopen Motion Control standard is explained as well as how PLCopen Function Blocks can be used to create PLC Motion Control application programs.
● PLC-based Motion Control
This chapter explains how PLC-based Motion Control with AC500 can be realized and how it can be used in combination with the available PLCopen Function Blocks.

Functionality of each function block is defined in the integrated documentation of the library.

Motion control with PS5611-Motion
With PS5611-Motion different Motion Control system structures are possible. Independent of the system structure a typical Motion Control application consists of the following system elements:
• An application program which contains PLCopen Function Blocks that defines the general application behavior and logics.
• A profile generator which generates a position profile based on the dynamic specifications of the application program to guide the axis to the desired positions.
• A position control loop which outputs a speed reference signal to minimize the following error.

To achieve the best system structure for an application these components can be separated into different devices. Each type of structure has its own kind of interface and type of signals which need to be transferred between the interacting devices.

All shown Motion Control system structures (central Motion Control with or without position control loop) can be combined together in the same application program for a Motion Control project.

PLC-based motion control

With PS5611-Motion the application program and also the profile generator are realized in the PLC. The implementation of the profile generator is based on a set of Function Blocks which are named Central Motion Control (CMC).

The profile generator of many possible axes is centrally placed inside the AC500 PLC. Therefore multi-axis motion functionalities become easily available and can be accessed by PLCopen Function Blocks. As a result, Motion Control functionalities are almost drive independent.

Available motion control functionalities:
• Simple axis Movements
• Electronic Gearin
• Electronic CAMs
• Position Profiles
• Velocity Profiles
• Acceleration Profiles

Then the output is a position reference signal which the drive will follow. A new position reference value will be calculated with every cycle of the PLC and has to be transferred to the drive, which demands real time capabilities to the PLC and to the communication channel. A real time fieldbus like EtherCAT is needed. The feedback of the actual position can be used for supervision purposes during operation and is needed to adjust the value of the position reference before the drive will be enabled.

AC500 as Motion Controller (Central Motion Control)

![System structure of PLC-based Motion Control with AC500 PLC and PS5611-Motion](image)
With PLC-based Motion Control it is also possible to include the position control loop to the AC500 PLC. In this case a speed reference signal will be transferred to the drive, which makes it possible to perform the full range of motion functionalities with standard drives. To close the position control loop, feedback of the actual position is mandatory.

AC500 as Motion Controller (Central Motion Control)

**Application Program (PLCopen)**

**Profile Generator**

**Position Control**

**Standard Drive** (Analog drive)

**Speed Control**

**Torque Control**

**Actual Position**

**Speed Reference**

Fig. 26: PLC-based Motion Control with AC500 PLC and PS5611-Motion, closed position control loop

Fig. 27: Central Motion Control with AC500 PLC and PS5611-Motion, different axis implementations at the same time

**Overview of PLCopen function blocks**

The following tables give an overview of the defined Function Blocks, divided into administrative (not driving motion) and motion related sets. They give an overview which Function Block could be used for the different possible configurations.

These Function Blocks are part of the library ABB_MotionControl_AC500.library.

If there are restrictions concerning a certain drive ("XXX") which lead to a different or limited behaviour compared to the standard the respective chapter is supplemented with an additional paragraph "Notes for XXX".

The “KERNEL” Function Blocks are available in two variants.
The CMC_Estp_Kernel block is solely to be used in AC500-eCo V3 CPUs and to make use of the integrated stepper-IO. It connects automatically to the internal IOs.

The CMC_Basic_Kernel block is designed to be used in standard V3 PLCs, and can either work with drives connected to a fieldbus or IOs.

Table 368: Administrative function blocks

<table>
<thead>
<tr>
<th>Function Block</th>
<th>PLC-based Motion Control</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>CMC_Basic_Kernel</td>
</tr>
<tr>
<td>MC_Power</td>
<td>X</td>
</tr>
<tr>
<td>MC_ReadStatus</td>
<td>X</td>
</tr>
<tr>
<td>MC_ReadAxisError</td>
<td>X</td>
</tr>
<tr>
<td>MC_ReadParameter</td>
<td>X</td>
</tr>
<tr>
<td>MC_ReadBoolParameter</td>
<td>X</td>
</tr>
<tr>
<td>MC_WriteParameter</td>
<td>X</td>
</tr>
<tr>
<td>MC_WriteBoolParameter</td>
<td>X</td>
</tr>
<tr>
<td>MC_Reset</td>
<td>X</td>
</tr>
<tr>
<td>MC_ReadActualPosition</td>
<td>X</td>
</tr>
<tr>
<td>MC_ReadActualVelocity</td>
<td>X</td>
</tr>
<tr>
<td>MC_SetOverride</td>
<td>X</td>
</tr>
<tr>
<td>MC_SetPosition</td>
<td>X</td>
</tr>
<tr>
<td>MC_CamTableSelect</td>
<td>X</td>
</tr>
</tbody>
</table>

Table 369: Single-Axis function blocks

<table>
<thead>
<tr>
<th>Function Block</th>
<th>PLC-based motion Control</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>CMC_Basic_Kernel</td>
</tr>
<tr>
<td>MC_MoveAbsolute</td>
<td>X</td>
</tr>
<tr>
<td>MC_MoveRelative</td>
<td>X</td>
</tr>
<tr>
<td>MC_MoveAdditive</td>
<td>X</td>
</tr>
<tr>
<td>MC_MoveSuperimposed</td>
<td>X</td>
</tr>
<tr>
<td>MC_HaltSuperimposed</td>
<td>X</td>
</tr>
<tr>
<td>MC_MoveVelocity</td>
<td>X</td>
</tr>
<tr>
<td>MC_MoveContinuousAbsolute</td>
<td>X</td>
</tr>
<tr>
<td>MC_MoveContinuousRelative</td>
<td>X</td>
</tr>
<tr>
<td>MC_Stop</td>
<td>X</td>
</tr>
</tbody>
</table>
### Table 370: Multi-Axis function blocks

<table>
<thead>
<tr>
<th>Function Block</th>
<th>PLC-based motion control</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>CMC_Basic_Kernel</td>
<td>CMC_Estp_Kernel</td>
</tr>
<tr>
<td>MC_PositionProfile</td>
<td>X</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>MC_VelocityProfile</td>
<td>X</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>MC_AccelerationProfile</td>
<td>X</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>MC_Halt</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
</tbody>
</table>

### Table 371: Homing function blocks

<table>
<thead>
<tr>
<th>Function Block</th>
<th>PLC-based motion control</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>CMC_Basic_Kernel</td>
<td>CMC_Estp_Kernel</td>
</tr>
<tr>
<td>MC_Home</td>
<td>-</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>MC_StepAbsSwitch</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>MC_StepLimitSwitch</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>MC_StepRefPulse</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>MC_StepDirect</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
</tbody>
</table>

### Table 372: ABB specific function blocks

<table>
<thead>
<tr>
<th>Function Block</th>
<th>PLC-based motion control</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>CMC_Basic_Kernel</td>
<td>CMC_Estp_Kernel</td>
</tr>
<tr>
<td>MCA_Cam_Extra</td>
<td>X</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>MCA_Home</td>
<td>-</td>
<td>-</td>
<td></td>
</tr>
</tbody>
</table>
### Function Block

<table>
<thead>
<tr>
<th>Function Block</th>
<th>PLC-based motion control</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>CMC_Basic_Kernel</td>
</tr>
<tr>
<td>MCA_Indexing</td>
<td>X</td>
</tr>
<tr>
<td>MCA_JogAxis</td>
<td>X</td>
</tr>
<tr>
<td>MCA_MoveByExternalReference</td>
<td>X</td>
</tr>
<tr>
<td>MCA_MoveVelocityContinuous</td>
<td>X</td>
</tr>
<tr>
<td>MCA_Parameter</td>
<td>X</td>
</tr>
<tr>
<td>MCA_Power</td>
<td>-</td>
</tr>
<tr>
<td>MCA_ReadParameterList</td>
<td>X</td>
</tr>
<tr>
<td>MCA_WriteParameterList</td>
<td>X</td>
</tr>
<tr>
<td>MCA_SetPositionContinuous</td>
<td>X</td>
</tr>
<tr>
<td>MCA_GearInDirect</td>
<td>X</td>
</tr>
<tr>
<td>MCA_CamInDirect</td>
<td>X</td>
</tr>
<tr>
<td>MCA_SetOperatingMode</td>
<td>X</td>
</tr>
<tr>
<td>MCA_CamInfo</td>
<td>X</td>
</tr>
<tr>
<td>MCA_DriveBasedHome</td>
<td>X</td>
</tr>
<tr>
<td>MCA_MoveRelativeOpto</td>
<td>X</td>
</tr>
<tr>
<td>MCA_PhasingByMaster</td>
<td>X</td>
</tr>
</tbody>
</table>

### Overview of libraries

- Add the following libraries for the listed applications.
  - In some cases by adding a library, there will be other libraries added automatically.

<table>
<thead>
<tr>
<th>Application</th>
<th>Library to be added manually</th>
</tr>
</thead>
<tbody>
<tr>
<td>PLC-based motion control</td>
<td>ABB_MotionControl_AC500 compiled-library</td>
</tr>
<tr>
<td></td>
<td>ABB_MathFunctions_AC500 compiled-library</td>
</tr>
<tr>
<td>PLC-based motion control, optional for EtherCAT</td>
<td>ABB_Ecat_CiA402_AC500 library</td>
</tr>
<tr>
<td>Motion control with Eco V3 (CMC_Estp_Kernel)</td>
<td>ABB_MotionControlEco_AC500.library</td>
</tr>
</tbody>
</table>

The features of the Function Blocks provided with PS5611-Motion can be used from the PLC program according to PLCopen standard. Different drives and different Motion Control realizations could be used and can be combined with each other as well as different fieldbusses. ABB_Ecat_CiA402_AC500.library is editable and can be adapted based on the drive configuration and drive type.
Overview of data types

The following data types are used for the Motion Control library. The data types are defined in the library file ABB_MotionControl_AC500 compiled-library. The corresponding elements can be used for the Function Blocks inputs.

Table 373: Structures

<table>
<thead>
<tr>
<th>Data type</th>
<th>Elements</th>
<th>Element data type</th>
</tr>
</thead>
<tbody>
<tr>
<td>CMC_AXIS_IO</td>
<td>limitSwitchPos</td>
<td>BOOL</td>
</tr>
<tr>
<td></td>
<td>limitSwitchNeg</td>
<td>BOOL</td>
</tr>
<tr>
<td></td>
<td>absRefSwitch</td>
<td>BOOL</td>
</tr>
<tr>
<td>MC_PPROFILE</td>
<td>master_position</td>
<td>LREAL</td>
</tr>
<tr>
<td></td>
<td>interpolation_point</td>
<td>LREAL</td>
</tr>
<tr>
<td></td>
<td>velocity_ratio</td>
<td>LREAL</td>
</tr>
<tr>
<td></td>
<td>acceleration_ratio</td>
<td>LREAL</td>
</tr>
<tr>
<td>MC_TPROFILE</td>
<td>interpolation_point</td>
<td>LREAL</td>
</tr>
<tr>
<td></td>
<td>first_derivative</td>
<td>LREAL</td>
</tr>
<tr>
<td></td>
<td>second_derivative</td>
<td>LREAL</td>
</tr>
<tr>
<td></td>
<td>delta_time</td>
<td>TIME</td>
</tr>
</tbody>
</table>

Table 374: Enum

<table>
<thead>
<tr>
<th>Data type</th>
<th>Possible values</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_ABB iTYPES_ENUM</td>
<td>MCA_SPLINE_COMPLETE</td>
</tr>
<tr>
<td></td>
<td>MCA_SPLINE_NATURAL</td>
</tr>
<tr>
<td></td>
<td>MCA_POLY5</td>
</tr>
<tr>
<td></td>
<td>MCA_POLY3</td>
</tr>
<tr>
<td></td>
<td>MCA_LINEAR</td>
</tr>
<tr>
<td>MC_BUFFERMODE</td>
<td>mcABORTING</td>
</tr>
<tr>
<td></td>
<td>mcBUFFERED</td>
</tr>
<tr>
<td></td>
<td>mcBLENDINGlow</td>
</tr>
<tr>
<td></td>
<td>mcBLENDINGprevious</td>
</tr>
<tr>
<td></td>
<td>mcBLENDINGnext</td>
</tr>
<tr>
<td></td>
<td>mcBLENDINGhigh</td>
</tr>
<tr>
<td>MC_DIRECTION</td>
<td>DEFAULT</td>
</tr>
<tr>
<td></td>
<td>POSITIVE</td>
</tr>
<tr>
<td></td>
<td>SHORTEST</td>
</tr>
<tr>
<td></td>
<td>NEGATIVE</td>
</tr>
<tr>
<td></td>
<td>CURRENT</td>
</tr>
<tr>
<td></td>
<td>POSITIVE_STOP</td>
</tr>
<tr>
<td></td>
<td>NEGATIVE_STOP</td>
</tr>
<tr>
<td></td>
<td>CURRENT_STOP</td>
</tr>
<tr>
<td>MC_HOMING_DIRECTION</td>
<td>MC_SwitchNegative</td>
</tr>
<tr>
<td></td>
<td>MC_SwitchPositive</td>
</tr>
<tr>
<td>Data type</td>
<td>Possible values</td>
</tr>
<tr>
<td>---------------------------</td>
<td>------------------------------------------------------</td>
</tr>
<tr>
<td></td>
<td>MC_Positive</td>
</tr>
<tr>
<td></td>
<td>MC_Negative</td>
</tr>
<tr>
<td>MC_HOMING_EDGE</td>
<td>MC_EdgeOn</td>
</tr>
<tr>
<td></td>
<td>MC_EdgeOff</td>
</tr>
<tr>
<td></td>
<td>MC_On</td>
</tr>
<tr>
<td></td>
<td>MC_Off</td>
</tr>
<tr>
<td>MC_HOMING_MODE</td>
<td>MC_REFPULSE</td>
</tr>
<tr>
<td></td>
<td>MC_DIRECT</td>
</tr>
<tr>
<td>MC_SOURCE</td>
<td>mcActualValue</td>
</tr>
<tr>
<td></td>
<td>mcSetValue</td>
</tr>
<tr>
<td>ERROR_ID</td>
<td>MC_Ok</td>
</tr>
<tr>
<td></td>
<td>Wrong_State</td>
</tr>
<tr>
<td></td>
<td>Drive_Problem</td>
</tr>
<tr>
<td></td>
<td>Parameter_Exceeds_Limit</td>
</tr>
<tr>
<td></td>
<td>No_Field_Access</td>
</tr>
<tr>
<td></td>
<td>Bus_Problem</td>
</tr>
<tr>
<td></td>
<td>Abs_Switch_Error</td>
</tr>
<tr>
<td></td>
<td>Timeout</td>
</tr>
<tr>
<td></td>
<td>NAK</td>
</tr>
<tr>
<td></td>
<td>MC_TimeLimitExceeded</td>
</tr>
<tr>
<td></td>
<td>MC_DistanceLimitExceeded</td>
</tr>
<tr>
<td></td>
<td>MC_TorqueLimitExceeded</td>
</tr>
<tr>
<td></td>
<td>Not_Implemented</td>
</tr>
<tr>
<td></td>
<td>ErrorID_POSITION_FOLLOW</td>
</tr>
<tr>
<td></td>
<td>ErrorID_POSSW</td>
</tr>
<tr>
<td></td>
<td>ErrorID_NEGSW</td>
</tr>
<tr>
<td></td>
<td>ErrorID_VELOCITY_FAULT</td>
</tr>
<tr>
<td></td>
<td>ErrorID_INTERPOLATION_FAULT</td>
</tr>
<tr>
<td></td>
<td>ErrorID_WARNING_VELOCITYLIMIT</td>
</tr>
<tr>
<td></td>
<td>ErrorID_WARNING_POSITIONLIMITPOS</td>
</tr>
<tr>
<td></td>
<td>ErrorID_WARNING_POSITIONLIMITNEG</td>
</tr>
<tr>
<td></td>
<td>ErrorID_WARNING_POSITIONOVERRUN</td>
</tr>
<tr>
<td></td>
<td>ErrorID_WARNING_ABORT</td>
</tr>
<tr>
<td></td>
<td>ErrorID_WARNING_MOVEMENT_DIRECTION</td>
</tr>
</tbody>
</table>

**Naming of function blocks and data structures**

**PLCopen**

All Function Blocks named MC_xxx are implemented according to PLCopen definition and follow the PLCopen documentation. They may have additional inputs but according to PLCopen rules.

All Function Blocks named MCA_xxx are implemented corresponding to PLCopen rules with adaptations specific to AC500. They are AC500 specific extensions to the PLCopen library.
**PLC-based motion control**

All Function Blocks named CMC_xxx belong to the implementation of PLC-based Motion Control.

All data types named CMC_xxx belong to the implementation of PLC-based Motion Control.

All data types named MC_xxx are implemented according to PLCopen definition and follow the PLCopen documentation.

All data types named AXIS_xxx exist according to PLCopen definition. The content is ABB specific and not documented.

All Function Blocks named zCMC_xxx belong to the implementation of PLC-based Motion Control. These are not documented and not intended for customer use.

**Auxiliaries**

All Function Blocks named xxx_APP are not write protected and may be modified for adaptations.

### 1.4.2.4.3 PLCopen

Based on application requirements and project specifications, engineers are required to use or select a wide range of Motion Control hardware. In the past, this required unique software to be created for each application even though the functions are the same. PLCopen motion standard provides a way to have standard application libraries that are reusable for multiple hardware platforms. This lowers development, maintenance, and support costs while eliminating confusion. In addition, engineering becomes easier, training costs decrease, and the software is reusable across platforms. Effectively, this standardization is done by defining libraries of reusable components. In this way, the programming is less hardware dependent, the reusability of the application software increases, the cost involved in training and support reduced, and the application becomes scalable across different control solutions. Due to the data hiding and encapsulation, it is usable on different architectures, for instance ranging from centralized to distributed or integrated to networked control. It is not specifically designed for one application, but will serve as a basic layer for ongoing definitions in different areas. As such, it is open to existing and future technologies.

ABB is a member of the PLCopen organization. More information about PLCopen can be read on the [PLCopen website](https://www.plcopen.org).

![PLCopen logo](image)

**Fig. 28: PLCopen Motion Control logo**

Function Blocks according to PLCopen are designed for controlling axes via the language elements consistent with those defined in the IEC 61131-3 standard. It was decided by the task force that it would not be practical to encapsulate all the aspects of one axis into only one Function Block. The retained solution is to provide a set of command-oriented Function Blocks that have a reference to the axis, e.g., the abstract data type Axis, which offers flexibility, ease of use, and reusability.

Implementations based on IEC 61131-3 (for instance via Function Blocks and SFC) will be focused towards the interface (look-and-feel/proxy) of the Function Blocks. This specification does not define the internal operation of the Function Blocks.
PLCopen Motion Control Function Blocks can be used in any IEC 61131-3 programming language. The following picture shows an example of a Function Block used in Function Block Diagram (FBD) language.

![Diagram of HC_MoveAbsolute Function Block](image)

Fig. 29: Command for absolute positioning according to PLCopen standard

Application programs which use the manufacturer independent function blocks according to PLCopen will lead to the following advantages:

- Reuseable software structure for different platforms.
- Programming based on Function Blocks.
- Function Blocks can be used in any IEC 61131-3 language.

All function blocks which are defined by PLCopen will have the following qualities independently to the manufacturer of the motion control system:

- Same inputs/outputs
- Same functional behavior
- Same name

The following parts of the PLCopen motion control definition are completely or partly included in this product:

- Part 1: Function Blocks for Motion Control
- Part 2: Extensions
- Part 3: User Guidelines
- Part 4: Homing Procedures

**Programming guidelines**

This chapter explains some rules on the usage of the libraries and the structure Axis_Ref.

- A general rule is that all Function Blocks which are used for a specific drive are to be used in the same PLC-task. When multitasking is used for the PLC, it is allowed to have different drives in different tasks, but all blocks belonging to a specific drive need to be in the same task. There is no multithreading protection for the Axis_Ref instance.

- If Axis_Ref is used as input on a user defined Function Block or program or function, always use it as VAR_IN_OUT and never use it as VAR_INPUT or VAR_OUTPUT. The reason is that this would
  - Break the consistency and destroy data.
  - Consume a lot of computing power by copying data.

- The "Min update time" update time for the fieldbus, defined under PLC Configuration > Couplers[FIX] > [fieldbus master type] must not exceed the half of the scantime of the PLC-task. For example, if scantime of PLC-task is 5 ms, "Min update time" should not be greater than 2 ms.

- Any instance of a Function Block should be called only once per cycle and in only one specific task.

If the instance is used in several tasks, it has to be checked that is not called several times. Because this could corrupt the handshake from Function Block to Axis_Ref to CMC_Basic_Kernel and vice versa.
- Some PLCopen Function Blocks are only allowed to be called within the same task as the CMC_Basic_Kernel Function Block. This is mentioned in the Function Block descriptions.
- If PLCopen Function Blocks are called from a different task, they cycle time should be at least 2 times the cycle time for CMC_Basic_Kernel Function Block is.

### Axis data type Axis_Ref

The Axis_Ref is a structure that contains information on the corresponding axis. It is used as a VAR_IN_OUT in all Motion Control Function Blocks defined in this document. The content of this structure is implementation dependent and can ultimately be empty. If there are elements in this structure, the supplier shall support the access to them, but this is outside of the scope of this document. The refresh rate of this structure is also implementation dependent. According to IEC 61131-3 it is allowed to switch the Axis_Ref for an active Function Block, for instance from Axis1 to Axis2. However, the behavior of this can vary across different platforms, and is not encouraged to do.

**Axis_Ref data type declaration:**

```plaintext
TYPE Axis_Ref : STRUCT

(Content is implementation dependent)
END_STRUCT
```

### Example

```plaintext
TYPE Axis_Ref : STRUCT
AxisNo: UINT; AxisName: STRING (255);
……..
END_STRUCT
```

### The single axis state diagram

The following diagram normatively defines the behavior of the axis at a high level when multiple motion control Function Blocks are simultaneously activated. This combination of motion profiles is useful in building a more complicated profile or to handle exceptions within a program. (In real implementations there may be additional states at a lower level defined). The basic rule is that motion commands are always taken sequentially, even if the PLC had the capability of real parallel processing. These commands act on the axis' state diagram.

The axis is always in one of the defined states (see diagram below). Any motion command that causes a transition changes the state of the axis and, as a consequence, modifies the way the current motion is computed. The single axis state diagram is an abstraction layer of what the real state of the axis is, comparable to the image of the I/O points within a cyclic (PLC) program. A change of state is reflected immediately when issuing the corresponding motion command.

The response time of immediately is system dependent, coupled to the state of the axis, or an abstraction layer in the software.

The diagram is focused on a single axis. The multiple axis Function Blocks, MC_CamIn, MC_GearIn and MC_Phasing, can be looked at, from a single axis state diagram point of view, as multiple single-axes all in specific states. For instance, the CAM-master can be in the state Continuous Motion. The corresponding slave is in the state Synchronized Motion. Connecting a slave axis to a master axis has no influence on the master axis.

The state Disabled describes the initial state of the axis. In this state the movement of the axis is not influenced by the Function Blocks. The axis feedback is operational. If the MC_Power Function Block is called with Enable=TRUE while being in state Disabled, this either leads to Standstill if there is no error inside the axis, or to ErrorStop if an error exists.
Calling MC_Power with Enable=FALSE in any state, the axis goes to the state Disabled, either directly or via any other state. If a motion generating Function Block controls an axis, while the MC_Power Function Block with Enable=FALSE is called, the motion generating Function Block is aborted (CommandAborted).

The intention of the state ErrorStop is that the axis goes to a stop, if possible. There are no further inputs from Function Blocks accepted until a reset has been done from the ErrorStop state.

The transition Error refers to errors from the axis and axis control, and not from the Function Block instances. These axis errors may also be reflected in the output of the Function Blocks instances errors.

Issuing MC_Home in any other state than StandStill will go to ErrorStop, even if MC_Home is issued from the state Homing itself.

Function Blocks which are not listed in the single axis state diagram do not affect the state of the axis, meaning that whenever they are called the state does not change. They are: MC_ReadStatus; MC_ReadAxisError; MC_ReadParameter; MC_ReadBoolParameter; MC_WriteParameter; MC_WriteBoolParameter; MC_ReadActualPosition and MC_CamTableSelect.

Calling the Function Block MC_Stop in state StandStill changes the state to Stopping and back to Standstill when Execute = FALSE. The state Stopping is kept as long as the input Execute is true. The output Done is set when the stop ramp is finished.
1. In this state ErrorStop or Stopping, all Function Blocks can be called, although they will not be executed, except MC_Reset and Error – they will generate the transition to StandStill or ErrorStop respectively.

2. Power.Enable = TRUE and there is an error in the Axis.

3. Power.Enable = TRUE and there is no error in the Axis.

Visualizations

For usage with the PLCopen Library, a set of visualization objects is defined. These visualizations use the placeholder concept, which means that they could be used in an actual visualization several times and be instantiated by replacing the “placeholder” with an effective data-structure.

Two types of visualizations exist:

- As placeholder, an instance of Axis_Ref should be used. These are named: MC_VISU_Axis_name. Here the name could be statemachine or its actual.
- As placeholder, an instance of the respective PLCopen Function Block should be used. These visualizations are named MC_VISU_FB_name where "name" could be MoveAbsolute or MoveVelocity, so the complete element is named MC_VISU_FB_MoveAbsolute or MC_VISU_FB_MoveVelocity.

The background colour and the colour for the title of each element could be changed. The colours are defined in some global predefined variables in MC_VISU_COLOR_INFORMATION. By changing these values, different colours will be used.

**MC_VISU_COLOR_INFORMATION (GVL)**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Initial</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_VISU_BACKGROUND_COLOR</td>
<td>DWORD</td>
<td>16'FFFF8888</td>
<td>&quot;Color combination for the Grey color&quot;</td>
</tr>
<tr>
<td>MC_VISU_TITLE_COLOR</td>
<td>DWORD</td>
<td>16'FFFF00</td>
<td>&quot;Color combination for yellow&quot;</td>
</tr>
</tbody>
</table>

Below, some existing visualizations are shown.

**MC_VISU_Axis_StateMachine** This shows the statemachine of the axis according to PLCopen definition. The active state is shown green except the ErrorStop which is shown red. Usually, it starts with Disabled. When no remote connection to the drive is available, it will switch to ErrorStop immediately.

The Placeholder of this visualization has to be connected to an instance of the data type Axis_Ref.

**MC_VISU_Axis_actual** This object shows some actual values.

The Placeholder of this visualization has to be connected to an instance of the data type Axis_Ref.
**MC_VISU_Axis_FB_error**  
This object shows the error information connected to the PLCopen Function Blocks. This is NOT a drive error. If no error occurs in the execution of a Function Block, just the name is shown. If an error occurred, it shows the name of the Function Block as well as the error number and a short description. In the example below, the MC_Power Function Block recognized that no fieldbus connection to the drive was available.

The Placeholder of this visualization has to be connected to an instance of the data type Axis_Ref.

### Error codes

Besides the diagnosis information of the drive which is described in the respective drive documentation, there are a number of error codes directly related to the Function Blocks. These error codes are displayed at the output “ErrorID” of the Function Block.

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Mnemonic</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>MC_Ok</td>
<td>No Error</td>
</tr>
<tr>
<td>1</td>
<td>WRONG_STATE</td>
<td>A Function Block was activated not according to the statemachine, e.g. tried to start a movement while in state Disabled.</td>
</tr>
<tr>
<td>2</td>
<td>DRIVE_PROBLEM</td>
<td>The drive left the expected state, e.g. tripped.</td>
</tr>
<tr>
<td>3</td>
<td>PARAMETER_EXCEEDS_LIMIT</td>
<td>A parameter at the Function Block is outside the possible range. This does not refer to the parameter range which is allowed for the drive but just to the 32-Bit Integer which is used for internal calculation.</td>
</tr>
<tr>
<td>4</td>
<td>NO_FIELD_ACCESS</td>
<td>The field bus connection is faulty.</td>
</tr>
<tr>
<td>5</td>
<td>BUS_PROBLEM</td>
<td>Not used</td>
</tr>
<tr>
<td>6</td>
<td>ABS_SWITCH_ERROR</td>
<td>During Homing (when done by Function Blocks) a switch was wrong, e.g. the positive switch occurred when moving in negative direction.</td>
</tr>
<tr>
<td>Error Code</td>
<td>Mnemonic</td>
<td>Explanation</td>
</tr>
<tr>
<td>------------</td>
<td>-----------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>7</td>
<td>TIMEOUT</td>
<td>Any activity was not completed within an appropriate time.</td>
</tr>
<tr>
<td>8</td>
<td>NAK</td>
<td>Parameter access not applicable</td>
</tr>
<tr>
<td>9</td>
<td>MC_TimeLimitExceeded</td>
<td>Used by Function Blocks with TimeLimit.</td>
</tr>
<tr>
<td>10</td>
<td>MC_DistanceLimitExceeded</td>
<td>Used by Function Blocks with DistanceLimit.</td>
</tr>
<tr>
<td>11</td>
<td>MC_TorqueLimitExceeded</td>
<td>Used by Function Blocks with TorqueLimit.</td>
</tr>
<tr>
<td>12</td>
<td>NOT_IMPLEMENTED</td>
<td>Function Block is not implemented for the specific drive/motion application.</td>
</tr>
<tr>
<td>101</td>
<td>ErrorID_POSITION_FOLLOW</td>
<td>Following error, caused by &gt; position error =&gt; ERRORSTOP. (parameter POS_LAG_PERCENTAGE)</td>
</tr>
<tr>
<td>102</td>
<td>ErrorID_POSSW</td>
<td>Positive software limit switch =&gt; ERRORSTOP. The actual position did exceed the positive Software limit switch position. This supervision has to be activated with MC_WriteParameter.</td>
</tr>
<tr>
<td>103</td>
<td>ErrorID_NEGSW</td>
<td>Negative software limit switch =&gt; ERRORSTOP. The actual position did exceed the negative Software limit switch position. This supervision has to be activated with MC_WriteParameter.</td>
</tr>
<tr>
<td>104</td>
<td>ErrorID_VELOCITY_FAULT</td>
<td>The measured velocity and commanded velocity are &gt; 50% (related to maximum velocity) apart, for a certain time =&gt;ERRORSTOP (parameter V_CHECKTIME)</td>
</tr>
<tr>
<td>105</td>
<td>ErrorID_INTERPOLATION_FAULT</td>
<td>following error, caused by interpolation problem =&gt;ERRORSTOP. Position following error occurred, but reason most likely a interpolation problem, not drive problem (e.g. CAM Table, position step).</td>
</tr>
<tr>
<td>110</td>
<td>ErrorID_WARNING_VELOCITYLIMIT</td>
<td>Velocity or acceleration/deceleration are in limitation, set by parameter EnableLimitVelocity, MaxVelocityAppl, MaxDecelerationAppl</td>
</tr>
<tr>
<td>111</td>
<td>ErrorID_WARNING_POSITIONLIMITPOS</td>
<td>Position is in limitation towards position limit (SWLimit2DecPos), axis decelerates near positive software limit switch</td>
</tr>
<tr>
<td>112</td>
<td>ErrorID_WARNING_POSITIONLIMITNEG</td>
<td>Position is in limitation towards position limit (SWLimit2DecNeg), axis decelerates near negative software limit switch</td>
</tr>
<tr>
<td>113</td>
<td>ErrorID_WARNING_POSITIONOVERRUN</td>
<td>A linear axis created a 32bit position overrun (&gt; 2147483647 u=&gt;inc) =&gt;configure modulo</td>
</tr>
<tr>
<td>114</td>
<td>ErrorID_WARNING_ABORT</td>
<td>Axis aborted due to too large position gap due to velocity limitation</td>
</tr>
<tr>
<td>115</td>
<td>ErrorID_WARNING_MOTION_DIRECTION</td>
<td>Either positive or negative direction blocked by MC_Power</td>
</tr>
</tbody>
</table>
Error handling

All access to the drive/motion control is via Function Blocks. Internally these Function Blocks provide basic error checking on the input data. Exactly, how this is done is implementation dependent. For instance, if MaxVelocity is set to 6000, and the Velocity input to a Function Block is set to 10,000, a basic error report is generated. In the case where an intelligent drive is coupled via a network to the system, the MaxVelocity parameter is probably stored on the drive. The Function Block must take care of the errors generated by the drive internally. With another implementation, the MaxVelocity value could be stored locally. In this case the Function Block will generate the error locally.

Both centralized and decentralized error handling methods are possible when using the motion control Function Blocks.

Centralized error handling is used to simplify programming of the Function Block. Error reaction is the same independent of the instance in which the error has occurred.

Decentralized error handling gives the possibility of different reactions depending on the Function Block in which an error occurred.
Fig. 32: Function Blocks with decentralized error handling

PLCopen parameter

Additional parameters are available by ReadParameter and WriteParameter Function Blocks.

<table>
<thead>
<tr>
<th>Parameter number (PN)</th>
<th>Name</th>
<th>Data-type</th>
<th>Min.</th>
<th>Max.</th>
<th>Default</th>
<th>R/W</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>CommandedPosition</td>
<td>DINT</td>
<td></td>
<td></td>
<td>R</td>
<td></td>
<td>Commanded position.</td>
</tr>
<tr>
<td>2</td>
<td>SWLimitPos</td>
<td>DINT</td>
<td>-2147483647</td>
<td>2147483647</td>
<td>214748</td>
<td>R/W</td>
<td>Positive Software limit switch position.</td>
</tr>
<tr>
<td>Parameter number (PN)</td>
<td>Name</td>
<td>Datatype</td>
<td>Min.</td>
<td>Max.</td>
<td>Default</td>
<td>R/W</td>
<td>Comments</td>
</tr>
<tr>
<td>-----------------------</td>
<td>--------------------------</td>
<td>----------</td>
<td>-------------</td>
<td>-------------</td>
<td>---------</td>
<td>-----</td>
<td>---------------------------------------------</td>
</tr>
<tr>
<td>3</td>
<td>SWLimitNeg</td>
<td>DINT</td>
<td>-2147483647</td>
<td>2147483647</td>
<td>-2147483647</td>
<td>R/W</td>
<td>Negative Software limit switch position.</td>
</tr>
<tr>
<td>4</td>
<td>EnableLimitPos</td>
<td>BOOL</td>
<td>FALSE</td>
<td>TRUE</td>
<td>FALSE</td>
<td>R/W</td>
<td>Enable positive software limit switch.</td>
</tr>
<tr>
<td>5</td>
<td>EnableLimitNeg</td>
<td>BOOL</td>
<td>FALSE</td>
<td>TRUE</td>
<td>FALSE</td>
<td>R/W</td>
<td>Enable negative software limit switch.</td>
</tr>
<tr>
<td>6</td>
<td>EnablePosLagMonitoring</td>
<td>BOOL</td>
<td>FALSE</td>
<td>TRUE</td>
<td>TRUE</td>
<td>R/W</td>
<td>Enable monitoring of position lag (following error).</td>
</tr>
<tr>
<td>7</td>
<td>MaxPositionLag</td>
<td>DINT</td>
<td>1</td>
<td>2147483647</td>
<td>***</td>
<td>R</td>
<td>Maximal position lag.</td>
</tr>
<tr>
<td>8</td>
<td>MaxVelocitySystem</td>
<td>DINT</td>
<td></td>
<td>32767</td>
<td>R</td>
<td></td>
<td>Maximal allowed velocity of the axis in the motion system.</td>
</tr>
<tr>
<td>9</td>
<td>MaxVelocityAppl</td>
<td>DINT</td>
<td>0***</td>
<td>32767</td>
<td>32767</td>
<td>R/W</td>
<td>Maximal allowed velocity of the axis in the application.</td>
</tr>
<tr>
<td>10</td>
<td>ActualVelocity</td>
<td>DINT</td>
<td>-32767</td>
<td>32767</td>
<td>R</td>
<td></td>
<td>Actual velocity.</td>
</tr>
<tr>
<td>11</td>
<td>CommandedVelocity</td>
<td>DINT</td>
<td>-32767</td>
<td>32767</td>
<td>R</td>
<td></td>
<td>Commanded velocity.</td>
</tr>
<tr>
<td>12</td>
<td>MaxAccelerationSystem</td>
<td>DINT</td>
<td></td>
<td>32767</td>
<td>R</td>
<td></td>
<td>Maximal allowed acceleration of the axis in the motion system.</td>
</tr>
<tr>
<td>13</td>
<td>MaxAccelerationAppl</td>
<td>DINT</td>
<td>10</td>
<td>32767</td>
<td>32767</td>
<td>R/W</td>
<td>Maximal allowed acceleration of the axis in the application.</td>
</tr>
<tr>
<td>14</td>
<td>MaxDecelerationSystem</td>
<td>DINT</td>
<td></td>
<td>32767</td>
<td>R</td>
<td></td>
<td>Maximal allowed deceleration of the axis.</td>
</tr>
<tr>
<td>15</td>
<td>MaxDecelerationAppl</td>
<td>DINT</td>
<td>10</td>
<td>32767</td>
<td>32767</td>
<td>R/W</td>
<td>Maximal allowed deceleration of the axis.</td>
</tr>
<tr>
<td>16</td>
<td>MaxJerk</td>
<td>DINT</td>
<td>0*</td>
<td>2147483647</td>
<td>2147483647</td>
<td>R/W</td>
<td>Maximal allowed jerk of the axis.</td>
</tr>
<tr>
<td>Parameter number (PN)</td>
<td>Name</td>
<td>Data-type</td>
<td>Min.</td>
<td>Max.</td>
<td>Default</td>
<td>R/W</td>
<td>Comments</td>
</tr>
<tr>
<td>----------------------</td>
<td>-----------------------</td>
<td>-----------</td>
<td>------</td>
<td>------------</td>
<td>---------</td>
<td>-----</td>
<td>---------------------------------------------------------------------------</td>
</tr>
<tr>
<td>2001</td>
<td>MODULO_NO_MINATOR</td>
<td>DINT</td>
<td>1</td>
<td>2147483647</td>
<td>1</td>
<td>R/W</td>
<td>ABB specific parameter. Used for PLC-based Motion Control implementation: Gearbox modifier to MODULO_RANGE</td>
</tr>
<tr>
<td>2002</td>
<td>MODULO_DEN_MINATOR</td>
<td>DINT</td>
<td>1</td>
<td>2147483647</td>
<td>1</td>
<td>R/W</td>
<td>ABB specific parameter. Used for PLC-based Motion Control implementation: Gearbox modifier to MODULO_RANGE</td>
</tr>
<tr>
<td>2003</td>
<td>Enable-Limit2Decelerate</td>
<td>BOOL</td>
<td>FALSE</td>
<td>TRUE</td>
<td>FALSE</td>
<td>R/W</td>
<td>Enable software limit switches to decelerate</td>
</tr>
<tr>
<td>2004</td>
<td>Enable-LimitAbort</td>
<td>BOOL</td>
<td>FALSE</td>
<td>TRUE</td>
<td>FALSE</td>
<td>R/W</td>
<td>Enable that software limit switches will abort ongoing movement. FALSE = Limits position and velocity, decelerates and shows a warning until the position limit is reached, then ERROR STOP TRUE = Switches off any ongoing motion and decelerates to the position limit, then ERROR STOP</td>
</tr>
<tr>
<td>2005</td>
<td>Enable-LimitVelocity</td>
<td>BOOL</td>
<td>FALSE</td>
<td>TRUE</td>
<td>FALSE</td>
<td>R/W</td>
<td>If the velocity is limited the unmoved position will be covered whenever possible</td>
</tr>
<tr>
<td>2006</td>
<td>SWLimit2DecPos</td>
<td>LREAL</td>
<td>-2147483647</td>
<td>2147483647</td>
<td>2147483647</td>
<td>R/W</td>
<td>Used as end position for EnableLimit2Decelerate</td>
</tr>
</tbody>
</table>
### Parameter number (PN)

<table>
<thead>
<tr>
<th>Parameter number</th>
<th>Name</th>
<th>Data-type</th>
<th>Min.</th>
<th>Max.</th>
<th>Default</th>
<th>R/W</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>2007</td>
<td>SWLimit2DecNeg</td>
<td>LREAL</td>
<td>-2147483647</td>
<td>2147483647</td>
<td>214748 3647</td>
<td>R/W</td>
<td>Used as end position for EnableLimit2Decel-erate</td>
</tr>
<tr>
<td>2008</td>
<td>MaxPosition-GapLL</td>
<td>LREAL</td>
<td>0</td>
<td>2147483647</td>
<td>00</td>
<td>R/W</td>
<td>Used to stop the ongoing movement if position is behind</td>
</tr>
</tbody>
</table>

0* means: no limitation of jerk is performed.

**Axis will stay in stop.

***is modified by CMC_AXIS_CONTROL_PARAMETER, the max. Value is calculated in increments, the value which is delivered by ReadParameter will be given in [u].

In addition to the above parameters certain other operation can be done using the below parameters from the data type "Axis_Parameter"

### Name | Type | Initial | Comment
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>paraFilterVariant</td>
<td>INT</td>
<td>0</td>
<td>Filter for actual velocity&lt;br&gt;0 = PT1&lt;br&gt;1 = LinearRegression</td>
</tr>
<tr>
<td>paraFilterTime</td>
<td>INT</td>
<td>10</td>
<td>Time in PLC cycles, used with paraFilter-Variant</td>
</tr>
<tr>
<td>paraFilterForecast</td>
<td>INT</td>
<td>0</td>
<td>Time in PLC cycles, used with paraFilter-Variant = 1</td>
</tr>
<tr>
<td>paraReverseDirection</td>
<td>INT</td>
<td>0</td>
<td>Changes the direction for actual and reference positions based on the mode selected.&lt;br&gt;0 = normal direction&lt;br&gt;1 = reverse input position&lt;br&gt;2 = reverse output position and speed reference&lt;br&gt;3 = reverse both</td>
</tr>
</tbody>
</table>
Limits

Table 375: Limitations for the inputs of PLCopen Function Blocks when used with CMC_Basic_Kernel

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Min.</th>
<th>Max.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Velocity</td>
<td>0</td>
<td>x</td>
</tr>
<tr>
<td>Acceleration, Deceleration</td>
<td>0</td>
<td>x</td>
</tr>
<tr>
<td>Position</td>
<td>-2147483647</td>
<td>2147483647</td>
</tr>
</tbody>
</table>

General restrictions

Restrictions for the available function blocks

- As buffered mode, MC_Aborting is realized as a default. This does NOT mean that the axis stops when another movement is started while an ongoing movement is still active. It means instead that the new movement will take control immediately and change the velocity to its own velocity by using its own acceleration or deceleration.
- The buffered mode MC_Buffered could be reached with using the axis state StandStill as enable signal for the Execute of the next block.
- From the Extended Inputs and Outputs at the Function Blocks, the following are not realized:
  - BufferedMode: The realization just supports the MC_Aborting mode.
  - The following Outputs at ReadStatus are not supported: ConstantVelocity, Accelerating and Decelerating.
  - TorqueLimit for Homing Function Blocks.
**MC_Aborting Mode**

The diagram shows the behavior with BufferMode MC_Aborting, which is the only available BufferMode. When the second Block is activated, it will take control and will continue on its own velocity. The velocity is changed by using the acceleration value from the second Function Block. The movement will not be stopped in between. The first Function Block shows CommandAborted when the second Function Block is activated.

**MC_Buffered**

A behavior according to BufferMode MC_Buffered could be reached by using the Done output from the first Function Block to enable the Execute of the second Function Block.

### Behavior of the function block inputs and outputs

#### General rules

**Table 376: General rules**

| Output exlusivity | The outputs Busy, Done, Error, and CommandAborted are mutually exclusive:
<p>|                  | Only one of them can be TRUE on one Function Block. If Execute is TRUE, one of these outputs has to be TRUE. Only one of the outputs Active, Error, Done and CommandAborted is set at the same time. |
| Output status     | The outputs Done, InGear, InSync, InVelocity, Error, ErrorID and CommandAborted are reset with the falling edge of Execute. However, the falling edge of Execute does not stop or even influence the execution of the actual Function Block. It must be guaranteed that the corresponding outputs are set for at least one cycle if the situation occurs, even if execute was reset before the Function Block completed. If an instance of a Function Block receives a new execute before it has finished (as a series of commands on the same instance), the Function Block will not return any feedback, like Done or CommandAborted, for the previous action. |
| Input parameters  | The parameters are used with the rising edge of the execute input. To modify any parameter, it is necessary to change the input parameter(s) and to trigger the motion again. |
| Missing input parameters | According to IEC 61131-3, if any parameter of a Function Block input is missing (open) then the value from the previous invocation of this instance will be used. In the first invocation the initial value is applied. |
| Position versus distance | Position is a value defined within a coordinate system. Distance is a relative measure related to technical units. Distance is the difference between two positions. |
| Sign rules | Velocity, Acceleration, Deceleration and Jerk are always positive values. Position and Distance can be both positive and negative. |
| Error Handling Behavior | All Function Blocks have two outputs, which deal with errors that can occur while executing that Function Block. These outputs are defined as follow: |
|                  | Error Rising edge of Error informs that an error occurred during the execution of the Function Block. |
|                  | ErrorID: Error number |</p>
<table>
<thead>
<tr>
<th><strong>Function Block Naming</strong></th>
<th>In case of multiple libraries within one system (to support multiple drive/motion control systems), the Function Block naming may be changed to MC_FunctionBlockName_SupplierID.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Behavior of Done output</strong></td>
<td>The outputs Done, InGear, InSync... are set when the commanded action has been completed successfully. With multiple Function Blocks working on the same axis in a sequence, the following applies: When one movement on an axis is interrupted with another movement on the same axis without having reached the final goal, Done of the first Function Block will not be set.</td>
</tr>
<tr>
<td><strong>Behavior of CommandAborted output</strong></td>
<td>CommandAborted is set, when a commanded motion is interrupted by another motion command. The reset-behavior of CommandAborted is like that of Done. When CommandAborted occurs, the other output-signals such as InVelocity are reset.</td>
</tr>
<tr>
<td><strong>Inputs exceeding application limits</strong></td>
<td>If a Function Block is commanded with parameters which result in a violation of application limits, the instance of the Function Block generates an error. The consequences of this error for the axis are application specific and thus should be handled by the application program.</td>
</tr>
<tr>
<td><strong>Behavior of Busy output</strong></td>
<td>Every Function Block can have an output Busy, reflecting that the Function Block is not finished. Busy is SET at the rising edge of Execute and RESET when one of the outputs Done, Aborted, or Error is set. It is recommended that this Function Block should be kept in the active loop of the application program for at least as long as Busy is true, because the outputs may still change. For one axis, several Function Blocks might be busy, but only one can be active at a time. Exceptions are MC_SuperImposed and MC_Phasing, where more than one Function Block related to one axis can be active.</td>
</tr>
<tr>
<td><strong>Output Active</strong></td>
<td>The output Active is required on buffered Function Blocks. This output is set at the moment the Function Block takes control of the motion of the according axis. For un-buffered mode the outputs Active and Busy can have the same value.</td>
</tr>
<tr>
<td><strong>Enable and Valid/Status</strong></td>
<td>The input Enable is coupled to output Valid. Enable is level sensitive, and Valid shows that a valid set of outputs is available at the Function Block. The output Valid is TRUE as long as an output value of Valid is available and the input Enable is TRUE. The relevant output value can be refreshed as long as the input Enable is TRUE. If there is a Function Block error, the output is not Valid (Valid set to FALSE). When the error condition disappears, the values will reappear and output Valid will be set again.</td>
</tr>
</tbody>
</table>
Why is the command input edge sensitive?

The input Execute for the different Function Blocks described in this document always triggers the function with its rising edge. The reason for this is that with edge triggered Execute new input values may be commanded during execution of a previous command. The advantage of this method is a precise management of the instant a motion command is performed. Combining different Function Blocks is then easier in both centralized and decentralized models of axis management. The output Done can be used to trigger the next part of the movement. The example given below is intended to explain the behavior of the Function Block execution.

The following figure illustrates the sequence of three Function Blocks First, Second and Third controlling the same axis. These three Function Blocks could be for instance various absolute or relative move commands. When First is completed the motion its rising output First.Done triggers Second.Execute. The output Second.Done AND In13 triggers the Third.Execute.
The input ContinuousUpdate

Like described in the previous chapter, the input Execute triggers a new movement. With a rising edge of this input the values of the other Function Block inputs are defining the movement. Until version 1.1 of PLCopen there was the general rule that a later change in these input parameters does not affect the ongoing motion.

Nevertheless, there are numerous application examples, where a continuous change of the parameters are needed. The user could retrigger the input Execute of the Function Block, but this complicated the application.

Therefore, the input ContinuousUpdate has been introduced. It is an extended input to all applicable Function Blocks. If it is TRUE, when the Function Block is triggered (rising Execute), it will - as long as it stays TRUE – make the Function Block use the current values of the input variables and apply it to the ongoing movement. This does not influence the general behavior of the Function Block nor does it impact the single axis state diagram. In other words it only influences the ongoing movement and its impact ends as soon as the Function Block is no longer Busy or the input ContinuousUpdate is set to FALSE.
It can be that certain inputs like BufferMode are not really intended to change every cycle. However, this has to be dealt with in the application, and is not forbidden in the specification.

If ContinuousUpdate is FALSE with the rising edge of the input Execute, a change in the input parameters is ignored during the whole movement and the original behavior of previous versions is applicable. The ContinuousUpdate is not a retriggering of the input Execute of the Function Block. A retriggering of a Function Block which was previously aborted, stopped, or completed, would regain control on the axis and modify its single axis state diagram. Opposite to this, the ContinuousUpdate only effects an ongoing movement. Also, a ContinuousUpdate of relative inputs (e.g. Distance in MC_MoveRelative) always refers to the initial condition (at rising edge of Execute).

Example

- MC_MoveContinuousRelative is started at Position 0 with Distance 100, Velocity 10 and ContinuousUpdate set TRUE. Execute is Set and so the movement is started to position 100.
- While the movement is executed (let the drive be at position 50), the input Distance is changed to 130, Velocity 20.
- The axis will accelerate (to the new Velocity 20) and stop at Position 130 and set the output Done and does not accept any new values.

Unit of length

The only specification for physical quantities is made on the unit of length (noted as [u]) that is to be coherent with its derivatives i.e. (velocity [u/s]; acceleration [u/s²]; jerk [u/s³]). Nevertheless, the unit [u] is not specified (manufacturer dependent). Only its relations with others are specified.

Aborting versus buffered modes

Some of the Function Blocks have an input called BufferMode. With this input, the Function Block can either work in a Non-buffered mode (default behavior) or in a Buffered mode. The difference between those modes is when they should start their action:

- A command in a non-buffered mode acts immediately, even if this interrupts another motion,
- A command in a buffered mode waits till the current Function Block sets its output Done (or InPosition, InVelocity...).
- The library just supports the mode "aborting" (MCAborting)

The following examples describe the different behavior of these modes:
Example 1:
Standard behaviour of two following absolute movements

Fig. 35: Basic example with two MC_MoveAbsolute on same axis
Fig. 36: Timing diagram for example above without interference between Function Block 1 and Function Block 2
Example 2: Aborting motion

<table>
<thead>
<tr>
<th>Axis</th>
<th>Execute</th>
<th>Done</th>
<th>Done_1</th>
<th>Start_2</th>
<th>MC_MoveAbsolute</th>
<th>Axis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Start_1</td>
<td>Position</td>
<td>CommandAborted</td>
<td>CA_1</td>
<td>2000</td>
<td>MC_MoveAbsolute</td>
<td>Axis</td>
</tr>
<tr>
<td>1000</td>
<td>Velocity</td>
<td>Busy</td>
<td>Busy_1</td>
<td>50</td>
<td>Velocity</td>
<td>Busy</td>
</tr>
<tr>
<td>100</td>
<td>Acceleration</td>
<td>Active</td>
<td>Active_1</td>
<td>50</td>
<td>Acceleration</td>
<td>Active</td>
</tr>
<tr>
<td>100</td>
<td>Deceleration</td>
<td>Error</td>
<td>ErrorID</td>
<td>50</td>
<td>Deceleration</td>
<td>Error</td>
</tr>
<tr>
<td>Aborting</td>
<td>Direction</td>
<td>BufferMode</td>
<td>Aborting</td>
<td>BufferMode</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Fig. 37: Timing diagram for example above with Function Block 2 interrupting Function Block 1 (McAbort Mode)
If an on-going motion is aborted by another movement, it can occur that the braking distance is not sufficient due to deceleration limits.

In rotary axis, a modulo can be added. A modulo axis could go to the earliest repetition of the absolute position specified, in cases where the axis should not change direction and reverse to attain the target position.

In linear systems, the resulting overshoot can be resolved by reversing, as each position is unique and therefore there is no need to add a modulo to reach the correct position.
Examples

Example: A function block instance controls different motions of an axis

The following figure shows an example where the Function Block 1 is used to control AxisX with three different values of Velocity. In a Sequential Function Chart (SFC) the velocity 10, 20, and 0 is assigned to V. To trigger the input Execute with a rising edge the variable E is step-wise set and reset.

Fig. 38: Single Function Block with SFC

The following timing diagram explains how it works:
Fig. 39: Timing diagram for a usage of single Function Block

The second InVelocity is set for only one cycle because the Execute has gone low before the ActualVelocity equals CommandedVelocity.
Different instances related to the same axis can control the motions on an axis. Each instance will then be responsible for one part of the global profile.

**Example: Different function blocks instances control the motions of an axis**

![Cascaded Function Blocks Diagram]

*Fig. 40: Cascaded Function Blocks*

The timing diagram:
Fig. 41: Cascaded Function Blocks timing diagram
A corresponding solution written in LD looks like:

Fig. 42: Cascaded Function Blocks with LD
1.4.2.4 PLC-based motion control

PLC-based motion control architecture

With the Function Blocks of motion library a Motion Control profiler can be used inside the PLC. As shown in the following figure it is needed to provide the actual position of the drive. The output can be either a position or a velocity reference signal. The used output signal will then be used to move the axis in the desired way.

![Diagram of Compact Motion Kernel Function Block](image)

There are 2 possibilities to send a reference value to the drive:

- When the position control loop is closed by the PLC by a CMC_Basic_Kernel Function Block, the output Speed_Reference should be connected to the drive. The value of Speed_Reference can be scaled with the axis parameters Max_Rpm and Ref_Max.
- When the position control loop is closed by the drive, the output Position_Reference should be connected to the drive. The unit for the output Position_Reference is incremented as well as the input Drive_ActualPosition.

![Diagram of Application and Implementation Layers](image)

Fig. 43: Architecture for centralized Motion Control

In general the programming of a machine consists of two layers as shown in the figure above. In the application layer Function Blocks according to PLCopen Motion Control are used to program the application sequences with all necessary types of movements and administrative commands. Due to the standard PLCopen Motion Control this can be reused in any other machine programs that used PLCopen Function Blocks.
The axis implementation layer is responsible for the execution of the commands from the application layer and can be programmed for each axis in a different way depending on the used hardware components.

### Table 377: Needed function blocks for an application with PLC-based Motion Control

<table>
<thead>
<tr>
<th>Library</th>
<th>Content</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABB_MotionControl_AC500.library</td>
<td>Kernel Function Block, Parameters Function Block, Axis Simulation Function Block</td>
</tr>
<tr>
<td></td>
<td>Data types for AC500 Motion Control</td>
</tr>
<tr>
<td></td>
<td>Motion Control Function Blocks according to PLCopen</td>
</tr>
</tbody>
</table>

*For a central motion axis implementation the use of the Function Blocks CMC_Basic_Kernel and CMC_Axis_Control_Parameter are mandatory.*

The library design is independent from any bus architecture or any specific drive features.

<table>
<thead>
<tr>
<th>Example for a possible system architecture</th>
<th>System</th>
<th>Velocity reference</th>
<th>Position feedback</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>System A</td>
<td>Output via analog output channel as voltage or current</td>
<td>From incremental encoder connected to CD522 IO module</td>
</tr>
<tr>
<td></td>
<td>System B</td>
<td>Output via EtherCAT network</td>
<td>Input via EtherCAT network</td>
</tr>
<tr>
<td></td>
<td>System C</td>
<td>Output as frequency signal of CD522 IO module</td>
<td>From incremental encoder connected to CD522 IO module</td>
</tr>
<tr>
<td></td>
<td>System D</td>
<td>Output via PROFINET IO network</td>
<td>Input via PROFINET IO network</td>
</tr>
</tbody>
</table>

In case the velocity reference value is used from the kernel Function Block the position control loop is closed inside the drive. In this case, it is necessary to adjust the related parameters from the parameters Function Block. When the position reference will be used the position control loop is closed inside the drive. In this case, the internal control loop is just used to monitor the position and velocity.

*When the position reference is used for the drive the following aspects have to be taken care of:*
  - It is necessary to use a real time fieldbus, like EtherCAT.
  - The PLC cycle has to be synchronized to the fieldbus cycle.
  - The task calculation times may not exceed the used cycle time.

The drive’s status should be managed by a specialized Function Block that supports the used type of drive as shown in the figure above. The kernel Function Block is the main Function Block which is needed to operate an axis with PLC-based Motion Control. It must be used with the parameter Function Block which is the interface to input parameters which are used to setup the axis.
The drive has to be accessed outside the CMC_Basic_Kernel Function Block. Actual values and reference values might be transferred by a synchronized fieldbus or by I/Os. The Function Block CMC_Basic_Kernel has to be called every cycle and at least once before any Function Block MC... is activated.

The following figure shows an example with an axis simulation. The main data signals are drawn in bold lines. Here, the drive will receive a speed reference signal which means that the position control loop is closed inside the PLC by the Compact Motion Function Blocks. The time behaviour of the simulated drive can be set by the parameter T1 at the axis simulation Function Block. If the time constant is too slow and the axis parameter Control_Time is too short the simulated axis will run into instability – like a real drive. Sample values:

A different option to create a virtual or simulated axis is to engage the Enable_Virtual input at CMC_Basic_Kernel. This virtual axis will follow the speed reference without additional delay, whereas the CMC_Axis_Simu creates a first order delay.
The following figure shows an example with a CiA402 drive on an EtherCAT network. The main data signals are drawn in bold lines. Here, the drive will receive a position reference signal which means that the position control loop is closed inside the drive.

In this example the main signals are to be transferred via EtherCAT network. The drive control Function Block for the Microflex e190 can be found in the ABB_Ecat_CiA402_AC500.library.

**Kernel function block**

**Kernel Arithmetic**

The “KERNEL” Function Blocks are available in two variants.

The CMC_Estp_Kernel block is solely to be used in eco-V3 CPUs and to make use of the integrated stepper-IO. It connects automatically to the internal IOs.

The CMC_Basic_Kernel block is designed to be used in any V3 PLCs and can either work with drives connected to a fieldbus or IOs.
How does the parameter for jerk influence the axis movements

The diagram shows the result with different jerk values and the same velocity and acceleration. The time needed for acceleration with jerk=0 is:

\[ \text{Time1}=\frac{\text{velocity}}{\text{acceleration}}=(20/100)\text{s}=0.2\text{s} \]

The additional time with jerk=500 will be:

\[ \text{Time2}=\frac{\text{acceleration}}{\text{jerk}}=(100/500)\text{s}=0.2\text{s} \]

So the total time is:

\[ \text{Time}=\text{Time1}+\text{Time2}=0.2\text{s}+0.2\text{s}=0.4\text{s} \]

In the last example with jerk=100, the velocity and acceleration values are not reached.

Basic functionalities

How to connect a drive

The connection to a drive must be done with the inputs and outputs of the Function Block CMC_Basic_Kernel. All inputs and outputs of the kernel Function Block with the prefix “Drive_” are intended to be used with a drive, but in some cases not all of them are needed. In all cases the input Drive_ActualPosition has to be connected with the actual position of the axis. This value can be received by an IO module of the PLC or via a fieldbus.
Depending on which device closes the position control loop either the output Speed_Reference or Position_Reference output has to be used. The value of Speed_Reference can be connected to an analog output module or be transferred via a fieldbus. The value of Position_Reference should be exclusively sent via a real-time fieldbus like EtherCAT.

Example 1: Analog drive - Motor with incremental encoder

In this example the position control loop will be closed by the PLC, therefore the input Drive_ActualPosition and the output Speed_Reference are to be used.

In combination with the IO module CD522 and the corresponding Function Block CD522Encoder32Bit the position of the encoder can be used. For the effective resolution of the encoder parameter Inc_Per_R of the parameter Function Block has to be used.

The output Speed_Reference can be written directly to the global variable of an output channel of an analog module but can also be transferred via a fieldbus. The scaling of this output value can be done with the parameters Ref_Max and Max_Rpm of the Function Block CMC_Axis_Control_Parameter_Rial.

The scaling of the Speed_Reference value can be set with the inputs Ref_Max and Max_Rpm of the parameter Function Block.

In order to finish a homing sequence which is done by the Function Block MC_StepRefPulse the outputs Drive_Set_Ref and Drive_Set_Position from the kernel Function Block have to be connected with the inputs EN_RPI and START_VALUE of the CD552 IO module Function Block. Also the output RdyRpi of the CD552 IO module Function Block has to be connected with Drive_Ref_Ok from the kernel Function Block.

To enable and disable the drive Drive_Release could be connected to a binary output to activate the drive. Drive_InOperation could be connected to a binary input to get the information that Drive_Release was successful.
In this example the position control loop will be closed by the drive, therefore the input 
Drive_ActualPosition and the output Position_Reference are to be used. The inputs referring to 
the position control loop of the parameter Function Block do not have to be set.

For the effective resolution of the motor’s encoder parameter Inc_Per_R of the parameter 
Function Block has to be adjusted.

To enable and disable the drive Drive_Release and Drive_InOperation have to be connected to 
the control Function Block ECAT_CiA402_Control_App of the library 
ABB_Ecat_CiA402_AC500.library, which controls the status and control word of the drive.

All Function Blocks from this library are not password protected and free to be changed in 
order to be adapted for different drives. The library and the Function Blocks are marked with 
the ending _APP.

To enable and disable a drive

In order to enable a drive the Function Block MC_Power has to be used within the applicational 
layer. The kernel Function Block will then, if possible, output a rising edge on the output 
Drive_Release which can be connected to the drive-control Function Block which performs the 
needed actions on the drives control word to enable the drive. As soon the drive states enabled, 
this signal can be connected to the input Drive_In_Operation of the kernel Function Block. The 
axis state according to the single axis state diagram of PLCopen will then switch from Disabled 
to Standstill.

Fig. 45: Enabling sequence of a drive
As long as the drive is in state Disabled or ErrorStop the input Drive_Actual_Position will be copied to the output Position_Reference of the kernel Function Block. The output Speed_Reference will be zero.

When the axis is in operation, which means it is not in state Disabled or ErrorStop, then the output Position_Reference will be calculated by the kernel Function Block and the position control loop will be closed, which outputs non zero value for the output Speed_Reference in case of a following error. The input Actual_Position should then follow the position reference. The difference of both values is the following error and will be supervised by the kernel Function Block.

In case of drive problem, Drive_Inoperation should be reset. The Function Block will open the position control loop and Speed_Reference will be set to zero.

For the most drives the status is control by the drives control word whereas the drives status word represents its actual status. In order to enable the drive it might be necessary to pass through several drives states according a defined scheme which depends on the used drive. Therefore the library ABB_Ecat_CiA402_AC500.library is added to PS5611-Motion which contains Function Blocks to operate with different drives on an EtherCAT network. There is also the PS5605-DRIVES library package which can be used to control the state of a drive.

How to use the axis simulation

It is possible to use a simulated axis instead of a real drive. The axis simulation can be used in the following use cases:

- When the real drive is not available the simulation can be used to test all available motion functionalities to verify the application program.
- The simulation can be used to create a virtual master axis and synchronize other axes to it.

The simulation is realized by the Function Block CMC_Axis_Simu or Enable_Virtual = TRUE can be used at the KERNEL-block.

Homing will be possible if the limit-switches (data type CMC_Axis_IO) are simulated also. This is not done by CMC_Axis_Simu but could be realized in the PLC program.
Fig. 46: Example for Simulation

The drive velocity is simulated by PT1-Characteristic. The input $T_1$ gives the time constant for this PT1 as multiple of the cycle time. All other properties are simulated according to the CMC_Axis_Control_Parameter.

The value of the time behaviour from the axis simulation Function Block set by the input $T_1$ has to be at least four times smaller than the value of the axis parameter Control_Time from the parameter Function Block. If Enable_Virtual = TRUE is used, no delay will be applied to the simulated drive speed, and it will not be possible to test the position-control loop, but it will be fine to be used as virtual axis.

How to perform a homing

The homing of an axis is a procedure which consists of up to two phases. For each phase there are different Function Blocks available. The available Function Blocks are according to PLCopen and belong to the application layer. Available Function Blocks for each phase are listed in the table below.

Table 378: Overview of the available homing function blocks

<table>
<thead>
<tr>
<th>Function Block</th>
<th>Phase 1</th>
<th>Phase 2/Finish Homing</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_StepAbsSwitch</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>MC_StepDirect</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Phase 1</td>
<td>Phase 2/Finish Homing</td>
<td></td>
</tr>
<tr>
<td>-------------------------</td>
<td>-----------------------</td>
<td></td>
</tr>
<tr>
<td>MC_StepLimitSwitch</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>MC_StepRefPulse</td>
<td>X</td>
<td></td>
</tr>
</tbody>
</table>

In order to create a complete homing sequence one Function Block of each phase can be used.

**First phase**

The used Function Blocks will change the axis state to Homing and will move the axis to approach installed limit switches or a dedicated absolute switch in the desired directions. No manipulation of a position value will be done in this phase. The use of Function Blocks of this phase is optional for a homing.

The signals of the installed limit switches have to be written to a variable of the data type CMC_Axis_IO.

**Second phase**

Function Blocks from this phase will also change the axis state to Homing if this has not already happen and will finish the homing. Therefore a new position will be set to the axis. The axis state will then switch back to Standstill.

The use of a Function Block of the second phase is mandatory for a homing.

In general with AC500 PLC-based Motion Control there are two position values: One position value will represent the encoder counts of a drive or the CD522 module which is connected to the input Drive_ActualPosition of the kernel Function Block. The other position is a user defined scaled unit which is used for PLCopen Function Blocks.

There are different ways to finish the homing by manipulate and adjust a position value. Which value should be manipulated depends on the used drive or module and its capabilities. See the following types A, B and C.

**Type A**

The user defined position unit will be changed only. The Function Block MC_StepDirect must be used here. This type of homing is less complex than the other types but also less precise.

![Fig. 47: Homing Type A](image)

**Type B**

The drive or the CD522 module will change its own position value, the encoder counts.
The process will be started by the execution of the Function Block MC_StepRefPulse. The axis will start to move. The output Drive_Set_Ref of the kernel Function Block will then set the drive to sense for a digital signal. At the same time the kernel Function Block outputs a preset value which will replace the actual encoder count value at the moment the digital signal occurs.

This signal can be a Z-pulse of an incremental encoder but also any other signal from a sensor. This functionality may require a configuration of the drive or the CD522 module in order to be used.

In the same cycle when the new position value is set there also has to be a boolean signal stating a new position value at the input Drive_Ref_Ok of the kernel Function Block. The user defined position value will then be shifted accordingly.

Example of type B for phase 2: *Chapter 1.4.2.4.4.2.1 “How to connect a drive” on page 2066*

**Type C**

The encoder count position value will not be changed but involves registration capabilities of a drive or the CD522 module.
The process will be started by the execution of the Function Block ECAT_HomingOnTouchProbe_APP (ABB_Ecat_CiA402_AC500.library). The axis will start to move. The output Drive_Set_Ref of the kernel Function Block will then command the drive or the CD522 module to activate the Touch Probe functionality. This will configure the drive to latch a position at the moment a digital signal occurs. The digital signal can be a Z-pulse of an incremental encoder but also any other signal from a sensor. This functionality may require a configuration of the drive or the CD522 module in order to be used. In combination with the latched position value there is a boolean signal which states that a new latch value has been received. In case of the module CD522 this encoder count position value has to be converted from encoder counts to equivalent user scaled units by the use of the function “CMC_Get_Units_From_Inc” (ABB_MotionControl_AC500.library) before it can be connected to the Function Block ECAT_HomingOnTouchProbe_APP.

To manage the Touch Probe objects of a drive within the CiA402 profile (e.g. Microflex e190) the Function Block ECAT_HomingOnTouchProbe_APP (ABB_Ecat_CiA402_AC500.library) can be used. This will also cover the conversion from encoder counts to user scaled units. At the end of the process the Function Block ECAT_HomingOnTouchProbe_APP will manipulate the user scaled position value according to the latched position from the drive and the users settings.

For further information see:
AN00220-001 - AC500 and MicroFlex e190 - EtherCAT Homing Methods

**How to Use a CAM curve**

The CAM functionality is only available in combination with the kernel Function Block CMC_Basic_Kernel.
The usage of a CAM function is based on the following elements:

- CAM table defined with the data type MC_PProfile.
- An instance of the Function Block MC_CamTableSelect.
- An instance of the Function Block MCA_Cam_Extra (optional).
- An instance of Function Block MC_CamIn.
- An instance of Function Block MC_CamOut.

1. Declare a CAM table as an array of the data type MC_PProfile in the program.
2. Write data to this array.
3. Use the address of the CAM table at the input CamTable of the Function Block MC_CamTableSelect.
4. Execute the Function Block MC_CamTableSelect to process the data of the CAM table with the Function Block's input parameters.
5. Additionally you can execute the Function Block MCA_Cam_Extra for optional parameters after the processing of the Function Block MC_CamTableSelect.
6. Execute the Function Block MC_CamIn to start the slave axis movement according to the CAM table data and parameters.
   - The axis will operate in the axis state Synchronized Motion.
7. To leave the axis state you can execute the Function Block MC_CamOut.
   - The axis state will switch to state Continuous Motion and maintains its last velocity as long as there is no other command.
8. You can also use any other motion command to interrupt the Synchronized Motion.

CAM table

CAM data is done with one table (two dimensional – describing master and slave positions together).

The data of the elements (array of data type MC_PProfile) can either be assigned within the declaration or can be assigned during runtime before the execution of the Function Block MC_CamTableSelect.

It can be filled with data in the following ways:

- To use a predefined variable list.
- To calculate the values within the program (before using the MC_CamTableSelect).
- To send values by any communication access to the PLC.

In order to use the new data it is necessary to execute the Function Block MC_CamTableSelect again. In case the CAM table is executed the Function Block MC_CamTableSelect may not be executed.

Elements of the data type MC_PProfile:

- The inputs MasterSyncPosition and MasterSyncDistance of the Function Block MC_CamIn can be used to define a distance to synchronize the slave axis onto the CAM table during the start. In case master axis moves with negative velocity the parameter MasterSyncDistance can be negative. The MasterSyncPosition should always be within the range of the CAM table master position.
- MasterSyncDistance = 0 will deactivate the synchronization. In this case the slave axis should be moved on the CAM curve before MC_CamIn is executed, otherwise a following error can occur.
The master position in the CAM table must be strictly monotonic rising.

The length of a CAM table is just restricted by the memory size of the PLC. When long tables are used, it is recommended to call CamTableSelect in a task with lower priority as it will need a considerable computing time.

It is possible to hold several CamTables as a pool and to switch from one to another. This has to be done at matching positions as no means for synchronization are available.

The offset and scaling values (except the time-scale) are transferred continuously. This will allow to follow a "Moving Target" by adjusting these values.

The parameters at MC_CamTableSelect, MC_CamIn and function and MCA_Cam_Extra also modify the behavior:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Default value</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>MasterAbsolute</td>
<td>BOOL</td>
<td>FALSE</td>
<td>TRUE=Master_position from MC_PProfile equals the master axis absolute position.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>FALSE=CAM is executed relative to the master axis actual position at start.</td>
</tr>
<tr>
<td>SlaveAbsolute</td>
<td>BOOL</td>
<td>FALSE</td>
<td>TRUE=interpolation_point from MC_PProfile equals the slave axis absolute position.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>FALSE=CAM is started from actual slave position. The values &quot;interpolation_point&quot; are relative to the slave axis position at start.</td>
</tr>
<tr>
<td>iType</td>
<td>ENUM</td>
<td></td>
<td>Interpolationtype.</td>
</tr>
<tr>
<td>Number_of_pairs</td>
<td>INT</td>
<td></td>
<td>Number of points used in TimePosition Array.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Default value</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>MasterOffset</td>
<td>LREAL</td>
<td>0</td>
<td>Just used with MasterAbsolute=TRUE, ignored otherwise. Used position for cam-table is: Master axis position-MasterOffset.</td>
</tr>
<tr>
<td>SlaveOffset</td>
<td>LREAL</td>
<td>0</td>
<td>Just used with SlaveAbsolute=TRUE, ignored otherwise. Used position is slave axis position=interpolation_point+Slaveoffset.</td>
</tr>
<tr>
<td>Parameter</td>
<td>Type</td>
<td>Default value</td>
<td>Comment</td>
</tr>
<tr>
<td>----------------------</td>
<td>----------</td>
<td>---------------</td>
<td>--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>MasterScaling</td>
<td>LREAL</td>
<td>1</td>
<td>The position used for interpolation is multiplied by MasterScaling, e.g., MasterScaling=2: the scaled master will pass the position range with double velocity and within the half distance compared to its real velocity and position.</td>
</tr>
<tr>
<td>SlaveScaling</td>
<td>LREAL</td>
<td>1</td>
<td>Interpolation result is multiplied by SlaveScaling, e.g., SlaveScaling=2: Slave axis will run twice the distance.</td>
</tr>
<tr>
<td>MasterSyncPosition</td>
<td>LREAL</td>
<td>0</td>
<td>Start synchronization at master axis position=MasterSyncPosition-MasterStartDistance+MasterOffset, meet the CamTable at master axis position=MasterSyncPosition. In case of MasterAbsolute=FALSE: start at actualPosition+MasterSyncPosition-MasterAbsolute, meet the CamTable at actualPosition+MasterSyncPosition!!! It is just possible to use the &quot;sync&quot; mechanism when the axis is in StandStill on start.</td>
</tr>
<tr>
<td>MasterStartDistance</td>
<td>LREAL</td>
<td>0</td>
<td>A negative value will create a reverse synchronization mode, which means the master should move in negative direction to synchronize. It is independent from the ReverseBit which indicates how to end the movement.</td>
</tr>
</tbody>
</table>

These 2 parameters are "extras" to be written with the MCA_Cam_Extra function. When the parameters are used, the MCA_Cam_Extra has to be called after the MC_CamTableSelect.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Default value</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Periodic</td>
<td>BOOL</td>
<td>TRUE for master &quot;Modulo&quot;, FALSE for master linear axis</td>
<td>CamTable will not reach &quot;EndOfProfile&quot; but will be repeated periodically. When the master is a linear axis, it has to move forward and backward within the CamTable position range, but even when it leaves this position range, the CamTable will stay active.</td>
</tr>
<tr>
<td>Reverse</td>
<td>BOOL</td>
<td>FALSE</td>
<td>Just necessary when a CamTable is NOT &quot;periodic&quot; and will run in reverse direction (master with negative velocity) Reverse=FALSE, the CamTable is ready when the master leaves the position range in positive direction, e.g., when it moves from 359° to 0° on a rollover axes Reverse=TRUE, the CamTable is ready when the master leaves the position range in negative direction.</td>
</tr>
</tbody>
</table>
In this example, the slave will run from 0 to 2000 while the master runs from 0 to 1000. The slave will start and end with velocity \( v = 0 \), no matter which velocity the master has during start. The slave will reach the maximum velocity when it is at position 1000 and the master is at position 500.
How to use an external axis

To use multiaxis PLCopen Function Blocks with an externally sensed axis as master axis the following structure can be used for the axis implementation:

![Diagram of axis implementation structure]

**Fig. 51: Structure synchronization to an external axis**

The use of a feed forward filter Function Block is needed if the slave axis has to follow the position of the external axis. In this case there will be a time delay between sensing the position of the external axis and moving the follower axis along the sensed position. The filter Function Block will then add a certain distance to the external axis’ position depending of its speed.

The filter Function Block MATH_LINEAR_REGRESSION from the library ABB_MathFunctions_AC500.library can be used here.

**Fig. 52: Filter Function Block to feed forward an externally sensed position**

For an axis which is following the external axis, the value “mcActualValue” for the input “MasterValueSource” for multi-axis PLCopen Function Blocks has to be used.

When the filter Function Block MATH_LINEAR_REGRESSION is used to process an actual position, 2 different purposes are fulfilled:

- A jitter or noise can be compensated
- It is possible to calculate a forecast-position to compensate for a delay in position measurement

Process the actual position or any other master axis always before the slave axis.

Otherwise, an additional 1 cycle-delay is introduced.

The MATH_LINEAR_REGRESSION-block calculates the progress for a variable which is captured in equidistant periods of time and is assumed to follow a linear curve. It uses the Gauss “least squares” -algorithm to do so. The line is calculated in a way that the sum of squares for the distances from the measured points to the assumed straight line is minimized.

A noise or jitter influence of the value is compensated and a predictive value for the variable with an adjustable forecast horizon can be calculated.

Linear equation:

\[ \text{Line[i]} = \text{gradient} \times i + \text{offset} \]
Sum of squares:

\[
\text{sum} = \sum_{i=1}^{\text{history}} (x[i] - \text{line}[i])^2
\]

The gradient and offset for the line are calculated in a way that “sum” is minimized. Then these 2 values are used to calculate the forecast value:

\[\text{NEXT_VALUE} = \text{gradient} \times \text{FORECAST} + \text{offset}\]

FORECAST=0 would mean: value right now, no future or past considered.

When the ACTUAL value is a modulo value, for example a single turn encoder or a rollover axis, this has to be considered in the calculation. The 2 input values POSITIVE_LIMIT and NEGATIVE_LIMIT can be used to configure this. They define the upper and lower limit for ACTUAL. Also, the NEXT_BINARY will as a result be limited to these borders.

**Example**

![Fig. 53: Next Value_Forecast](image)

**How to use an encoder/drive with <> 32 bit position overrun**

The incremental position as actual position at the Function Block CMC_Basic_Kernel is usually assumed as position with a 32 bit position overrun. As well as it is the reference position which is sent to the drive.

Any modulo-axis configuration should be done inside the PLC.

Some drives are requested to correct their positions themselves for a non-linear axis which should constantly run into the same direction.

In this case, the drive has to be configured as a modulo-axis and the Function Block CMC_Basic_Kernel needs some additional Function Blocks to create the 32 bit value. See *Chapter 1.4.2.4.4.3.4 “Roll-Over axis” on page 2090.*
The Function Block CMC_Modulo2Binary will convert any position with any Modulo_Range to a 32 bit binary position.

The actual_position is assumed to run between 0 to Modulo_Range.

The actual_position should not change > 1/4 Modulo_Range between two scan cycles.

The Function Block CMC_Modulo2Binary will convert the 32 bit binary position reference from CMC_Basic_Kernel to a position reference which runs from 0 to Modulo_Range.

How to do position correction “on the fly”

Sometimes it is required to have a position correction "on the fly". For example, it can happen that a position is wrong due to mechanical slip and that a switch which is passed by during the movement is used to capture a position value.

In other cases, it is required to synchronize the position to a print mark, so an actual_position has to be corrected, but not the movement of the printed material.

For both applications, the Function Block MCA_SetPositionContinuous can be used. It will use ramps and a limited velocity for the correction, so it will be tolerable to execute it during an ongoing movement and while the axis is activated in a multi-axis movement.

The block can be used in any axis state except ERRORSTOP and HOMING.

Two different operation modes are possible:

1. SuperImp=FALSE
   - The actual_position will be modified.
   - The block will not cause any movement.
   - If a PLCopen block in DISCRETE_MOTION (positioning) is active during the execution, this block will not reach Done as the actual_position is modified.
   - If a slave axis is coupled to an axis while MCA_SetPositionContinuous is executed (with SuperImp=FALSE) it will follow.
   - This mode is possible while the axis is in state DISABLED.
2. SuperImp=TRUE
   - The actual_position will stay constant.
   - A mechanical movement is executed (without changing the axis state machine).
   - A slave axis will not follow.
   - This behavior is similar to a superimposed movement.
   - It is not possible when the axis is in state DISABLED.

The block can just be aborted by another MCA_SetPositionContinuous.

How to limit the movement

It is possible to limit the movement by position (software limit switches) and by velocity. By default, no software limit switches are activated in PS5611-Motion. It is possible to activate them by accessing some PLCopen parameter.

The functionality described below is just available with linear axes.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Minimum</th>
<th>Maximum</th>
<th>Default</th>
<th>R/W</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SWLimitPos</td>
<td>DINT</td>
<td>2147483647</td>
<td>2147483647</td>
<td>2147483647</td>
<td>R/W</td>
<td>Positive Software limit switch position.</td>
</tr>
<tr>
<td>SWLimitNeg</td>
<td>DINT</td>
<td>2147483647</td>
<td>2147483647</td>
<td>2147483647</td>
<td>R/W</td>
<td>Negative Software limit switch position.</td>
</tr>
<tr>
<td>EnableLimitPos</td>
<td>BOOL</td>
<td>FALSE</td>
<td>TRUE</td>
<td>FALSE</td>
<td>R/W</td>
<td>Enable positive software limit switch.</td>
</tr>
<tr>
<td>EnableLimitNeg</td>
<td>BOOL</td>
<td>FALSE</td>
<td>TRUE</td>
<td>FALSE</td>
<td>R/W</td>
<td>Enable negative software limit switch.</td>
</tr>
<tr>
<td>EnableLimit2Decelerate</td>
<td>BOOL</td>
<td>FALSE</td>
<td>TRUE</td>
<td>FALSE</td>
<td>R/W</td>
<td>Enable software limit switches to decelerate.</td>
</tr>
<tr>
<td>Parameter</td>
<td>Data type</td>
<td>Minimum</td>
<td>Maximum</td>
<td>Default</td>
<td>R/W</td>
<td>Description</td>
</tr>
<tr>
<td>---------------</td>
<td>-----------</td>
<td>---------</td>
<td>---------</td>
<td>---------</td>
<td>-----</td>
<td>----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>2004</td>
<td>EnableLimitAbort</td>
<td>BOOL</td>
<td>FALSE</td>
<td>TRUE</td>
<td>FALSE</td>
<td>R/W</td>
</tr>
<tr>
<td>2005</td>
<td>EnableLimitVelocity</td>
<td>BOOL</td>
<td>FALSE</td>
<td>TRUE</td>
<td>FALSE</td>
<td>R/W</td>
</tr>
<tr>
<td>2006</td>
<td>SWLimit2DecPos</td>
<td>LREAL</td>
<td>-2147483647</td>
<td>2147483647</td>
<td>2147483647</td>
<td>R/W</td>
</tr>
</tbody>
</table>
The following different behavior is possible:

- No limitation at all (default)
- Limit position with ERRORSTOP:
  - Limit position between SWLimitNeg to SWLimitPos, axis to state ERRORSTOP in case the position range is left.
- Limit velocity and acceleration:
  - Limit velocity to paraMaxVelocityAppl and acceleration/deceleration to paraMaxDecelerationAppl, create WARNING_VELOCITY, not state changes for axis, abort movement is optional when MaxPositionGap is reached due to limitation.
- Limit Position with ramp-down:
  - In addition, it is possible to limit the position between SWLimit2DecNeg and SWLimit2DecPos. paraMaxDecelerationAppl is used to ramp down.

When activated with EnableLimitPos or EnableLimitNeg, the reaction will be as follows:

- When the control position reaches the respective limit switch, the axis will go to state ERRORSTOP, and Drive_Release will be switched off. The actual position might be behind, depending on the following error. It is assumed that a drive or application specific braking is performed. The axis will be stopped behind the limit.
- The axis could be switched on again by MC_Power. A movement in the opposite direction will be possible.
- The functionality of EnableLimitPos and EnableLimitNeg is unchanged.

You can use the limitation of movement to achieve a soft or adjustable braking in advance before reaching the software limit switch. The limitation is activated by three boolean parameter and will calculate a position distance to the limit switch, which depends on the actual velocity and given deceleration ramp. “paraMaxDecelerationAppl” is used for deceleration. It will decelerate the axis by the given deceleration ramp when the calculated position is reached and stop at the software limit switch. The original behavior is not modified, so if also these software limit switches are activated, the axis might be set to state ERRORSTOP.
There are 2 different modes:

- **EnableLimitAbort = TRUE**
  
  Any ongoing motion will be aborted immediately (when the distance to stop is reached, as shown in the above diagram), a warning is shown.
  
  The axis will be decelerated to reach the software limit switch.

- **EnableLimitAbort = FALSE, EnableLimitDecelerate=TRUE**

  A warning is shown and the velocity is reduced, with respect to the given deceleration and position limit.
  
  The ongoing motion is not aborted. If it was just a "tight fit", e.g. in a master slave movement and the direction is turned soon enough, it might be possible to continue the movement.
  
  As the ongoing movement is not interrupted, an activated movement might not be completed, for example a MC_MoveAbsolute will never reach its target position. A warning is shown at function block CMC_Basic_Kernel.

When **EnableLimitPos = TRUE or EnableLimitNeg = TRUE**, and the values for SWLimitPos or SWLimitNeg are set, the axis will be set to state ERRORSTOP when these position limits are reached.

In addition, the function block will allow to limit the velocity. With **EnableLimitVelocity = TRUE**, it will monitor the velocity demand from the position reference and limit the position reference, so the given velocity limit will not be exceeded. A warning will be shown. The velocity used for limitation is MaxVelocityAppl.
The velocity limitation can be used to prevent short-term velocity peeks. The limited position will be catched up later, whenever possible. This can result in not-expected behavior. The WARNING issued by CMC_Basic_Kernel can be checked and used to stop a movement. The movement will be aborted automatically when the position is by MaxPositionGap behind.

- For a single axis movement, the commanded velocity is limited at the beginning. No position gap will occur.
- In a multi-axis movement, the slave axis follows a master. This can result in a position gap. A velocity peek from the master axis can be reduced by using the limitation. If the master is too fast because of the value for MaxPositionGap, the movement will be aborted.

When EnableLimit2Decelerate or EnableLimitAbort are used, the velocity is limited to MaxVelocitySystem with EnableLimitVelocity = FALSE. The function modifies the position reference. This modified position reference is used to control the drive. Whenever the limitation interferes the kernel will show a warning or an error. The warning or error message will disappear when the situation is cleared.

<table>
<thead>
<tr>
<th></th>
<th>EnableLimitPos</th>
<th>TRUE</th>
<th>ERRORSTOP when positions exceed, no previous warning or deceleration.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>EnableLimitNeg</td>
<td>TRUE</td>
<td>Reduce the velocity when reaching a position limit within the deceleration distance calculated by using MaxDecelerationAppl. Display a warning at CMC_Basic_Kernel. The underlying movement stays active. With EnableLimitPos = TRUE or EnableLimitNeg = TRUE: When the Position limit is reached, the axis is set to mode ERRORSTOP also if EnableLimitPos or EnableLimitNeg are used. Otherwise, just the movement is limited, without affecting the statemachine. An activated positioning movement will not reach its target. Velocity is limited to MaxVelocitySystem.</td>
</tr>
<tr>
<td>4</td>
<td>EnableLimitPos</td>
<td>FALSE/TRUE</td>
<td>Reduce the velocity when reaching a position limit within the deceleration distance calculated by using MaxDecelerationAppl. Display a warning at CMC_Basic_Kernel. The underlying movement stays active. With EnableLimitPos = TRUE or EnableLimitNeg = TRUE: When the Position limit is reached, the axis is set to mode ERRORSTOP also if EnableLimitPos or EnableLimitNeg are used. Otherwise, just the movement is limited, without affecting the statemachine. An activated positioning movement will not reach its target. Velocity is limited to MaxVelocitySystem.</td>
</tr>
<tr>
<td>5</td>
<td>EnableLimitNeg</td>
<td>FALSE/TRUE</td>
<td>Reduce the velocity when reaching a position limit within the deceleration distance calculated by using MaxDecelerationAppl. Display a warning at CMC_Basic_Kernel. The underlying movement stays active. With EnableLimitPos = TRUE or EnableLimitNeg = TRUE: When the Position limit is reached, the axis is set to mode ERRORSTOP also if EnableLimitPos or EnableLimitNeg are used. Otherwise, just the movement is limited, without affecting the statemachine. An activated positioning movement will not reach its target. Velocity is limited to MaxVelocitySystem.</td>
</tr>
<tr>
<td></td>
<td>EnableLimitPos</td>
<td>FALSE/TRUE</td>
<td>Reduce the velocity when reaching a position limit within the deceleration distance calculated by using MaxDecelerationAppl. Display a warning at CMC_Basic_Kernel. The underlying movement stays active. With EnableLimitPos = TRUE or EnableLimitNeg = TRUE: When the Position limit is reached, the axis is set to mode ERRORSTOP also if EnableLimitPos or EnableLimitNeg are used. Otherwise, just the movement is limited, without affecting the state machine. An activated positioning movement will not reach its target. Velocity is limited to MaxVelocitySystem. The active PLCopen function block is aborted as soon as the warning is issued. With EnableLimitPos = TRUE or EnableLimitNeg = TRUE: When the Position limit is reached, the axis is set to mode ERRORSTOP.</td>
</tr>
<tr>
<td>---</td>
<td>-------------</td>
<td>------------</td>
<td>---------------------------------------------------</td>
</tr>
<tr>
<td>5</td>
<td>EnableLimitNeg</td>
<td>FALSE/TRUE</td>
<td>The velocity is checked and also limited to the value MaxVelocityAppl. A warning is shown. The active movement is not aborted. This functionality works independent from software limit switches.</td>
</tr>
<tr>
<td>2003</td>
<td>EnableLimit2Decelerate</td>
<td>---</td>
<td></td>
</tr>
<tr>
<td>2004</td>
<td>EnableLimitAbort</td>
<td>TRUE</td>
<td></td>
</tr>
<tr>
<td>2005</td>
<td>EnableLimitVelocity</td>
<td>FALSE</td>
<td></td>
</tr>
</tbody>
</table>

**Axis parameters**

The parameters for axis configuration and adjustment are set by the Function Blocks CMC_Axis_Control_Parameters.

Depending on the version of the kernel Function Block the corresponding version of the parameters Function Block has to be used. The instance will then be connected to the kernel Function Block by its instance name.
In this example the control structure is a simple position control loop with just proportional gain. When the application does not require minimized position following error it should be used this way as it is simple to adjust, robust and requires minimal performance. The proportional gain is then adjusted by Control_Time. Just change values at CMC_Axis_Control_Parameter when the position control loop is open (Drive_Release=FALSE, the axis state is Disabled). The values are sending to the control loop with a positive edge at "Enable". The CMC_Basic_Kernel block needs to be already enabled.

**Supervision**

**Pos_Lag_Percentage**

This parameter configures the position window for the supervision of the following error. The default value is 150[\%]. A value of 0[\%] will deactivate the supervision function.

The size of the position window depends on the setting of the parameters Control_Time and Max_Rpm. The size of the position window depends on the following equation:

\[
\text{Position Window [Increments]} = \left( \frac{\text{Inc Per R}}{\text{Max Rpm}} \right) \times \left( \frac{\text{Control_Time}}{1000} \right)
\]

\[
\text{Position Window [Units]} = \left( \frac{\text{U_per_Rev_Nominator}}{\text{U_per_Rev_Denominator}} \right) \times \left( \frac{\text{Max_Rpm}}{60} \right) \times \left( \frac{\text{Control_Time}}{1000} \right)
\]

**Example**

Position Window [Increments] = (10000) * (6000/60) * (50/1000) = 50000 [Increments]

Position Window [Units] = (1/1) * (6000/60) * (50/1000) = 5 [Units]

A value of 100\% will result in a position window which corresponds to the expected following error with the giving Control_Time at Max_Rpm. Therefore it is recommended to use values higher than 100[\%]. In case the parameter FF_Percentage is used smaller values can be used.

If the supervised position window is exceeded the axis state will change to ErrorStop.

**V_Check_Time**

After the configured time the drive’s actual velocity has to be at least 50 \% of the commanded velocity. This function can also be used in case the Position Reference is transferred to the drive.

A value of 0 will deactivate this supervision function.
If the supervised velocity window is exceeded the axis state will change to ErrorStop.

**Position control loop**

The default value is 100 which leads to a proportional gain of 10.

*In case the value of Control Time is too short the position control loop will run into instability.*

*In case the position control loop is not used this parameter must not be set to 0.*

The static following error depends on the axis velocity and can be calculated easily: Control Time multiplied by the axis velocity (\( p_{\text{error}} = v \times CT \)).

In general it should be aimed to reach a high position control loop gain with a short Control Time to achieve a small following error. As the reaction times take account in the possible Control Time of the complete system (parameters of the drive control loops, PLC cycle time as well as the communication fieldbus) should be considered.

As a basic rule the Control Time should be at least four times longer than the reaction time between the output of the Speed Reference and the input of actual position.
When the time $T_s$ and $T_t$ is measured, a control_time of $4(T_s+T_t)$ will result in an aperiodic damping of the position control loop. It is important to measure the values from inside the PLC (e.g. Trace) to have the complete reaction times included. Practical values for Control_Time might be from 50 - 500ms. The PLC cycle time as well as bus cycle times and mechanical reaction will influence the value.

**FF_Percentage**

The default value is 0.

In case a velocity feedforward has to be configured a value of up to 80 is recommended. For larger values than 80 the parameter Horizon needs to be used as the resulted position will overshoot otherwise.

A value of 100 adds a velocity to the Speed Reference output which corresponds exactly to the ongoing Position Reference value.

**Integral_Part**

The integral part of the position control loop can be used to eliminate a permanent positioning error, e.g. in case of hanging loads.

The time value can be regarded as the time the integrator needs to sum up the input value to reach the same value for its output.

> In case the Integral Part Time is too short the position control loop will run into instability.

**Horizon**

A communication delay of the Speed Reference value to the drive system can cause an overshoot during positioning caused by the velocity feedforward gain.

This function will compensate this communication delay to prevent an overshoot by time shifting the signals Velocity Feed Forward and Position Reference relatively to each other.

The value of Horizon can be approximately assumed to be the time delay of the communication delay.

The delay time might be caused by the cycle time of the control loop and by any delay in sending the speed reference, delay in the drive to build up the torque and delay to receive the actual position. To overcome this delay, a Horizon > 0 might be used. The feed forward reference will be created in advance, while the proportional gain is applied to the original motion profile. The delay is then compensated.

This function should not be used if the feed forward parameter FF_Percentage is 0.

A value of 0 will deactivate this function, which is the default value.

While this function is used, it will increase the needed PLC calculation time for this axis.
PLC cycle time

Cycle

This parameter represents the cycle time in which the kernel Function Block of the axis is called. If the configured cycle time is not correct the resulting acceleration and speed of an axis will be not correct also.

In case the task execution of the axis is synchronized to a fieldbus (e.g. EtherCAT) the cycle time of the fieldbus has to be used.

Roll-Over axis

When a rollover axis is used, the position control loop has to be closed inside the PLC, the drive should run in speed mode using the Speed Reference value of the Kernel Function Block.
If the Position Reference value is used, the drive must be able to perform a position over-run after 32 bit. If the drive’s position over-run is different, it can be adapted with the function blocks CMC_Binary2Modulo and CMC_Modulo2Binary from the library ABB_MotionControl_AC500.library. Incompatibility can cause an axis to trip after hours of operation.

The possible position following error has to be smaller than \( \frac{1}{2} \) Modulo_Range. Make sure that the modulo range is large enough.

Position following error = \((100 - \text{FF Percentage}) \times \text{Max Rpm} \times \text{Inc Per R} \times \text{Control Time} / 6000000\). This is the maximum value at constant velocity.

**En_Modulo**  
With this parameter the axis can be configured as a roll-over axis.

**Modulo_Range**  
The modulo range will be defined in drive position counts (DINT). It will result that the scaled unit position which is used by the PLCopen function blocks will stay within the defined range.

**Example**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>En_Modulo</td>
<td>TRUE</td>
</tr>
<tr>
<td>Modulo_Range</td>
<td>20000</td>
</tr>
<tr>
<td>Inc_Per_Rev</td>
<td>10000</td>
</tr>
<tr>
<td>U_Per_Rev_Nominator</td>
<td>360 (e.g. degree)</td>
</tr>
<tr>
<td>U_Per_Rev_Denominator</td>
<td>1</td>
</tr>
</tbody>
</table>

The scaled units position will cover the range from 0 to 720 (degrees).

In some cases it is not suitable to set the modulo range of an application with the DINT value of the parameter Modulo_Range only. In such cases the parameters 2001 Modulo_Nominator and 2002 Modulo_Denominator can be used to scale the parameter Modulo_Range to a more precise value.

**Parameter Modulo_Nominator and Modulo_Denominator (supported with CMC_Basic_Kernel)**

These parameters can be used to modify the Modulo_Range in a way that fractions of an increment could be used for 1 modulo (=rollover) distance.

- Default: Modulo_Nominator=1 and Modulo_Denominator=1: the actual position for an axis is limited between 0 and Modulo_Range increments.
- Limitations: Modulo_Range*Modulo_Nominator < 2147483647. Otherwise: default values will be used.
- When modifying these parameters, the position control loop should be opened.
Example

En_Modulo            = TRUE
Modulo_Range         = 1024
Modulo_Nominator         = 10
Modulo_Denominator         = 3
Inc_Per_R            = 1024
U_Per_Rev_Nominator     = 80*5*3
U_Per_Rev_Denominator     = 10

Result of parameters Modulo_Range, Modulo_Nominator and Modulo_Denominator: The modulo range will cover one revolution of the toothed belt wheel.

Result of parameters U_Per_Rev_Nominator and U_Per_Rev_Denominator: One scaled unit corresponds to one mm of the tooth belt.

Example:  
Gearbox 10.1

<table>
<thead>
<tr>
<th></th>
<th>Option1</th>
<th>Option2</th>
</tr>
</thead>
<tbody>
<tr>
<td>En_Modulo</td>
<td>TRUE</td>
<td>TRUE</td>
</tr>
<tr>
<td>Modulo_Range</td>
<td>10240</td>
<td>10240</td>
</tr>
<tr>
<td>Modulo_Nominator</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Modulo_Denominator</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Inc_Per_R</td>
<td>1024</td>
<td>10240</td>
</tr>
<tr>
<td>U_Per_Rev_Nominator</td>
<td>36</td>
<td>360</td>
</tr>
<tr>
<td>U_Per_Rev_Denominator</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Max_Rpm</td>
<td>3000</td>
<td>300</td>
</tr>
</tbody>
</table>

The two options above describe exactly the same configuration. The Modulo_Range is equivalent to 10 motor revolutions and is 10240 increments. For the position, 1u means 1° and the resolution is 360°/10240inc = 0,035°/Inc = 1°/28,44 Inc.
The two options above describe exactly the same configuration. The gearbox is 10:3, so the Modulo_Range is equivalent to 1024*10/3 = 3413 + 1/3 increments. For the first option, the resulting modulo range is calculated 1024*10/3, for option2, it is 10240*1/3. For the position, 1u means 1° and the resolution is 108°/1024inc = 0,105°/Inc = 1°/9.481 Inc.

Scaling of the unit of length

Inc_Per_R  
With this parameter the number of the drive position counts each revolution of the motor (DINT) have to be entered.

U_Per_Rev_Denominator & U_Per_Rev_Nominator  
With these two parameters the number of units which correspond to one revolution of the motor have to be entered.

The units of length can be scaled to values like: mm, inch, degree, …

All dynamic paramters of the PLCopen function blocks like velocity, acceleration and jerk are based on seconds. Velocity [units/s], acceleration [units/s²], jerk [units/s³]

Example 1

\[
\begin{align*}
\text{Inc Per Rev} & = 10000 \\
\text{U Per Rev Nominator} & = 360 \\
\text{U Per Rev Denominator} & = 1
\end{align*}
\]

This will scale one unit to one degrees of the motor shaft. Correspondingly a velocity [units/s] of 360 will turn the motor shaft one revolution per second.

Example 2

In this example one unit will be scaled to one millimeter of the conveyor.

5 mm each tooth

80 teeth

Gearbox 1:5 ratio

Motor/Encoder 1024 counts per revolution

Fig. 60: Scaling units

How many units will pass after one revolution of the motor? (80*5mm) / 5 = 80

\[
\begin{align*}
\text{Inc Per Rev} & = 1024 \\
\text{U Per Rev Nominator} & = 80 \\
\text{U Per Rev Denominator} & = 1
\end{align*}
\]
Example 3  In this example one unit will be scaled to one millimeter of the conveyor.

5 mm each tooth

80 teeth

1:32 ratio

1024 counts per revolution

Fig. 61: Scaling units

How many units will pass after one revolution of the motor? (80*5mm) / 32 = 12,5 = 125 / 10

Inc_Per_Rev = 1024
U_Per_Rev_Nominator = 125
U_Per_Rev_Denominator = 10

Scaling of the speed reference output

These two parameters are used to scale Speed Reference output of the kernel FB in order to reach the intended velocity by the output value and to limit the highest possible output value.

Ref_Max  Highest possible output value of the Speed Reference output. The Speed Reference value that corresponds to the parameter Max_Rpm should be used.

Max_Rpm  Maximum speed of the motor in revolutions per minute.

Example

- Analog Drive: 1000 rpm at 2 Volts, 3200 rpm at 6,4 Volts (max.)
- Analog output module: 10 Volts output at digital value 27648
- Ref_Max = 17695 (= 27648 / 10 * 6,4)
- Max_Rpm = 3200

Access and modify parameters

All modifications will be effective immediately. There is no extra plausibility check and values are not checked for limitations.

Use this functionality with care.

Some parameters are collected inside a structure in Axis_Ref, and can be accessed and modified immediately. They are the same parameters as used with function blocks MC_WriteParameter and MC_ReadParameter © Chapter 1.4.2.4.3.6 “PLCopen parameter” on page 2042.

The differences are:
- Only available with CMC_Basic_Kernel
- The parameter values are LREAL instead of DINT and can be used with decimals.
- The parameters will be effective immediately.
- There is no check for consistency or limits.
- The parameters for position control can be checked and modified by accessing the structure Axis.parameter.position_control in addition.
### Parameter for position control

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>KP</td>
<td>Proportional gain in positive direction. Used directly to multiply the following error and create the Reference_Prop.</td>
</tr>
<tr>
<td>KF</td>
<td>Feed forward in positive direction. Used directly to multiply the speed reference and create the Reference_FF.</td>
</tr>
<tr>
<td>KP_BACK</td>
<td>Proportional gain in negative direction. Used directly to multiply the following error and create the Reference_Prop.</td>
</tr>
<tr>
<td>KF_BACK</td>
<td>Feed forward in negative direction. Used directly to multiply the speed reference and create the Reference_FF.</td>
</tr>
<tr>
<td>TI</td>
<td>Integration time. When parameter is used the position control loop has an additional integral part. In TI cycle, the Reference_ITG will reach the value of Reference_Prop, when KI=100*KP.</td>
</tr>
<tr>
<td>KI</td>
<td>Proportional gain, used for integral part of position control loop.</td>
</tr>
<tr>
<td>KF_100</td>
<td>Value for feed forward gain, if 100% would be used.</td>
</tr>
<tr>
<td>Max_Time</td>
<td>Delay time used for supervision of velocity. With Max_Time=0, no supervision is executed.</td>
</tr>
<tr>
<td>D_XS_Max</td>
<td>Maximum possible velocity in [u/cycle]. The maximum allowed following error is part of the parameter structure, PLCopen parameter paraMaxPositionLag.</td>
</tr>
<tr>
<td>Ref_Max</td>
<td>Limit for Speed_Reference.</td>
</tr>
</tbody>
</table>

**Element actual of Axis_Ref**

The element actual represents actual values from inside the position control loop.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Position</td>
<td>Actual position in [u] to control the axis.</td>
</tr>
<tr>
<td>Control_Position</td>
<td>Reference position in [u] which is actually used for control loop.</td>
</tr>
<tr>
<td>D_XS</td>
<td>Distance in [u] to be moved per cycle.</td>
</tr>
<tr>
<td>D_XSS</td>
<td>Following error in [u].</td>
</tr>
<tr>
<td>Reference_Prop</td>
<td>Proportional part for Speed_Reference.</td>
</tr>
<tr>
<td>Reference_FF</td>
<td>Feed forward part for Speed_Reference.</td>
</tr>
<tr>
<td>Reference_ITG</td>
<td>Integral part for Speed_Reference.</td>
</tr>
</tbody>
</table>

**Possible to use different gain for forward/ backward movement, possible improvement for hydraulic axis or vertical movement**

See parameter KP/KP_BACK and KF/KF_BACK.

**Limitation for velocity and acceleration and deceleration**

From library version 3.1 on, these values are not limited to the 16-bit value range (32767). The limit for velocity is calculated by the values given at CMC_Axis_Control_Parameter and the acceleration is limited such that this velocity can not be reached faster than 1 cycle.
Programming guidelines

To achieve the best results for Motion Control the actual position has to be transferred in best possible quality (with minimal jitter) to the PLC. The position feedback is expected to be in increments as the data type is a DINT.

The kernel Function Block (CMC_Basic_Kernel or CMC_Estp_Kernel) has to be called every cycle and its task requires a fixed cycle time.

A variable of type Axis_Ref is used to connect to the PLCopen Function Blocks and their kernel Function Block.

The Function Block CMC_Axis_Control_Parameter has to be used for the axis configuration. Chapter 1.4.2.4.4.3 “Axis parameters” on page 2086

The signal of the limits switches and the absolute switch should be connected to the elements of the data type CMC_Axis_IO. The signal of the absolute switch must be TRUE in case the axis hits the sensor. The signal of a corresponding limit switch has to be true when the axis leaves the area surrounded by the limit switches. If needed the signal has to be inverted before it is connected to the elements of the data type.

Task configuration

The kernel Function Block and the transfer of axis IO data should be processed in a cyclic task. This task should be as short and real-time as possible to achieve the best motion control performance. It might be useful to set the task priority to 10 and the priority of all other task will be set to 11 or higher.

In order to save PLC processing time the most PLCopen function blocks as well as the application logic can also be processed in a task which runs on a lower priority than the real-time task with the axis implementation as shown in the figure below.

Optional:
Unsynchronized Task Execution

Synchronized Task Execution

All PLCopen function blocks which must be called in the same task than the kernel function block:

- MC_CombineAxes
- MCA_MoveByExternalReference

In case the position reference is transferred to the drive the task of the axis implementation should be synchronized to the fieldbus cycle. The following pictures show an example for EtherCAT:

Fig. 62: Task of axis layer
Fig. 63: Task of application implementation

**Visualization**

The structure of the position control loop is also as visualization element CMC_Visu_FB_Basic_Kernel. included in ABB_MotionControl_AC500.library. As placeholder, an instance of CMC_Basic_Kernel has to be used. The visualization shows all numbers as they are really used inside the block, the adjustment for different resolution or cycle times is already included.

**ABB specific data structures**

Not all data structures are defined by PLCopen. Some specific structures are described in the following chapter. In addition to the data in these arrays, the movement is modified by offset and scaling values at the respective Function Block. These offset and scaling values (except the time-scale) are transferred continuously. This will allow to follow a "Moving Target" by adjusting these values.
PositionProfile

The data type MC_PProfile is used for CamTable. An array has to be defined and provided at MC_CamTableSelect. Several CamTables could be defined and the axis could change between them on the fly. There is no routine of smooth movement from one table to the next so the user has to take care just to switch on appropriate positions. Details are described in the documentation included with the library.

```plaintext
ARRAY[1..3] OF MC_PProfile:=
  (Master_position:= 0 ,interpolation_point := 0 ,Velocity_ratio:= 0 ,Acceleration_ratio:= 0 ),
  (Master_position:= 50 ,interpolation_point := 25 ,Velocity_ratio:= 0 ,Acceleration_ratio:= 0 ),
  (Master_position:= 100 ,interpolation_point := 0 ,Velocity_ratio:= 0 ,Acceleration_ratio:= 0 );
```

Declaration example

**CAM_table**

This structure is used for time based profiles, e.g. MC_PositionProfile:

Interpolation types for profiles

The curves defined by an array of MC_PProfile hold master position points and according slave positions. When the master position is between 2 points, the according position for the slave is interpolated. Different types of interpolation are possible. The type is defined in MC_ABB_iTypes_Enum. The master could be a real axis or some virtual axis which could be created by just writing values for position and velocity to the Axis_Master variable as shown in the example. The same interpolation types could be used on MC_TProfile.

![Fig. 64: usage_centralized10_V3](image_url)

Fig. 64: usage_centralized10_V3
Table 379: Overview of different interpolations

<table>
<thead>
<tr>
<th>Interpolation Types</th>
<th>Results in</th>
<th>Requires</th>
</tr>
</thead>
<tbody>
<tr>
<td>MCA_LINEAR</td>
<td>Linear interpolation with constant velocity between interpolation points.</td>
<td>profile.MC_PProfile_Array[x].master_position, profile.MC_PProfile_Array[x].interpolation_point</td>
</tr>
<tr>
<td>MCA_SPLINE_NATURAL</td>
<td>Cubic spline interpolation without jerk.</td>
<td>profile.MC_PProfile_Array[x].master_position, profile.MC_PProfile_Array[x].interpolation_point</td>
</tr>
<tr>
<td>MCA_SPLINE_COMPLETE</td>
<td>Cubic spline interpolation without jerk, start and end of profile with velocity=0.</td>
<td>profile.MC_PProfile_Array[x].master_position, profile.MC_PProfile_Array[x].interpolation_point</td>
</tr>
<tr>
<td>MCA_POLY3</td>
<td>Polynomial interpolation with linear velocity between interpolation points.</td>
<td>profile.MC_PProfile_Array[x].master_position, profile.MC_PProfile_Array[x].interpolation_point, profile.MC_PProfile_Array[x].velocity_ratio</td>
</tr>
<tr>
<td>MCA_POLY5</td>
<td>Polynomial interpolation with linear acceleration between interpolation points.</td>
<td>profile.MC_PProfile_Array[x].master_position, profile.MC_PProfile_Array[x].interpolation_point, profile.MC_PProfile_Array[x].velocity_ratio, profile.MC_PProfile_Array[x].acceleration_ratio</td>
</tr>
</tbody>
</table>

The interpolations allow to run on smooth curves without the need to define a large number of points. The following chapter shows the results with different interpolation modes for a sinus-curve with 10 interpolation points. The following table gives the mean deviation.

<table>
<thead>
<tr>
<th>Interpolation Type</th>
<th>Mean deviation [ppm]</th>
</tr>
</thead>
<tbody>
<tr>
<td>MCA_LINEAR</td>
<td>19686 =1.9%</td>
</tr>
<tr>
<td>MCA_SPLINE_NATURAL</td>
<td>151 =0.0151%</td>
</tr>
<tr>
<td>MCA_SPLINE_COMPLETE</td>
<td>25510 =2.5%</td>
</tr>
<tr>
<td>MCA_POLY3</td>
<td>131 =0.0131%</td>
</tr>
<tr>
<td>MCA_POLY5</td>
<td>0.37</td>
</tr>
</tbody>
</table>

The original curve is represented by y_sinus for position and v_sinus for velocity. The diagrams show the result which is achieved by different interpolation types.

MCA_LINEAR
Fig. 65: Results from linear interpolation

The velocity is constant between the interpolation points.

MCA_POLY3

Fig. 66: Results from polynomial interpolation

The result looks almost identical to the original curve. The mean deviation shows that MCA_POLY3, MCA_POLY5 and MCA_SPLINE_NATURAL produce results which follow the original curve really good and are almost identical. The spline interpolation produces a jerk-free curve without the need of providing velocity values and acceleration values in advance.
Fig. 67: Results from complete spline interpolation

In the beginning and the end, the curve does not follow the original curve. The reason is that it starts with velocity $=0$ and produces a jerk free result.

So the favoured result has to be considered in advance to choose the right interpolation method. With these different methods it is not necessary to provide a large number of interpolation points to get good results and smooth acceleration and deceleration ramps.

1.4.2.4.5 Examples

Example projects for the libraries can be found in the folder: \Users\Public\Documents\AutomationBuilder\Examples

1.4.2.5 High Availability Modbus TCP

1.4.2.5.1 HA-Modbus TCP - System Technology

The AC500 High Availability System

The AC500 High Availability system is designed for the demand of automation systems that require a higher availability, which is realized by redundant devices and communications. The redundancy concept reduces the risk of losing production due to failure of parts of the automation system and thereby minimizes scheduled idle times.

For instance, control can be taken over by the secondary station automatically if the primary station fails.

AC500 High Availability system implements redundancy based on standard AC500 PLCs:

- PLC
- Field communication
- SCADA communication
General differences in high availability and redundancy systems are in which way and how fast the switchover between redundancies happens. AC500 High Availability systems are hot-standby.

Redundant systems or redundancy functions:
- Always up and running (hot)
- Failure detection
- Switch over automatically in very short time

The AC500 High Availability system HA-Modbus TCP is the Ethernet based variant of HA-CS31 redundancy system.

![Figure 68: Principle AC500 HA-Modbus TCP architecture example based on Ethernet redundancy](image)

- For the data exchange/synchronization (=Sync) between the PLC A and the PLC B the protocol **UDP** is used e.g. via one of the Ethernet ports.  
  **Note:** To differ a "sync link" failure from an "other PLC" failure a so called *lifecom2* communication is used, which then should be routed via a different communication path than the Sync, e.g. the field or SCADA network.
- The field I/O connection is performed via the Ethernet protocol '*ModbusTCP' - connecting the CI52x devices (**CI521-MODTCP** or **CI522-MODTCP**).  
  For High availability/redundancy in a decentralized I/O station communication the Ethernet network redundancy mechanisms can be used (realized by external, managed switches). CI52x modules are not actively participating in ring recovery, however, smaller systems can be daisy chained if **MRP (Media Redundancy Protocol)** is used.
- SCADA connection is redundant by nature of the two Ethernet ports and can be extended with further redundancy level as well by managed switches. SCADA itself can also switch the primary PLC to ensure communication to the active PLC in case of a simple connection and a connection failure. If the redundancy mechanism of the OPC DA Server is not used, SCADA level itself has to be able to handle and differentiate primary and secondary PLC.
In most PLC applications the critical components to fail are, beneath PLC, typically the power supply or communication components such as wires or switches. Therefore a SPOF (Single Point Of Failure) has to be avoided by adding redundant devices or redundancy functions wherever a failure likelihood is high and failures are not tolerable.

HA core functionality typically can tolerate only a single failure in the different levels. Then, a repair of the failed part is highly advised to achieve and ensure redundancy again. As shown in the above figure, the I/O-network already provides a second independent redundancy layer by its redundancy mechanism (e.g. ring), which can keep up communication despite a break in the ring itself, without switching the PLCs: There a second failure in the PLC level could be tolerated, but it is highly advised to repair immediately anyway.

With a well-planned communication network, the PLCs can operate geographically separated. So even in catastrophic events with full mechanical destruction still one PLC will be available to control the process or infrastructure.

In order to achieve high availability, the CODESYS application must be enhanced with HA function blocks, from the HA-Modbus TCP library and the CI52x library.

- **HA control** function blocks manage the core HA functionality by collecting diagnosis and switching if necessary.
- **HA utility** function blocks provide standard functions in the application program with internal sync for integral data e.g. timers, counters, PI control.
- **CI52x library** contains a function block to configure and communicate to the CI modules and ensures that only the primary PLC writes to the outputs. The inputs are read by both PLCs.
- For both PLCs the same application must be used/downloaded.

For configuration of the CI52x Modbus TCPs, a separate Bulk Data Manager tool (BDM) is provided. Especially in larger systems usage of BDM is recommended to comfortably engineer and create CI52x related data in one place:

- Configuration and parameters of the used I/O modules
- Program code for variable naming, configuration, communication and HA functionality

The BDM tool can serve SCADA programming and documentation as well in an efficient manner.

### Differences HA-Modbus TCP (V2/ V3) and HA-CS31

The following table gives an overview of the different High Availability variants possible with AC500.

The figures are indicative, depend on chosen architectures, system size, network and CPU/CM modules used.

<table>
<thead>
<tr>
<th>Library version</th>
<th>HA-CS31</th>
<th>HA-Modbus TCP</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CPU version</strong></td>
<td>V2 CPUs</td>
<td>V2 CPUs</td>
</tr>
<tr>
<td></td>
<td></td>
<td>V3 CPUs</td>
</tr>
<tr>
<td><strong>IO Communication</strong></td>
<td>Parallel serial</td>
<td>Ethernet</td>
</tr>
<tr>
<td><strong>CPUs</strong></td>
<td>PM573 - 595</td>
<td>PM573</td>
</tr>
<tr>
<td></td>
<td>PM595</td>
<td>PM5630</td>
</tr>
<tr>
<td></td>
<td>PM5670</td>
<td></td>
</tr>
<tr>
<td><strong>Max. system size CI52x</strong></td>
<td>3 - 50</td>
<td>3</td>
</tr>
</tbody>
</table>

1. Indicates a reduced failure rate
2. Indicates a reduced system size
### IO Modules

<table>
<thead>
<tr>
<th>Switch-over times</th>
<th>CPU</th>
<th>Field</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>CI590: S500 usable</td>
<td>25 - 120 ms (^3)</td>
</tr>
<tr>
<td></td>
<td>CI52x: S500 and S500-eCo usable (^4)</td>
<td>15 - 120 ms (^3)</td>
</tr>
</tbody>
</table>

|       | 70 ms \(^6\) | Depends mainly on network size, redundancy mechanism of external switches and TCP: e.g. MRP typ. <150 ms \(^7\) |

### SCADA connectivity

|       | OPC DA, IEC60870, ... | OPC DA, IEC60870, ... | OPC DA, OPC UA, IEC60870, ... |

### Programming

|       | Codesys V2.3 | Codesys V2.3 | Codesys V3.5 |

### Interfaces

|       | Several CS31 and Ethernet interfaces | Several ETH ports, via CM597 | 2 ETH ports \(^5\) (+ 1 CAN Interface) |

### Overview of AC500 HA system alternatives

![Diagram of AC500 HA system alternatives]

1) Number of CI52x recommendation based on performance or max. number of sockets (CPU and CM modules)

2) Limited by CPU performance, number of CM574 modules number of CS31 clients and process data limits.

3) Depends on system size and CPU type

4) Some S500-eCo modules not yet usable, see release notes

5) Communication Module CM597 not available

6) see Chapter 1.4.2.5.1.6.1 “Task Configuration Recommendations for HA System” on page 2113

7) In switchover moment some telegrams delayed by \(~ 250 \text{ ms} (500\text{ms on V2 CPU ETH})\)

---

How to get and install the AC500 High Availability System Package

The AC500 HA-Modbus TCP is a zip file named `PS5601-HA-MTCP`. It contains a batch file and the package. The batch file installs the package with the help of Automation Builder and results in the following being available:
● Libraries for the different CPU versions (V2, V3). See installation path: C:\Program Files (x86)\Common Files\CAA-Targets\ABB_AC500\AC500_V12\library\ApplicationLibraries.

● Documentation. See documentation for HA-Modbus TCP system technology, function block documentation, Bulk Data Manager documentation, CI52x device documentation (see the following picture).

● AC500/ CP600 example projects, their PDF documentation (see the following picture).

● Bulk Data Manager (BDM) tool which helps efficient engineering in larger projects (see the following picture). This requires a separate installation.

This package is available only for sales registered customers/end-customers and only via ABB’s sales/support.

---

**Fig. 69: Libraries installation path and documentation, tool, examples installation path**

---

**Requirements**

**Hardware**

Two same type AC500 PLCs are required as central hardware components. Each PLC is equipped with at least two Ethernet ports in CPU or in a communication module. The two PLCs, called PLC A and PLC B, are linked by Ethernet to exchange and synchronize information (Sync). Connections to the AC500 peripheral field devices (I/O) are performed via Ethernet as well.

Peripheral devices for I/O usable are CI521-MODTCP or CI522-MODTCP which communicate via the Modbus TCP protocol with the PLCs. Based on the life systems status exchanged between PLC A and PLC B over the Sync, in the event of a fault in the system the primary PLC may switch automatically to the secondary in a fast, bumpless way: Allowing the process control and status data exchange to continue unnoticed by the process. Faults will result in error messages so that repair can be initialized.

HA-Modbus TCP library supports currently up to 100 CI52x, depending on CPU type. Each CI52x can support up to maximum of 10 S500-I/O modules on it.
Local I/O on a CPU can signal / interact for diagnosis or service with / from this CPU.

This local I/O is NOT redundant and will be not available in CPU failure case.

- SCADA/PC connection is done using ETH ports of both PLCs and one or several managed Ethernet switches depending on the redundancy requirements in the Ethernet levels.
- HA communication between PLC A and PLC B is done via two physical communications between PLC A and PLC B, using Sync (UDP) and lifecom2 (Modbus TCP or CAN).
- Field devices (CI52x modules) will be connected via Ethernet switches, forming a redundant network (if requested). For details on network configuration see Chapter 1.4.2.5.1.6.4 “Field I/O Network Topologies” on page 2115.

High availability system without connecting field devices (CI52x modules)
For a system without any CI521-MODTCP or CI522-MODTCP module connected in the network, set the global variable "xNoCiBus“ defined in HA_GLOBAL_VARIABLES to TRUE.

Effects of xNoCiBus = TRUE:
- CI modules are not connected in the high availability system. One of the PLCs will be primary.
- Data sync between the PLCs will be established.
- Change over of the PLC only happens if either the PLC is stopped or powered off.
- User has to define the HA and Application in different tasks Chapter 1.4.2.5.1.6.1 “Task Configuration Recommendations for HA System” on page 2113.

Example
HA hardware configuration based on V3 PLC. The Sync connection is via SCADA Network, lifecom2 is via field network (can be the other way round as well).
Support of I/O modules (S500/S500-eCo) depends on the version of the library package. See the version details of the library in the release notes.

Software

In order to achieve High Availability, the application must be enhanced with HA function blocks from the HA-Modbus TCP library. There is a separate specific bulk engineering tool called Bulk Data Manager (BDM), which enables a comfortable pre-engineering of the CI52x, all signal names and their communication, resulting in export files for the Automation Builder application. The same data and approach can also be used to start SCADA side setup efficiently from the same data source.
Functionality

Failures, Use Cases

The AC500 High Availability system performs a switch-over when the primary PLC is powered off, crashed or stopped or if the primary PLC loses fieldbus communication completely while the secondary still has it. In the following the different use cases and reaction times are outlined.
### HA use cases – Failures

<table>
<thead>
<tr>
<th>Case</th>
<th>Use case</th>
<th>Reaction</th>
<th>Diagnosis message on *)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Primary PLC power off, crash or stop</td>
<td>Switch over to secondary PLC. CI52x outputs are frozen during switch over period</td>
<td>Secondary</td>
</tr>
<tr>
<td>2</td>
<td>Secondary PLC power off, crash or stop</td>
<td>No switch over, process continues</td>
<td>Primary</td>
</tr>
<tr>
<td>3</td>
<td>Primary PLC loses connection to fieldbus CI52x modules while secondary still has connection.</td>
<td>Switch over to secondary PLC. CI52x outputs are frozen during switch over period</td>
<td>Primary</td>
</tr>
<tr>
<td>4</td>
<td>Secondary PLC loses connection to one or more CI52x modules</td>
<td>No switch over, process continues</td>
<td>Secondary</td>
</tr>
<tr>
<td>5</td>
<td>CI52x communication interface module is stopped/ powered off</td>
<td>No switch over, process continues</td>
<td>Primary and secondary</td>
</tr>
<tr>
<td>6</td>
<td>Connection lost in Field Ethernet network</td>
<td>Depending on Ethernet network structure, and redundancy mechanisms used a reconfiguration time exists</td>
<td>Depending on Sync or lifecom2 routing via the same network: case 4 and/ or case 7 may result</td>
</tr>
<tr>
<td>7</td>
<td>Sync and/ or lifecom2 broken</td>
<td>No switch over, process continues</td>
<td>Primary and secondary</td>
</tr>
</tbody>
</table>
### Case Use case Reaction Diagnosis message on *)

| 8  | Primary PLC loses connection to SCADA | SCADA is responsible to detect and switch over |  |
| 9  | Secondary PLC loses connection to SCADA | SCADA is responsible to detect and switch over |  |
| 10 | SCADA is broken | SCADA is responsible to detect and switch over |  |
| 11 | Manual switch over by user | Switch over to secondary PLC. CI52x outputs are frozen during switch over period |  |

*) Detailed diagnosis, see function block description

---

If lifecom2 is lost and if the PLC is in STOP mode: RUNTIME ERROR will not be TRUE. This is because Modbus is still responding even if PLC is in STOP mode.

---

The networks for larger systems are often seen as a separate entity and done by a separate company. Make sure to have the redundancy status information of the network at least in SCADA, to repair in time.

If the I/O field network responsibility is with the automation/PLC part, the redundancy status should be also monitored by the PLC. A warning to initiate repair may be created from the managed switches in the I/O field network.

Examples:

- Alarm output(s) wired e.g. to a CI52x input and related settings of the switch(es)
- Settings of the switch(es) to send e.g. SNMP traps, which can be received in PLC (AC500 SNMP library)
- Use of “automation switches” which can also communicate their status directly via Modbus.

---

Normally the HA-Modbus TCP library takes care of communication supervision, nevertheless if communication is cut completely, the CI52x communication interfaces and its I/O modules have to react on their own to achieve a bumpless or desired behaviour: The following parameters for the CI52x communication interfaces and I/O modules need to be considered:

- **CI52x: parameter “Timeout for Bus supervision”:**
  
  Allows to detect errors from CI side as well and take action to ensure a fail safe behavior if communication is cut. It can be set in 10 ms steps, if set to 0 no bus supervision is active [proposed value: 50 = 500 ms = default in Bulk data manager; should be set to a higher value of e.g. 65 if V2 CPU ports are used for field communication].

- **“Behaviour Outputs” at “Timeout for Bus supervision”:**
  
  This fail-safe parameter has to be consciously set: separate settings are possible for each module (and CI): “off”; “last” or “substitute”: 5 s, 10 s, ∞ s ¹)

Remarks:

¹) The parameters “Behaviour Outputs at comm. Error” is only analyzed if the Failsafe-mode is ON.

²) Both are CI52x parameters set e.g. via Bulk data manager tool in the program.
**System Structure**

A HA-Modbus TCP system is characterized by two AC500 PLCs with the following features:

- Identical application program with additional HA function blocks that are loaded in both PLCs
- Communication interface modules CI52x-MODTCP are connected via Modbus TCP
- Synchronization of both PLCs (sync and lifecom2 logical connections)

The application program should be divided into at least three main tasks/programs, which then communicate via internal structures of the libraries and dedicated internal memory areas for HA-Sync array and the Modbus CI52x memory(ies) CIModDataxx:

**Modbus Task**

- CIModCI52x updates INPUT and OUTPUT in each Modbus cycle for primary CPU. Secondary CPU only reads INPUT.

**Application Task**

- At the start of the Application task the InputRefresh program has to be called. It copies data from the Modbus via the structure CIModDataxx to the user variables, which were defined in BDM as signals (refer to BDM documentation, chapter 7)
- The main application programs should be in this task only and use these variables for the user defined functions. E.g. here the user programs and logic should be called and use utility blocks (which sync their historic data automatically) and HA_MOD_DATA_SYNC blocks for further user data which should be synced.
- Utility and HA_MOD_DATA_SYNC blocks data are stored in the HA Sync array.
- OutputRefresh program is called as a last step. It copies data from variables via structure CIModDataxx to Modbus, which were defined in BDM as signals (refer to BDM doc, chapter 7)

**HA task**

- HA_MOD_CONTROL uses UDP sync mechanism to share HA Sync array data with secondary CPU.

When inputs and outputs are accessed by Modbus, it will not allow application to read / write data because of the controlled access to common resources by multiple processes, this is implemented based on the concept of semaphore and ensures consistent data.
Sync Data

- User can use HA_MOD_DATA_SYNC_function block to synchronize data between Primary and secondary CPU.
- HA utility blocks also sync data between two CPUs (sync mechanism is inbuilt in these blocks). Both data will then be copied to HA Sync Array.
- HA SYNC Array is transferred via UDP to secondary CPU.

1 Eth Frame copies approx. 1412 data bytes of it. The number of Eth Frames needed to sync complete HA Sync Array depends on number of data sync bytes. Global variable iNoOfEthFrames gives the user this information, which should be used to calculate Application task cycle time. Check task configuration recommendation for details.

V2 needs one HA cycle to send 1 Eth frame data from Primary to secondary CPU and receive acknowledge from Secondary CPU. V3 CPU needs 2 HA cycle for above operation

- The HA program/ task should be called at highest priority as it is responsible for the core HA functionality and should be the fastest task.
- The Modbus task contains the Modbus communication function blocks at lower priority and (depending on CPU performance) also a faster cycle time to ensure sufficient update rates on the Modbus without overloading the CPU with communication.
- The application program parts should be called in the application task with even lower priority and a larger cycle time than above tasks.

The following picture indicates the placement of the main HA-Modbus TCP libraries function blocks to realize a High Availability System – with the tasks as described above.

Fig. 73: HA programming with the libraries: Red blocks are mandatory HA function blocks, dotted lines show optional function blocks

The main/ application program does not need to consider HA_MOD_DATA_SYNC separately, if it uses the HA utility function blocks from the HA-Modbus TCP library, as they internally synchronize necessary data.

Only for additional own calculated variables or function blocks where the user wants synchronization the use of the HA_MOD_DATA_SYNC function block is needed.
All the prepared sync data is synchronized with the secondary PLC. Typically only integral values (timers, counters, PID, ...) or settings which might have been received have to be synchronized, e.g. for fast start-up cases when a secondary CPU was restarted, as both PLCs are running and calculating closely in parallel and based on the same input values. For further information, please refer to library function block documentation and the example projects.

Following steps are performed:

- Status of the inputs connected to CI52x decentralized I/O stations is transferred to both PLCs simultaneously in every PLC cycle. They are received by the CI52x function block.
- In the HA program the HA_MOD_CONTROL function block collects all diagnosis, sync and lifecom2 data from the field and/or the other PLC. Then, it decides if a switchover is necessary based on a simple decision matrix.
- Data is synchronized as explained above.
- At the end of the program, the generated output values are send, by transferring from the primary PLC respective buffers to the CI52x-MODTCP communication interface module(s) via CI52x function block and Modbus TCP. The secondary PLC is prepared to send but stays “silent” (not sending output values).

### Task Configuration Recommendations for HA System

For a better performance of the HA system please consider the following recommendations in your project task configuration:

- Use the real time priorities for all HA related tasks (as proposed in the following table).
- Use the cycle times based on the formula (see the following table).
- If local I/O modules or coupler modules are used, consider their bus cycle options.
- Measure PLC and CPU load during operation. If the PLC load is higher than 40 % or CPU load higher than 60% then try the following steps (Note: watchdog stops the PLC at 80 % PLC load in V3 CPU):
  - Increase HA cycle time (e.g. to 8 ms / 12 ms / 24 ms, ...) and all others according to below formulas.
  - After each step check PLC load (must be < 40 %) and CPU load (must be < 60%).
  - Use a higher performant PLC type.
<table>
<thead>
<tr>
<th>Task</th>
<th>Priority</th>
<th>Cycle Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>HA</td>
<td>10 (high)</td>
<td>4 ms or higher (based on the target to keep CPU load &lt;60% or PLC load &lt;40% as HA base load)</td>
</tr>
<tr>
<td>Modbus</td>
<td>11 (medium)</td>
<td>Max ((HA cycle time *2), (3ms + roundup (#CI modules/2)))</td>
</tr>
<tr>
<td>Application</td>
<td>12 (low)</td>
<td>V2 CPU: Max ((Modbus cycle time *2), (iNoOfEthFrames * HA cycle time))</td>
</tr>
<tr>
<td></td>
<td></td>
<td>V3 CPU: 2 * Max ((Modbus cycle time *2), (iNoOfEthFrames * HA cycle time))</td>
</tr>
</tbody>
</table>

*iNoOfEthFrames* variable is defined in the Global variables of HA-Modbus TCP library.

**CPU choice, system size and performance indications (under above conditions):**

The graphs indicate values for the system configuration when using above formulas and assumptions for PLC- and CPU Load (PLC- and CPU Load are only different with V3 CPUs, where communication mainly increase CPU load).

![Modbus_cycle_time](image_url)
Example of a Utility Function Block – to enable bumpless Switch over

Consider the on-delay timer HA_MOD_TON (V2)/ HaModTon (V3).

Both PLCs require the same function block called in the program. Under normal operating conditions the elapsed time ET and output, Q of the timer is synchronized internally for both PLCs. ET and Q data are available and can be attached to local or global variables in the program as per application requirements. If PLC A shuts down due to a fault, the primary status switches over to PLC B.

In the event of a switch over, the moment the PLC B becomes the primary, the timer on this PLC will keep running. Till the time of PLC A failure the timer on the PLC B was synchronized, the actual process remains unaffected by the switch over.

Fault on Primary and Secondary PLC

The AC500 High Availability System itself only takes care of the first fault. In case of a second fault the primary PLC remains primary PLC until the second fault occurs. This results in no further switch overs (manual switch overs included). As the network with redundancy is expected to manage itself a second layer of independent redundancy exists without switching PLCs, but e.g. a ring also only can tolerate only one error, then repair is necessary.

Field I/O Network Topologies

Modbus TCP communication between PLC and communication interface modules CI521-MODTCP or CI522-MODTCP can be done using different network topologies. In the following subchapters different simple combinations with their pros and cons are explained.
If a CI52x module of a daisy chain is powered off, next following modules will lose connection/data provided there is no redundancy in the Ethernet network (e.g. ring and managed switch).

Star Topology (Non-redundant field device communication)

If no redundant field communication is required (e.g. for testing purposes) a star topology with a simple switch can be used. The switch (if managed and operating a ring or not) is a single point of failure, but PLCs are redundant.

Each CI52x module and PLCs in the network are connected to using one to one cable between device and Ethernet switch or a daisy chain of several.

Simple Ring Topology (smaller systems)

In a simple configuration, CI52x modules can be part of a ring if MRP protocol is used in the managed switches. Then the CI52x are connected from one to another device ("daisy chained") through e.g. two network switches. The redundancy protocol detects a closed ring and opens one port of a managed switch to avoid the ring. The user has to configure the necessary ring configurations and enable the ring manager for the used ring ports in one switch.

It is recommended that time interval between ETH cable disconnection and re-connection should be greater than 2-3 seconds.
**Standard Network Topology (large systems)**

In the standard redundant network, which is often done by 3rd party dedicated telecom companies, managed switches are used for every connection point to “the network”. It’s the networks (and operator) responsibility to repair any failure fast enough so that no influence on the HA system or its outputs occur.

The network can use other fast redundancy algorithms, also having other than ring structures, if redundancy links are activated fast enough.
Getting Started
HA-Modbus TCP Preconditions

HA-Modbus TCP is supported as of Automation Builder 2.0 and the before mentioned corresponding AC500 CPUs.

AC500 V3 PLC is not supporting external couplers (e.g. Ethernet or CAN) therefore onboard ETH1, ETH2 and CAN ports are to be used for communication.

3000 is the sync array size: Either 3000 HA_DATA_SYNC function block instances alone or together 3000 instances of HA_DATA_SYNC + HA utility function block can be used.

The HA-Modbus TCP system takes care of the first fault only. This fault must be visualized by the programmer and overall system (e.g. HMI, SCADA) to the operator, to plan a repair as soon as possible as redundancy might be lost. If more than one error occurs in the same PLC, restart the system to reset the error.

The SCADA/ HMI has to be used to only read/ use data from the primary PLC. Parameters and control data should be always written to both PLCs or has to be synchronized via the function block.

Quick Start list and Guidelines for Usage

Configuration of the HA system is done by the following steps (for details see the example documentations):

- Install the hardware ➤ Chapter 1.4.2.5.1.4 “Requirements” on page 2105
- Install Automation Builder including the latest libraries ➤ Chapter 1.4.2.5.1.4 “Requirements” on page 2105

Redundancy switch over timing should match the settings in the program and CI modules for time-out and freeze periods.
• Install the Bulk Data Manager tool (BDM) ☞ Chapter 1.4.2.5.1.3 “How to get and install the AC500 High Availability System Package” on page 2104
• Create a new project in Automation Builder for the chosen CPUs
• Assign the IP addresses in ≥ 2 different Ethernet networks:
  – SCADA network: SCADA, connected PLC A and PLC B
  – Field network: connected CI52x communication interface(s)
• Configure a network switch in the field network (if managed / redundant)
• Run BDM tool to configure CI52x network, export the files
• Import the Bulk data export files to the Automation Builder project
• Add Modbus TCP configuration
• Add Callback stop function HA_MOD_CALLBACK_STOP and call it in the system event “stop”
• Add optional HA utility function blocks or function block HA_MOD_DATASYNC
• Add the task configuration ☞ Chapter 1.4.2.5.1.6.1 “Task Configuration Recommendations for HA System” on page 2113
• Compile and download to both PLCs (simplified in V3 via integrated download manager)
• Create a boot project and RUN
• Operation: Test use cases e.g. by putting the primary PLC to STOP mode and observe the switch over

Library Overview

The libraries in Codesys V3 have an integrated documentation, where all technical details can be found. For Codesys V2 a separate library documentation exists:

HA-Modbus TCP Library

The following function blocks are contained in the libraries:
HA-Modbus TCP
Library
Primary CPU currently can read-out the diagnosis info (CI52x Function block outputs) from CI module only once, hence secondary PLC will not be able to read the diagnosis information from the CI52x module. 

So if any change happens in CI52x diagnosis it is not reflected in secondary CPU.

This can lead to different diagnosis information of CI52x module in Primary and Secondary CPU.

Hence it is recommended to customers that diagnosis information should be handled in the application (e.g. SCADA).

### 1.4.2.6 CAA libraries

#### 1.4.2.6.1 General

1.4.2.6.1.1 CAA Guidelines

CAA Guidelines

The guidelines for the CAA libraries correspond to the general guidelines for library development. For a detailed description see help chapter *Guidelines for Library Development*.

### 1.4.2.6.2 CAA_CallBack.Library

1.4.2.6.2.1 Overview Callback.Library

1.4.2.6.2.2 Data types / Enumerations

1.4.2.6.2.3 Modules

1.4.2.6.2.4 Examples
Overview Callback.Library

1.4.2.6.2.1.1 Overview CAA_Callback.Library............................................... 2122
1.4.2.6.2.1.2 CAA_Callback.Library............................................................... 2122
1.4.2.6.2.1.3 Sequence of execution.............................................................. 2123
1.4.2.6.2.1.4 Sequence of triggering.............................................................. 2123

Overview CAA_Callback.Library

The CODESYS runtime system provides different events for different purposes. Each event can be assigned to an event class. Further on each event arises from a certain source. With the functionality of this library it is possible to register user specific functions to handle these events.

CAA_Callback.Library

In the runtime system different events occur. Each event can be assigned to an event class. Further on each event arises from a certain source. The runtime system has an internal list of registered callback definitions. At each occurring event the runtime system checks its list of callback definitions and if applicable calls a callback function.

A callback definition consists of:
- Event
- Event class
- Event source
- Index of the function which should be called as soon as the event occurs.

Event e of class c from source s triggers the callback (e, c, s), if

\[
(e = e \text{ OR } e = \text{CB.ALL_EVENTS}) \text{ AND } \\left((c \text{ AND } c) > 0 \text{ OR } c = \text{CB.ALL_CLASSES}\right) \text{ AND } \\left(s = s \text{ OR } s = \text{CB.ALL_SOURCES}\right)
\]

A function which should be used as callback function, must have the following interface:

FUNCTION Callback<Name> : BOOL
VAR_INPUT
  dwSpec  : DWORD; (* CB.Event and CB.EventClass *)
  dwParam : DWORD; (* Application specific Parameter *)
  dwSource : DWORD; (* CB.EventSource *)
END_VAR

The data types are DWORDs in order to be compatible to CoDeSys V2.3. These however refer to the enumerations specified in brackets. Parameter dwSpec via function % CB.DecodeEvent and % CB.DecodeClass can be split in two variables of type % CB.Event and % CB.EventClass. The parameter dwSource can be converted to data type % CB.EventSource with the help of the DWORD_TO_INT conversion.

Remarks on the structure an the usage of callback functions:

The function name must have a prefix „Callback“.
The function must not contain any local variables.
The function must not be checked with breakpoints, ....
The function must be parameterized with property Enable system call,* !
Multiple callbacks may be assigned to one event. A callback (event, class, source, function index) however can get registered only once; if one tries to assign an identical callback, `CB.RegisterCallback` will dump an error.

After a reset of the PLC all registered callbacks will be deregistered immediately. Callbacks, which are possible registered on event AFTER_RESET (see `CB.Event`), will be called once before the automatic deregistration.

Regard that callback functions can be called immediately after the event has occurred. Thereby an currently running IEC task can be interrupted. The programmer must be aware that this might raise similar problems like there are in multitasking systems (data consistency etc.); besides that the run time of a callback function should be kept as short as possible, because it blocks the whole system.

The return value of the callback function is of no significance[1].

[1] Explanation: If the return value was of importance, then in case of multiple callback functions which are registered on the same event, it would not be clear which return value has priority.

**Sequence of execution**

( `CAA_Callback.Library` )

The execution of callbacks which are triggered by the same event follows the reversed order of definitions.[2]

[2] The reason: System components use the same mechanism and normally register callbacks before the application. Often the RESET event causes a system component to deallocate memory. If the application also has defined a callback on the same event, it might occur that the application tries to operate on deallocated memory area.

**Sequence of triggering**

( `CAA_Callback.Library` )

Some incidents trigger a sequence of events. These will be called in the following designated order:

<table>
<thead>
<tr>
<th>Event</th>
<th>STOP (only if PLC is running)</th>
</tr>
</thead>
<tbody>
<tr>
<td>PLC reset</td>
<td>BEFORE_RESET</td>
</tr>
<tr>
<td></td>
<td>AFTER_RESET</td>
</tr>
<tr>
<td>Shutdown</td>
<td>SHUTDOWN</td>
</tr>
<tr>
<td></td>
<td>STOP (only if PLC is running)</td>
</tr>
<tr>
<td>Online Change</td>
<td>ONLINECHANGE</td>
</tr>
<tr>
<td>Program download</td>
<td>STOP (only if PLC is running)</td>
</tr>
<tr>
<td></td>
<td>BEFORE_DOWNLOAD</td>
</tr>
</tbody>
</table>
Data types / Enumerations

1.4.2.6.2.2.1 CB.Event (ENUM) ................................................................................. 2124
1.4.2.6.2.2.2 CB.EventClass (ENUM) .......................................................................... 2126
1.4.2.6.2.2.3 CB.EventSource (ENUM) ........................................................................ 2126
1.4.2.6.2.2.4 CB.Error (ENUM) .................................................................................... 2127
1.4.2.6.2.2.5 CB.Callback (STRUCT; Prefix cb) ......................................................... 2127
1.4.2.6.2.2.6 CB.gc_cbNull (VAR GLOBAL CONSTANT) ........................................... 2127

CB.Event (ENUM)

This enumeration describes all standardized events of the CAA_Callback.Library. An X in the table shown below indicates that each CAA PLC supports the respective event. The source of those events is CB.RUNTIME.

PLC-specific events can be used in the following number areas:
900-999, 1900-1999, 2900-2999, 3900-3999, 4900-4999, 5900-5999, 6900-6999, 7000-7999, from 10000

The value of each event at the same results in a classification:

| 1000-1999 | ONLINE_EVENTS |
| 2000-2999 | INFOS |
| 3000-3999 | WARNINGS |
| 4000-4999 | RTS_ERRORS |
| 5000-5999 | SYSTEM_EXCEPTIONS |
| 6000-6999 | INTERRUPTS |
| 7000-7499 | IO |
| 8000-9899 | FIELDBUS |
| 9900-9999 | TIMERS |
| ab 10000 | MANUF_SPEC |

<p>| CB.NO_EVENT | 0 | X |
| CB.ALL_EVENTS | -1 | X |
| CB.START | 1000 | start | X |
| CB.STOP | 1001 | stop | X |
| CB.BEFORE_RESET | 1002 | before reset | X |
| CB.AFTER_RESET | 1003 | after reset | X |
| CB.SHUTDOWN | 1004 | shutdown | X |
| CB.ONLINE_CHANGE | 1005 | after CodeInit at Online Change |
| CB.BEFORE_DOWNLOAD | 1006 | before program download | X |
| CB.TASKCODE_NOT_CALLED | 1007 | task code will not be called |
| CB.TIMER | 1008 | scheduler tick (not in case of stop) |
| CB.DEBUG_LOOP | 1009 | communication timer tick (if PLC is stopped on breakpoint) | X |
| CB.SCHEDULE | 1010 | scheduler tick (even at stop) |</p>
<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
<th>Errors</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>CB.ERR_WATCHDOG</td>
<td>software watchdog</td>
<td>4000</td>
<td></td>
</tr>
<tr>
<td>CB.ERR_HARDWARE_WATCHDOG</td>
<td>hardware watchdog</td>
<td>4001</td>
<td></td>
</tr>
<tr>
<td>CB.ERR_FIELDBUS</td>
<td>field bus watchdog</td>
<td>4002</td>
<td>X</td>
</tr>
<tr>
<td>CB.ERR_IOUPDATE</td>
<td>E/A error</td>
<td>4003</td>
<td>X</td>
</tr>
<tr>
<td>CB.ERR_POWERFAIL</td>
<td>power failure</td>
<td>4004</td>
<td></td>
</tr>
<tr>
<td>CB.EXCPT_ILLEGAL_INSTRUCTION</td>
<td>invalid instruction</td>
<td>5000</td>
<td></td>
</tr>
<tr>
<td>CB.EXCPT_ACCESS_VIOLATION</td>
<td>access violation</td>
<td>5001</td>
<td></td>
</tr>
<tr>
<td>CB.EXCPT_PRIV_INSTRUCTION</td>
<td>privileged instruction</td>
<td>5002</td>
<td></td>
</tr>
<tr>
<td>CB.EXCPT_IN_PAGE_ERROR</td>
<td>access violation</td>
<td>5003</td>
<td></td>
</tr>
<tr>
<td>CB.EXCPT_STACK_OVERFLOW</td>
<td>stack overflow</td>
<td>5004</td>
<td></td>
</tr>
<tr>
<td>CB.EXCPT_MISALIGNMENT</td>
<td>misalignment</td>
<td>5005</td>
<td></td>
</tr>
<tr>
<td>CB.EXCPT_ARRAYBOUNDS</td>
<td>array bounds</td>
<td>5006</td>
<td></td>
</tr>
<tr>
<td>CB.EXCPT_DIVIDE_BYZERO</td>
<td>division by zero</td>
<td>5007</td>
<td></td>
</tr>
<tr>
<td>CB.EXCPT_OVERFLOW</td>
<td>overflow</td>
<td>5008</td>
<td></td>
</tr>
<tr>
<td>CB.EXCPT_NONCONTINUABLE</td>
<td>not continuable exception</td>
<td>5009</td>
<td></td>
</tr>
<tr>
<td>CB.EXCPT_NO_FPU_AVAILABLE</td>
<td>no FPU available</td>
<td>5500</td>
<td></td>
</tr>
<tr>
<td>CB.EXCPT_FPU_ERROR</td>
<td>FPU error</td>
<td>5501</td>
<td></td>
</tr>
<tr>
<td>CB.EXCPT_FPU_DENORMAL_OPERAND</td>
<td>invalid operand</td>
<td>5502</td>
<td></td>
</tr>
<tr>
<td>CB.EXCPT_FPU_DIVIDE_BYZERO</td>
<td>division by zero</td>
<td>5503</td>
<td></td>
</tr>
<tr>
<td>CB.EXCPT_FPU_INVALID_OPERATION</td>
<td>invalid operation</td>
<td>5504</td>
<td></td>
</tr>
<tr>
<td>CB.EXCPT_FPU_OVERFLOW</td>
<td>overflow</td>
<td>5505</td>
<td></td>
</tr>
<tr>
<td>CB.EXCPT_FPU_STACK_CHECK</td>
<td>FPU-specific exception</td>
<td>5506</td>
<td></td>
</tr>
<tr>
<td>CB.INTERRUPT_0</td>
<td></td>
<td>6000</td>
<td></td>
</tr>
<tr>
<td>CB.INTERRUPT_1</td>
<td></td>
<td>6001</td>
<td></td>
</tr>
<tr>
<td>CB.INTERRUPT_2</td>
<td></td>
<td>6002</td>
<td></td>
</tr>
<tr>
<td>CB.INTERRUPT_3</td>
<td></td>
<td>6003</td>
<td></td>
</tr>
<tr>
<td>CB.INTERRUPT_4</td>
<td></td>
<td>6004</td>
<td></td>
</tr>
<tr>
<td>CB.INTERRUPT_5</td>
<td></td>
<td>6005</td>
<td></td>
</tr>
<tr>
<td>CB.INTERRUPT_6</td>
<td></td>
<td>6006</td>
<td></td>
</tr>
<tr>
<td>CB.INTERRUPT_7</td>
<td></td>
<td>6007</td>
<td></td>
</tr>
<tr>
<td>CB.INTERRUPT_8</td>
<td></td>
<td>6008</td>
<td></td>
</tr>
<tr>
<td>CB.INTERRUPT_9</td>
<td></td>
<td>6009</td>
<td></td>
</tr>
<tr>
<td>CB.INTERRUPT_10</td>
<td></td>
<td>6010</td>
<td></td>
</tr>
<tr>
<td>CB.INTERRUPT_11</td>
<td></td>
<td>6011</td>
<td></td>
</tr>
<tr>
<td>CB.INTERRUPT_12</td>
<td></td>
<td>6012</td>
<td></td>
</tr>
<tr>
<td>CB.INTERRUPT_13</td>
<td></td>
<td>6013</td>
<td></td>
</tr>
<tr>
<td>CB.INTERRUPT_14</td>
<td></td>
<td>6014</td>
<td></td>
</tr>
<tr>
<td>CB.INTERRUPT_15</td>
<td></td>
<td>6015</td>
<td></td>
</tr>
<tr>
<td>CB.INTERRUPT_255</td>
<td></td>
<td>6255</td>
<td></td>
</tr>
<tr>
<td>CB.AFTER_READING_INPUTS</td>
<td>after reading the inputs</td>
<td>7000</td>
<td></td>
</tr>
<tr>
<td>CB.BEFORE_WRITING_OUTPUTS</td>
<td>before writing the outputs</td>
<td>7001</td>
<td></td>
</tr>
<tr>
<td>Address</td>
<td>Description</td>
<td></td>
<td></td>
</tr>
<tr>
<td>-----------</td>
<td>---------------------------------</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8000-809 9</td>
<td>reserved for CAN library</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8100-819 9</td>
<td>reserved for Profinet library</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8200-829 9</td>
<td>reserved for Ethernet library</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CB.TIMER_1</td>
<td>reserved for CAA.Timer.lib</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9900-999 9</td>
<td>reserved for CAA.Timer.lib</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**CB.EventClass (ENUM)**

This enumeration describes the event classes of the %CAA_Callback.Library.%

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CB.ALL_CLASSES</td>
<td>-1</td>
</tr>
<tr>
<td>CB.NO_CLASS</td>
<td>0</td>
</tr>
<tr>
<td>CB.ONLINE_EVENTS</td>
<td>16#0001, common online events</td>
</tr>
<tr>
<td>CB.INfos</td>
<td>16#0002, information events</td>
</tr>
<tr>
<td>CB.Warnings</td>
<td>16#0004, warnings</td>
</tr>
<tr>
<td>CB.RTS_ERRORS</td>
<td>16#0008, runtime errors</td>
</tr>
<tr>
<td>CB.SYSTEM_EXCEPTIONS</td>
<td>16#0010, system exception error</td>
</tr>
<tr>
<td>CB.INTERRUPTS</td>
<td>16#0020, interrupts</td>
</tr>
<tr>
<td>CB.IO</td>
<td>16#0040, I/O events</td>
</tr>
<tr>
<td>CB.FIELDBUS</td>
<td>16#0080, field bus events</td>
</tr>
<tr>
<td>CB.TIMERS</td>
<td>16#0100, timer event</td>
</tr>
<tr>
<td>CB.MANUF_SPEC</td>
<td>16#0200, PLC-specific events</td>
</tr>
</tbody>
</table>

**CB.EventSource (ENUM)**

This enumeration describes the triggers of an event.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CB.ALL_SOURCES</td>
<td>-1</td>
</tr>
<tr>
<td>CB.NO_SOURCE</td>
<td>0</td>
</tr>
<tr>
<td>CB.RUNTIME</td>
<td>16#0001, runtime system</td>
</tr>
<tr>
<td>CB.SYSTEM</td>
<td>16#0002, (operating)system</td>
</tr>
<tr>
<td>CB.IECTASK</td>
<td>16#0004, task management</td>
</tr>
<tr>
<td>CB.IEC_PROGRAM</td>
<td>16#0008, IEC program</td>
</tr>
<tr>
<td>CB.DRIVER</td>
<td>16#0010, driver</td>
</tr>
</tbody>
</table>
CB.Error (ENUM)

This data structure describes errors which might occur when using functions of theCAA_Callback library. Error range 20000 – 20099 is reserved for this library in the prefix registration for libraries.

<table>
<thead>
<tr>
<th>CB.Error Code</th>
<th>Error Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CB.NO_ERROR</td>
<td>0 no error</td>
</tr>
<tr>
<td>CB.FIRST_ERROR</td>
<td>20000 start of standard error area</td>
</tr>
<tr>
<td>CB.HANDLE_INVALID</td>
<td>20001 invalid handle</td>
</tr>
<tr>
<td>CB.UNKNOWN_EVENT</td>
<td>20002 unknown event</td>
</tr>
<tr>
<td>CB.CALL_BACK_NOT_REMOVABLE</td>
<td>20003 callback cannot be deactivated</td>
</tr>
<tr>
<td>CB.WRONG_ARGUMENT</td>
<td>20004 for example: a null pointer was passed</td>
</tr>
<tr>
<td>CB.NO_MEMORY</td>
<td>20005</td>
</tr>
<tr>
<td>CB.EVENT_EXISTS</td>
<td>20006</td>
</tr>
<tr>
<td>CB.CALL_CALLBACKS_FAILED</td>
<td>20007</td>
</tr>
<tr>
<td>CB.NO_SYSTEM_EVENT</td>
<td>20008</td>
</tr>
<tr>
<td>CB.BUFFER_NOT_AVAILABLE</td>
<td>20009</td>
</tr>
<tr>
<td>CB.SYSTEM_EVENT</td>
<td>20010 This event cannot be sent by the user, it is sent by the system</td>
</tr>
<tr>
<td>CB.FIRST_MF</td>
<td>20050 manufacturer-specific error</td>
</tr>
<tr>
<td>CB.LAST_ERROR</td>
<td>20099 end of the standard error area</td>
</tr>
</tbody>
</table>

CB.Callback (STRUCT; Prefix cb)

This data structure of theCAA_Callback.Library describes the callback conditions and -function.

<table>
<thead>
<tr>
<th>Member</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pPOUFunc</td>
<td>CAA.PVOID</td>
<td>address of the function to be called</td>
</tr>
<tr>
<td>eEvent</td>
<td>CB.Event</td>
<td>trigger event or CB.ALL_EVENTS</td>
</tr>
<tr>
<td>eClass</td>
<td>CB.EventClass</td>
<td>event class (mask)</td>
</tr>
<tr>
<td>eSource</td>
<td>CB.EventSource</td>
<td>event source</td>
</tr>
</tbody>
</table>

CB.gc_cbNull (VAR GLOBAL CONSTANT)

This constant of theCAA_Callback.Library can be used for the initialization of not used inputs of type CB.CALLBACK.
CB.gc_cbNull | CB.Callback | pPOUFunc := CAA.gc_pNULL;  
event := CB.NO_EVENT; 
eClass := CB.NO_CLASS; 
eSource := CB.NO_SOURCE;

| Modules |
|------------------|------------------|
| 1.4.2.6.2.3.1 | CB.RegisterCallback (FUN) | 2128 |
| 1.4.2.6.2.3.2 | CB.RegisterCallbackInterface (FUN) | 2128 |
| 1.4.2.6.2.3.3 | CB.UnregisterCallback (FUN) | 2129 |
| 1.4.2.6.2.3.4 | CB.GetCallback (FUN) | 2129 |
| 1.4.2.6.2.3.5 | CB.IsHandleValid (FUN) | 2129 |
| 1.4.2.6.2.3.6 | CB.GetNumberOfActiveCallbacks (FUN) | 2129 |
| 1.4.2.6.2.3.7 | CB.GetHandleOfCallback (FUN) | 2130 |
| 1.4.2.6.2.3.8 | CB.PostEvent (FUN) | 2130 |
| 1.4.2.6.2.3.9 | CB.SendEvent (FUN) | 2130 |
| 1.4.2.6.2.3.10 | CB.CallFunctionByIndex (FUN) | 2131 |
| 1.4.2.6.2.3.11 | CB.DecodeEvent (FUN) | 2131 |
| 1.4.2.6.2.3.12 | CB.DecodeClass (FUN) | 2131 |
| 1.4.2.6.2.3.13 | CB.EncodeSpec (FUN) | 2132 |

CB.RegisterCallback (FUN)  
( © CAA_Callback.Library)  

With this function new callbacks can be activated.

Table 381: Input:  
<table>
<thead>
<tr>
<th>cbNew</th>
<th>CB.Callback</th>
<th>description of the callback</th>
</tr>
</thead>
<tbody>
<tr>
<td>peError</td>
<td>POINTER TO CB.Error</td>
<td>pointer on error ID</td>
</tr>
</tbody>
</table>

Table 382: Output:  
| hCallback | CAA.HANDLE | handle of the new callback; 0 in case of error. |

CB.RegisterCallbackInterface (FUN)  
( © CAA_Callback.Library)  

Not yet implemented.

Table 383: Input:  
<table>
<thead>
<tr>
<th></th>
<th>Interface</th>
</tr>
</thead>
<tbody>
<tr>
<td>method</td>
<td></td>
</tr>
<tr>
<td>eEvent</td>
<td>CB.Event</td>
</tr>
<tr>
<td>eClass</td>
<td>CB.EventClass</td>
</tr>
<tr>
<td>eSource</td>
<td>CB.EventSource</td>
</tr>
</tbody>
</table>
CB.UnregisterCallback (FUN)
This function of the %CAA_Callback.Library can be used to delete callbacks.

Table 385: Input:
<table>
<thead>
<tr>
<th>hHandle</th>
<th>CAA.HANDLE</th>
<th>callback handle</th>
</tr>
</thead>
</table>

Table 386: Output:
<table>
<thead>
<tr>
<th>eError</th>
<th>CB.Error</th>
<th>error description</th>
</tr>
</thead>
</table>

CB.GetCallback (FUN)
This function of the %CAA_Callback.Library serves to get a function via its handle.

Table 387: Input:
<table>
<thead>
<tr>
<th>hHandle</th>
<th>CAA.HANDLE</th>
<th>callback handle</th>
</tr>
</thead>
<tbody>
<tr>
<td>pCallback</td>
<td>POINTER TO CB.CALLBACK</td>
<td>return: description of the callback conditions and the callback function</td>
</tr>
</tbody>
</table>

Table 388: Output:
<table>
<thead>
<tr>
<th>eError</th>
<th>CB.Error</th>
<th>error description</th>
</tr>
</thead>
</table>

CB.IsHandleValid (FUN)
This function of the %CAA_Callback.Library serves to check whether a callback handle is valid.

Table 389: Input:
<table>
<thead>
<tr>
<th>hHandle</th>
<th>CAA.HANDLE</th>
<th>callback handle</th>
</tr>
</thead>
<tbody>
<tr>
<td>peError</td>
<td>POINTER TO CB.Error</td>
<td>pointer on error ID</td>
</tr>
</tbody>
</table>

Table 390: Output:
<table>
<thead>
<tr>
<th>xValid</th>
<th>BOOL</th>
<th>TRUE: handle is valid; FALSE: handle is invalid</th>
</tr>
</thead>
</table>

CB.GetNumberActiveCallbacks (FUN)
This function of the %CAA_Callback.Library provides the number of active callbacks.
Table 391: Input:

| peError | POINTER TO CB.Error | pointer on error ID |

Table 392: Output:

| ctNumCallbacks | CAA.COUNT | number of active callbacks |

**CB.GetHandleOfCallback (FUN)**

This function of the %CAA_Callback.Library gets a callback handle via the callback number.

Table 393: Input:

| ctNumber | UINT | callback number (value between 1 and %CB.GetNumberActiveCallbacks) |
| peError | POINTER TO CB.Error | pointer on error ID |

Table 394: Output:

| hCallback | CAA.HANDLE | callback handle |

**CB.PostEvent (FUN)**

This function of the %CAA_Callback.Library serves to trigger an event in order to call one or multiple callbacks.

The function returns immediately. The callbacks will be executed in the background. For this reason the function is suitable for triggering events e.g. in interrupt service routines.

Table 395: Input:

| eEvent | CB.EVENT | event |
| eSource | CB.EventSource | event source |
| dwParam | DWORD | parameter for the receiver of the event |

Table 396: Output:

| eError | CB.Error | error description |

**CB.SendEvent (FUN)**

This function of the %CAA_Callback.Library serves to trigger an event in order to call one or multiple callbacks with this event. It is recommended that IEC programs using this function do set IECPROGRAM as event source and ALL_CLASSES as event class. If the function is used by drivers, it is advisable to use DRIVER as event source. The function will not return before all callbacks have been executed. Thus it is made sure, that all interested parties have terminated their respective reaction until the function returns.
Table 397: Input:

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>eEvent</td>
<td>CB.EVENT</td>
<td>event</td>
</tr>
<tr>
<td>eSource</td>
<td>CB.EventSource</td>
<td>event source</td>
</tr>
<tr>
<td>dwParam</td>
<td>DWORD</td>
<td>parameter for the receiver of the event</td>
</tr>
</tbody>
</table>

Table 398: Output:

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>eError</td>
<td>CB.Error</td>
<td>error ID</td>
</tr>
</tbody>
</table>

**CB.CallFunctionByIndex (FUN)**

This function of the `%CAA_Callback.Library` serves to call an IEC function indirectly via its POU index. The following constraints have to be regarded:

- The name of the function must have a prefix „Callback“.
- The function must have a return value of a data type of 4 byte length.
- The function must have exactly three parameters of a data type of 4 byte length.

Table 399: Input:

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>pPOUFunc</td>
<td>CAA.PVOID</td>
<td>POU-Index of the function to be called</td>
</tr>
<tr>
<td>dwParam1</td>
<td>DWORD</td>
<td>first parameter of the function to be called</td>
</tr>
<tr>
<td>dwParam2</td>
<td>DWORD</td>
<td>second parameter of the function to be called</td>
</tr>
<tr>
<td>dwParam3</td>
<td>DWORD</td>
<td>third parameter of the function to be called</td>
</tr>
<tr>
<td>peError</td>
<td>POINTER TO CB.Error</td>
<td>pointer on error ID</td>
</tr>
</tbody>
</table>

Table 400: Output:

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>dwValue</td>
<td>DWORD</td>
</tr>
</tbody>
</table>

**CB.DecodeEvent (FUN)**

This function of the `%CAA_Callback.Library` serves to extract value eEvent of type `%CB.Event` from parameter dwSpec of the callback function.

Table 401: Input:

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>dwSpec</td>
<td>DWORD</td>
</tr>
</tbody>
</table>

Table 402: Output:

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>eEvent</td>
<td>CB.Event</td>
<td>event name for the callback</td>
</tr>
</tbody>
</table>

**CB.DecodeClass (FUN)**

This function serves to extract class eClass vom Type `%CB.EventClass` from parameter dwSpec of the callback function.
**Table 403: Input:**

| dwSpec | DWORD | parameter of the callback function |

**Table 404: Output:**

| eClass | CB.EventClass | event class of the callback |

**CB.EncodeSpec (FUN)**

This function of the `CAA_Callback.Library` serves to create parameter `dwSpec` of type `DWORD`, which is necessary for callback functions, out of two values of type `CB.EventClass` and `CB.EventSource`.

**Table 405: Input:**

<table>
<thead>
<tr>
<th>eEvent</th>
<th>CB.EventEvent</th>
<th>parameter of the callback function</th>
</tr>
</thead>
<tbody>
<tr>
<td>eClass</td>
<td>CB.EventClass</td>
<td>parameter of the callback function</td>
</tr>
</tbody>
</table>

**Table 406: Output:**

| dwSpec | DWORD | parameter of the callback function |

**Examples**

1.4.2.6.2.4.1 Inserting Callbacks (EXAMPLE)................................................ 2132
1.4.2.6.2.4.2 Deleting Callbacks..................................................................... 2133
1.4.2.6.2.4.3 Triggering callbacks (EXAMPLE).............................................. 2134
1.4.2.6.2.4.4 Indirect function call (EXAMPLE).............................................. 2134

**Inserting Callbacks (EXAMPLE)**

( `CAA_Callback.Library`) The following code lines insert two callbacks. The first calls function BeforeResetFunction, which is called before the PLC is reset; the second calls function ErrorFunction as soon as an event of class RTS_ERRORS or SYSTEM_EXCEPTIONS occurs.

```c
cbNew.eEvent := CB.BEFORE_RESET;
cbNew.eClass := CB.ALL_CLASSES;
cbNew.eSource := CB.ALL_SOURCES;
cbNew.pPOUFunc := ADR(CallbackBeforeReset);
CB.RegisterCallback(cbNew);

cbNew.eEvent := CB.ALL_EVENTS;
```

```c
cbNew.eClass := CB.RTS_ERRORS OR CB.SYSTEM_EXCEPTIONS;
```

```c
cbNew.eSource := CB.ALL_SOURCES;
```

```c
cbNew.iPOUFunc := ADR(CallbackError);
```

```c
CB.RegisterCallback(cbNew);
```
Deleting Callbacks

1.4.2.6.2.4.2.1 Handleknown (EXAMPLE) ..................................................... 2133
1.4.2.6.2.4.2.2 HandleUnknown (EXAMPLE) ................................................. 2133

Handleknown (EXAMPLE)

This example of the `CAA_Callback.Library` assumes that the handle, which was returned during the registration of the callback, was stored:

(* define callback *)
caNew.eEvent := CB.BEFORE_RESET;
caNew.eClass := CB.ALL_CLASSES;
caNew.eSource := CB.ALL_SOURCES;
caNew.iPOUFunc := ADR(CallbackBeforeReset);

(* insert callback and store handle *)
ha := CB.RegisterCallback(caNew);

(* delete callback *)
IF CB.IsHandleValid(ha) THEN
CB.UnregisterCallback(ha);
END_IF

HandleUnknown (EXAMPLE)

Example of the `CAA_Callback.Library`:
If the handle is not known any more, the callback can be deleted as shown in the following example:

(* define callback *)
caDelete.eEvent := CB.BEFORE_RESET;
caDelete.eClass := CB.ALL_CLASSES;
caDelete.eSource := CB.ALL_SOURCES;
caDelete.iPOUFunc := ADR(CallbackBeforeReset);

(* browse all callbacks *)
FOR i:=1 TO CB.GetNumberActiveCallbacks() DO
 (* get handle *)
ha := CB.GetHandleOfCallback(i);

 (* if searched callback *)
IF CB.GetCallback(ha)=caDelete THEN
 (* delete callback *)
CB.UnregisterCallback(ha);
EXIT;
END_IF
Triggering callbacks (EXAMPLE)

The following example of the \texttt{CAA\textunderscore Callback.Library} simulates an event, which effects that the system will trigger the callbacks registered on this event:

\begin{verbatim}
CB.PostEvent
(     eEvent := CB.EXCPT\_OVERFLOW,
     eSource := CB.IECPROGRAM,
     dwParam := 0
);
\end{verbatim}

Indirect function call (EXAMPLE)

\begin{verbatim}
( \texttt{CAA\textunderscore Callback.Library})
1. Implementation of a function:
FUNCTION CallbackTestFunction : DWORD
    VAR_INPUT
    tTime    : TIME;
    udiCount : UDINT;
    dwState  : DWORD;
END_VAR

2. Indirect call of this function:
IF xCallTest THEN
    xCallTest := FALSE;
    dwValue := CB.CallFunctionByIndex
(     pPOUFunc := ADR(CallbackTestFunction),
     dwParam1  := TIME\_TO\_DWORD(TIME()),
     dwParam2  := UDINT\_TO\_DWORD(udiCount),
     dwParam3 := dwState
);
END_IF
\end{verbatim}
Overview CAA_CiA405.Library

This specification uses the standardized CANopen interface for IEC 61131-3 programmable devices like PLC.

The devices use communication techniques which conform to those described in the CiA Draft Standard DS-301 (Application Layer and Communication Profile), Draft Standard Proposal DSP-302 (Framework for Programmable CANopen Devices) and Draft Standard Proposal DS-405 (Interface and Device Profile for IEC 61131-3 Programmable Devices). These documents should be consulted in parallel to this specification.

References:
- /DS-301/ CiA Draft Standard DS 301 V4.02 CANopen Application Layer and Communication Profile
- /DS-405/ CiA Draft Standard Proposal DS 405 V2.0 CANopen interface and Device Profile for IEC 61131-3 Programmable Devices

In general, generating an application implements the handling of up to five interfaces:

**Fig. 80: Generating an application /handling interfaces**

This description just covers the interface:

A. that means, the access to a CANopen communication system from within an IEC 61131-3 program based on calls to function blocks

This description does not (yet) cover the interfaces:
B. utility functions for debugging, monitoring and network management
C. interface between CANopen tools and IEC 61131-3 programming environment
D. manufacturer specific interface between programming environment and PLC runtime
E. CANopen tool specific interface between configuration tool and network

NOTICE!
The figure describes not necessarily the use of different tools for programming and configuration. One tool may handle both functionality and hide the interfaces. An example is given by the programming tool CODESYS (see dotted line), where the CANopen configuration and the runtime software on the PLC is included.

Application interface
The needs of any application from within an IEC 61131-3 level it is to read from the device object directory of a CANopen connected node or to write to it (SDO). Further elements of a CANopen communication are Process Data Objects (PDO) which are supported in the process image within the IEC application as inputs (receive PDO; i.e.: %IX.. ) or outputs (transmit PDO; i.e.: %QW.. ). Received asynchronous error messages (EMCY) are registered in a buffer for each node. The network status is controlled by the Network management (NMT). Node guarding functionality is available with CANopen as well as synchronized communication for process data.

The chapters of the current description describe the needs of a function block library for CANopen communication.

Data types / Enumerations

1.4.2.6.3.2.1 CIA405.DEVICE_STATE (ENUM) .................................................. 2136
1.4.2.6.3.2.2 CIA405.SDO_MODE (ENUM) .................................................. 2136
1.4.2.6.3.2.3 CIA405.TRANSITION_STATE (ENUM) ..................................... 2138
1.4.2.6.3.2.4 CIA405.TRANSACTION_STATE (ENUM) .................................. 2138
1.4.2.6.3.2.5 CIA405.ERROR (ENUM) ......................................................... 2138
1.4.2.6.3.2.6 CIA405.EMCY_DATA (STRUCT) .............................................. 2138
1.4.2.6.3.2.7 CIA405.EMCY_ERROR (STRUCT) .......................................... 2138
1.4.2.6.3.2.8 CIA405.KERNEL_ERROR_INFO (STRUCT)............................ 2139
1.4.2.6.3.2.9 CIA405.DEVICE........................................................................ 2139
1.4.2.6.3.10 CIA405.SDO_ERROR............................................................. 2139
1.4.2.6.3.11 CIA405.CANOPEN_KERNEL_ERROR................................... 2139

Data types / Enumerations

The data types of the CIA405.Library described in this chapter are mandatory by means of function library needs.

CIA405.DEVICE_STATE (ENUM)

( CIA405.Library)
Type CIA405.STATE describes the state of the CANopen network layer, as defined in /DS-301/. The states INIT, RESET_COMM, RESET_APP, PRE_OPERATIONAL, STOPPED, OPERATIONAL correspond to the same states in /DS-301/. The state UNKNOWN must be used, if the actual state of the device is not known (in example, if no guarding of the device is performed). The state NOT_AVAIL must be used, if it is known, that the device is not available (in example, if guarding is performed and the device does not answer).

<table>
<thead>
<tr>
<th>State</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>INIT</td>
<td>0</td>
</tr>
<tr>
<td>RESET_COMM</td>
<td>1</td>
</tr>
<tr>
<td>RESET_APP</td>
<td>2</td>
</tr>
<tr>
<td>PRE_OPERATIONAL</td>
<td>3</td>
</tr>
<tr>
<td>STOPPED</td>
<td>4</td>
</tr>
<tr>
<td>OPERATIONAL</td>
<td>5</td>
</tr>
<tr>
<td>UNKNOWN</td>
<td>6</td>
</tr>
<tr>
<td>NOT_AVAIL</td>
<td>7</td>
</tr>
</tbody>
</table>

The state, delivered by $\text{CIA405.GET\_STATE}$, is:

- If node guarding / heartbeat is not active for the selected DEVICE: UNKNOWN (6) (even if a boot-up message is received)
- If heartbeat is active but the NMT-master is not the Heartbeat-Consumer: UNKNOWN (6).
- If node guarding / heartbeat is active and the device reports its NMT state: reported NMT state (value, defined in DS301 V4.02 chap. "Error Guarding Protocol")
- If node guarding / heartbeat is active and the device does not report its NMT state correctly before timeout, i.e. "node life time" (Nodes Guarding Protocol) respective "Heartbeat Consumer Time" (Heartbeat Protocol): NOT_AVAIL (7)

### CIA405.SDO_MODE (ENUM)

( $\text{CAA\_Cia405.Library}$)

The Parameter mode for $\text{CIA405.SDO\_WRITE}$ is one value of the following set:

<table>
<thead>
<tr>
<th>Mode</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>AUTO</td>
<td>0</td>
</tr>
<tr>
<td>EXPEDITED</td>
<td>1</td>
</tr>
<tr>
<td>SEGMENTED</td>
<td>2</td>
</tr>
<tr>
<td>BLOCK</td>
<td>3</td>
</tr>
</tbody>
</table>

In “Auto” mode the client uses the most efficient transfer protocol to transfer the data from or to the server. He queries the possibilities and compromise the best way for both parties. The SDO protocol is defined in /DS 301/.
CIA405.TRANSITION_STATE (ENUM)
( % CAA_Cia405.Library)
Type CIA405.TRANSITION_STATE describes the state transitions of the CANopen network layer, as defined in /DS-301/.

<table>
<thead>
<tr>
<th>State Description</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>STOP_REMOTE_NODE</td>
<td>16#04</td>
</tr>
<tr>
<td>START_REMOTE_NODE</td>
<td>16#05</td>
</tr>
<tr>
<td>RESET_NODE</td>
<td>16#06</td>
</tr>
<tr>
<td>RESET_COMMUNICATION</td>
<td>16#07</td>
</tr>
<tr>
<td>ENTER_PRE_OPERATIONAL</td>
<td>16#7F</td>
</tr>
<tr>
<td>ALL_EXCEPT_NMT_AND_SENDER</td>
<td>16#800</td>
</tr>
</tbody>
</table>

A flag to use in combination. Must be used if NMT command should be addressed to all devices (DEVICE=0) except to the CANopen-Manager.

CIA405._ERROR (ENUM)
( % CAA_Cia405.Library)
The error codes occur on the ERROR output on some building blocks are part of this set:

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>no error</td>
</tr>
<tr>
<td>10300</td>
<td>first library specific error</td>
</tr>
<tr>
<td>10301</td>
<td></td>
</tr>
<tr>
<td>10302</td>
<td></td>
</tr>
<tr>
<td>10303</td>
<td></td>
</tr>
<tr>
<td>10316</td>
<td></td>
</tr>
<tr>
<td>10317</td>
<td></td>
</tr>
<tr>
<td>10332</td>
<td>first manufacturer specific error</td>
</tr>
<tr>
<td>10399</td>
<td>last library specific error</td>
</tr>
</tbody>
</table>

CIA405.EMCY_DATA (STRUCT)
( % CAA_Cia405.Library)

<table>
<thead>
<tr>
<th>Device</th>
<th>DEVICE</th>
</tr>
</thead>
</table>

CIA405.EMCY_ERROR (STRUCT)
Type CIA405.EMCY_ERROR contains emergency error information, as specified in /DS-301/:
<table>
<thead>
<tr>
<th>EMCY_ERROR_CODE</th>
<th>WORD</th>
</tr>
</thead>
<tbody>
<tr>
<td>ERROR_REGISTER</td>
<td>BYTE</td>
</tr>
<tr>
<td>ERROR_FIELD</td>
<td>ARRAY [1..5] OF BYTE</td>
</tr>
</tbody>
</table>

**CIA405.KERNEL_ERROR_INFO** (STRUCT)

( % CAA_CiA405.Library)

<table>
<thead>
<tr>
<th>wError</th>
<th>CANOPENKERNEL_ERROR</th>
</tr>
</thead>
<tbody>
<tr>
<td>usIDevice</td>
<td>DEVICE</td>
</tr>
</tbody>
</table>

**CIA405.DEVICE**

( % CAA_CiA405.Library)
Type CIA405.DEVICE represents the Node ID of a device.

<table>
<thead>
<tr>
<th>DEVICE</th>
<th>USINT (0..127)</th>
</tr>
</thead>
</table>

**CIA405.SDO_ERROR**

( % CAA_Cia405.Library)
Type CIA405.SDO_ERROR represents error information as defined in /DS-301/:

<table>
<thead>
<tr>
<th>SDO_ERROR</th>
<th>UDINT</th>
</tr>
</thead>
</table>

**CIA405.CANOPEN KERNEL_ERROR**

( % CAA_CiA405.Library)
Type CIA405.CANOPEN KERNEL_ERROR contains error information about the CANopen Kernel.

<table>
<thead>
<tr>
<th>CANOPENKERNEL_ERROR</th>
<th>USINT</th>
</tr>
</thead>
</table>

Global Constants

1.4.2.6.3.3.1 CIA405.CANOPEN_KERNEL_ERROR_CODES..................... 2139

**CIA405.CANOPEN_KERNEL_ERROR_CODES**
<table>
<thead>
<tr>
<th><strong>CIA405.CANOPEN_KERNEL_ERROR</strong></th>
<th><strong>value/ range</strong></th>
<th><strong>Description</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CANOPEN_KERNEL_NO_ERROR</strong></td>
<td>16#0</td>
<td>No error</td>
</tr>
<tr>
<td></td>
<td></td>
<td>There is no error notified by the CANopen kernel. The output DEVICE is invalid and may contain any value. Nevertheless it is possible, that a SDO-Abort or EMCY message was received from some device; see CIA405.SDO_READ/WRITE and CIA405.RECV_EMCY, -DEV.</td>
</tr>
<tr>
<td><strong>CANOPEN_KERNEL_OTHER_ERROR</strong></td>
<td>16#1</td>
<td>Other error</td>
</tr>
<tr>
<td></td>
<td></td>
<td>If there is a &quot;further error register&quot; - output at the called function block, this output contains more specific information, e.g. at function blocks CIA405.SDO_READ and CIA405.SDO_WRITE, were the contents of the SDO-Abort message is delivered. At function blocks CIA405.RECV_EMCY_DEV and CIA405.RECV_EMCY, the contents of the received EMCY-message is delivered. At the function block CIA405.GET_CANOPEN_KERNEL_STATE this value never is delivered, because there is no &quot;further error register&quot; - output available.</td>
</tr>
<tr>
<td><strong>CANOPEN_KERNEL_DATA_OVERFLOW</strong></td>
<td>16#2</td>
<td>Data overflow</td>
</tr>
<tr>
<td></td>
<td></td>
<td>In case of implemented send- or receive-buffers for CAN objects, it is possible that these buffers get an overflow.</td>
</tr>
<tr>
<td><strong>CANOPEN_KERNEL_TIMEOUT</strong></td>
<td>16#3</td>
<td>Timeout</td>
</tr>
<tr>
<td></td>
<td></td>
<td>A timeout occurs when - for example - a node does not respond to a SDO-request. The time measurement is started when the application puts the request, i.e. at rising edge of ENABLE signal. See also Additional parameters.</td>
</tr>
<tr>
<td></td>
<td>16#4 – 16#F</td>
<td>Reserved for further SDO errors</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Reserved by CiA.</td>
</tr>
<tr>
<td><strong>CANOPEN_KERNEL_CANBUS_OFF</strong></td>
<td>16#10</td>
<td>CAN Bus off</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The node is disconnected from the bus.</td>
</tr>
<tr>
<td><strong>CANOPEN_KERNEL_CAN_ERROR_PASSIVE</strong></td>
<td>16#11</td>
<td>CAN Error Passive</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The node is error passive. The node is able to communicate but not allowed to send an active error flag in case of error detection, i.e. not to disturb the bus communication. Further it waits for 8 Bit-times more than other nodes before it starts another transmission.</td>
</tr>
<tr>
<td></td>
<td>16#12 – 16#1F</td>
<td>Reserved for further internal errors</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Reserved by CiA.</td>
</tr>
<tr>
<td><strong>CANOPEN_INTERNAL_FB_ERROR</strong></td>
<td>16#21</td>
<td>Manufacturer specific</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Up to 222 manufacturer specific errors (16#21 – 16#FF). This is the value range free to use for the error codes coming up from the lower level software, e.g. from the kernel, drivers, services, operating system, ... .</td>
</tr>
<tr>
<td></td>
<td>16#100 – 16#FFFF</td>
<td>Reserved by CIA</td>
</tr>
</tbody>
</table>
This chapter describes standard function blocks to access CANopen from IEC 61131-3. The function blocks described in this chapter are mandatory by means of function library needs. Optional function blocks in the sense of CANopen standardization are described too in order to assimilate all function blocks defined within DS-405. The following table can be used by IEC 61131-3 compliant systems to state the features covered.

<table>
<thead>
<tr>
<th>Function block</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>CIA405.RECV_EMCY_DEV</td>
<td>mandatory</td>
</tr>
<tr>
<td>CIA405.RECV_EMCY</td>
<td>mandatory</td>
</tr>
<tr>
<td>CIA405.SDO_WRITE4</td>
<td>mandatory</td>
</tr>
<tr>
<td>CIA405.SDO_READ4</td>
<td>mandatory</td>
</tr>
<tr>
<td>CIA405.GET_LOCAL_NODE_ID</td>
<td>mandatory</td>
</tr>
<tr>
<td>CIA405.GET_STATE</td>
<td>mandatory</td>
</tr>
<tr>
<td>CIA405.GET_CANOPEN_KERNEL_STATE</td>
<td>mandatory</td>
</tr>
<tr>
<td>CIA405.NMT</td>
<td>mandatory</td>
</tr>
<tr>
<td>CIA405.SDO_WRITE</td>
<td>optional</td>
</tr>
<tr>
<td>CIA405.SDO_READ</td>
<td>optional</td>
</tr>
</tbody>
</table>

General function block design issues

1.4.2.6.3.4.2.1 Data types vs. Data length......................................................... 2141
1.4.2.6.3.4.2.2 Timeout................................................................................... 2141
1.4.2.6.3.4.2.3 Additional parameters............................................................ 2142
1.4.2.6.3.4.2.4 Control mechanism.................................................................... 2143

Data types vs. Data length

(© CAA_Cia405.Library)

It is possible to represent transmission of data of arbitrary length within IEC 61131-3. However, the interfaces to function blocks allowing for that may be quite more difficult to use and understand. Therefore different function blocks are defined herein, for the transmission of a fixed (maximum) amount of data each as well as blocks for arbitrary length.

Timeout

(© CAA_Cia405.Library)
The interface is designed such that it is possible to wrap the calls with a timer block to implement timeouts. Additionally, it is allowed that lower level CANopen software implement their own timeouts and report such as errors to the caller.

### Additional parameters

Some systems may support or require additional information in the function blocks. For example, a PLC may have several CAN channels. In that case the function blocks require the channel number of the service. Another requirement will be to limit the time a function block takes until it reports to the caller if the function block is executed with or without success. Therefore a Timeout input (milliseconds) is required. Therefore these parameters are added to the function blocks.

In DS405 it is allowed and within `CAA_CIA405.Library` it is mandatory to add the following parameters to the function blocks:

**NETWORK: USINT; (*CAN channel number*)**

Selects the CAN channel on which the requested service has to be executed (range 1..64). A device with only one CAN channel available must accept either the number 1 only or must accept all numbers, i.e. must ignore the value at this input.

**CHANNEL: USINT; (*SDO channel number*)**

Selects the SDO channel on which the requested service has to be executed (starting from number 1). A device with only one SDO channel available must accept either the number 1 only or must accept all numbers, i.e. must ignore the value at this input.

The CAN channel number is not the same like the Net ID specified in the CANBus configurator. For calculating the CAN channel number use following formula: \( \text{Net ID} + 1 \)

**TIMEOUT: UDINT; (*milliseconds; maximum wait time for response*)**

The timeout defines the maximum wait time for a response of the function block. The time measurement starts with the rising edge on input ENABLE of the called function block. When the defined maximum time is reached without being a response to the requested service available, the function block terminates the execution and signals error timeout. For an infinite wait time for response the value 0 is used at this input.

Within the `CAA_CIA405.Library` one additional output to the function block `CIA405.GET_CANOPEN_KERNEL_STATE` is defined:

**DEVICE: CIA405.DEVICE; (*source of STATE *)**

DEVICE contains either the device number of the device to which the STATE is connected to (for example in case of reception of an unknown data format) or the number of the device where the kernel is located.

Within the `CAA_CIA405.Library` one additional input to the function block `CIA405.SDO_WRITE` is defined:

**MODE: CIA405.SDO_MODE; (*type of protocol *)**

MODE select the method for sending/receiving the data from the server regarding the SDO protocol specification in /CIA301/.
Control mechanism

In general the task of each defined function block of the CAA_Cia405.Library is to read or to write one or more parameters. Such parameters can be variables or states at CANopen connected devices (SDO, EMCY, NMT-state) or at the local node (own node number, CANopen kernel state). Therefore each function block performs the same behavior in general.

A typical behavior of the control signals ENABLE, CONFIRM and ERROR is shown at the SDO access:

![Request/Response diagram](image)

- **Fig. 81: Request/Response diagram**
  - The peak at the symbol ✗ shows the time when the event occurs. The symbol □ shows that the message is ignored. High level at ENABLE and CONFIRM signals TRUE; low level signals FALSE. High level at ERROR signals a value > 0; low level signals value = 0.
  - PLC cycles: the diagram shows, that there is no relation between the signal timing and the PLC cycles, i.e. the response can be available immediately or some PLC cycles later.
  - Timeout: the time control starts with a rising edge at ENABLE. Timeout event occurs, when there is no response available before time control reaches the timeout value. When timeout occurs, ERROR = 3 and ERRORINFO may contain additional error information using the SDO-Abort data format. If there is no additional error information available, ERRORINFO is 0.

**Request:**

The "no error" as well as the "error" condition shows, that immediately with the rising edge detected at Input ENABLE the Request event occurs, i.e. it is put to a lower level software. The outputs stay at their initial values. The program control is given back to the caller if the Response or Abort is not available immediately.

**A:Response:** As soon as the SDO-Response message is available the output CONFIRM is set TRUE and stays TRUE as long as the function block is called with input ENABLE=TRUE. When the function block is called with ENABLE=FALSE, CONFIRM is set FALSE, i.e. the initial value.
B: Abort: As soon as the SDO-Abort message is available the output ERROR is set to value 1 and more error information is available in the output ERRORINFO ("further error register" / DS405/). The outputs stay stable as long as the function block is called with input ENABLE = TRUE. When the function block is called with ENABLE = FALSE, ERROR and ERRORINFO are set to 0, these are their initial values.

C: ENABLE = FALSE before Response or Abort available: As soon as the function block is called with ENABLE = FALSE, the outputs are set to their initial values, i.e. CONFIRM = FALSE, ERROR = 0, ERRORINFO = 0. A possibly available Response or Abort is ignored as well as a later arriving Response or Abort.

D: Timeout: When the function block is called and timeout is detected and Response or Abort is not available, ERROR = 3 and ERRORINFO may contain additional error information using the SDO-Abort data format. If there is no additional error information available, ERRORINFO is 0.

Flow control

The function block stays in idle, i.e. all outputs are set to their initial values and it executes no operation, as long as input ENABLE is FALSE. The number of calls does not matter. As soon as ENABLE performs a rising edge, the function block will start execution, i.e. it will read the input values and put a request to a lower level software, e.g. to read or write a selected parameter. If the response to the request is available immediately, like the return value on a function call, the function block will set the outputs immediately and finish its task. For example, a function which gets the local node id, may work in that way. If the response is not available immediately, the function block gives the program control back to the caller with all output signals at busy state, e.g. CONFIRM = FALSE and ERROR = 0.

On each call of the function block, the availability of the response is checked and the timeout, if applicable. Changing input values except ENABLE are ignored.

If the response signals an error condition or the timeout is reached, the function block set the outputs to error state, e.g. CONFIRM = FALSE and ERROR <> 0 and finishes its task.

If the response signals success, the function block sets the outputs to confirm state, e.g. CONFIRM = TRUE and ERROR = 0 and, if it was a read request, the data outputs to the actual requested values and finishes its task.

NOTICE!

In /DS405/ it is not clearly defined if the function block starts execution again a) immediately, b) on the next call or c) after the next rising edge at input ENABLE. To give other function blocks the chance to access the same device on the same SDO-channel, it is recommended to wait at least one PLC cycle (b) before starting execution again. This avoids, that one function block, called with input ENABLE = TRUE forever, could block the communication to one device. A disadvantage may be in case a) and b), that the function block does not signalize when a new response, e.g. new data, is available. The user does not know how old the delivered data are. On write requests, the user does not know if the input data are transmitted to the device before they are changed. Further it is doubtful to transmit the same data several times. For this reasons it is recommended to do not start execution again when ENABLE stays TRUE (c). When ENABLE becomes FALSE all outputs are set to their initial values immediately. A possibly available response is ignored as well as an ongoing processed request. A rising edge at ENABLE starts the flow control from the beginning.
NOTICE!

In /DS301/ (V 4.02 chap. 9.2.2.1.7) is defined regarding the SDO-Abort: "The service may be executed at any time by both the client and the server of a SDO. If the client of a SDO has confirmed service outstanding, the indication of the abort is taken to be the confirmation of that service."

CIA405_CIA405BASE

1.4.2.6.3.4.3.1 CIA405.CIA405BASE (FB)......................................................... 2145
1.4.2.6.3.4.3.2 CIA405.SDO_WRITE4 (FB)................................................... 2146
1.4.2.6.3.4.3.3 CIA405.SDO_WRITE (FB)..................................................... 2148
1.4.2.6.3.4.3.4 CIA405.SDO_READ4 (FB)..................................................... 2148
1.4.2.6.3.4.3.5 CIA405.SDO_READ (FB)....................................................... 2150

CIA405.CIA405BASE (FB)

( % CAA_CIA405.Library )

To implement the common behavior model for all the CIA405 function blocks this function block is used as a base class. All other CIA405 function blocks within this document extend this function block.

Table 407: Input:

<table>
<thead>
<tr>
<th>NETWORK</th>
<th>USINT</th>
<th>CAN channel number; defines on which CAN-Bus a CiA405 function block performs its action.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>range: 1 … MAX NETWORKS (max. 64)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>initial: 1</td>
</tr>
<tr>
<td>ENABLE</td>
<td>BOOL</td>
<td>Enable execution</td>
</tr>
<tr>
<td>TIMEOUT</td>
<td>UDINT</td>
<td>Maximum execution time in ms; 0 means disable timeout</td>
</tr>
</tbody>
</table>

Table 408: Output:

<table>
<thead>
<tr>
<th>CONFIRM</th>
<th>BOOL</th>
<th>Executed with success</th>
</tr>
</thead>
<tbody>
<tr>
<td>ERROR</td>
<td>CIA405.CANOPEN_KERNEL_ERROR</td>
<td>Error code</td>
</tr>
<tr>
<td></td>
<td></td>
<td>range: 0000 … FFFF</td>
</tr>
<tr>
<td></td>
<td></td>
<td>initial: 0</td>
</tr>
</tbody>
</table>
### CIA405.SDO_WRITE4 (FB)

( % CAA_Cia405.Library)

Extends % CIA405.CIA405BASE.

**Table 409: Input:**

<table>
<thead>
<tr>
<th>Input</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NETWORK</td>
<td>USINT</td>
<td>see CIA405.CIA405BASE</td>
</tr>
<tr>
<td>ENABLE</td>
<td>BOOL</td>
<td>see CIA405.CIA405BASE</td>
</tr>
<tr>
<td>TIMEOUT</td>
<td>UDINT</td>
<td>see CIA405.CIA405BASE</td>
</tr>
<tr>
<td>DEVICE</td>
<td>CIA405.DEVICE</td>
<td>Addressed device; 0 means local device</td>
</tr>
<tr>
<td></td>
<td></td>
<td>range: 0 … 127</td>
</tr>
<tr>
<td></td>
<td></td>
<td>initial: 0</td>
</tr>
<tr>
<td>CHANNEL</td>
<td>USINT</td>
<td>SDO channel number</td>
</tr>
<tr>
<td></td>
<td></td>
<td>range: 1 … 128</td>
</tr>
<tr>
<td></td>
<td></td>
<td>initial: 1</td>
</tr>
<tr>
<td>INDEX</td>
<td>WORD</td>
<td>Object index</td>
</tr>
<tr>
<td></td>
<td></td>
<td>range: 0000 … FFFF</td>
</tr>
<tr>
<td></td>
<td></td>
<td>initial: 0</td>
</tr>
<tr>
<td>SUBINDEX</td>
<td>BYTE</td>
<td>Object subindex</td>
</tr>
<tr>
<td></td>
<td></td>
<td>range: 00 … FF</td>
</tr>
<tr>
<td></td>
<td></td>
<td>initial: 0</td>
</tr>
<tr>
<td>DATA</td>
<td>POINTER TO BYTE</td>
<td>Object data</td>
</tr>
<tr>
<td></td>
<td></td>
<td>range: 00 00 00 00 … FF FF FF FF</td>
</tr>
<tr>
<td></td>
<td></td>
<td>initial: 0</td>
</tr>
<tr>
<td>DATALENGTH</td>
<td>USINT</td>
<td>Length of valid data in bytes</td>
</tr>
<tr>
<td></td>
<td></td>
<td>range: 1 … 4</td>
</tr>
</tbody>
</table>

**Table 410: Output:**

<table>
<thead>
<tr>
<th>Output</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONFIRM</td>
<td>BOOL</td>
<td>see CIA405.CIA405BASE</td>
</tr>
<tr>
<td>ERROR</td>
<td>CIA405.CANOPEN_KERNEL_ERROR</td>
<td>see CIA405.CIA405BASE</td>
</tr>
<tr>
<td>ERRORINFO</td>
<td>CIA405.SDO_ERROR</td>
<td>SDO Abort code</td>
</tr>
<tr>
<td></td>
<td></td>
<td>range: 0 … 2^{32} - 1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>initial: 0</td>
</tr>
</tbody>
</table>
The figure above shows the timing diagram for CIA405.SDO_WRITE4. After all data is provided to the inputs, ENABLE is set to TRUE (1). The SDO will be sent, and when the CANopen software reports success to the function block, output CONFIRM will be changed to TRUE (2). The caller will see this and change ENABLE to FALSE (3), which in turn will cause output CONFIRM to change to FALSE (4).

- With a rising edge on input ENABLE, the function block will sample the inputs and initiate the transmission of the SDO specified in DATA and DATALENGTH to the recipient specified with DEVICE, INDEX and SUBINDEX. The value of DEVICE is limited to a range of 1 to 127. For access to the local object dictionary it is allowed to use the value 0 for DEVICE.
- With TRUE on input ENABLE, the function block is allowed to continue execution. If a result is reported on the call by lower level CANopen software, outputs CONFIRM, ERROR and ERRORINFO are set accordingly.
- With a falling edge on input ENABLE the function block will terminate. If the transmission did not finish yet, it will be aborted if possible. Outputs CONFIRM and ERROR will both be set to FALSE respectively "no error".
- With a FALSE on input ENABLE, the function block will return immediately and not take any action.
- For behavior of ENABLE, CONFIRM and ERROR see chapter "Control mechanism" on page 2143.
- For inputs NETWORK and TIMEOUT see chapter "Additional parameters .
- The input DATA defined as ARRAY[1..4] OF BYTE contains the least significant data byte at the lowest array index. Example: the value 16#01020304 (hexadecimal) is assigned as ARRAY[1..4] OF BYTE := 04, 03, 02, 01

The result of a writing operation is reported (immediately after the writing call or afterwards) in outputs CONFIRM, ERROR and ERRORINFO with a rising edge on either CONFIRM or ERROR. In case of ERROR is equal to 1 (other error), ERRORINFO will give more specified information on the cause. This is especially true for the occurrence of an SDO Abort. Then ERRORINFO contains the Abort code.

It cannot be assumed in general that this function can synchronously complete; rather, it should be possible to have this function continue to be executed while the call to CIA405.SDO_WRITE4 returns and the PLC program is continued. Therefore, the result of a call may be available only several cycles after a rising edge has been applied to ENABLE.
CIA405.SDO_WRITE (FB)

( % CAA_Cia405.Library)

This function block can be used as alternative to the CIA_SDO_WRITEx function blocks. It supports the arbitrary length of data. The behavior and timing is the same as with % CIA405.SDO_WRITE4. The only difference is the data type of the variables DATA and DATA-LENGTH.

Extends % CIA405.CIA405BASE.

Table 411: Input:

<table>
<thead>
<tr>
<th>FIELD</th>
<th>DATA TYPE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>NETWORK</td>
<td>USINT</td>
<td>see CIA405.CIA405BASE</td>
</tr>
<tr>
<td>ENABLE</td>
<td>BOOL</td>
<td>see CIA405.CIA405BASE</td>
</tr>
<tr>
<td>TIMEOUT</td>
<td>UDINT</td>
<td>see CIA405.CIA405BASE</td>
</tr>
<tr>
<td>DEVICE</td>
<td>CIA405.DEVICE</td>
<td>Addressed device; 0 means local device</td>
</tr>
<tr>
<td></td>
<td></td>
<td>range: 0 … 127</td>
</tr>
<tr>
<td></td>
<td></td>
<td>initial: 0</td>
</tr>
<tr>
<td>CHANNEL</td>
<td>USINT</td>
<td>SDO channel number</td>
</tr>
<tr>
<td></td>
<td></td>
<td>range: 1 … 128</td>
</tr>
<tr>
<td></td>
<td></td>
<td>initial: 1</td>
</tr>
<tr>
<td>INDEX</td>
<td>WORD</td>
<td>Object index</td>
</tr>
<tr>
<td></td>
<td></td>
<td>range: 0000 … FFFF</td>
</tr>
<tr>
<td></td>
<td></td>
<td>initial: 0</td>
</tr>
<tr>
<td>SUBINDEX</td>
<td>BYTE</td>
<td>Object subindex</td>
</tr>
<tr>
<td></td>
<td></td>
<td>range: 00 … FF</td>
</tr>
<tr>
<td></td>
<td></td>
<td>initial: 0</td>
</tr>
<tr>
<td>MODE</td>
<td>CIA405.SDO_MODE</td>
<td></td>
</tr>
<tr>
<td>DATA</td>
<td>POINTER TO BYTE</td>
<td>Object data</td>
</tr>
<tr>
<td></td>
<td></td>
<td>range:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>initial: 0</td>
</tr>
<tr>
<td>DATALENGTH</td>
<td>UINT</td>
<td>Length of valid data in bytes</td>
</tr>
<tr>
<td></td>
<td></td>
<td>range: 1 …</td>
</tr>
</tbody>
</table>

Table 412: Output:

<table>
<thead>
<tr>
<th>FIELD</th>
<th>DATA TYPE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONFIRM</td>
<td>BOOL</td>
<td>see CIA405.CIA405BASE</td>
</tr>
<tr>
<td>ERROR</td>
<td>CIA405.CANOPEN_KERNEL_ERROR</td>
<td>see CIA405.CIA405BASE</td>
</tr>
<tr>
<td>ERRORINFO</td>
<td>CIA405.SDO_ERROR</td>
<td>SDO Abort code</td>
</tr>
<tr>
<td></td>
<td></td>
<td>range: 0 … 2^{32}-1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>initial: 0</td>
</tr>
</tbody>
</table>

CIA405.SDO_READ4 (FB)

( % CAA_Cia405.Library)

Extends % CIA405.CIA405BASE.
### Table 413: Input:

<table>
<thead>
<tr>
<th>Input</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NETWORK</td>
<td>USINT</td>
<td>see CIA405.CIA405BASE</td>
</tr>
<tr>
<td>ENABLE</td>
<td>BOOL</td>
<td>see CIA405.CIA405BASE</td>
</tr>
<tr>
<td>TIMEOUT</td>
<td>UDINT</td>
<td>see CIA405.CIA405BASE</td>
</tr>
<tr>
<td>DEVICE</td>
<td>CIA405.DEVICE</td>
<td>Addressed device; 0 means local device range: 0 … 127 \ initial: 0</td>
</tr>
<tr>
<td>CHANNEL</td>
<td>USINT</td>
<td>SDO channel number range: 1 … 128 \ initial: 1</td>
</tr>
<tr>
<td>INDEX</td>
<td>WORD</td>
<td>Object index range: 0000 … FFFF \ initial: 0</td>
</tr>
<tr>
<td>SUBINDEX</td>
<td>BYTE</td>
<td>Object subindex range: 00 … FF \ initial: 0</td>
</tr>
</tbody>
</table>

### Table 414: Output:

<table>
<thead>
<tr>
<th>Output</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONFIRM</td>
<td>BOOL</td>
<td>see CIA405.CIA405BASE</td>
</tr>
<tr>
<td>ERROR</td>
<td>CIA405.CANOPEN_ KERNEL_ERROR</td>
<td>see CIA405.CIA405BASE</td>
</tr>
<tr>
<td>DATA</td>
<td>ARRAY OF BYTE</td>
<td>Object data range: 00 00 00 00 … FF FF FF FF \ initial: 00 00 00</td>
</tr>
<tr>
<td>DATALENGTH</td>
<td>USINT</td>
<td>Length of valid data in bytes range: 0 … 4 \ initial: 0</td>
</tr>
<tr>
<td>ERRORINFO</td>
<td>CIA405.SDO_ERRO R</td>
<td>SDO Abort code range: 0 … 2^{32}-1 \ initial: 0</td>
</tr>
</tbody>
</table>
The figure above shows the timing diagram for CIA405.SDO_READ4. After all data is provided to the inputs, ENABLE is set to TRUE (1). The SDO will be received, and when the CANopen software reports success to the function block, output CONFIRM will be changed to TRUE (2). The caller will see this and change ENABLE to FALSE (3), which in turn will cause output CONFIRM to change to FALSE (4).

- With a rising edge on input ENABLE, the function block will sample the inputs and initiate the transmission of the SDO specified with DEVICE, INDEX and SUBINDEX. The value of DEVICE is limited to a range of 1 to 127. For access to the local object dictionary it is allowed to use the value 0 for DEVICE.
- With TRUE on input ENABLE, the function block is allowed to continue execution. If a result is reported on the call by lower level CANopen software, outputs CONFIRM, DATA and DATALENGTH (in case of success) or ERROR and ERRORINFO (in case of failure) are set accordingly.
- With a falling edge on input ENABLE the function block will terminate. If the transmission did not finish yet, it will be aborted if possible. Outputs CONFIRM and ERROR will both be set to FALSE respectively "no error".
- With a FALSE on input ENABLE, the function block will return immediately and not take any action.
- For behavior of ENABLE, CONFIRM and ERROR see chapter "Control mechanism" on page 2143.
- For inputs NETWORK and TIMEOUT see chapter "Additional parameters."
- The output DATA defined as ARRAY[1..4] OF BYTE contains the least significant data byte at the lowest array index. Example: the value 16#01020304 (hexadecimal) is assigned as ARRAY[1..4] OF BYTE := 04, 03, 02, 01

CIA405.SDO_READ (FB)
( % CAA.CiA405.Library)
This function block can be used as alternative to the CIA_SDO_READx function blocks. It supports the arbitrary length of data. The behavior and timing is the same as with CIA405.SDO_WRITE4. The only difference is the data type of the variables DATA and DATALENGTH.

Extends % CIA405.CIA405BASE.
### Table 415: Input:

<table>
<thead>
<tr>
<th>NETWORK</th>
<th>USINT</th>
<th>see CIA405.CIA405BASE</th>
</tr>
</thead>
<tbody>
<tr>
<td>ENABLE</td>
<td>BOOL</td>
<td>see CIA405.CIA405BASE</td>
</tr>
<tr>
<td>TIMEOUT</td>
<td>UDINT</td>
<td>see CIA405.CIA405BASE</td>
</tr>
<tr>
<td>DEVICE</td>
<td>CIA405.DEVICE</td>
<td>Addressed device; 0 means local device range: 0 … 127 initial: 0</td>
</tr>
<tr>
<td>CHANNEL</td>
<td>USINT</td>
<td>SDO channel number range: 1 … 128 initial: 1</td>
</tr>
<tr>
<td>INDEX</td>
<td>WORD</td>
<td>Object index initial: 0</td>
</tr>
<tr>
<td>SUBINDEX</td>
<td>BYTE</td>
<td>Object subindex initial: 0</td>
</tr>
</tbody>
</table>

### Table 416: Output:

<table>
<thead>
<tr>
<th>CONFIRM</th>
<th>BOOL</th>
<th>see CIA405.CIA405BASE</th>
</tr>
</thead>
<tbody>
<tr>
<td>ERROR</td>
<td>CIA405.CANOPEN_KERNEL_ERROR</td>
<td>see CIA405.CIA405BASE</td>
</tr>
<tr>
<td>DATA</td>
<td>POINTER TO BYTE</td>
<td>Object data initial: 0</td>
</tr>
<tr>
<td>ERRORINFO</td>
<td>CIA405.SDO_ERROR</td>
<td>SDO Abort code initial: 0</td>
</tr>
</tbody>
</table>

**In/Output:**

| DATALENGTH | UINT | Input: Capacity of the Buffer Output: Length of valid data in bytes initial: 0 |

**Own node ID**

1.4.2.6.3.4.4.1  CIA405.GET_LOCAL_NODE_ID (FB)................................. 2151

**CIA405.GET_LOCAL_NODE_ID (FB)**

( CAA_CIA405.Library)

Function block CIA405.GET_LOCAL_NODE_ID returns the own node ID.

Extends CAA_CIA405.Library.
Table 417: Input:

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>NETWORK</td>
<td>USINT</td>
<td>see CIA405.CIA405BASE</td>
</tr>
<tr>
<td>ENABLE</td>
<td>BOOL</td>
<td>see CIA405.CIA405BASE</td>
</tr>
<tr>
<td>TIMEOUT</td>
<td>UDNT</td>
<td>see CIA405.CIA405BASE</td>
</tr>
</tbody>
</table>

Table 418: Output:

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>CONFIRM</td>
<td>BOOL</td>
<td>see CIA405.CIA405BASE</td>
</tr>
<tr>
<td>ERROR</td>
<td>CIA405.CANOPEN_KERNEL_ERROR</td>
<td>see CIA405.CIA405BASE</td>
</tr>
<tr>
<td>DEVICE</td>
<td>CIA405.DEVICE</td>
<td>Local node id; 0 means error range: 0 … 127 initial: 0</td>
</tr>
</tbody>
</table>

Fig. 84: Request/Response diagram

- Data output DEVICE is valid only if CONFIRM is TRUE. If CONFIRM is FALSE, the data output is invalid and may contain any value.
- The request for data is initiated only once with the rising edge at ENABLE. Another request is initiated not before another rising edge at ENABLE is recognized.
- In case of an error during execution of the function block it is defined to deliver the invalid node number DEVICE=0 and ERROR = 0.
- For behavior of ENABLE and CONFIRM see chapter Chapter 1.4.2.6.3.4.2.4 "Control mechanism" on page 2143.
- For inputs NETWORK and TIMEOUT see chapter Additional parameters.

Query state

1.4.2.6.3.4.5.1 CIA405.GET_STATE (FB)................................. 2152
1.4.2.6.3.4.5.2 CIA405.GET_CANOPEN_KERNEL_STATE (FB)............. 2153

CIA405.GET_STATE (FB)

(CAA_Cia405.Library)
Function block CIA405.GET_STATE returns the current state of a CANopen network device.
Extends CIA405.CIA405BASE.

Table 419: Input:

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>NETWORK</td>
<td>USINT</td>
<td>see CIA405.CIA405BASE</td>
</tr>
<tr>
<td>ENABLE</td>
<td>BOOL</td>
<td>see CIA405.CIA405BASE</td>
</tr>
</tbody>
</table>
### CIA405.DEVICE
Addressed device; 0 means local device
range: 1 ... 127
initial: 0

### Table 420: Output:

<table>
<thead>
<tr>
<th>CONFIRM</th>
<th>BOOL</th>
<th>see CIA405.CIA405BASE</th>
</tr>
</thead>
<tbody>
<tr>
<td>ERROR</td>
<td>CIA405.CANOPEN_KERNEL_ERROR</td>
<td>see CIA405.CIA405BASE</td>
</tr>
</tbody>
</table>
| STATE   | CIA405.DEVICE_STATE | Device state
range: 0 ... 7
initial: 0 |

### Table 421: Input:

<table>
<thead>
<tr>
<th>NETWORK</th>
<th>USINT</th>
<th>see CIA405.CIA405BASE</th>
</tr>
</thead>
<tbody>
<tr>
<td>ENABLE</td>
<td>BOOL</td>
<td>see CIA405.CIA405BASE</td>
</tr>
<tr>
<td>TIMEOUT</td>
<td>UDINT</td>
<td>see CIA405.CIA405BASE</td>
</tr>
</tbody>
</table>

---

**CIA405.GET_CANOPEN_KERNEL_STATE (FB)**

( `CAA_CiA405.Library` )

Function block CIA405.GET_CANOPEN_KERNEL_STATE returns the current state of the CANopen Kernel.

Extends `CIA405.CIA405BASE`.

---

If the state is not known, the function block returns with the state UNKNOWN. This will occur, if the device is not guarded and Heartbeat is not running. If Guarding is performed and the device does not answer or it is known for other reasons, that the device is not available, the function returns the state NOT_AVAIL. If `DEVICE` is 0 or equal to the own Node-ID, the function block returns the state of the local communication process.

- Data output STATE is valid only if CONFIRM is TRUE. If CONFIRM is FALSE, the data output is invalid and may contain any value.
- The request for data is initiated only once with the rising edge at ENABLE. Another request is initiated not before the next rising edge at ENABLE is recognized.
- For behavior of ENABLE and CONFIRM see chapter `Chapter 1.4.2.6.3.4.2.4 "Control mechanism"` on page 2143.
- For inputs NETWORK and TIMEOUT see chapter `Additional parameters`.
- In case of an error during execution of the function block it is defined to deliver STATE = UNKNOWN (6) with CONFIRM = TRUE and ERROR = 0.
### Table 422: Output:

<table>
<thead>
<tr>
<th>CONFI RM</th>
<th>BOOL</th>
<th>see CIA405.CIA405BASE</th>
</tr>
</thead>
<tbody>
<tr>
<td>ERROR</td>
<td>CIA405.CANOPEN_KERNEL_ERROR</td>
<td>see CIA405.CIA405BASE</td>
</tr>
<tr>
<td>STATE</td>
<td>CIA405.CANOPEN_KERNEL_ERROR</td>
<td>Kernel state</td>
</tr>
<tr>
<td></td>
<td></td>
<td>range: 0000 … FFFF</td>
</tr>
<tr>
<td></td>
<td></td>
<td>initial: 0</td>
</tr>
<tr>
<td>DEVICE</td>
<td>CIA405.DEVICE</td>
<td>Additional device information connected to STATE; 0 means local device</td>
</tr>
<tr>
<td></td>
<td></td>
<td>range: 0 … 127</td>
</tr>
<tr>
<td></td>
<td></td>
<td>initial: 0</td>
</tr>
</tbody>
</table>

**Fig. 86: Request/Response diagram**

- Data output STATE is valid only if CONFI RM is TRUE. If CONFI RM is FALSE, the data output is invalid and may contain any value.
- The request for data is initiated only once with the rising edge at ENABLE. Another request is initiated not before the next rising edge at ENABLE is recognized.
- For behavior of ENABLE and CONFI RM see chapter § Chapter 1.4.2.6.3.4.2.4 “Control mechanism” on page 2143.
- For inputs NETWORK and TIMEOUT and output DEVICE see chapter § Additional parameters.
- In case of an error during execution of the function block it is defined to deliver DEVICE = 0 (local device) with STATE <> 0 and ERROR = 0.
- The function block reports any error except a) SDO-Abort-indications created by an SDO-Client and b) EMCY messages received from a CAN device. SDO-Abort indications are handled by the function blocks § CIA405.SDO_READ and § CIA405.SDO_WRITE. EMCY messages are handled by the function blocks § CIA405.RECV_EMCY and § CIA405.RECV_EMCY_DEV.
- As soon as the STATE is reported, it is acknowledged to the lower level error handling. That means for example to clear the error message in the memory in order to enable the entry of another error message which will be reported on the next request for data.

**Network management**

1.4.2.6.3.4.6.1 CIA405.NMT (FB) ........................................................................ 2154
1.4.2.6.3.4.6.2 CIA405.RECV_EMCY_DEV (FB) .................................................. 2156
1.4.2.6.3.4.6.3 CIA405.RECV_EMCY (FB) ......................................................... 2157

**CIA405.NMT (FB)**

( § CAA_CIA405.Library)

Function block CIA405.NMT controls network management functions of one or all CANopen nodes.

Extends § CIA405.CIA405BASE.
Table 423: Input:

<table>
<thead>
<tr>
<th>INPUT</th>
<th>TYPE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>NETWORK</td>
<td>USINT</td>
<td>see CIA405.CIA405BASE</td>
</tr>
<tr>
<td>ENABLE</td>
<td>BOOL</td>
<td>see CIA405.CIA405BASE</td>
</tr>
<tr>
<td>TIMEOUT</td>
<td>UDINT</td>
<td>see CIA405.CIA405BASE</td>
</tr>
<tr>
<td>STATE</td>
<td>CIA405.TRANSTION_STATE</td>
<td>initial: 0</td>
</tr>
<tr>
<td>DEVICE</td>
<td>CIA405.DEVICE</td>
<td>Node-ID of the device; 0 means all devices range: 0 … 127 initial: 0</td>
</tr>
</tbody>
</table>

Table 424: Output:

<table>
<thead>
<tr>
<th>OUTPUT</th>
<th>TYPE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONFIRM</td>
<td>BOOL</td>
<td>see CIA405.CIA405BASE</td>
</tr>
<tr>
<td>ERROR</td>
<td>CIA405.CANOPEN_KERNEL_ERROR</td>
<td>see CIA405.CIA405BASE</td>
</tr>
</tbody>
</table>

Fig. 87: NMT service diagram

- For behavior of ENABLE and CONFIRM see chapter § Chapter 1.4.2.6.3.4.2.4 “Control mechanism” on page 2143.
- For inputs NETWORK and TIMEOUT see chapter § Additional parameters .
- NMTservice: the NMTservice event is the request to the addressed node (DEVICE=0 means: local device) to perform the requested state transition.
- confirm: the event confirm is the positive acknowledgement that the service is processed without error; for example the CANopen kernel returns no error to the function block.
- error: the event error is the negative acknowledgement that the service is processes with error; for example the CANopen kernel returns an error to the function block.
- Timeout: the time control starts with a rising edge at ENABLE. Timeout event occurs, when there is no acknowledgement returned to the function block before time control reaches the timeout value. When timeout occurs, ERROR = 3.
Function block CIA405.RECV_EMCY_DEV checks if an emergency object has been received from DEVICE.

Extends % CIA405.CIA405BASE.

**Table 425: Input:**

<table>
<thead>
<tr>
<th>NETWORK</th>
<th>USINT</th>
<th>see % CIA405.CIA405BASE</th>
</tr>
</thead>
<tbody>
<tr>
<td>ENABLE</td>
<td>BOOL</td>
<td>see % CIA405.CIA405BASE</td>
</tr>
<tr>
<td>TIMEOUT</td>
<td>UDINT</td>
<td>see % CIA405.CIA405BASE</td>
</tr>
<tr>
<td>DEVICE</td>
<td>CIA405.DEVICE</td>
<td>Additional device information connected to STATE; 0 means local device range: 0 ... 127 initial: 0</td>
</tr>
</tbody>
</table>

**Table 426: Output:**

<table>
<thead>
<tr>
<th>CONFIRM</th>
<th>BOOL</th>
<th>see % CIA405.CIA405BASE</th>
</tr>
</thead>
<tbody>
<tr>
<td>ERROR</td>
<td>CIA405.CANOPEN KERNEL_ERROR</td>
<td>see % CIA405.CIA405BASE</td>
</tr>
<tr>
<td>ERRORINFO</td>
<td>CIA405.EMCY_ERR OR EMCY code</td>
<td></td>
</tr>
</tbody>
</table>

**Fig. 88: Request/Response diagram**

CIA405.RECV_EMCY_DEV will check if an emergency object (/DS-301/) has been received from DEVICE. If the function block has finished its action without any error, output CONFIRM is changed to TRUE and ERROR to 0. ERRORINFO contains the information about the received emergency. An emergency was received if the error code of ERRORINFO is unequal 0. An error code with value 0 means that the device has currently no emergency.
A received emergency is not cleared and it is delivered with each call of the function block until the device sends an emergency reset.

If an error occurred while checking for an emergency, CONFIRM is set to FALSE and ERROR is set to the corresponding error value.

- For behaviour of ENABLE and CONFIRM see chapter “Control mechanism” on page 2143.
- For inputs NETWORK and TIMEOUT see chapter Additional parameters.
- Response: this event is the positive acknowledgement that the request is processed without error, i.e. the value of EMCY is available at the output ERRORINFO. If no EMCY was received, ERRORINFO = 0, i.e. all members of the structure CIA405.EMCY_ERROR are zero. For detailed description of CIA405.EMCY_ERROR see CIA405.EMCY_ERROR (STRUCT)
- error: this event is the negative acknowledgement and means that the request is processed with error, for example the CANopen kernel returns an error to the function block when accessing the value of EMCY. In this case ERRORINFO = 0. For detailed description of CIA405.CANOPEN_KERNEL_ERROR see CIA405.CANOPEN_KERNEL_ERROR
- Timeout: the time control starts with a rising edge at ENABLE. Timeout event occurs, when there is no acknowledgement returned to the function block before time control reaches the timeout value. When timeout occurs, ERROR = 3 and ERRORINFO = 0.

Recommendation: The function block should deliver the EMCY of the checked device on each call. Often the earliest EMCY contains the originally error information; later EMCYs often just results from following errors. A lower level software may work in the way, that a received EMCY does not overwrite an EMCY which is different from "No Error" ( is equal to "Reset Error" ). On the other hand, a received "No Error"- EMCY overwrites the stored EMCY in any case.

CIA405.RECV_EMCY (FB)

Function block CIA405.RECV_EMCY checks if an emergency object has been received from any DEVICE.

<table>
<thead>
<tr>
<th>Table 427: Input:</th>
</tr>
</thead>
<tbody>
<tr>
<td>NETWORK</td>
</tr>
<tr>
<td>ENABLE</td>
</tr>
<tr>
<td>TIMEOUT</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Table 428: Output:</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONFIRM</td>
</tr>
<tr>
<td>ERROR</td>
</tr>
<tr>
<td>DEVICE</td>
</tr>
<tr>
<td>ERRORINFO</td>
</tr>
</tbody>
</table>
CIA405.RECV_EMCY will check if an emergency object (/DS-301/) has been received from any device. If the function block has finished its action without any error, output CONFIRM is changed to TRUE and ERROR to 0. ERRORINFO contains the information about the received emergency and DEVICE is set to the ID of the device sending the emergency object. An emergency was received if both DEVICE and error code of ERRORINFO are unequal 0.

If an error occurred while checking for an emergency, CONFIRM is set to FALSE and ERROR is set to the corresponding error value.

- For behavior of ENABLE and CONFIRM see chapter “Control mechanism” on page 2143.
- For inputs NETWORK and TIMEOUT see chapter “Additional parameters.”
- Response: this event is the positive acknowledgement that the request is processed without error, i.e. the value of EMCY received from DEVICE is available at the output ERRORINFO. If no EMCY was received, ERRORINFO = 0, i.e. all members of the structure CIA405.EMCY_ERROR are zero.
- error: this event is the negative acknowledgement and means that the request is processed with error, for example the CANopen kernel returns an error to the function block when accessing the value of EMCY. In this case ERRORINFO = 0.
- Timeout: the time control starts with a rising edge at ENABLE. Timeout event occurs, when there is no acknowledgement returned to the function block before time control reaches the timeout value. When timeout occurs, ERROR = 3 and ERRORINFO = 0.

Recommendation: In order to deliver all pending EMCY messages with each call of the function block in a loop and also to notify a "Reset Error"-EMCY at least once, a lower level software may work in the way, that for each device a storage for the received EMCY messages is managed as described in the recommendation for CIA405.RECV_EMCY_DEV (FB). The function block CIA405.RECV_EMCY scans the EMCY storages in a loop over all existing devices. If a device is found during the scan, for which the EMCY-storage contains an EMCY different from "No Error", the EMCY is delivered. Else if a device is found, for which an EMCY different from "No Error" was delivered at the last scan and for which the EMCY-storage now contains "No Error", this "No Error"-EMCY is delivered. Each device, for which a "No Error" was delivered at last, is skipped during the scan, that is a "No Error"-EMCY is delivered only once. On the next execution the search is continued. In order to get the information that a pending EMCY was cleared by receiving "Reset Error" from a device, this "Reset Error"-EMCY is delivered only once, i.e. DEVICE > 0 while ERRORINFO = 0. In order to get the information that all devices are error-free, the function block delivers DEVICE = 0 and ERRORINFO = 0 after a complete scan over the EMCY-storages of all existing devices.
1.4.2.6.4 CAA_DTUtil.library

1.4.2.6.4.1 Overview CAA_DTUtil.library

1.4.2.6.4.2 Data types / Enumerations

1.4.2.6.4.3 Global Constants

1.4.2.6.4.4 Modules

1.4.2.6.4.5 Examples CAA_DTUtil.library

Overview CAA_DTUtil.library

This library provides functions for handling date and time.

Data types / Enumerations

1.4.2.6.4.2.1 DTU.ERROR (ENUM)

DTU.ERROR (ENUM)

This data structure describes errors which might occur when using the functions of the CAA_DTUtility library. Error number range 5750 – 5799 is reserved for this library in the prefix registration for libraries (LibraryRegistry.chm).

<table>
<thead>
<tr>
<th>NO_ERROR</th>
<th>0</th>
<th>No error</th>
</tr>
</thead>
<tbody>
<tr>
<td>_FIRST_ERROR</td>
<td>5750</td>
<td>First library-specific error</td>
</tr>
<tr>
<td>TIME_OUT</td>
<td>5751</td>
<td>Time limit exceeded</td>
</tr>
<tr>
<td>NOTAVAILABLE</td>
<td>5752</td>
<td>Not available</td>
</tr>
<tr>
<td>INPUT_INVALID</td>
<td>5753</td>
<td>Invalid input values</td>
</tr>
<tr>
<td>ERROR_UNKNOWN</td>
<td>5754</td>
<td>Unknown error</td>
</tr>
<tr>
<td>WRONG_PARAMETER</td>
<td>5755</td>
<td>Wrong parameter</td>
</tr>
<tr>
<td>TZI_NOT_SET</td>
<td>5756</td>
<td>Timezone information not initialized</td>
</tr>
<tr>
<td>FIRST_MF</td>
<td>5770</td>
<td>First manufacturer-specific error</td>
</tr>
<tr>
<td>LAST_ERROR</td>
<td>5799</td>
<td>Last library-specific error</td>
</tr>
</tbody>
</table>

Global Constants

1.4.2.6.4.3.1 DTU.gc_tziTimeZoneCET (VAR_GLOBAL CONSTANT)

DTU.gc_tziTimeZoneCET (VAR_GLOBAL CONSTANT)

The global constant „DTU.gc_tziTimeZoneCET“ defines the time zone as well as the switch between summer and standard time of the Central European EU countries.

The switch is done according to the following rules:

- The switch to summer time (CEST) takes place on the last Sunday of March. The clock is set ahead by one hour at 2:00 a.m..
- The switch to standard time (CET) takes place on the last Sunday of October. The clock is set back by one hour at 3:00 a.m.
Reading and setting the realtime clock

1.4.2.6.4.4.1.1 Reading and setting the realtime clock............................... 2160
1.4.2.6.4.4.1.2 DTU.GetDateAndTime (FB prefix: gdt)............................ 2160
1.4.2.6.4.4.1.3 DTU.SetDateAndTime (FB prefix: sdt)............................ 2161
1.4.2.6.4.4.1.4 DTU.SetTimeZoneInformation (FB prefix: sti).................... 2161
1.4.2.6.4.4.1.5 DTU.GetTimeZoneInformation (FB prefix: gti).................... 2162

Reading and setting the realtime clock

(CAA_DTUtil.library)
Reading and setting of the realtime clock is already supported by modules of the library CAA_RealTimeClock.lib. These modules, however, might not take into account the time zone and a possible switch between summer and standard time.

If an automatic switch is desired all the same, the function blocks of the CAA.DTUtility library can be used for reading and setting the realtime clock. In this case the time switch is done via this library.

The switch between summer and standard time is not done until one of the function blocks DTU.GetDateAndTime or DTU.SetDateAndTime is called. For this reason it is recommended to read the system time cyclically via function block „DTU.GetDateAndTime“. If the runtime system supports the automatic switch between summer and standard time, the calls of the library will be forwarded directly to the corresponding functions of library CAA.RealTimeClock and the clock switch will be done by the runtime system.

DTU.GetDateAndTime (FB prefix: gdt)

This function block of the CAA_DTUtil.library reads the realtime clock. It returns the time of the local time zone (i.e. a possible switch to summer/standard time will be regarded)

The current time zone switch can be defined by „DTU.SetTimeZoneInformation“.

If a switch to summer/standard time has become necessary since the last polling of the time, this is done automatically.

At the start of the module the outputs „dtDateAndTime“ and „eTimeZone“ are reset. (DT#1970-01-01-00:00 and RTCLK_UNKNOWN). After successful operation they will contain the current local time.

Table 429: Input:

<table>
<thead>
<tr>
<th>xExecute</th>
<th>BOOL</th>
<th>A rising edge will start the action</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>see: Guidelines for CAA libs</td>
</tr>
</tbody>
</table>

Table 430: Output:

<table>
<thead>
<tr>
<th>dtDateAndTime</th>
<th>DATE_AND_TIME</th>
<th>Current local time</th>
</tr>
</thead>
<tbody>
<tr>
<td>eTimeZone</td>
<td>RTCLK_PERIODE</td>
<td>Current time zone</td>
</tr>
<tr>
<td>xDone</td>
<td>BOOL</td>
<td>Action terminated successfully</td>
</tr>
<tr>
<td></td>
<td></td>
<td>see: Guidelines for CAA libs</td>
</tr>
</tbody>
</table>
**DTU.SetDateAndTime (FB prefix: sdt)**

This function block of the CAA_DTUtil.library sets the realtime clock. The current local time is passed in the parameter „dtDateAndTime“.

> **NOTICE!**
> If the time is within the not-unique interval shortly after the switch to standard (winter) time, standard time will be assumed. This case occurs in the Central European countries, if for example on the last Sunday of October the time is set at 2:30 a.m.. Here it is not clear, whether 2:30 a.m. is still summer time or already standard (winter) time. In this case the module assumes standard (winter) time.

**Table 431: Input:**

<table>
<thead>
<tr>
<th>xExecute</th>
<th>BOOL</th>
<th>A rising edge will start the action</th>
</tr>
</thead>
<tbody>
<tr>
<td>dtDateAndTime</td>
<td>DATE_AND_TIME</td>
<td>Current time</td>
</tr>
</tbody>
</table>

**Table 432: Output:**

<table>
<thead>
<tr>
<th>xDone</th>
<th>BOOL</th>
<th>Action terminated successfully</th>
</tr>
</thead>
<tbody>
<tr>
<td>xBusy</td>
<td>BOOL</td>
<td>Module is active</td>
</tr>
<tr>
<td>xError</td>
<td>BOOL</td>
<td>Error occurred</td>
</tr>
<tr>
<td>eError</td>
<td>WORD</td>
<td>Error ID for diagnosis (see DTU.ERROR)</td>
</tr>
</tbody>
</table>

**DTU.SetTimeZoneInformation (FB prefix: sti)**

With this function block of the CAA_DTUtil.library the specific information of the time zone can be modified.

The time zone setting will be kept even after a “Reset-Origin” of the controller and can only be modified by being overwritten again. If no automatic switch is desired, all elements of „tziTimeZoneInformation.stStandardDate“ and „tziTimeZoneInformation.stDaylightDate“ must be set to 0. (for further information see library CAA_RealTimeClock)

**Table 433: Input:**

<table>
<thead>
<tr>
<th>xExecute</th>
<th>BOOL</th>
<th>A rising edge will start the action</th>
</tr>
</thead>
<tbody>
<tr>
<td>tziTimeZoneInformation</td>
<td>RTCLK.TIME_ZONE_INFO</td>
<td>Information on the time zone</td>
</tr>
</tbody>
</table>
DTU.GetTimeZoneInformation (FB prefix: gti)

This function block of the CAA_DTUtil.library reads the specific information on the time zone. (for further information see library CAA_RealTimeClock)

<table>
<thead>
<tr>
<th>Table 434: Output:</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>xDone</td>
<td>BOOL</td>
<td>Action terminated successfully</td>
</tr>
<tr>
<td>xBusy</td>
<td>BOOL</td>
<td>Module is active</td>
</tr>
<tr>
<td>xError</td>
<td>BOOL</td>
<td>Error occurred</td>
</tr>
<tr>
<td>eError</td>
<td>WORD</td>
<td>Error ID for diagnosis (see RTCLK.ERROR)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Table 435: Input:</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>xExecute</td>
<td>BOOL</td>
<td>A rising edge will start the action</td>
</tr>
<tr>
<td>tziTimeZoneInforma-</td>
<td>RTCLK.TIME_ZONE</td>
<td>Information on the time zone</td>
</tr>
<tr>
<td>tion</td>
<td>_INFO</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Table 436: Output:</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>xDone</td>
<td>BOOL</td>
<td>Action terminated successfully</td>
</tr>
<tr>
<td>xBusy</td>
<td>BOOL</td>
<td>Module is active</td>
</tr>
<tr>
<td>xError</td>
<td>BOOL</td>
<td>Error occurred</td>
</tr>
<tr>
<td>eError</td>
<td>WORD</td>
<td>Error ID for diagnosis (see RTCLK.ERROR)</td>
</tr>
</tbody>
</table>

Handling Functions

1.4.2.6.4.4.2.1  DTU.GetDayOfWeek (FUN) ......................................................... 2162
1.4.2.6.4.4.2.2  DTU.DateConcat (FUN) ......................................................... 2163
1.4.2.6.4.4.2.3  DTU.DateSplit (FB prefix: ds) ............................................. 2163
1.4.2.6.4.4.2.4  DTU.TODConcat (FUN) ......................................................... 2163
1.4.2.6.4.4.2.5  DTU.TODSplit (FB prefix: tods) ............................................. 2163
1.4.2.6.4.4.2.6  DTU.DTConcat (FUN) ......................................................... 2164
1.4.2.6.4.4.2.7  DTU.DTSplit (FB prefix: dts) ............................................. 2164

DTU.GetDayOfWeek (FUN)

This function of the CAA_DTUtil.library determines the day of the week of a specified date.

<table>
<thead>
<tr>
<th>Table 437: Input:</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>dateDate</td>
<td>DATE</td>
<td>Date</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Table 438: Output:</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>eWeekday</td>
<td>RTCLK.WEEKDAY</td>
<td>Day of the week (0..6) 0=Sunday</td>
</tr>
</tbody>
</table>

2162 3ADR010583, 1, en_US 2020/12/10
DTU.DateConcat (FUN)

This function of the CAA_DTUtil.library converts year, month and day to the IEC 6-1131-3 format "DATE". In case of invalid entries D#1970-01-01 will be returned.

Table 439: Input:

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>uiYear</td>
<td>UINT</td>
<td>Year (1970..2099)</td>
</tr>
<tr>
<td>uiMonth</td>
<td>UINT</td>
<td>Month (1..12)</td>
</tr>
<tr>
<td>uiDay</td>
<td>UINT</td>
<td>Day (1..31)</td>
</tr>
</tbody>
</table>

Table 440: Output:

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>dateDate</td>
<td>DATE</td>
</tr>
<tr>
<td></td>
<td>Date</td>
</tr>
</tbody>
</table>

DTU.DateSplit (FB prefix: ds)

This function block of the CAA_DTUtil.library provides year, month and day from the IEC 6-1131-3 format "DATE".

Table 441: Input:

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>dateDate</td>
<td>DATE</td>
</tr>
</tbody>
</table>

Table 442: Output:

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>uiYear</td>
<td>UINT</td>
<td>Year (1970..2099)</td>
</tr>
<tr>
<td>uiMonth</td>
<td>UINT</td>
<td>Month (1..12)</td>
</tr>
<tr>
<td>uiDay</td>
<td>UINT</td>
<td>Day (1..31)</td>
</tr>
</tbody>
</table>

DTU.TODConcat (FUN)

This function of the CAA_DTUtil.library converts hour, minute and second to the IEC 6-1131-3 format TIME_OF_DAY. In case of invalid inputs TOD#00:00 will be returned.

Table 443: Input:

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>uiHour</td>
<td>UINT</td>
<td>Hour (0..23)</td>
</tr>
<tr>
<td>uiMinute</td>
<td>UINT</td>
<td>Minute (0..59)</td>
</tr>
<tr>
<td>uiSecond</td>
<td>UINT</td>
<td>Second (0..59)</td>
</tr>
<tr>
<td>uiMillisecond</td>
<td>UINT</td>
<td>Second (0..999)</td>
</tr>
</tbody>
</table>

Table 444: Output:

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>todTime</td>
<td>TIME_OF_DAY</td>
</tr>
<tr>
<td></td>
<td>Time</td>
</tr>
</tbody>
</table>

DTU.TODSplit (FB prefix: tods)

The function block of the CAA_DTUtil.library FB provides hour, minute and second from the IEC 6-1131-3 format TIME_OF_DAY.
### Table 445: Input:

<table>
<thead>
<tr>
<th>todTime</th>
<th>TIME_OF_DAY</th>
<th>Time</th>
</tr>
</thead>
</table>

### Table 446: Output:

<table>
<thead>
<tr>
<th>uiHour</th>
<th>UINT</th>
<th>Hour (0..23)</th>
</tr>
</thead>
<tbody>
<tr>
<td>uiMinute</td>
<td>UINT</td>
<td>Minute (0..59)</td>
</tr>
<tr>
<td>uiSecond</td>
<td>UINT</td>
<td>Second (0..59)</td>
</tr>
<tr>
<td>uiMillisSecond</td>
<td>UINT</td>
<td>Millisecond (0..999)</td>
</tr>
</tbody>
</table>

#### DTU.DTConcat (FUN)

The function of the CAA_DTUtil.library converts year, month, hour, minute and second to the IEC 6-1131-3 format DATE_AND_TIME. In case of invalid inputs DT#1970-01-01-00:00 will be returned.

### Table 447: Input:

<table>
<thead>
<tr>
<th>uiYear</th>
<th>UINT</th>
<th>Year (1970..2099)</th>
</tr>
</thead>
<tbody>
<tr>
<td>uiMonth</td>
<td>UINT</td>
<td>Month (1..12)</td>
</tr>
<tr>
<td>uiDay</td>
<td>UINT</td>
<td>Day (1..31)</td>
</tr>
<tr>
<td>uiHour</td>
<td>UINT</td>
<td>Hour (0..23)</td>
</tr>
<tr>
<td>uiMinute</td>
<td>UINT</td>
<td>Minute (0..59)</td>
</tr>
<tr>
<td>uiSecond</td>
<td>UINT</td>
<td>Second (0..59)</td>
</tr>
</tbody>
</table>

### Table 448: Output:

<table>
<thead>
<tr>
<th>dtDateAndTime</th>
<th>DATE_AND_TIME</th>
<th>Date / Time</th>
</tr>
</thead>
</table>

#### DTU.DTSplit (FB prefix: dts)

The function block of the CAA_DTUtil.library provides year, month, hour, minute and second from the IEC 6-1131-3 format DATE_AND_TIME.

### Table 449: Input:

<table>
<thead>
<tr>
<th>dtDateAndTime</th>
<th>DATE_AND_TIME</th>
<th>Date / Time</th>
</tr>
</thead>
</table>

### Table 450: Output:

<table>
<thead>
<tr>
<th>uiYear</th>
<th>UINT</th>
<th>Year (1970..2099)</th>
</tr>
</thead>
<tbody>
<tr>
<td>uiMonth</td>
<td>UINT</td>
<td>Month (1..12)</td>
</tr>
<tr>
<td>uiDay</td>
<td>UINT</td>
<td>Day (1..31)</td>
</tr>
<tr>
<td>uiHour</td>
<td>UINT</td>
<td>Hour (0..23)</td>
</tr>
<tr>
<td>uiMinute</td>
<td>UINT</td>
<td>Minute (0..59)</td>
</tr>
<tr>
<td>uiSecond</td>
<td>UINT</td>
<td>Second (0..59)</td>
</tr>
</tbody>
</table>
Examples CAA_DTUtil.library

**GetDateAndTime:**

```plaintext
VAR
    dtu_getdt : DTU.GetDateAndTime;
    dtDate : DATE_AND_TIME;
    eTZ : RTCLK.PERIODE;
END_VAR

dtu_getdt.xExecute := TRUE;
dtu_getdt();

IF dtu_getdt.xError THEN
    ; // error handling
END_IF

IF dtu_getdt.xDone THEN
    // done without error
    dtDate := dtu_getdt.dtDateAndTime;
    eTZ := dtu_getdt.eTimezone;
END_IF
```

**SetDateAndTime:**

```plaintext
VAR
    dtu_setdt : DTU.SetDateAndTime;
END_VAR

dtu_setdt.dtDateAndTime := DATE_AND_TIME#2008-05-19-20:00:00;
dtu_setdt.xExecute := TRUE;
dtu_setdt();

IF dtu_setdt.xError THEN
    ; // error handling
END_IF

IF dtu_setdt.xDone THEN
    ; // done without error
END_IF
```

**GetTimeZoneInformation:**

```plaintext
VAR
    dtu_gettzi : DTU.GetTimeZoneInformation;
```
tziInfo : RTCLK.TIME_ZONE_INFO;
END_VAR

dtu_gettzi.xExecute := TRUE;
dtu_gettzi();

IF dtu_gettzi.xError THEN
    // error handling
END_IF

IF dtu_gettzi.xDone THEN
    // done without error
    tziInfo := dtu_gettzi.tziInfo;
END_IF

SetTimeZoneInformation
VAR
    dtu_settzi : DTU.SetTimeZoneInformation;
END_VAR

dtu_settzi.tziInfo := DTU.gc_tziTimeZoneCET;
dtu_settzi.xExecute := TRUE;
DTU.settzi();

IF DTU.settzi.xError THEN
    // error handling
END_IF

IF DTU.settzi.xDone THEN
    // done without error
END_IF

DateConcat:
VAR
    eError : DTU.ERROR;
    datDate := DATE;
END_VAR

datDate := DTU.DateConcat
  (uiYear := 2005,
   uiMonth := 2,
uiDay := 1,
peError := ADR(eError)
);

DateSplit
VAR
  eError : DTU.ERROR;
  uiYear : UINT;
  uiMonth : UINT;
  uiDay : UINT;
END_VAR

eError := DateSplit
{
  datDate := DATE#2008-05-15,
  puiYear := ADR(uiDateSplitYear),
  puiMonth := ADR(uiDateSplitMonth),
  puiDay := ADR(uiDateSplitDay)
};

DTConcat:
VAR
  eError : DTU.ERROR;
  dtDate : DATE_AND_TIME;
END_VAR

dtDate := DTConcat
{
  uiYear := 2007,
  uiMonth := 6,
  uiDay := 4,
  uiHour := 15,
  uiMinute := 22,
  uiSecond := 1,
  peError := ADR(eError)
};

DTSplit:
VAR
  eError : DTU.ERROR;
  uiYear : UINT;
  uiMonth : UINT;


```
uiDay : UINT;
uiHour : UINT;
uiMinute : UINT;
uiSecond : UINT;
END_VAR

eError := DTSplit
{
  dtDateAndTime := DATE_AND_TIME#2005-08-15-22:06:00,
  puiYear := ADR(uiYear),
  puiMonth := ADR(uiMonth),
  puiDay := ADR(uiDay),
  puiHour := ADR(uiHour),
  puiMinute := ADR(uiMinute),
  puiSecond := ADR(uiSecond)
};

TODConcat
VAR
  eError : DTU.ERROR;
  todTime : TIME_OF_DAY;
END_VAR

todTime := TODConcat
{
  uiHour := 5,
  uiMinute := 4,
  uiSecond := 3,
  uiMillisecond := 2,
  peError := ADR(eError)
};

TODSplit:
VAR
  eError : DTU.ERROR;
  uiHour : UINT;
  uiMinute : UINT;
  uiSecond : UINT;
  uiMillisecond : UINT;
END_VAR
```
eError := TODSplit
{
    todTime := TIME_OF_DAY#22:05:11.003,
    puiHour := ADR(uiTODSplitHour),
    puiMinute := ADR(uiTODSplitMinute),
    puiSecond := ADR(uiTODSplitSecond),
    puiMillisecond := ADR(uiTODSplitMillisecond)
};

GetDayOfWeek
VAR
    eError : DTU.ERROR;
    eWeekday : RTCLK_WEEKDAY;
END_VAR

eWeekday := GetDayOfWeek
{
    dtDate := DATE#2008-5-15,
    peError := ADR(eError)
};

1.4.2.6.5 CAA_File.library
1.4.2.6.5.1 Overview CAA_File.library............................................................ 2169
1.4.2.6.5.2 Data types / Enumerations........................................................... 2170
1.4.2.6.5.3 Modules........................................................................................ 2172
1.4.2.6.5.4 Examples...................................................................................... 2185

Overview CAA_File.library
1.4.2.6.5.1.1 Overview CAA_File.Library....................................................... 2169
1.4.2.6.5.1.2 Directory paths and protocols................................................... 2169

Overview CAA_File.Library
This library provides function blocks for accessing file directory systems and files. The standar-
dized inputs and outputs of the function blocks are described in detail in the © CAA Guidelines
and will therefore not be described here. File access is only supported in binary mode. All input
values of the function blocks are stored in local internal variables. This does not apply for the
contents of memory structures where a pointer is passed on as an input for example "pBuffer"of
"FILE.Read" or "FILE.Write".

Directory paths and protocols
CAA_FILENAME which is described in “CAA_Types.library” should be usable for library
"CAA_File" in the following way. Separators between directory paths in "CAA.FILENAME" are
the slash (\") or the back-slash (\"). The interpretation of the directory path is tolerant to the
effect that either one or the other of these characters can be used.
Mixed usage should be avoided by an error message. The part of a “CAA.FILENAME” character string behind the last separator is the file name. The rest specifies the directory path. Only in the first part of the directory path may a series of colons be used for the specification of drive letters or for device identification. The function block $FILE.SetRoot$ provides for an enhanced portability of the applications, regardless of the many different options available to display a directory path. The starting point within the directory tree can thus be set centrally and it is then possible to navigate through the application relative to this starting point.

- Absolute paths in the local file system: C:/dir/Test.txt or /ram:2/home/Test.txt
- Relative paths in the local file system: ../Log/Test.txt or 'Log/Test.txt
- Complete path specification for a file in the local file system: file://C:/dir/Test.txt
- Complete path specification for a file on a FTP Server: ftp://<user>:<passwd>@<ip-addr>[:<port>]/home/user/Test.txt
- Complete path specification for a file on a HTTP Server: http://<ip-addr>[:<port>]/dir/Test.txt

Whether the appropriate resolver functions for FTP- or HTTP-Servers are available in the corresponding implementation, can be evaluated via function $FILE.GetProperty$. Only IP addresses are valid, the resolution of a name is not intended for in the functional scope of the library “CAA.File”.

Data types / Enumerations

1.4.2.6.5.2.1 $FILE.MODE$ (ENUM)................................................................. 2170
1.4.2.6.5.2.2 $FILE.ATTRIB$ (ENUM)............................................................... 2170
1.4.2.6.5.2.3 $FILE.ERROR$ (ENUM)............................................................... 2171
1.4.2.6.5.2.4 $FILE.DIR_ENTRY$ (STRUCT; Prefix de).................................... 2171

$FILE.MODE$ (ENUM)

( $CAA_File.Library$)

<table>
<thead>
<tr>
<th>Mode</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MWRITE</td>
<td>0</td>
<td>Write access, file will be overwitten or created</td>
</tr>
<tr>
<td>MREAD</td>
<td>1</td>
<td>Read access, file will only be opened for reading</td>
</tr>
<tr>
<td>MRDWR</td>
<td>2</td>
<td>Read- and Write access, the file will be overwritten or created</td>
</tr>
<tr>
<td>MAPPD</td>
<td>3</td>
<td>File will be opened in WRITE mode, but the written data will be appended at the end of the file.</td>
</tr>
</tbody>
</table>

$FILE.ATTRIB$ (ENUM)

( $CAA_File.Library$)

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ARCHIVE</td>
<td>0</td>
<td>Archive file</td>
</tr>
<tr>
<td>HIDDEN</td>
<td>1</td>
<td>Hidden file</td>
</tr>
<tr>
<td>NORMAL</td>
<td>2</td>
<td>File, for which none of the other attributes has been set</td>
</tr>
<tr>
<td>READONLY</td>
<td>3</td>
<td>File can only be read</td>
</tr>
</tbody>
</table>
FILE.ERROR (ENUM)

This data structure describes errors, which might occur when handling functions of the CAA.File.library. The error range 5100 – 5199 is reserved for the CAA.File.library in the prefix registration for libraries (LibraryRegistry.chm).

<table>
<thead>
<tr>
<th>Enum</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NO_ERROR</td>
<td>0</td>
<td>No error</td>
</tr>
<tr>
<td>FIRST_ERROR</td>
<td>5100</td>
<td>First library-specific error</td>
</tr>
<tr>
<td>TIME_OUT</td>
<td>5101</td>
<td>Time limit exceeded</td>
</tr>
<tr>
<td>ABORT</td>
<td>5102</td>
<td>Order has been aborted by activating input xAbort</td>
</tr>
<tr>
<td>HANDLE_INVALID</td>
<td>5103</td>
<td>Invalid handle</td>
</tr>
<tr>
<td>NOT_EXIST</td>
<td>5104</td>
<td>Directory or file does not exist</td>
</tr>
<tr>
<td>EXIST</td>
<td>5105</td>
<td>Directory or file already exists</td>
</tr>
<tr>
<td>NO_MORE_ENTRIES</td>
<td>5106</td>
<td>No further entries are available</td>
</tr>
<tr>
<td>NOT_EMPTY</td>
<td>5107</td>
<td>File or directory is not empty</td>
</tr>
<tr>
<td>READ_ONLY_CAA</td>
<td>5108</td>
<td>File or directory is write protected</td>
</tr>
<tr>
<td>WRONG_PARAMETER</td>
<td>5109</td>
<td>Wrong parameters</td>
</tr>
<tr>
<td>ERROR_UNKNOWN</td>
<td>5110</td>
<td>Unknown error</td>
</tr>
<tr>
<td>WRITE_INCOMPLETE</td>
<td>5111</td>
<td>Not all data have been written</td>
</tr>
<tr>
<td>NOT_IMPLEMENTED</td>
<td>5112</td>
<td>Not implemented</td>
</tr>
<tr>
<td>FIRST_MF</td>
<td>5150</td>
<td>First manufacturer-specific error</td>
</tr>
<tr>
<td>LAST_ERROR</td>
<td>5199</td>
<td>Latest library-specific error</td>
</tr>
</tbody>
</table>

FILE.DIR_ENTRY (STRUCT; Prefix de)

(CAA.File.Library)

<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sEntry</td>
<td>CAA.FILENAME</td>
<td>Name of the directory or file</td>
</tr>
<tr>
<td>szSize</td>
<td>CAA.SIZE</td>
<td>File size</td>
</tr>
<tr>
<td>xDirectory</td>
<td>BOOL</td>
<td>Directory or file:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>True: - Directory</td>
</tr>
<tr>
<td></td>
<td></td>
<td>False: - File</td>
</tr>
<tr>
<td>xExclusive</td>
<td>BOOL</td>
<td>Access mode on file:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>True: - exclusive access on the file</td>
</tr>
<tr>
<td></td>
<td></td>
<td>False: - multiple access on this file possible</td>
</tr>
<tr>
<td>dtLastModification</td>
<td>DT</td>
<td>Date and time of last modification</td>
</tr>
</tbody>
</table>
**Modules**

1.4.2.6.5.3.1 Function blocks - Root directory ................................................ 2172
1.4.2.6.5.3.2 Function blocks - File directory services .................................. 2173
1.4.2.6.5.3.3 Function blocks - File services .................................................. 2176

**Function blocks - Root directory**

1.4.2.6.5.3.1.1 FILE.SetRoot (FB; Prefix filsrt) ............................................... 2172
1.4.2.6.5.3.1.2 FILE.GetRoot (FB; Prefix filgrt) ............................................... 2172

**FILE.SetRoot (FB; Prefix filsrt)**

This function block can be used to determine the root directory for all function blocks of the
CAA_File.library in case of relative path specification.

**Table 451: Input:**

<table>
<thead>
<tr>
<th>xExecute</th>
<th>BOOL</th>
<th>see: % Chapter 1.4.2.6.1.1 “CAA Guidelines” on page 2121</th>
</tr>
</thead>
<tbody>
<tr>
<td>sFileName</td>
<td>CAA.FILENAME</td>
<td>Name of root directory:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>no special characters,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>separator for sub-directories “/”,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>absolute or relative path specification</td>
</tr>
<tr>
<td></td>
<td></td>
<td>No empty spaces</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Use capital letters</td>
</tr>
</tbody>
</table>

**Table 452: Output:**

<table>
<thead>
<tr>
<th>xDone</th>
<th>BOOL</th>
<th>see: % Chapter 1.4.2.6.1.1 “CAA Guidelines” on page 2121</th>
</tr>
</thead>
<tbody>
<tr>
<td>xBusy</td>
<td>BOOL</td>
<td>see: % Chapter 1.4.2.6.1.1 “CAA Guidelines” on page 2121</td>
</tr>
<tr>
<td>xError</td>
<td>BOOL</td>
<td>see: % Chapter 1.4.2.6.1.1 “CAA Guidelines” on page 2121</td>
</tr>
<tr>
<td>eError</td>
<td>FILE.ERROR</td>
<td>Error ID</td>
</tr>
</tbody>
</table>

**FILE.GetRoot (FB; Prefix filgrt)**

This function block of the % CAA_File.library returns the path for the root directory.

**Table 453: Input:**

<table>
<thead>
<tr>
<th>xExecute</th>
<th>BOOL</th>
<th>see: % Chapter 1.4.2.6.1.1 “CAA Guidelines” on page 2121</th>
</tr>
</thead>
</table>

**Table 454: Output:**

<table>
<thead>
<tr>
<th>xDone</th>
<th>BOOL</th>
<th>see: % Chapter 1.4.2.6.1.1 “CAA Guidelines” on page 2121</th>
</tr>
</thead>
<tbody>
<tr>
<td>xBusy</td>
<td>BOOL</td>
<td>see: % Chapter 1.4.2.6.1.1 “CAA Guidelines” on page 2121</td>
</tr>
<tr>
<td>xError</td>
<td>BOOL</td>
<td>see: % Chapter 1.4.2.6.1.1 “CAA Guidelines” on page 2121</td>
</tr>
<tr>
<td>eError</td>
<td>FILE.ERROR</td>
<td>Error ID</td>
</tr>
<tr>
<td>sFileName</td>
<td>CAA.FILENAME</td>
<td>Root directory name with drive specification</td>
</tr>
</tbody>
</table>
Function blocks - File directory services

1.4.2.6.5.3.2.1 FILE.DirCreate (FB; Prefix fildcr) ............................................ 2173
1.4.2.6.5.3.2.2 FILE.DirOpen (FB; Prefix fildop) ............................................. 2173
1.4.2.6.5.3.2.3 FILE.DirList (FB; Prefix fildls) ............................................. 2174
1.4.2.6.5.3.2.4 FILE.DirRemove (FB; Prefix fildrm) ....................................... 2174
1.4.2.6.5.3.2.5 FILE.DirRename (FB; Prefix fildrn) ......................................... 2175
1.4.2.6.5.3.2.6 FILE.DirClose (FB; Prefix fildcl) ............................................ 2175

FILE.DirCreate (FB; Prefix fildcr)

This function block of the $CAA_File.library$ creates a sub-directory in the standard directory. If the sub-directory already exists, an error message is generated. There may be restrictions concerning the specification of the directory name, for example only capital letters allowed, for different targets. The function "FILE.GetProperty" shows these restrictions.

Table 455: Input:

<table>
<thead>
<tr>
<th>xExecute</th>
<th>BOOL</th>
<th>see: $Chapter 1.4.2.6.1.1 “CAA Guidelines” on page 2121</th>
</tr>
</thead>
<tbody>
<tr>
<td>sDirName</td>
<td>CAA.FILENAME</td>
<td>Directory name:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>No special characters,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Separator for sub-directories “/”,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Absolute or relative path specifications</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Recommendation:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>No empty spaces</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Use capital letters</td>
</tr>
<tr>
<td>xParents</td>
<td>BOOL</td>
<td>TRUE: - missing sub-directories will be created automatically,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>FALSE: - missing sub-directories will cause an error message</td>
</tr>
</tbody>
</table>

Table 456: Output:

<table>
<thead>
<tr>
<th>xDone</th>
<th>BOOL</th>
<th>see: $Chapter 1.4.2.6.1.1 “CAA Guidelines” on page 2121</th>
</tr>
</thead>
<tbody>
<tr>
<td>xBusy</td>
<td>BOOL</td>
<td>see: $Chapter 1.4.2.6.1.1 “CAA Guidelines” on page 2121</td>
</tr>
<tr>
<td>xError</td>
<td>BOOL</td>
<td>see: $Chapter 1.4.2.6.1.1 “CAA Guidelines” on page 2121</td>
</tr>
<tr>
<td>eError</td>
<td>FILE.ERROR</td>
<td>Error ID</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5104 - FILE.NOT_EXIST:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Directory does not exist</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5105 - FILE.EXIST:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Directory already exists</td>
</tr>
</tbody>
</table>

FILE.DirOpen (FB; Prefix fildop)

This function block of the $CAA_File.library$ opens a directory the entries (files and sub-directories) of which should be read with the help of the function block $FILE.DirList$. The return value is a handle. There may be restrictions concerning the specification of the directory name, e.g. only capital letters allowed, for different targets. The function $FILE.GetProperty$ shows these restrictions.
There is no natural text content in the image provided.
### Table 461: Input:

<table>
<thead>
<tr>
<th>Input</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>xExecute</td>
<td>BOOL</td>
<td>see: % Chapter 1.4.2.6.1.1 “CAA Guidelines” on page 2121</td>
</tr>
<tr>
<td>xAbort</td>
<td>BOOL</td>
<td>see: % Chapter 1.4.2.6.1.1 “CAA Guidelines” on page 2121</td>
</tr>
<tr>
<td>udiTimeOut</td>
<td>UDINT</td>
<td></td>
</tr>
<tr>
<td>sDirName</td>
<td>CAA.FILENAME</td>
<td>Directory name</td>
</tr>
<tr>
<td>xRecursive</td>
<td>BOOL</td>
<td>Condition for deleting the directory:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>TRUE: Directory is deleted including all files,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>FALSE: Directory is only deleted, if no files and sub-directories are</td>
</tr>
<tr>
<td></td>
<td></td>
<td>included; otherwise an error will be dumped</td>
</tr>
</tbody>
</table>

### Table 462: Output:

<table>
<thead>
<tr>
<th>Output</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>xDone</td>
<td>BOOL</td>
<td>see: % Chapter 1.4.2.6.1.1 “CAA Guidelines” on page 2121</td>
</tr>
<tr>
<td>xBusy</td>
<td>BOOL</td>
<td>see: % Chapter 1.4.2.6.1.1 “CAA Guidelines” on page 2121</td>
</tr>
<tr>
<td>xError</td>
<td>BOOL</td>
<td>see: % Chapter 1.4.2.6.1.1 “CAA Guidelines” on page 2121</td>
</tr>
<tr>
<td>xAborted</td>
<td>BOOL</td>
<td>see: % Chapter 1.4.2.6.1.1 “CAA Guidelines” on page 2121</td>
</tr>
<tr>
<td>eError</td>
<td>FILE.ERROR</td>
<td>Error ID</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5107 - FILE.NOT_EMPTY:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Directory not empty; will be generated only if xRecursive = FALSE</td>
</tr>
</tbody>
</table>

### FILE.DirRename (FB; Prefix fildrn)

This function block of the %CAA_File.library can be used to rename a directory and is optional for the different targets. Whether this function block is implemented or not is explicitly stated and encoded in the return value of the function "FILE.GetProperty" (see description of %FILE.GetProperty). The directory cannot be renamed until access on the properties has been terminated via "FILE.DirClose". The function block can be used to move files or directories!

### Table 463: Input:

<table>
<thead>
<tr>
<th>Input</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>xExecute</td>
<td>BOOL</td>
<td>see: % Chapter 1.4.2.6.1.1 “CAA Guidelines” on page 2121</td>
</tr>
<tr>
<td>sDirNameOld</td>
<td>CAA.FILENAME</td>
<td>Old directory name</td>
</tr>
<tr>
<td>sDirNameNew</td>
<td>CAA.FILENAME</td>
<td>New directory name</td>
</tr>
</tbody>
</table>

### Table 464: Output:

<table>
<thead>
<tr>
<th>Output</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>xDone</td>
<td>BOOL</td>
<td>see: % Chapter 1.4.2.6.1.1 “CAA Guidelines” on page 2121</td>
</tr>
<tr>
<td>xBusy</td>
<td>BOOL</td>
<td>see: % Chapter 1.4.2.6.1.1 “CAA Guidelines” on page 2121</td>
</tr>
<tr>
<td>xError</td>
<td>BOOL</td>
<td>see: % Chapter 1.4.2.6.1.1 “CAA Guidelines” on page 2121</td>
</tr>
<tr>
<td>eError</td>
<td>FILE.ERROR</td>
<td>Error ID</td>
</tr>
</tbody>
</table>

### FILE.DirClose (FB; Prefix fildcl)

This function block of the %CAA_File.library closes the access on the specified directory.
Table 465: Input:

<table>
<thead>
<tr>
<th>xExecute</th>
<th>BOOL</th>
<th>see: % Chapter 1.4.2.6.1.1 “CAA Guidelines” on page 2121</th>
</tr>
</thead>
<tbody>
<tr>
<td>hDir</td>
<td>CAA.HANDLE</td>
<td>Directory handle</td>
</tr>
</tbody>
</table>

Table 466: Output:

<table>
<thead>
<tr>
<th>xDone</th>
<th>BOOL</th>
<th>see: % Chapter 1.4.2.6.1.1 “CAA Guidelines” on page 2121</th>
</tr>
</thead>
<tbody>
<tr>
<td>xBusy</td>
<td>BOOL</td>
<td>see: % Chapter 1.4.2.6.1.1 “CAA Guidelines” on page 2121</td>
</tr>
<tr>
<td>xError</td>
<td>BOOL</td>
<td>see: % Chapter 1.4.2.6.1.1 “CAA Guidelines” on page 2121</td>
</tr>
<tr>
<td>eError</td>
<td>FILE.ERROR</td>
<td>Error ID</td>
</tr>
</tbody>
</table>

Function blocks - File services

1.4.2.6.5.3.3.1 FILE.Open (FB; Prefix filop)................................................... 2176
1.4.2.6.5.3.3.2 FILE.Close (FB; Prefix filcl).................................................... 2177
1.4.2.6.5.3.3.3 FILE.Write (FB; Prefix filwr).................................................... 2177
1.4.2.6.5.3.3.4 FILE.Read (FB; Prefix filrd).................................................... 2178
1.4.2.6.5.3.3.5 FILE.Rename (FB; Prefix filrn)............................................... 2179
1.4.2.6.5.3.3.6 FILE.Copy (FB; Prefix filcp).................................................... 2179
1.4.2.6.5.3.3.7 FILE.Delete (FB; Prefix fildl)................................................... 2180
1.4.2.6.5.3.3.8 FILE.Flush (FB; Prefix filfl)...................................................... 2180
1.4.2.6.5.3.3.9 FILE.GetAttribute (FB; Prefix filga)......................................... 2181
1.4.2.6.5.3.3.10 FILE.GetPos (FB; Prefix filgp)................................................ 2181
1.4.2.6.5.3.3.11 FILE.SetPos (FB; Prefix filsp)................................................ 2182
1.4.2.6.5.3.3.12 FILE.EOF (FB; Prefix fileof).................................................. 2182
1.4.2.6.5.3.3.13 FILE.GetSize (FB; Prefix filgs)............................................. 2182
1.4.2.6.5.3.3.14 FILE.GetTime (FB; Prefix filgt)............................................. 2183
1.4.2.6.5.3.3.15 FILE.SetAttribute (FB; Prefix fisla)....................................... 2183
1.4.2.6.5.3.3.16 FILE.GetProperty (FUN).......................................................... 2184

FILE.Open (FB; Prefix filop)

This function block of the %CAA_File.library opens an already existing file or creates a new one. The return value is a file handle, which can then be used as an input "hFile" in the function blocks %FILE.Read, %FILE.Write, %FILE.Close. There may be restrictions concerning the specification of the directory name, e.g. only capital letters allowed, for different targets. The function %FILE.GetProperty shows these restrictions.

Table 467: Input:

<table>
<thead>
<tr>
<th>xExecute</th>
<th>BOOL</th>
<th>see: % Chapter 1.4.2.6.1.1 “CAA Guidelines” on page 2121</th>
</tr>
</thead>
<tbody>
<tr>
<td>sFileName</td>
<td>CAA.FILENAME</td>
<td>Directory name:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>No special characters,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Separator for sub-directories “/”,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Absolute or relative path specifications</td>
</tr>
<tr>
<td></td>
<td></td>
<td>No empty spaces</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Use capital letters</td>
</tr>
</tbody>
</table>
eFileMode | FILE.MODE | Mode in which the file should be opened,
- FILE.MWRITE:
  - Write access; file will be overwritten or created
- FILE.MREAD:
  - Read access; file will be opened for reading only
- FILE.MRDWR:
  - Read- and write access; the file will be overwritten or created
- FILE.MAPPD:
  - File will be opened in WRITE mode, but the written data will be appended at the end of the file

xExclusive | BOOL | File access mode:
- TRUE: exclusive data access
- FALSE: multiple data access possible

**Table 468: Output:**

| xDone | BOOL | see: % Chapter 1.4.2.6.1.1 “CAA Guidelines” on page 2121 |
| xBusy | BOOL | see: % Chapter 1.4.2.6.1.1 “CAA Guidelines” on page 2121 |
| xError | BOOL | see: % Chapter 1.4.2.6.1.1 “CAA Guidelines” on page 2121 |
| eError | FILE.ERROR | Error ID |
| hFile | CAA.HANDLE | File handle |

**FILE.Close (FB; Prefix filcl)**

This function block of the %CAA_File.library terminates the file access, i.e. closes the file.

**Table 469: Input:**

| xExecute | BOOL | see: % Chapter 1.4.2.6.1.1 “CAA Guidelines” on page 2121 |
| hFile | CAA.HANDLE | File handle |

**Table 470: Output:**

| xDone | BOOL | see: % Chapter 1.4.2.6.1.1 “CAA Guidelines” on page 2121 |
| xBusy | BOOL | see: % Chapter 1.4.2.6.1.1 “CAA Guidelines” on page 2121 |
| xError | BOOL | see: % Chapter 1.4.2.6.1.1 “CAA Guidelines” on page 2121 |
| eError | FILE.ERROR | Error ID |

**FILE.Write (FB; Prefix filwr)**

This function block of the %CAA_File.library writes data into the file, which was previously opened via %FILE.Open. The contents of the memory area indicated by pointer "pBuffer" should not be modified during the write action! The size of the structure of the memory containing the bytes to be written as well as the number of bytes to be written will not be checked.

If the stability of the pointer on the data structures and their contents cannot be guaranteed in case of an online change, an online change must be avoided with the help of functions from the "CAA_Application.library".
### Table 471: Input:

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>xExecute</td>
<td>BOOL</td>
<td>see: % Chapter 1.4.2.6.1.1 “CAA Guidelines” on page 2121</td>
</tr>
<tr>
<td>xAbort</td>
<td>BOOL</td>
<td>see: % Chapter 1.4.2.6.1.1 “CAA Guidelines” on page 2121</td>
</tr>
<tr>
<td>udiTimeOut</td>
<td>UDINT</td>
<td></td>
</tr>
<tr>
<td>hFile</td>
<td>CAA.HANDLE</td>
<td>File handle</td>
</tr>
<tr>
<td>pBuffer</td>
<td>CAA.PVOID</td>
<td>The address from where the data can be obtained, can be determined with the help of operator ADR.</td>
</tr>
<tr>
<td>szSize</td>
<td>CAA.SIZE</td>
<td>Number of bytes to be written (can be determined via operator sizeof)</td>
</tr>
</tbody>
</table>

### Table 472: Output:

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>xDone</td>
<td>BOOL</td>
<td>see: Guidelines for CAA libs</td>
</tr>
<tr>
<td>xBusy</td>
<td>BOOL</td>
<td>see: Guidelines for CAA libs</td>
</tr>
<tr>
<td>xError</td>
<td>BOOL</td>
<td>see: Guidelines for CAA libs</td>
</tr>
<tr>
<td>xAborted</td>
<td>BOOL</td>
<td>see: Guidelines for CAA libs</td>
</tr>
<tr>
<td>eError</td>
<td>FILE.ERROR</td>
<td>Error ID</td>
</tr>
</tbody>
</table>

**FILE.Read (FB; Prefix filrd)**

This function block of the %CAA_File.library reads the file, which was previously opened via %FILE.Open. If less characters can be read than specified in "szBuffer", the function block returns an active "xDone" and indicates the current number of characters in "szSize". The size of the target memory structure for the bytes to be read and the number of bytes to be read will not be checked.

If the stability of the pointer on the data structures and their contents cannot be guaranteed in case of an online change, an online change must be avoided with the help of functions from the “CAA.Application library”.

### Table 473: Input:

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>xExecute</td>
<td>BOOL</td>
<td>see: % Chapter 1.4.2.6.1.1 “CAA Guidelines” on page 2121</td>
</tr>
<tr>
<td>xAbort</td>
<td>BOOL</td>
<td>see: % Chapter 1.4.2.6.1.1 “CAA Guidelines” on page 2121</td>
</tr>
<tr>
<td>udiTimeOut</td>
<td>UDINT</td>
<td></td>
</tr>
<tr>
<td>hFile</td>
<td>CAA.HANDLE</td>
<td>File handle</td>
</tr>
<tr>
<td>pBuffer</td>
<td>CAA.PVOID</td>
<td>Target address for the first byte to be read; can be retrieved via operator ADR</td>
</tr>
<tr>
<td>szBuffer</td>
<td>CAA.SIZE</td>
<td>Maximum number of bytes to be read; can be retrieved via operator sizeof</td>
</tr>
</tbody>
</table>

### Table 474: Output:

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>xDone</td>
<td>BOOL</td>
<td>see: Guidelines for CAA libs</td>
</tr>
<tr>
<td>xBusy</td>
<td>BOOL</td>
<td>see: Guidelines for CAA libs</td>
</tr>
<tr>
<td>xError</td>
<td>BOOL</td>
<td>see: Guidelines for CAA libs</td>
</tr>
<tr>
<td>xAborted</td>
<td>BOOL</td>
<td>see: Guidelines for CAA libs</td>
</tr>
</tbody>
</table>
FILE.Rename (FB; Prefix filrn)

This function block modifies the file name and is optional for the different targets. This does not work if the file is open via FILE.Open. Whether this function block is implemented or not is explicitly stated and encoded in the return value of the function "FILE.GetProperty" (see description of FILE.GetProperty). The function block cannot be used to move files or directories!

Table 475: Input:

<table>
<thead>
<tr>
<th>xExecute</th>
<th>BOOL</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>sFileNameOld</td>
<td>CAA.FILENAME</td>
<td>Old file name</td>
</tr>
<tr>
<td>sFileNameNew</td>
<td>CAA.FILENAME</td>
<td>New file name</td>
</tr>
</tbody>
</table>

Table 476: Output:

<table>
<thead>
<tr>
<th>xDone</th>
<th>BOOL</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>xBusy</td>
<td>BOOL</td>
<td></td>
</tr>
<tr>
<td>xError</td>
<td>BOOL</td>
<td></td>
</tr>
<tr>
<td>eError</td>
<td>FILE.ERROR</td>
<td>Error ID</td>
</tr>
</tbody>
</table>

FILE.Copy (FB; Prefix filcp)

This function block of the CAA_File.library copies a file. The target file is created or - if already existent - overwritten. The overwrite mode can be defined with the input "xOverWrite". If the overwrite mode is activated, no write protection may be activated in the existing file.

The function block is optional for different targets. Whether this function block is implemented or not is explicitly stated and encoded in the return value of the function "FILE.GetProperty" (see description of FILE.GetProperty).

Table 477: Input:

<table>
<thead>
<tr>
<th>xExecute</th>
<th>BOOL</th>
<th>see: Chapter 1.4.2.6.1.1 “CAA Guidelines” on page 2121</th>
</tr>
</thead>
<tbody>
<tr>
<td>xAbort</td>
<td>BOOL</td>
<td>see: Chapter 1.4.2.6.1.1 “CAA Guidelines” on page 2121</td>
</tr>
<tr>
<td>udiTimeOut</td>
<td>UDINT</td>
<td></td>
</tr>
<tr>
<td>sFileNameDest</td>
<td>CAA.FILENAME</td>
<td>File to which data should be copied; for restrictions concerning the file- and directory names see FILE.Open (FB)</td>
</tr>
<tr>
<td>sFileNameSource</td>
<td>CAA.FILENAME</td>
<td>File from which data should be copied, for restrictions concerning the file- and directory names see FILE.Open (FB)</td>
</tr>
<tr>
<td>xOverWrite</td>
<td>BOOL</td>
<td>Overwriting an already existing file with equal name TRUE: File will be overwritten FALSE: File will not be overwritten, an error message will be created</td>
</tr>
</tbody>
</table>
## FILE.Delete (FB; Prefix fildl)

This function block of the File library deletes a file. This is not possible if the file is open via FILE.Open.

### Table 478: Output:

<table>
<thead>
<tr>
<th>xDone</th>
<th>BOOL</th>
<th>see: Guidelines for CAA libs</th>
</tr>
</thead>
<tbody>
<tr>
<td>xBusy</td>
<td>BOOL</td>
<td>see: Guidelines for CAA libs</td>
</tr>
<tr>
<td>xError</td>
<td>BOOL</td>
<td>see: Guidelines for CAA libs</td>
</tr>
<tr>
<td>xAborted</td>
<td>BOOL</td>
<td>see: Guidelines for CAA libs</td>
</tr>
<tr>
<td>eError</td>
<td>FILE.ERROR</td>
<td>Error ID</td>
</tr>
<tr>
<td>szSize</td>
<td>CAA.SIZE</td>
<td>Current number of successfully copied bytes; this value is already valid before xDone is set</td>
</tr>
</tbody>
</table>

### Table 479: Input:

<table>
<thead>
<tr>
<th>xExecute</th>
<th>BOOL</th>
<th>see: Chapter 1.4.2.6.1.1 “CAA Guidelines” on page 2121</th>
</tr>
</thead>
<tbody>
<tr>
<td>sFileName</td>
<td>CAA.FILENAME</td>
<td>Name of the file to be deleted.</td>
</tr>
</tbody>
</table>

### Table 480: Output:

<table>
<thead>
<tr>
<th>xDone</th>
<th>BOOL</th>
<th>see: Chapter 1.4.2.6.1.1 “CAA Guidelines” on page 2121</th>
</tr>
</thead>
<tbody>
<tr>
<td>xBusy</td>
<td>BOOL</td>
<td>see: Chapter 1.4.2.6.1.1 “CAA Guidelines” on page 2121</td>
</tr>
<tr>
<td>xError</td>
<td>BOOL</td>
<td>see: Chapter 1.4.2.6.1.1 “CAA Guidelines” on page 2121</td>
</tr>
<tr>
<td>eError</td>
<td>FILE.ERROR</td>
<td>Error ID</td>
</tr>
</tbody>
</table>

## FILE.Flush (FB; Prefix filfl)

This function block flushes data from the file system cache to the disk for the file with given file handle hFile. Calling this function block guarantees that data will be stored on the disk.

### Table 481: Input:

<table>
<thead>
<tr>
<th>xExecute</th>
<th>BOOL</th>
<th>Rising edge: Action start</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Falling edge: Resets outputs If a falling edge occurs before the function block has completed its action, the outputs operate in the usual manner and are only reset if either the action is completed or in the event of an error. In this case, the corresponding output values (xDone, xError, iError) are present at the outputs for exactly one cycle.</td>
</tr>
<tr>
<td>hFile</td>
<td>CAA.HANDLE</td>
<td>Handle of the directory or the file</td>
</tr>
</tbody>
</table>

### Table 482: Output:

<table>
<thead>
<tr>
<th>xDone</th>
<th>BOOL</th>
<th>Action successfully completed</th>
</tr>
</thead>
<tbody>
<tr>
<td>xBusy</td>
<td>BOOL</td>
<td>Function block active</td>
</tr>
</tbody>
</table>
FILE.GetAttribute (FB; Prefix filga)

With this function block of the CAA_File.library directory and file attributes can be retrieved. It is optional for the different targets. Whether this function block is implemented or not is explicitly stated and encoded in the return value of the function "FILE.GetProperty" (see description of § FILE.GetProperty).

<table>
<thead>
<tr>
<th>Table 483: Input:</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>xExecute</strong></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td><strong>hFile</strong></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Table 484: Output:</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>xDone</strong></td>
</tr>
<tr>
<td><strong>xBusy</strong></td>
</tr>
<tr>
<td><strong>xError</strong></td>
</tr>
<tr>
<td><strong>eError</strong></td>
</tr>
<tr>
<td><strong>eFileAttrib</strong></td>
</tr>
</tbody>
</table>

FILE.GetPos (FB; Prefix filgp)

This function block of the § CAA_File.library returns the offset position currently set in the file. The file must be opened via § FILE.Open.

<table>
<thead>
<tr>
<th>Table 485: Input:</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>xExecute</strong></td>
</tr>
<tr>
<td><strong>hFile</strong></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Table 486: Output:</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>xDone</strong></td>
</tr>
<tr>
<td><strong>xBusy</strong></td>
</tr>
<tr>
<td><strong>xError</strong></td>
</tr>
<tr>
<td><strong>eError</strong></td>
</tr>
<tr>
<td><strong>udiPos</strong></td>
</tr>
</tbody>
</table>
FILE.SetPos (FB; Prefix filsp)
This function block of the $CAA_File.library modifies the current offset for a file access. The file must be opened via $FILE.Open.

Table 487: Input:

<table>
<thead>
<tr>
<th>xExecute</th>
<th>BOOL</th>
<th>see: $Chapter 1.4.2.6.1.1 “CAA Guidelines” on page 2121</th>
</tr>
</thead>
<tbody>
<tr>
<td>hFile</td>
<td>CAA.HANDLE</td>
<td>File handle</td>
</tr>
<tr>
<td>udiPos</td>
<td>UDINT</td>
<td>Offset position, Number of bytes from the start of the file</td>
</tr>
</tbody>
</table>

Table 488: Output:

<table>
<thead>
<tr>
<th>xDone</th>
<th>BOOL</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>xBusy</td>
<td>BOOL</td>
<td></td>
</tr>
<tr>
<td>xError</td>
<td>BOOL</td>
<td></td>
</tr>
<tr>
<td>eError</td>
<td>FILE.ERROR</td>
<td>Error ID</td>
</tr>
</tbody>
</table>

FILE.EOF (FB; Prefix fileof)
( $CAA_File library)
This function block sets “xEOF” on “TRUE” if the current offset is equal to the end of the file. If the end of the file has not yet been reached, “FALSE” is returned.

Table 489: Input:

<table>
<thead>
<tr>
<th>xExecute</th>
<th>BOOL</th>
<th>see: $Chapter 1.4.2.6.1.1 “CAA Guidelines” on page 2121</th>
</tr>
</thead>
<tbody>
<tr>
<td>hFile</td>
<td>CAA.HANDLE</td>
<td>File handle</td>
</tr>
</tbody>
</table>

Table 490: Output:

<table>
<thead>
<tr>
<th>xDone</th>
<th>BOOL</th>
<th>see: $Chapter 1.4.2.6.1.1 “CAA Guidelines” on page 2121</th>
</tr>
</thead>
<tbody>
<tr>
<td>xBusy</td>
<td>BOOL</td>
<td>see: $Chapter 1.4.2.6.1.1 “CAA Guidelines” on page 2121</td>
</tr>
<tr>
<td>xError</td>
<td>BOOL</td>
<td>see: $Chapter 1.4.2.6.1.1 “CAA Guidelines” on page 2121</td>
</tr>
<tr>
<td>eError</td>
<td>FILE.ERROR</td>
<td>Error ID</td>
</tr>
<tr>
<td>xEOF</td>
<td>BOOL</td>
<td>End of file reached</td>
</tr>
</tbody>
</table>

FILE.GetSize (FB; Prefix filgs)
This function block of the $CAA_File library returns the size of the file specified by “sFileName”.

Table 491: Input:

<table>
<thead>
<tr>
<th>xExecute</th>
<th>BOOL</th>
<th>see: $Chapter 1.4.2.6.1.1 “CAA Guidelines” on page 2121</th>
</tr>
</thead>
<tbody>
<tr>
<td>sFileName</td>
<td>STRING</td>
<td>File name</td>
</tr>
</tbody>
</table>
Table 492: Output:

<table>
<thead>
<tr>
<th>xDone</th>
<th>BOOL</th>
<th>see: Chapter 1.4.2.6.1.1 “CAA Guidelines” on page 2121</th>
</tr>
</thead>
<tbody>
<tr>
<td>xBusy</td>
<td>BOOL</td>
<td>see: Chapter 1.4.2.6.1.1 “CAA Guidelines” on page 2121</td>
</tr>
<tr>
<td>xError</td>
<td>BOOL</td>
<td>see: Chapter 1.4.2.6.1.1 “CAA Guidelines” on page 2121</td>
</tr>
<tr>
<td>eError</td>
<td>FILE.ERROR</td>
<td>Error ID</td>
</tr>
<tr>
<td>szSize</td>
<td>CAA.SIZE</td>
<td>File size in Bytes</td>
</tr>
</tbody>
</table>

FILE.GetTime (FB; Prefix filgt)

This function block of the CAA_File.library returns date and time of the last modification of the file specified by “sFileName”.

Table 493: Input:

<table>
<thead>
<tr>
<th>xExecute</th>
<th>BOOL</th>
<th>see: Chapter 1.4.2.6.1.1 “CAA Guidelines” on page 2121</th>
</tr>
</thead>
<tbody>
<tr>
<td>sFileName</td>
<td>CAA.FILENAME</td>
<td>File name</td>
</tr>
</tbody>
</table>

Table 494: Output:

<table>
<thead>
<tr>
<th>xDone</th>
<th>BOOL</th>
<th>see: Chapter 1.4.2.6.1.1 “CAA Guidelines” on page 2121</th>
</tr>
</thead>
<tbody>
<tr>
<td>xBusy</td>
<td>BOOL</td>
<td>see: Chapter 1.4.2.6.1.1 “CAA Guidelines” on page 2121</td>
</tr>
<tr>
<td>xError</td>
<td>BOOL</td>
<td>see: Chapter 1.4.2.6.1.1 “CAA Guidelines” on page 2121</td>
</tr>
<tr>
<td>eError</td>
<td>FILE.ERROR</td>
<td>Error ID</td>
</tr>
<tr>
<td>dtLastModification</td>
<td>DT</td>
<td>Date and time of the last file modification, for example dt#2006-05-08-00:00:00</td>
</tr>
</tbody>
</table>

FILE.SetAttribute (FB; Prefix filsa)

With this function block of the CAA_File.library directory and file attributes can be modified. It is optional for the different targets. Whether this function block is implemented or not is explicitly stated and encoded in the return value of the function “FILE.GetProperty” (see description of FILE.GetProperty).
Table 495: Input:

<table>
<thead>
<tr>
<th>xExecute</th>
<th>BOOL</th>
<th>see: $ Chapter 1.4.2.6.1.1 “CAA Guidelines” on page 2121</th>
</tr>
</thead>
<tbody>
<tr>
<td>hFile</td>
<td>CAA.HANDLE</td>
<td>Handle of the directory or the file</td>
</tr>
<tr>
<td>eFileAttrib</td>
<td>FILE.ATTRIB</td>
<td>File attributes to be set:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>FILE.ATTRIB_ARCHIVE - Archive attribute for storage purposes</td>
</tr>
<tr>
<td></td>
<td></td>
<td>FILE.ATTRIB_HIDDEN - Attribute for hidden files</td>
</tr>
<tr>
<td></td>
<td></td>
<td>FILE.ATTRIB_NORMAL - Attribute, if no other attribute is set</td>
</tr>
<tr>
<td></td>
<td></td>
<td>FILE.ATTRIB_READONLY - Write protection attribute; the file cannot be opened in FILE.WRITE_MODE or FILE.RDWR_MODE mode; an error message will be created</td>
</tr>
</tbody>
</table>

Table 496: Output:

<table>
<thead>
<tr>
<th>xDone</th>
<th>BOOL</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>xBusy</td>
<td>BOOL</td>
<td></td>
</tr>
<tr>
<td>xError</td>
<td>BOOL</td>
<td></td>
</tr>
<tr>
<td>eError</td>
<td>FILE.ERROR</td>
<td>Error ID</td>
</tr>
</tbody>
</table>

FILE.GetProperty (FUN)

This function of the $ CAA_File.library $ provides information on target-specific settings and functionalities, which concern the file- or directory system or the implementation of optional function blocks.

Table 497: Input:

<table>
<thead>
<tr>
<th>wProperty</th>
<th>WORD</th>
<th>Number of the property, see table below</th>
</tr>
</thead>
</table>

Table 498: Output:

<table>
<thead>
<tr>
<th>dwProperty</th>
<th>DWORD</th>
<th>Return value describes the property, see table below</th>
</tr>
</thead>
</table>

Table 499: Properties:

<table>
<thead>
<tr>
<th>Nr.</th>
<th>Constant name</th>
<th>Meaning</th>
<th>Return value</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>function block - support</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>FILE.gc_PDIRRENAME</td>
<td>Support of function block FILE.DirRename</td>
<td>0: No 1: Yes</td>
</tr>
<tr>
<td>2</td>
<td>FILE.gc_PRENAME</td>
<td>Support of function block FILE.Rename</td>
<td>0: No 1: Yes</td>
</tr>
<tr>
<td></td>
<td>Description</td>
<td>Support of function block</td>
<td>0: No</td>
</tr>
<tr>
<td>---</td>
<td>------------------------------------------------------------------------------</td>
<td>---------------------------------</td>
<td>-------</td>
</tr>
<tr>
<td>3</td>
<td>FILE.gc_PCOPY Support of function block FILE.Copy</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>FILE.gc_PSETATTRIBUTE Support of function block FILE.SetAttribute</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Restrictions on directories and files**

<table>
<thead>
<tr>
<th></th>
<th>Description</th>
<th>Support of the directory and file name format to 8.3 format</th>
<th>0: No</th>
<th>1: Yes</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>FILE.gc_PFFILENAME83 Restriction on capital letters for directory and file names</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Path specification**

<table>
<thead>
<tr>
<th></th>
<th>Description</th>
<th>Support of absolute path specification</th>
<th>0: No</th>
<th>1: Yes</th>
</tr>
</thead>
<tbody>
<tr>
<td>200</td>
<td>FILE.gc_PABSOLUTEPATH Restriction on absolute path specification</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>201</td>
<td>FILE.gc_PDRIVELETTER Support of drive letters</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>210</td>
<td>FILE.gc_PRESOLVER Support of Resolver Syntax for file names</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>211</td>
<td>FILE.gc_PFTPRESOLVER Support of FTP-Client functions</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>212</td>
<td>FILE.gc_PHTTPRESOLVER Support of http-Client functions</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Network**

<table>
<thead>
<tr>
<th></th>
<th>Description</th>
<th>Support of network drives</th>
<th>0: No</th>
<th>1: Yes</th>
</tr>
</thead>
<tbody>
<tr>
<td>300</td>
<td>FILE.gc_PNETWORKDRIVES Support of network drives</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Examples**

1.4.2.6.5.4.1 Root (EXAMPLE).................................................................................. 2185
1.4.2.6.5.4.2 Directory (EXAMPLE)............................................................................. 2187
1.4.2.6.5.4.3 File - Standard (EXAMPLE)...................................................................... 2188
1.4.2.6.5.4.4 File - Modification (EXAMPLE).......................................................... 2190
1.4.2.6.5.4.5 File – internal pointer (EXAMPLE).................................................... 2191
1.4.2.6.5.4.6 File – Properties (EXAMPLE)............................................................ 2194

**Root (EXAMPLE)**

(* example of how to use the CAA_FILE.library - root directory or path function blocks *)

```pascal
PROGRAM ROOT_PRG
VAR
  xRootInit:     BOOL := FALSE;
  uiRootState:   UINT := 0;
  sStandarddir:  CAA.FILENAME:='C:/PLC_DIR';
  sDirName:      CAA.FILENAME;
```
filsrt: FILE.SetRoot;
filgrt: FILE.GetRoot;
END_VAR

IF NOT xRootInit THEN
  filsrt(xExecute:=FALSE);
  filgrt(xExecute:=FALSE);
  xRootInit:=TRUE;
  uiRootState:=0;
ELSE
  CASE uiRootState OF
  0: (* read root directory *)
    filgrt(xExecute:=TRUE);
    IF filgrt.xDone THEN
      sDirName:=filgrt.sFileName;
      uiRootState:=1;
    END_IF
    IF filgrt.xError THEN
      (* error handling*)
      ;
    END_IF
  1: (* compare root directory *)
    IF sDirName <> sStandarddir THEN
      uiRootState:=2;       (* set the new root *)
    ELSE
      uiRootState:=3;       (* standard root directory already is set *)
    END_IF
  2: (* set new root directory *)
    filsrt.sFileName:=sStandarddir;
    filsrt(xExecute:=TRUE);
    IF filsrt.xDone THEN
      uiRootState:=3;
    END_IF
    IF filsrt.xError THEN
      (* error handling*)
      ;
    END_IF
  3: (* end of example*)
    ;
  END_CASE
END_IF
Directory (EXAMPLE)

(* example of how to use the CAA_FILE library - directory function blocks *)

PROGRAM DIR_PRG

VAR
    xDirInit:       BOOL := FALSE;
    uiDirState:     UINT := 0;
    sDirNewName:    STRING:='./TestDirectory';
    sDirNextName:   STRING:='./NewDirectory';
    hDir:           CAA.HANDLE;
    deNewDirectory: FILE.FILE_DIR_ENTRY;
    fildcr:         FILE.DirCreate;
    fildop:         FILE.DirOpen;
    fildcl:         FILE.DirClose;
    fildls:         FILE.DirList;
    fildrn:         FILE.DirRename;
    fildrm:         FILE.DirRemove;
END_VAR

IF NOT xDirInit THEN
    fildcr(xExecute:=FALSE);
    fildcl(xExecute:=FALSE);
    fildls(xExecute:=FALSE);
    fildrm(xExecute:=FALSE);
    xDirInit:=TRUE;
    uiDirState:=0;
END_IF

CASE uiDirState OF
    0: (* create a new directory *)
        fildcr.sDirName:=sDirNewName;
        fildcr.xParent:=FALSE;
        fildcr(xExecute:=TRUE);
        IF fildcr.xDone THEN
            uiDirState:=1;
        END_IF
        IF fildcr.xError THEN
            (* error handling*)
        END_IF
    1: (* open directory *)
        fildop.sDirName:=sDirNewName;
        fildop(xExecute:=TRUE);
        IF fildop.xDone THEN
            hDir := fildop.hDir;
            uiDirState:=2;
        END_IF
        IF fildop.xError THEN
            (* error handling*)
        END_IF
    2: (* get directory property list *)
        fildls.hDir:=hDir;
        fildls(xExecute:=TRUE);
        IF fildls.xDone THEN
            deNewDirectory.sEntry := fildls.deDirEntry.sEntry;
            deNewDirectory.szSize := fildls.deDirEntry.szSize;
            deNewDirectory.xDirectory := fildls.deDirEntry.xDirectory;
            deNewDirectory.xExclusive := fildls.deDirEntry.xExclusive;
            deNewDirectory.dtLastModification := fildls.deDirEntry.dtLastModification;
            uiDirState:=3;
        END_IF
        IF fildop.xError THEN
(* error handling*)

; END_IF

3: (* close directory *)
fildcl.hDir:=hDir;
fildcl(xExecute:=TRUE);
IF fildcl.xDone THEN
   uiDirState:=4;
END_IF

IF fildcl.xError THEN
   (* error handling*)

; END_IF

4: (* rename directory *)

fildrn.sDirNameOld:=sDirNewName;
fildrn.sDirNameNew:=sDirNextName;
fildrn(xExecute:=TRUE);
IF fildrn.xDone THEN
   uiDirState:=5;
END_IF

IF fildrn.xError THEN
   (* error handling*)

; END_IF

5: (* remove directory *)

fildrm.sDirName:=sDirNextName;
fildrm.udiTimeOut:=100000; (* 100ms TimeOut *)
fildrm.xRecursive:=FALSE;
fildrm(xExecute:=TRUE);
IF fildrm.xDone THEN
   uiDirState:=6;
END_IF

IF fildrm.xError THEN
   (* error handling*)

; END_IF

6: (* end of example *)

; END_CASE
END_IF

File - Standard (EXAMPLE)

(* example of how to use the CAA_FILE.library - file standard function blocks *)

PROGRAM FILE.STANDARD_PRG

VAR
   
xFileStdInit: BOOL:=FALSE;
uFileStdState: UINT:=0;
sFileName: CAA.FILENAME:= 'TestFile.txt';
hFile: CAA.HANDLE;
sFileTestString: STRING:='Hello caa library user';
sFileString: STRING(100):='';
szFileSize1: CAA.SIZE := 0;
szFileSize2: CAA.SIZE := 0;
filop: FILE.Open;
filwr: FILE.Write;
filrd: FILE.Read;
filcl: FILE.Close;
filsp: FILE.SetPos;
END_VAR
IF NOT xFileStdInit THEN
    filop(xExecute:=FALSE);
    filcl(xExecute:=FALSE);
    filwr(xExecute:=FALSE);
    filrd(xExecute:=FALSE);
    xFileStdInit:=TRUE;
    uiFileStdState:=0;
ELSE
    CASE uiFileStdState OF
        0:(* create a new file *)
            filop.sFileName:=sFileName;
            filop.eFileMode:=FILE.MODE.MRDWR;
            filop.xExclusive:=TRUE;
            filop(xExecute:=TRUE);
            IF filop.xDone THEN
                hFile:=filop.hFile;
                uiFileStdState:=1;
            END_IF
            IF filop.xError THEN
                (* error handling*)
            END_IF
        1:(* write text in the file *)
            filwr.hFile:=hFile;
            filwr.pBuffer:=ADR(sFileTestString);
            szFileSize1:=SIZEOF(sFileTestString);
            filwr.szSize:=szFileSize1;
            filwr.udiTimeOut:=100000;    (* 100ms Timeout *)
            filwr(xExecute:=TRUE);
            IF filwr.xDone THEN
                uiFileStdState:=2;
            END_IF
            IF filwr.xError THEN
                (* error handling*)
            END_IF
        2:(* set the internal positon file pointer *)
            filsp.hFile:= hFile;
            filsp.udiPos:=0;
            filsp(xExecute:=TRUE);
            IF filsp.xDone THEN
                uiFileStdState:=3;
            END_IF
            IF filsp.xError THEN
                (* error handling*)
            END_IF
        3:(* read file - TestFile.txt*)
            filrd.hFile:=hFile;
            filrd.udiTimeOut:=100000;    (* 100ms Timeout *)
            filrd.pBuffer:=ADR(sFileString);
            filrd.szBuffer:=SIZEOF(sFileString);
            filrd(xExecute:=TRUE);
            IF filrd.xDone THEN
                szFileSize2:=filrd.szSize;
                IF szFileSize2 = szFileSize1 THEN
                    uiFileStdState:=4;
                ELSE
                    (* error handling*)
                END_IF
            END_IF
File - Modification (EXAMPLE)

(* example of how to use the $CAA_FILE.library - file change function blocks *)

PROGRAM FILE.CHANGE_PRG

VAR

xFileChgInit: BOOL:=FALSE;
uiFileChgState: UINT:=0;
sFileOldName: CAA.FILENAME:= 'TestFile.txt';
sFileNewName: CAA.FILENAME:= 'NewFile.txt';
szCopiedFileSize: CAA.SIZE := 0;
filcp: FILE.Copy;
film: FILE.Rename;
fildl: FILE.Delete;

END_VAR

IF NOT xFileChgInit THEN
  fildl(xExecute:=FALSE);
  film(xExecute:=FALSE);
  filcp(xExecute:=FALSE);
  xFileChgInit:=TRUE;
ELSE
  CASE uiFileChgState OF
    0: (*copy file *)
      filcp.sFileNameSource:=sFileNewName;
      filcp.sFileNameDest:='DestFile.txt';
  END_CASE
END_IF

4: (* close file - TestFile.txt *)
filcl.hFile:=hFile;
filcl( xExecute:=TRUE);
IF filcl.xDone THEN
  uiFileStdState:=4;
END_IF
IF filcl.xError THEN
  (* error handling*)
END_IF

5: (* end of example *)
filcp.udITimeOut:=100000; (* 100ms Timeout *)
filcp.xOverWrite:=TRUE; (* overwrite the existing file *)
filcp( xExecute:=TRUE);
IF filcp.xDone THEN
    szCopiedFileSize := filcp.szSize;
    uiFileChgState:=1;
END_IF
IF filcp.xError THEN
    (* error handling *)
;
END_IF
1: (* rename file *)
filrn.sFileNameOld:='DestFile.txt';
filrn.sFileNameNew:=sFileNewName;
filrn( xExecute:=TRUE);
IF filrn.xDone THEN
    uiFileChgState:=2;
END_IF
IF filrn.xError THEN
    (* error handling *)
;
END_IF
2: (* delete file *)
fildl.sFileName:=sFileNewName;
fildl( xExecute:=TRUE);
IF fildl.xDone THEN
    uiFileChgState:=3;
END_IF
IF fildl.xError THEN
    (* error handling *)
;
END_IF
3:
(* end of example *)
;
END_CASE
END_IF

File – internal pointer (EXAMPLE)
(* example of how to use the \ FILE.library - file position function blocks *)
PROGRAM FILE.POINT_PRG
VAR
  xFilePosInit: BOOL:=FALSE;
  uiFilePosState: UINT:=0;
  udiActualPosition: UDINT:=0;
  udiActualEoFPosition: UDINT:=0;
  sFileName: CAA.FILENAME:='TestFile.txt';
  hFile: CAA.HANDLE;
  filop: FILE.Open;
  filcl: FILE.Close;
  filgp: FILE.GetPos;
  filsp: FILE.SetPos;
  fileof: FILE.EOF;
END_VAR

IF NOT xFilePosInit THEN
  filop(xExecute:=FALSE);
  filcl(xExecute:=FALSE);
  filgp(xExecute:=FALSE);
  filsp(xExecute:=FALSE);
  fileof(xExecute:=FALSE);
  xFilePosInit:=TRUE;
  uiFilePosState:=0;
ELSE
  CASE uiFilePosState OF
  0:(^ open file ^)
    filop.sFileName:= sFileName;
    filop.eFileMode:=FILE.MRDWR;
    filop.xExclusive:=TRUE;
    filop( xExecute:=TRUE);
    IF filop.xDone THEN
      hFile:=filop.hFile;
      uiFilePosState:=1;
    END_IF
    IF filop.xError THEN
      (* error handling*)
    END_IF
  1: (^ get actual internal positon file pointer ^)
    filgp.hFile:= hFile;
    filgp( xExecute:=TRUE);
    IF filgp.xDone THEN
      udiActualPosition:=filgp.udiPos;
    END_IF
  END_CASE
END_IF
uiFilePosState:=2;
END_IF
IF filgp.xError THEN
 (* error handling*)
;
END_IF
2: (* query - end of file is reached *)
fileof.hFile:= hFile;
fileof(xExecute:=TRUE);
IF fileof.xDone THEN
IF fileof.xEOF THEN
   udiActualEoFPosition:=udiActualPosition;
   END_IF
   uiFilePosState:=3;
   END_IF
END_IF
IF filgp.xError THEN
 (* error handling*)
;
END_IF

3: (* set the internal position file pointer *)
filsp.hFile:= hFile;
filsp.udiPos:=udiActualEoFPosition - 5;
IF filsp.udiPos < 0 THEN
   filsp.udiPos:=0;
END_IF
filsp(xExecute:=TRUE);
IF filsp.xDone THEN
   uiFilePosState:=4;
END_IF
IF filsp.xError THEN
 (* error handling*)
;
END_IF
4: (* close file *)
filcl.hFile:=hFile;
filcl(xExecute:=TRUE);
IF filcl.xDone THEN
   uiFilePosState:=5;
END_IF
IF filcl.xError THEN
(* error handling*)

END_IF

5: (* end of example*)

END_CASE

END_IF

File – Properties (EXAMPLE)

(* example of how to use the CAA_FILE.library - file property function blocks *)

PROGRAM FILE.PROP_PRG

VAR

  xFilePropInit: BOOL:=FALSE;
  uiFilePropState: UINT:=0;
  sFileName: CAA.FILENAME:= 'TestFile.txt';
  hFile: CAA.HANDLE;
  szFileSize: CAA.SIZE := 0;
  dtLastFileMod: DT;
  sLastFileModification: STRING:="";
  filop: FILE.Open;
  filcl: FILE.Close;
  filsa: FILE.SetAttribute;
  filgs: FILE.GetSize;
  filgt: FILE.GetTime;

END_VAR

IF NOT xFilePropInit THEN
  filop(xExecute:=FALSE);
  filcl(xExecute:=FALSE);
  filsa(xExecute:=FALSE);
  filgs(xExecute:=FALSE);
  filgt(xExecute:=FALSE);
  xFilePropInit:=TRUE;
  uiFilePropState:=0;
ELSE
  CASE uiFilePropState OF
    0: (* get file size *)
      filgs.sFileName:=sFileName;
      filgs( xExecute:=TRUE);
      IF filgs.xDone THEN
        szFileSize:=filgs.szSize;
  END_CASE
END_IF
uiFilePropState:=1;
END_IF
IF filgs.xError THEN
  (* error handling*)
  ;
END_IF
1: (* get file date of last modification *)
filgt.sFileName:=sFileName;
filgt(xExecute:=TRUE);
IF filgt.xDone THEN
  dtLastFileMod:=filgt.dtLastModification;
  sLastFileModification:=DT_TO_STRING(dtLastFileMod);
  uiFilePropState:=2;
END_IF
IF filgt.xError THEN
  (* error handling*)
  ;
END_IF
2: (* open file *)
filop.sFileName:=sFileName;
filop.eFileMode:=FILE.MRDWR;
filop.xExclusive:=TRUE;
filop(xExecute:=TRUE);
IF filop.xDone THEN
  hFile:=filop.hFile;
  uiFilePropState:=3;
END_IF
IF filop.xError THEN
  (* error handling*)
  ;
END_IF
3:(* set file attribute - read only *)
filsa.hFile:=hFile;
filsa.eFileAttrib:=FILE.READONLY;
filsa(xExecute:=TRUE);
IF filsa.xDone THEN
  uiFilePropState:=4;
END_IF
IF filsa.xError THEN
  (* error handling*)
  ;
END_IF

4:(* close file *)
filcl.hFile:=hFile;
filcl( xExecute:=TRUE);
IF filcl.xDone THEN
    uiFilePropState:=5;
END_IF
IF filcl.xError THEN
   (* error handling*)
END_IF
5:(* end of example*)

END_CASE
END_IF

1.4.2.7 MQTT client library

1.4.2.7.1 Function blocks

The function blocks in this library can only be executed in RUN mode of the Processor Module, not in simulation mode.

MQTT client library is based on OASIS MQTT specification (v3.1.1).

MQTT client protocol For further information see MQTT Client Protocol Chapter 1.5.6.3.5 "MQTT client protocol" on page 3429
The function block `MqttConnectWithCertBuffer` establishes a connection to a MQTT broker. This function block only has to be called once per connection. The input parameters of the function block are used to access to the broker.

Comparing to function block `MqttConnectWithCertFile` it is possible to establish a TLS connection with certificates from buffer. In this case copy the content of the certificate `.pem` file into a `STRING` variable. All line breaks from the file must be replace with the ST specific line break character `$n$`.

Each time a TLS connection is established to a broker, the AC500 calculates some key pairs. For AC500-eCo PLCs, this calculation can take up to 7 seconds. Hence, ensure correct configuration of your watchdog.
Input description

A rising edge starts the operation, the output Busy goes to TRUE. In the first cycle all other inputs are read and stored, afterwards they are ignored. A falling edge does not stop the operation. After Done = TRUE or Error = TRUE and Execute = FALSE all outputs will be reset.

Set a reference to the connection.

MQTT_CONNECTION is used for other function blocks to reference to the established connection.

Set the IP address of the MQTT Broker. The IP address needs to be a string like ‘192.168.0.1’.

Set the port number of the MQTT broker.

Decide if using secure channel (TLS, Secure := TRUE) for communication.
### ServerCert

<table>
<thead>
<tr>
<th>Data type</th>
<th>Default value</th>
<th>Range</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>POINTER TO STRING(MQTT_MAX_PEM_CERT_LEN)</td>
<td>0</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

Set server certificate in PEM format. Only necessary if using a TLS connection. MQTT_MAX_PEM_CERT_LEN is an internal constant which is set to 3072.

### ClientCert

<table>
<thead>
<tr>
<th>Data type</th>
<th>Default value</th>
<th>Range</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>POINTER TO STRING(MQTT_MAX_PEM_CERT_LEN)</td>
<td>0</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

Set (optional) client certificate in PEM format. Only necessary if using a TLS connection. MQTT_MAX_PEM_CERT_LEN is an internal constant which is set to 3072.

### ClientKey

<table>
<thead>
<tr>
<th>Data type</th>
<th>Default value</th>
<th>Range</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>POINTER TO STRING(MQTT_MAX_PEM_CERT_LEN)</td>
<td>0</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

Set (optional) client certificate in PEM format. Only necessary if using a TLS connection and if using a client certificate. MQTT_MAX_PEM_CERT_LEN is an internal constant which is set to 3072.

### ClientId

<table>
<thead>
<tr>
<th>Data type</th>
<th>Default value</th>
<th>Range</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>POINTER TO STRING(MQTT_MAX_PEM_CERT_LEN)</td>
<td>0</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

Set client identifier which is used to identify the client to the server. Only necessary if using a TLS connection. MQTT_MAX_CLIENT_ID_LEN is an internal constant which is set to 250.

### Username

<table>
<thead>
<tr>
<th>Data type</th>
<th>Default value</th>
<th>Range</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>STRING(MQTT_MAX_USERNAME_LEN)</td>
<td>Empty string</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

Set username for MQTT broker. MQTT_MAX_USERNAME_LEN is an internal constant which is set to 250.

### Password

<table>
<thead>
<tr>
<th>Data type</th>
<th>Default value</th>
<th>Range</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>STRING(MQTT_MAX_PASSWORD_LEN)</td>
<td>Empty string</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

Set password for MQTT broker. MQTT_MAX_PASSWORD_LEN is an internal constant which is set to 250.

### LastWill

<table>
<thead>
<tr>
<th>Data type</th>
<th>Default value</th>
<th>Range</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>MQTT_MESSAGE</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

Optional Last Will message.
### KeepAlive

<table>
<thead>
<tr>
<th>Data type</th>
<th>Default value</th>
<th>Range</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>WORD</td>
<td>600</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

Time in seconds, which will be passed to the MQTT Broker and which is normally used by the Broker to disconnect clients if no communication was made for 1,5 * KeepAlive.

### Timeout

<table>
<thead>
<tr>
<th>Data type</th>
<th>Default value</th>
<th>Range</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>WORD</td>
<td>30000</td>
<td>-</td>
<td>Millisecond</td>
</tr>
</tbody>
</table>

Timeout for connect and all subsequent calls (ping, subscribe, unsubscribe, publish with QoS > 0).

### Output description

```plaintext
DECLARE
    Execute : BOOL
    Done : BOOL
    Busy : BOOL
    Error : BOOL
    ErrorID : MQTT_ERROR_ID
    IpAddress : STRING(MQTT_MAX_IP_ADDRESS_LEN)
    Port : WORD
    ServerCert : POINTER TO STRING(MQTT_MAX_PEM_CERT_LEN)
    ClientCert : POINTER TO STRING(MQTT_MAX_PEM_CERT_LEN)
    ClientKey : POINTER TO STRING(MQTT_MAX_PEM_KEY_LEN)
    ClientId : STRING(MQTT_MAX_CLIENT_ID_LEN)
    Username : STRING(MQTT_MAX_USERNAME_LEN)
    Password : STRING(MQTT_MAX_PASSWORD_LEN)
    LastWill : MQTT_MESSAGE
    KeepAlive : WORD
    Timeout : WORD
```

### Done

<table>
<thead>
<tr>
<th>Data type</th>
<th>Default value</th>
<th>Range</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>BOOL</td>
<td>FALSE</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

Operation is completed without error (while outputs Busy and Error are FALSE). This output is TRUE for at least one cycle or until Execute is set to FALSE.

### Busy

<table>
<thead>
<tr>
<th>Data type</th>
<th>Default value</th>
<th>Range</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>BOOL</td>
<td>FALSE</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

Operation is running.

### Error

<table>
<thead>
<tr>
<th>Data type</th>
<th>Default value</th>
<th>Range</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>BOOL</td>
<td>FALSE</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

Operation is stopped with error (while outputs Busy and Done are FALSE). This output is TRUE for at least one cycle or until Execute is set to FALSE.
ErrorID

<table>
<thead>
<tr>
<th>Data type</th>
<th>Default value</th>
<th>Range</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>MQTT_ERROR_ID</td>
<td>MQTT_ERR_NO_ERROR</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

Provides an error number from enumeration "MQTT_ERROR_ID (Enum) " on page 2216 if an error occurred while processing the function block.

MqttConnectWithCertFile

The function block MqttConnectWithCertFile establishes a connection to a MQTT broker. This function block only has to be called once per connection. The input parameters of the function block are used to access to the broker.

Comparing to function block MqttConnectWithCertBuffer it is possible to establish a TLS connection with certificates from file.

If your PLC does not have a persistent file storage like sdcard or flashdisk, consider using MqttConnectWithCertBuffer.
Input description

**MQTTCONNECTWITHCERTFILE**

- **Execute**: BOOL
- **Conn**: POINTER TO MQTT_CONNECTION
- **IpAddress**: STRING(MQTT_MAX_IP_ADDRESS_LEN)
- **Port**: WORD
- **Secure**: BOOL
- **ServerCert**: STRING(MQTT_MAX_FILE_PATH_LEN)
- **ClientCert**: STRING(MQTT_MAX_FILE_PATH_LEN)
- **ClientKey**: STRING(MQTT_MAX_FILE_PATH_LEN)
- **ClientId**: STRING(MQTT_MAX_CLIENT_ID_LEN)
- **Username**: STRING(MQTT_MAX_USERNAME_LEN)
- **Password**: STRING(MQTT_MAX_PASSWORD_LEN)
- **LastWill**: MQTT_MESSAGE
- **KeepAlive**: WORD
- **Timeout**: WORD

**Execute**

<table>
<thead>
<tr>
<th>Data type</th>
<th>Default value</th>
<th>Range</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>BOOL</td>
<td>FALSE</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

A rising edge starts the operation, the output Busy goes to TRUE. In the first cycle all other inputs are read and stored, afterwards they are ignored. A falling edge does not stop the operation. After Done = TRUE or Error = TRUE and Execute = FALSE all outputs will be reset.

**Conn**

<table>
<thead>
<tr>
<th>Data type</th>
<th>Default value</th>
<th>Range</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>POINTER TO MQTT_CONNECTION</td>
<td>0</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

Set a reference to the connection.

MQTT_CONNECTION is used for other function blocks to reference to the established connection.

**IpAddress**

<table>
<thead>
<tr>
<th>Data type</th>
<th>Default value</th>
<th>Range</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>STRING</td>
<td>Empty string</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

IP address of the MQTT Broker. The IP address needs to be a string like ‘192.168.0.1’.

**Port**

<table>
<thead>
<tr>
<th>Data type</th>
<th>Default value</th>
<th>Range</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>WORD</td>
<td>8883</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

Set the port number of the MQTT broker.

**Secure**

<table>
<thead>
<tr>
<th>Data type</th>
<th>Default value</th>
<th>Range</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>BOOL</td>
<td>TRUE</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

Decide if using secure channel (TLS, Secure := TRUE) for communication.
<table>
<thead>
<tr>
<th>Data type</th>
<th>Default value</th>
<th>Range</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>ServerCert</td>
<td>STRING(MQTT_MAX_FILE_PATH_LEN)</td>
<td>Empty string</td>
<td>-</td>
</tr>
<tr>
<td>ClientCert</td>
<td>STRING(MQTT_MAX_FILE_PATH_LEN)</td>
<td>Empty string</td>
<td>-</td>
</tr>
<tr>
<td>ClientKey</td>
<td>STRING(MQTT_MAX_FILE_PATH_LEN)</td>
<td>Empty string</td>
<td>-</td>
</tr>
<tr>
<td>ClientId</td>
<td>POINTER TO STRING(MQTT_MAX_PEM_CERT_LEN)</td>
<td>0</td>
<td>-</td>
</tr>
<tr>
<td>Username</td>
<td>STRING(MQTT_MAX_USERNAME_LEN)</td>
<td>Empty string</td>
<td>-</td>
</tr>
<tr>
<td>Password</td>
<td>STRING(MQTT_MAX_PASSWORD_LEN)</td>
<td>Empty string</td>
<td>-</td>
</tr>
<tr>
<td>LastWill</td>
<td>MQTT_MESSAGE</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>KeepAlive</td>
<td>WORD</td>
<td>600</td>
<td>-</td>
</tr>
</tbody>
</table>

File name of server certificate in PEM format. Only necessary if using a TLS connection.

File name of client certificate in PEM format. Only necessary if using a TLS connection.

File name of client private key in PEM format (optional). Only necessary if using a TLS connection and if using a client certificate.

Set client identifier which is used to identify the client to the server. Only necessary if using a TLS connection. MQTT_MAX_CLIENT_ID_LEN is an internal constant which is set to 250.

Set username for MQTT broker. MQTT_MAX_USERNAME_LEN is an internal constant which is set to 250.

Set password for MQTT broker. MQTT_MAX_PASSWORD_LEN is an internal constant which is set to 250.

Optional Last Will message.
Time in seconds, which will be passed to the MQTT Broker and which is normally used by the Broker to disconnect clients if no communication was made for 1.5 * KeepAlive.

### Timeout

<table>
<thead>
<tr>
<th>Data type</th>
<th>Default value</th>
<th>Range</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>WORD</td>
<td>30000</td>
<td>-</td>
<td>Millisecond</td>
</tr>
</tbody>
</table>

Timeout for connect and all subsequent calls (ping, subscribe, unsubscribe, publish with QoS > 0).

### Output description

```plaintext
MQTTCONNECTWITHCERTFILE

- Execute : BOOL   Done : BOOL
- Conn : POINTER TO MQTT_CONNECTION  Busy : BOOL
- pAddress : STRING(MQTT_MAX_IP_ADDRESS_LEN)  Error : BOOL
- Port : WORD       ErrorID : MQTT_ERROR_ID
- Secure : BOOL
- ServerCert : STRING(MQTT_MAX_FILE_PATH_LEN)
- ClientCert : STRING(MQTT_MAX_FILE_PATH_LEN)
- ClientKey : STRING(MQTT_MAX_FILE_PATH_LEN)
- ClientId : STRING(MQTT_MAX_CLIENT_ID_LEN)
- Username : STRING(MQTT_MAX_USERNAME_LEN)
- Password : STRING(MQTT_MAX_PASSWORD_LEN)
- LastWill : MQTT_MESSAGE
- KeepAlive : WORD
- Timeout : WORD
```

### Done

<table>
<thead>
<tr>
<th>Data type</th>
<th>Default value</th>
<th>Range</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>BOOL</td>
<td>FALSE</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

Operation is completed without error (while outputs Busy and Error are FALSE). This output is TRUE for at least one cycle or until Execute is set to FALSE.

### Busy

<table>
<thead>
<tr>
<th>Data type</th>
<th>Default value</th>
<th>Range</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>BOOL</td>
<td>FALSE</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

Operation is running.

### Error

<table>
<thead>
<tr>
<th>Data type</th>
<th>Default value</th>
<th>Range</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>BOOL</td>
<td>FALSE</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

Operation is stopped with error (while outputs Busy and Done are FALSE). This output is TRUE for at least one cycle or until Execute is set to FALSE.
**ErrorID**

<table>
<thead>
<tr>
<th>Data type</th>
<th>Default value</th>
<th>Range</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>MQTT_ERROR_ID</td>
<td>MQTT_ERR_NO_ERROR</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

Provides an error number from enumeration “MQTT_ERROR_ID (Enum)” on page 2216 if an error occurred while processing the function block.

**MqttDisconnect**

![MqttDisconnect Diagram]

**Parameter**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Included in library</td>
<td>ABB_MqttClient_AC500.lib</td>
</tr>
<tr>
<td>Available as of firmware</td>
<td>V2.8</td>
</tr>
<tr>
<td>Type</td>
<td>Function block with historical values</td>
</tr>
<tr>
<td>Group</td>
<td>C interface</td>
</tr>
</tbody>
</table>

The function block MqttDisconnect is used to disconnect from the MQTT Broker. This function block resets the used socket. In case of an error during the connection, the used socket will be reset automatically.

**Input description**

![Input Diagram]

**Execute**

<table>
<thead>
<tr>
<th>Data type</th>
<th>Default value</th>
<th>Range</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>BOOL</td>
<td>FALSE</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

A rising edge starts the operation, the output Busy goes to TRUE. In the first cycle all other inputs are read and stored, afterwards they are ignored. A falling edge does not stop the operation. After Done = TRUE or Error = TRUE and Execute = FALSE all outputs will be reset.
Conn

<table>
<thead>
<tr>
<th>Data type</th>
<th>Default value</th>
<th>Range</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>POINTER TO MQTT_CONNECTION</td>
<td>0</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

Pointer to a valid connection structure created by MqttConnect.

Output description

```plaintext
MQTTDISCONNECT

Execute: BOOL
Done: BOOL
Conn: POINTER TO MQTT_CONNECTION
Busy: BOOL
Error: BOOL
ErrorID: MQTT_ERROR_ID
```

Done

<table>
<thead>
<tr>
<th>Data type</th>
<th>Default value</th>
<th>Range</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>BOOL</td>
<td>FALSE</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

Operation is completed without error (while outputs Busy and Error are FALSE). This output is TRUE for at least one cycle or until Execute is set to FALSE.

Busy

<table>
<thead>
<tr>
<th>Data type</th>
<th>Default value</th>
<th>Range</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>BOOL</td>
<td>FALSE</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

Operation is running.

Error

<table>
<thead>
<tr>
<th>Data type</th>
<th>Default value</th>
<th>Range</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>BOOL</td>
<td>FALSE</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

Operation is stopped with error (while outputs Busy and Done are FALSE). This output is TRUE for at least one cycle or until Execute is set to FALSE.

ErrorID

<table>
<thead>
<tr>
<th>Data type</th>
<th>Default value</th>
<th>Range</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>MQTT_ERROR_ID</td>
<td>MQTT_ERR_NO_ERROR</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

Provides an error number from enumeration "MQTT_ERROR_ID (Enum) " on page 2216 if an error occurred while processing the function block.
**MqttGetReceivedPacket**

The function block `MqttGetReceivedPacket` returns the first packet received for any subscribed topic. Done = TRUE with PayloadLen = 0 indicates that nothing new has been received since the last call.

**Input description**

A rising edge starts the operation, the output Busy goes to TRUE. In the first cycle all other inputs are read and stored, afterwards they are ignored. A falling edge does not stop the operation. After Done = TRUE or Error = TRUE and Execute = FALSE all outputs will be reset.

**Execute**

<table>
<thead>
<tr>
<th>Data type</th>
<th>Default value</th>
<th>Range</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>BOOL</td>
<td>FALSE</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

**Conn**

<table>
<thead>
<tr>
<th>Data type</th>
<th>Default value</th>
<th>Range</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>POINTER TO MQTT_CONNECTION</td>
<td>0</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

Pointer to a valid connection structure created by MqttConnect.
### Payload

<table>
<thead>
<tr>
<th>Data type</th>
<th>Default value</th>
<th>Range</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>POINTER TO BYTE</td>
<td>0</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

Pointer to the data area where the received packet can be stored.

### MaxPayloadSize

<table>
<thead>
<tr>
<th>Data type</th>
<th>Default value</th>
<th>Range</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>WORD</td>
<td>0</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

Size of the data area where the received packet can be stored.

### Output description

#### MQTTGETRECEIVEDPACKET

- **Execute**: BOOL
- **Conn**: POINTER TO MQTT_CONNECTION
- **Payload**: POINTER TO BYTE
- **MaxPayloadSize**: DWORD
- **ErrorID**: MQTT_ERROR_ID
- **Topic**: STRING(MQTT_MAX_TOPIC_LEN)
- **PayloadLen**: DWORD

#### Done

<table>
<thead>
<tr>
<th>Data type</th>
<th>Default value</th>
<th>Range</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>BOOL</td>
<td>FALSE</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

Operation is completed without error (while outputs Busy and Error are FALSE). This output is TRUE for at least one cycle or until Execute is set to FALSE.

#### Busy

<table>
<thead>
<tr>
<th>Data type</th>
<th>Default value</th>
<th>Range</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>BOOL</td>
<td>FALSE</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

Operation is running.

#### Error

<table>
<thead>
<tr>
<th>Data type</th>
<th>Default value</th>
<th>Range</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>BOOL</td>
<td>FALSE</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

Operation is stopped with error (while outputs Busy and Done are FALSE). This output is TRUE for at least one cycle or until Execute is set to FALSE.

#### ErrorID

<table>
<thead>
<tr>
<th>Data type</th>
<th>Default value</th>
<th>Range</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>MQTT_ERROR_ID</td>
<td>MQTT_ERR_NO_ERROR</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

Provides an error number from enumeration “MQTT_ERROR_ID (Enum)” on page 2216 if an error occurred while processing the function block.
### Topic

<table>
<thead>
<tr>
<th>Data type</th>
<th>Default value</th>
<th>Range</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>STRING(MQTT_MAX_TOPIC_LEN)</td>
<td>Empty string</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

Topic where the payload data belongs to.

### PayloadLen

<table>
<thead>
<tr>
<th>Data type</th>
<th>Default value</th>
<th>Range</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>STRING(MQTT_MAX_TOPIC_LEN)</td>
<td>Empty string</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

Actual length of the payload.

### MqttPing

![MqttPing diagram]

- **Parameter**
  - Included in library: ABB_MqttClient_AC500.lib
  - Available as of firmware: V2.8
  - Type: Function block with historical values
  - Group: C interface

MqttPing sends an MQTT Ping request to the broker and waits for the response packet. If output Done is TRUE and output Error is FALSE, the client still reaches the broker.

### Input description

![Input description diagram]

### Execute

<table>
<thead>
<tr>
<th>Data type</th>
<th>Default value</th>
<th>Range</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>BOOL</td>
<td>FALSE</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

![Execute diagram]
A rising edge starts the operation, the output Busy goes to TRUE. In the first cycle all other inputs are read and stored, afterwards they are ignored. A falling edge does not stop the operation. After Done = TRUE or Error = TRUE and Execute = FALSE all outputs will be reset.

### Conn

<table>
<thead>
<tr>
<th>Data type</th>
<th>Default value</th>
<th>Range</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>POINTER TO MQTT_CONNECTION</td>
<td>0</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

Pointer to a valid connection structure created by MqttConnect.

### Output description

**MQTTPING**

- **Execute**: BOOL
- **Done**: BOOL
- **Conn**: POINTER TO MQTT_CONNECTION
- **Busy**: BOOL
- **Error**: BOOL
- **ErrorID**: MQTT_ERROR_ID

### Done

<table>
<thead>
<tr>
<th>Data type</th>
<th>Default value</th>
<th>Range</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>BOOL</td>
<td>FALSE</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

Operation is completed without error (while outputs Busy and Error are FALSE). This output is TRUE for at least one cycle or until Execute is set to FALSE.

### Busy

<table>
<thead>
<tr>
<th>Data type</th>
<th>Default value</th>
<th>Range</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>BOOL</td>
<td>FALSE</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

Operation is running.

### Error

<table>
<thead>
<tr>
<th>Data type</th>
<th>Default value</th>
<th>Range</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>BOOL</td>
<td>FALSE</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

Operation is stopped with error (while outputs Busy and Done are FALSE). This output is TRUE for at least one cycle or until Execute is set to FALSE.

### ErrorID

<table>
<thead>
<tr>
<th>Data type</th>
<th>Default value</th>
<th>Range</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>MQTT_ERROR_ID</td>
<td>MQTT_ERR_NO_ERROR</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

Provides an error number from enumeration "MQTT_ERROR_ID (Enum) " on page 2216 if an error occurred while processing the function block.
MqttPublish

MqttPublish publishes a MQTT message to MQTT Broker. The properties of the message (Topic, Payload, QOS, Retain Flag) can be set with the message input.

The Timeout that was set on the MqttConnect POU’s is used when a QoS Level of 1 or 2 is set for a message.

This means that this POU will wait for an answer from the broker until the Timeout is reached when the message is either QoS Level 1 or Level 2.

It will not wait for any answer when QoS Level is 0.

The application using this FB has to define if the message shall be re-published and how often that shall be done.

Input description

Execute

A rising edge starts the operation, the output Busy goes to TRUE. In the first cycle all other inputs are read and stored, afterwards they are ignored. A falling edge does not stop the operation. After Done = TRUE or Error = TRUE and Execute = FALSE all outputs will be reset.
### Conn

<table>
<thead>
<tr>
<th>Data type</th>
<th>Default value</th>
<th>Range</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>POINTER TO MQTT_CONNECTION</td>
<td>0</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

Pointer to a valid connection structure created by MqttConnect.

### Message

<table>
<thead>
<tr>
<th>Data type</th>
<th>Default value</th>
<th>Range</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>MQTT_MESSAGE</td>
<td>See description of MQTT_MESSAGE</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

Defines the Message to be published.

### Output description

#### MQTT_PUBLIC

- **Execute**: BOOL
- **Done**: BOOL
- **Conn**: POINTER TO MQTT_CONNECTION
- **Busy**: BOOL
- **Message**: MQTT_MESSAGE
- **Error**: BOOL
- **ErrorID**: MQTT_ERROR_ID

#### Done

<table>
<thead>
<tr>
<th>Data type</th>
<th>Default value</th>
<th>Range</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>BOOL</td>
<td>FALSE</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

Operation is completed without error (while outputs Busy and Error are FALSE). This output is TRUE for at least one cycle or until Execute is set to FALSE.

#### Busy

<table>
<thead>
<tr>
<th>Data type</th>
<th>Default value</th>
<th>Range</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>BOOL</td>
<td>FALSE</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

Operation is running.

#### Error

<table>
<thead>
<tr>
<th>Data type</th>
<th>Default value</th>
<th>Range</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>BOOL</td>
<td>FALSE</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

Operation is stopped with error (while outputs Busy and Done are FALSE). This output is TRUE for at least one cycle or until Execute is set to FALSE.

#### ErrorID

<table>
<thead>
<tr>
<th>Data type</th>
<th>Default value</th>
<th>Range</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>MQTT_ERROR_ID</td>
<td>MQTT_ERR_NO_ERROR</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

Provides an error number from enumeration "MQTT_ERROR_ID (Enum) " on page 2216 if an error occurred while processing the function block.
MqttSubscribe

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Included in library</td>
<td>ABB_MqttClient_AC500.lib</td>
</tr>
<tr>
<td>Available as of firmware</td>
<td>V2.8</td>
</tr>
<tr>
<td>Type</td>
<td>Function block with historical values</td>
</tr>
<tr>
<td>Group</td>
<td>C interface</td>
</tr>
</tbody>
</table>

The function block MqttSubscribe is used to subscribe to a topic name like 'topic/name'. Wildcard symbols (#/+).

It can be used for subscribing like city/# or city/+/temperature. If the connection between the client and the broker is interrupted, the client needs to subscribe again to a topic after a new connect.

Input description

<table>
<thead>
<tr>
<th>Execute</th>
<th>Data type</th>
<th>Default value</th>
<th>Range</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>BOOL</td>
<td>FALSE</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

A rising edge starts the operation, the output Busy goes to TRUE. In the first cycle all other inputs are read and stored, afterwards they are ignored. A falling edge does not stop the operation. After Done = TRUE or Error = TRUE and Execute = FALSE all outputs will be reset.

<table>
<thead>
<tr>
<th>Conn</th>
<th>Data type</th>
<th>Default value</th>
<th>Range</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>POINTER TO MQTT_CONNECTION</td>
<td>0</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

Pointer to a valid connection structure created by MqttConnect.
### Topic

<table>
<thead>
<tr>
<th>Data type</th>
<th>Default value</th>
<th>Range</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>STRING(MQTT_MAX_TOPIC_LEN)</td>
<td>Empty string</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

Set the Topic name as a string. The topic name must look like ‘topic/name’. Wildcard symbols can also be used for subscribing (for example: ‘city/#’ or ‘city/+/temperature’). For further information see the MQTT specification.

### MaxQos

<table>
<thead>
<tr>
<th>Data type</th>
<th>Default value</th>
<th>Range</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>MQTT_QOS</td>
<td>QOS_0</td>
<td>BASC_NORMAL, BASC_REVERSE</td>
<td>-</td>
</tr>
</tbody>
</table>

This input is used to signal the server which is the highest QoS level which can be handled. Can be reduced to relieve the client from high network load.

### Output description

**MQTTSUBSCRIBE**

- Execute : BOOL
- Conn : POINTER TO MQTT_CONNECTION
- Topic : STRING(MQTT_MAX_TOPIC_LEN)
- MaxQos : MQTT_QOS
- ErrorID : MQTT_ERROR_ID

**Done**

<table>
<thead>
<tr>
<th>Data type</th>
<th>Default value</th>
<th>Range</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>BOOL</td>
<td>FALSE</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

Operation is completed without error (while outputs Busy and Error are FALSE). This output is TRUE for at least one cycle or until Execute is set to FALSE.

**Busy**

<table>
<thead>
<tr>
<th>Data type</th>
<th>Default value</th>
<th>Range</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>BOOL</td>
<td>FALSE</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

Operation is running.

**Error**

<table>
<thead>
<tr>
<th>Data type</th>
<th>Default value</th>
<th>Range</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>BOOL</td>
<td>FALSE</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

Operation is stopped with error (while outputs Busy and Done are FALSE). This output is TRUE for at least one cycle or until Execute is set to FALSE.

**ErrorID**

<table>
<thead>
<tr>
<th>Data type</th>
<th>Default value</th>
<th>Range</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>MQTT_ERROR_ID</td>
<td>MQTT_ERR_NO_ERROR</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

Provides an error number from enumeration “MQTT_ERROR_ID (Enum)” on page 2216 if an error occurred while processing the function block.
### MqttUnsubscribe

The function block MqttUnsubscribe is used to unsubscribe from a topic name like 'topic/name'. Wildcard symbols (#/+ can be used for subscribing like city/# or city/+temperature.

#### Parameter

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Included in library</td>
<td>ABB_MqttClient_AC500.lib</td>
</tr>
<tr>
<td>Available as of firmware</td>
<td>V2.8</td>
</tr>
<tr>
<td>Type</td>
<td>Function block with historical values</td>
</tr>
<tr>
<td>Group</td>
<td>C interface</td>
</tr>
</tbody>
</table>

#### Input description

- **Execute**
  - **Data type**: BOOL
  - **Default value**: FALSE
  - **Range**: -
  - **Unit**: -
  
  A rising edge starts the operation, the output Busy goes to TRUE. In the first cycle all other inputs are read and stored, afterwards they are ignored. A falling edge does not stop the operation. After Done = TRUE or Error = TRUE and Execute = FALSE all outputs will be reset.

- **Conn**
  - **Data type**: POINTER TO MQTT_CONNECTION
  - **Default value**: 0
  - **Range**: -
  - **Unit**: -

  Pointer to a valid connection structure created by MqttConnect.
Output description

**Done**

<table>
<thead>
<tr>
<th>Data type</th>
<th>Default value</th>
<th>Range</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>BOOL</td>
<td>FALSE</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

Operation is completed without error (while outputs Busy and Error are FALSE). This output is TRUE for at least one cycle or until Execute is set to FALSE.

**Busy**

<table>
<thead>
<tr>
<th>Data type</th>
<th>Default value</th>
<th>Range</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>BOOL</td>
<td>FALSE</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

Operation is running.

**Error**

<table>
<thead>
<tr>
<th>Data type</th>
<th>Default value</th>
<th>Range</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>BOOL</td>
<td>FALSE</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

Operation is stopped with error (while outputs Busy and Done are FALSE). This output is TRUE for at least one cycle or until Execute is set to FALSE.

**ErrorID**

<table>
<thead>
<tr>
<th>Data type</th>
<th>Default value</th>
<th>Range</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>MQTT_ERROR_ID</td>
<td>MQTT_ERR_NO_ERROR</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

Provides an error number from enumeration “MQTT_ERROR_ID (Enum)” on page 2216 if an error occurred while processing the function block.

### 1.4.2.7.2 Structures and enumerations

**MQTT_ERROR_ID (Enum)**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MQTT_ERR_NO_ERROR</td>
<td>0</td>
<td>No error.</td>
</tr>
<tr>
<td>MQTT_ERR_CONN_SERVICE_UNAVAIL</td>
<td>16#3001</td>
<td>The Network Connection has been made but the MQTT service is unavailable on the specified port.</td>
</tr>
<tr>
<td>MQTT_ERR_COMMUNICATION_TIMEOUT</td>
<td>16#3013</td>
<td>The timeout value for the communication has been exceeded.</td>
</tr>
<tr>
<td>MQTT_ERR_REC_PACKET_TOO_LONG</td>
<td>16#3017</td>
<td>Received topic is too long.</td>
</tr>
<tr>
<td>MQTT_ERR_PING_NO_ANSWER</td>
<td>16#301A</td>
<td>The MQTT broker did not answer the ping. MQTT client has passed the KeepAlive or MQTT broker is unreachable.</td>
</tr>
<tr>
<td>MQTT_ERR_CONN_CLIENT_ID_NOT_ALLOWED</td>
<td>16#301F</td>
<td>The Client identifier is correct UTF-8 but not allowed by the Server.</td>
</tr>
<tr>
<td>Parameter</td>
<td>Value</td>
<td>Description</td>
</tr>
<tr>
<td>-----------</td>
<td>---------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>MQTT_ERR_CONN_REFUSED_PROTOCOL</td>
<td>16#3020</td>
<td>The Server does not support the level of the MQTT protocol requested by the Client.</td>
</tr>
<tr>
<td>MQTT_ERR_CONN_REFUSED_CONNECTION</td>
<td>16#3025</td>
<td>Connection refused, maybe the IP address is malformed.</td>
</tr>
<tr>
<td>MQTT_ERR_UNSPECIFIED_ERROR</td>
<td>16#302B</td>
<td>Internal library returned an unspecified error.</td>
</tr>
<tr>
<td>MQTT_ERR_NETWORK_ERROR</td>
<td>16#302D</td>
<td>General network error.</td>
</tr>
<tr>
<td>MQTT_ERR_CONN_AUTH_FAILED</td>
<td>16#3217</td>
<td>Authentication failed: Bad username, password OR client id.</td>
</tr>
<tr>
<td>MQTT_ERR_CONN_TLS_HANDSHAKE_FAILED</td>
<td>16#3230</td>
<td>Error on TLS handshake.</td>
</tr>
<tr>
<td>MQTT_ERR_CONN_SERVER_CERT_NOT_VALID</td>
<td>16#3231</td>
<td>Server certificate not valid. Check if PLC date has been set correctly.</td>
</tr>
<tr>
<td>MQTT_ERR_CONN_SERVER_CERT_NOT_PEM</td>
<td>16#3232</td>
<td>Server certificate format is not formatted as PEM.</td>
</tr>
<tr>
<td>MQTT_ERR_CONN_SERVER_CERT_EXPIRED</td>
<td>16#3233</td>
<td>Server certificate has expired.</td>
</tr>
<tr>
<td>MQTT_ERR_CONN_CLIENT_CERT_NOT_VALID</td>
<td>16#3234</td>
<td>Client certificate not valid. Check if PLC date has been set correctly.</td>
</tr>
<tr>
<td>MQTT_ERR_CONN_CLIENT_CERT_NOT_PEM</td>
<td>16#3235</td>
<td>Client certificate or client key format is not formatted as PEM.</td>
</tr>
<tr>
<td>MQTT_ERR_CONN_CLIENT_CERT_EXPIRED</td>
<td>16#3236</td>
<td>Client certificate has expired.</td>
</tr>
<tr>
<td>MQTT_ERR_INPUT_02_0</td>
<td>16#4020</td>
<td>Function Block Input 02 error (error case 0), specific error depends on used function block:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>● MqttConnectWithCertBuffer (FB): Parameter Conn of Function Block was not set.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>● MqttConnectWithCertFile (FB): Parameter Conn of Function Block was not set.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>● MqttGetReceivedPacket (FB): Parameter Conn of Function Block was not set.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>● MqttPublish (FB): Parameter Conn of Function Block was not set.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>● MqttSubscribe (FB): Parameter Conn of Function Block was not set.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>● MqttUnsubscribe (FB): Parameter Conn of Function Block was not set.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>● MqttPing (FB): Parameter Conn of Function Block was not set.</td>
</tr>
<tr>
<td>Parameter</td>
<td>Value</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------</td>
<td>--------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
</tbody>
</table>
| MQTT_ERR_INPUT_03_0       | 16#4030| Function Block Input 03 error (error case 0), specific error depends on used function block:  
|                           |        | - MqttGetReceivedPacket (FB): Pointer payload not initialized.              |
|                           |        | - MqttPublish (FB): Publish topic name must not contain wildcard characters (+ or #). |
|                           |        | - MqttSubscribe (FB): Topic is missing.                                     |
|                           |        | - MqttUnsubscribe (FB): Topic is missing.                                   |
| MQTT_ERR_INPUT_03_1       | 16#4031| Function Block Input 03 error (error case 1), specific error depends on used function block:  
|                           |        | - MqttPublish (FB): Payload is not set in MQTT_MESSAGE.                    |
| MQTT_ERR_INPUT_04_0       | 16#4040| Function Block Input 04 error (error case 0), specific error depends on used function block:  
|                           |        | - MqttConnectWithCertBuffer (FB): Check if Port number has been set correctly (0 is not accepted). |
|                           |        | - MqttConnectWithCertFile (FB): Check if Port number has been set correctly (0 is not accepted). |
| MQTT_ERR_INPUT_06_0       | 16#4060| Function Block Input 06 error (error case 0), specific error depends on used function block:  
|                           |        | - MqttConnectWithCertFile (FB): Server certificate file was not found.      |
| MQTT_ERR_INPUT_07_0       | 16#4070| Function Block Input 07 error (error case 0), specific error depends on used function block:  
|                           |        | - MqttConnectWithCertFile (FB): Client certificate file was not found.      |
| MQTT_ERR_INPUT_08_0       | 16#4080| Function Block Input 08 error (error case 0), specific error depends on used function block:  
|                           |        | - MqttConnectWithCertFile (FB): Client key file was not found.              |
| MQTT_ERR_INPUT_12_0       | 16#4120| Function Block Input 12 error (error case 0), specific error depends on used function block:  
<p>|                           |        | - MqttConnectWithCertBuffer (FB): Couldn't initialize Last Will message because the topic is not set. |
|                           |        | - MqttConnectWithCertFile (FB): Couldn't initialize Last Will message because the payload is not set. |</p>
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MQTT_ERR_INPUT_12_1</td>
<td>16#4121</td>
<td>Function Block Input 12 error (error case 1), specific error depends on used function block:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>● MqttConnectWithCertBuffer (FB): Couldn't initialize Last Will message because the topic is not set.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>● MqttConnectWithCertFile (FB): Couldn't initialize Last Will message because the payload is not set.</td>
</tr>
<tr>
<td>MQTT_ERR_FATAL_ERROR</td>
<td>16#FFFF</td>
<td>Fatal error state machine.</td>
</tr>
</tbody>
</table>

### MQTT_QoS (Enum)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>QOS_0</td>
<td>-</td>
<td>Fire and forget (At most once delivered).</td>
</tr>
<tr>
<td>QOS_1</td>
<td>-</td>
<td>Simple acknowledgement (At least once delivered).</td>
</tr>
<tr>
<td>QOS_2</td>
<td>-</td>
<td>Complex acknowledgement (Exactly once delivered).</td>
</tr>
</tbody>
</table>

### MQTT_Message

This structure is used for messages which can be published or used for LastWill on MqttConnect(FB).

<table>
<thead>
<tr>
<th>Variable name</th>
<th>Data type</th>
<th>Default value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sTopic</td>
<td>STRING(MQTT_MAX_TOPIC_LEN)</td>
<td>Empty string</td>
<td>Topic where this message belongs to.</td>
</tr>
<tr>
<td>abyPayload</td>
<td>POINTER TO BYTE</td>
<td>0</td>
<td>Payload which should be sent.</td>
</tr>
<tr>
<td>dwLen</td>
<td>DWORD</td>
<td>0</td>
<td>Length of the payload.</td>
</tr>
<tr>
<td>eQos</td>
<td>MQTT_QOS</td>
<td>QOS_0</td>
<td>Quality of Service level.</td>
</tr>
<tr>
<td>xRetainFlag</td>
<td>BOOL</td>
<td>FALSE</td>
<td>True = message must be stored by the server, False = server must not store this message.</td>
</tr>
</tbody>
</table>

### MQTT_Connection

Internal data required by the library to operate. This structure allocates memory and it is used to identify the MQTT connection you want to work with.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>abyConn</td>
<td>Array</td>
<td>MQTT_CLIENT_STRUCT_SIZE</td>
</tr>
<tr>
<td>abyTxBuf</td>
<td>Array</td>
<td>MQTT_TX_BUF_SIZE</td>
</tr>
<tr>
<td>abyRxBuf</td>
<td>Array</td>
<td>MQTT_RX_BUF_SIZE</td>
</tr>
<tr>
<td>abyMsgBuf</td>
<td>Array</td>
<td>MQTT_MSG_BUF_SIZE</td>
</tr>
</tbody>
</table>
### 1.4.2.7.3 Global variables

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Datatype</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MQTT_MAX_IP_ADDRESS_LEN</td>
<td>Word</td>
<td>15</td>
<td>Maximum length of the IP address.</td>
</tr>
<tr>
<td>MQTT_MAX_PEM_KEY_LEN</td>
<td>Word</td>
<td>2048</td>
<td>Maximum length of the PEM key.</td>
</tr>
<tr>
<td>MQTT_MAX_PEM_CERT_LEN</td>
<td>Word</td>
<td>3072</td>
<td>Maximum length of the PEM certificate.</td>
</tr>
<tr>
<td>MQTT_MAX_FILE_PATH_LEN</td>
<td>Word</td>
<td>255</td>
<td>Maximum length of the file path to the certificate files.</td>
</tr>
<tr>
<td>MQTT_MAX_CLIENT_ID_LEN</td>
<td>Word</td>
<td>250</td>
<td>Maximum length of the client id.</td>
</tr>
<tr>
<td>MQTT_MAX_USERNAME_LEN</td>
<td>Word</td>
<td>250</td>
<td>Maximum length of the user-name.</td>
</tr>
<tr>
<td>MQTT_MAX_PASSWORD_LEN</td>
<td>Word</td>
<td>250</td>
<td>Maximum length of the password.</td>
</tr>
<tr>
<td>MQTT_MAX_TOPIC_LEN</td>
<td>Word</td>
<td>255</td>
<td>Maximum length of the topic.</td>
</tr>
<tr>
<td>MQTT_CLIENT_STRUCT_SIZE</td>
<td>Word</td>
<td>336</td>
<td>Size of the internal connection structure representing the connection state.</td>
</tr>
<tr>
<td>MQTT_TX_BUF_SIZE</td>
<td>Word</td>
<td>1024</td>
<td>Size of the internally used output buffer.</td>
</tr>
<tr>
<td>MQTT_RX_BUF_SIZE</td>
<td>Word</td>
<td>1024</td>
<td>Size of the internally used input buffer.</td>
</tr>
<tr>
<td>MQTT_MSG_BUF_SIZE</td>
<td>Word</td>
<td>2148</td>
<td>Size of the internally used message buffer.</td>
</tr>
</tbody>
</table>

### 1.5 PLC integration

#### 1.5.1 Product overview and comparison

#### 1.5.1.1 Comparison of AC500 V3 terminal bases

With the latest Automation Builder version the following terminal bases are compatible with the AC500 V3 processor modules:

**Terminal base compatibility**

<table>
<thead>
<tr>
<th>Terminal Base</th>
<th>PM5630</th>
<th>PM5650</th>
<th>PM5670</th>
<th>PM5675</th>
</tr>
</thead>
<tbody>
<tr>
<td>TB5600</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TB5610</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TB5620</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TB5640</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TB5660</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The number of slots that are available on a terminal base for connecting communication modules or AC500-S modules differs within the terminal base range.
Table 500: Combination of TB56xx-2ETH(-XC) and PM56xx(-XC)

<table>
<thead>
<tr>
<th>Processor module</th>
<th>PM5630</th>
<th>PM5650</th>
<th>PM5670</th>
<th>PM5675</th>
</tr>
</thead>
<tbody>
<tr>
<td>TB5600-2ETH</td>
<td>0 slot</td>
<td>0 slot</td>
<td>0 slot</td>
<td>0 slot</td>
</tr>
<tr>
<td>TB5610-2ETH</td>
<td>1 slot</td>
<td>1 slot</td>
<td>1 slot</td>
<td>1 slot</td>
</tr>
<tr>
<td>TB5620-2ETH</td>
<td>2 slots</td>
<td>2 slots</td>
<td>2 slots</td>
<td>2 slots</td>
</tr>
<tr>
<td>TB5640-2ETH</td>
<td>-</td>
<td>4 slots</td>
<td>4 slots</td>
<td>4 slots</td>
</tr>
<tr>
<td>TB5660-2ETH</td>
<td>-</td>
<td>-</td>
<td>6 slots(^1)</td>
<td>6 slots(^1)</td>
</tr>
</tbody>
</table>

Remarks:
The slots can be used for connecting communication modules or AC500-S modules. Note that only one AC500-S module can be connected at one terminal base.

\(^1\) PM567x must have an index \(\geq C0\).

Supported devices
The AC500 V3 terminal bases can be equipped with the following supported devices:

Table 501: Comparison: TB56xx

<table>
<thead>
<tr>
<th>Processor module</th>
<th>PM5630</th>
<th>PM5650</th>
<th>PM5670</th>
<th>PM5675</th>
</tr>
</thead>
<tbody>
<tr>
<td>Max. number of variables allowed for each communication module supported</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Input variables</td>
<td>4 kB</td>
<td>4 kB</td>
<td>5 kB</td>
<td>5 kB</td>
</tr>
<tr>
<td>Output variables</td>
<td>4 kB</td>
<td>4 kB</td>
<td>5 kB</td>
<td>5 kB</td>
</tr>
<tr>
<td>Type of communication module supported</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CM574-RS/RCOM - serial interface</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>CM582-DP - PROFIBUS DP V0/V1 slave</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>CM592-DP - PROFIBUS DP V0/V1 master</td>
<td>1(^1)</td>
<td>1(^1)</td>
<td>1(^1)</td>
<td>1(^1)</td>
</tr>
<tr>
<td>CM579-ETHCAT - EtherCAT master</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>CM579-PNIO - PROFINET IO RT controller</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>CM589-PNIO - PROFINET IO RT device</td>
<td>1(^1)</td>
<td>1(^1)</td>
<td>1(^1)</td>
<td>1(^1)</td>
</tr>
<tr>
<td>CM589-PNIO-4 - PROFINET IO RT with 4 devices</td>
<td>1(^1)</td>
<td>1(^1)</td>
<td>1(^1)</td>
<td>1(^1)</td>
</tr>
<tr>
<td>CM597-ETH - Ethernet interface</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>CM588-CN - CAN, CANopen slave</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>CM598-CN - CAN, CANopen master</td>
<td>only CAN 2A/2B</td>
<td>only CAN 2A/2B</td>
<td>only CAN 2A/2B</td>
<td>only CAN 2A/2B</td>
</tr>
<tr>
<td>Type of AC500-S module supported</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SM560-S - safety module</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>SM560-S-FD-1 - safety module with F-Device functionality for 1 PROFIsafe network</td>
<td>1(^1)</td>
<td>1(^1)</td>
<td>1(^1)</td>
<td>1(^1)</td>
</tr>
<tr>
<td>SM560-S -FD-4 - safety module with F-Device functionality for 1 PROFIsafe network</td>
<td>1(^1)</td>
<td>1(^1)</td>
<td>1(^1)</td>
<td>1(^1)</td>
</tr>
</tbody>
</table>

Remarks:
1\(^1\) in preparation
<table>
<thead>
<tr>
<th>Processor module</th>
<th>PM5630</th>
<th>PM5650</th>
<th>PM5670</th>
<th>PM5675</th>
</tr>
</thead>
<tbody>
<tr>
<td>Program and data memory (FLASH and DDR3 SDRAM)</td>
<td>8 MB</td>
<td>80 MB</td>
<td>160 MB</td>
<td>160 MB</td>
</tr>
<tr>
<td>Thereof user program and data (dynamically allocated)</td>
<td>1 MB</td>
<td>4 MB</td>
<td>16 MB</td>
<td>16 MB</td>
</tr>
<tr>
<td>Thereof user webserver data</td>
<td>7 MB</td>
<td>76 MB</td>
<td>144 MB</td>
<td>144 MB</td>
</tr>
<tr>
<td>Number of webvisu instances in a project limitation (web pages)</td>
<td>8</td>
<td>32</td>
<td>64</td>
<td>64</td>
</tr>
<tr>
<td>Remaining for all other usage (project save, infrastructure...)</td>
<td>30 MB</td>
<td>285 MB</td>
<td>643 MB</td>
<td>643 MB</td>
</tr>
<tr>
<td>Buffered (SRAM)</td>
<td>256 kB</td>
<td>256 kB</td>
<td>1.5 MB</td>
<td>1.5 MB</td>
</tr>
<tr>
<td>Thereof VAR retain persistent</td>
<td>128 kB</td>
<td>128 kB</td>
<td>1024 kB</td>
<td>1024 kB</td>
</tr>
<tr>
<td>Thereof %M memory (e.g. Modbus register)</td>
<td>128 kB</td>
<td>128 kB</td>
<td>512 kB</td>
<td>512 kB</td>
</tr>
<tr>
<td>Expandable memory</td>
<td>None</td>
<td>None</td>
<td>None</td>
<td>None</td>
</tr>
<tr>
<td>Integrated mass storage memory (FLASH)</td>
<td>None</td>
<td>None</td>
<td>None</td>
<td>8 GB</td>
</tr>
<tr>
<td>Slot for pluggable memory card</td>
<td>MC502</td>
<td>MC502</td>
<td>MC502</td>
<td>MC502</td>
</tr>
<tr>
<td>Processor speed</td>
<td>300 MHz</td>
<td>600 MHz</td>
<td>1 GHz</td>
<td>1 GHz</td>
</tr>
<tr>
<td>Cycle time for 1 instruction (minimum):</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Binary</td>
<td>Min. 0.02 µs</td>
<td>Min. 0.01 µs</td>
<td>Min. 0.002 µs</td>
<td>Min. 0.002 µs</td>
</tr>
<tr>
<td>Word</td>
<td>Min. 0.02 µs</td>
<td>Min. 0.01 µs</td>
<td>Min. 0.002 µs</td>
<td>Min. 0.002 µs</td>
</tr>
<tr>
<td>Floating point</td>
<td>Min. 0.12 µs</td>
<td>Min. 0.01 µs</td>
<td>Min. 0.002 µs</td>
<td>Min. 0.002 µs</td>
</tr>
<tr>
<td>Mathematic co-processor</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>Motion capability</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>No. synchronized axis per 1 ms on EtherCAT CM typically</td>
<td>-</td>
<td>4</td>
<td>16</td>
<td>16</td>
</tr>
<tr>
<td>No. synchronized axis per 2 ms on EtherCAT CM typically</td>
<td>-</td>
<td>8</td>
<td>32</td>
<td>32</td>
</tr>
<tr>
<td>No. synchronized axis per 4 ms on EtherCAT CM or CANopen onboard typically</td>
<td>2</td>
<td>8</td>
<td>&gt;32</td>
<td>&gt;32</td>
</tr>
<tr>
<td>Min. bus cycle time for EtherCAT using external CM579</td>
<td>2 ms</td>
<td>1 ms</td>
<td>0.5 ms</td>
<td>0.5 ms</td>
</tr>
<tr>
<td>Max. number of central inputs and outputs (10 exp. modules):</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Digital inputs</td>
<td>320</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Digital outputs</td>
<td>320</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Analog inputs</td>
<td>160</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Analog outputs</td>
<td>160</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Number of decentralized inputs and outputs</td>
<td>Depends on the used fieldbus</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Data backup</td>
<td>Battery</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Data buffering time at 25 °C</td>
<td>Typ. 3 years</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Battery low indication</td>
<td>via application program</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Processor module</td>
<td>PM5630</td>
<td>PM5650</td>
<td>PM5670</td>
<td>PM5675</td>
</tr>
<tr>
<td>------------------</td>
<td>--------</td>
<td>--------</td>
<td>--------</td>
<td>--------</td>
</tr>
<tr>
<td><strong>Real-time clock:</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>With battery back-up</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Accuracy</td>
<td>Typ. ±2 s / day at 25 °C</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Program execution:</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cyclic</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Time-controlled</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Multitasking</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Minimum cycle time configurable for cyclical task</td>
<td>1 ms</td>
<td>1 ms</td>
<td>0,5 ms</td>
<td>0,5 ms</td>
</tr>
<tr>
<td><strong>User program protection by password</strong></td>
<td>x (user management)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Internal interfaces for communication:</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Serial interface COM1:</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Physical link</td>
<td>Configurable for RS-232 or RS-485 (9.6 kb/s, 19.2 kb/s, 38.4 kb/s, 57.6 kb/s and 115.2 kb/s)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Connection</td>
<td>Pluggable terminal block, spring connection</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Usage</td>
<td>Serial ASCII communication, Modbus RTU</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>CAN interface:</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Physical link</td>
<td>CAN 2A/2B (from 50 kb/s to 1 Mb/s)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Connection</td>
<td>Pluggable terminal block, spring connection</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Usage</td>
<td>CANopen master communication, CAN 2A/2B, J1939 protocol, CAN sync</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Max. number of variables allowed</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Input variables</td>
<td>2 kB</td>
<td>4 kB</td>
<td>5 kB</td>
<td>5 kB</td>
</tr>
<tr>
<td>Output variables</td>
<td>2 kB</td>
<td>4 kB</td>
<td>5 kB</td>
<td>5 kB</td>
</tr>
<tr>
<td><strong>Network interface ETH1, ETH2:</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Usage</td>
<td>Ethernet</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Physical link</td>
<td>10/100 base-TX, configurable as internal switch or independent Interfaces</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Connection</td>
<td>2x RJ45 socket, provided on TB56xx-2ETH</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LEDs, LCD display, function keys</td>
<td>RUN / STOP, status, diagnosis, settings</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Number of timers</td>
<td>Unlimited</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Number of counters</td>
<td>Unlimited</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Programming languages:</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Structured text ST</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Instruction list IL</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Function block diagram FBD</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ladder diagram LD</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sequential function chart SFC</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Continuous function chart (CFC)</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

1.5.1.2 Comparison of features and protocols
### Table 503: OPC UA server / OPC DA server

<table>
<thead>
<tr>
<th>Processor module</th>
<th>PM5630</th>
<th>PM5650</th>
<th>PM5670</th>
<th>PM5675</th>
</tr>
</thead>
<tbody>
<tr>
<td>OPC UA server</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>Number of free tags + additional license for extension</td>
<td>1.000</td>
<td>5.000</td>
<td>30.000</td>
<td>30.000</td>
</tr>
<tr>
<td>Number of connections</td>
<td>10</td>
<td>20</td>
<td>50</td>
<td>50</td>
</tr>
<tr>
<td>Min. sampling rate (limit)</td>
<td>500 ms</td>
<td>100 ms</td>
<td>50 ms</td>
<td>50 ms</td>
</tr>
<tr>
<td>OPC DA server AE</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>Number of connections</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>8</td>
</tr>
</tbody>
</table>

Remarks:
1) in preparation

### Table 504: Modbus, Telecontrol

<table>
<thead>
<tr>
<th>Processor module</th>
<th>PM5630</th>
<th>PM5650</th>
<th>PM5670</th>
<th>PM5675</th>
</tr>
</thead>
<tbody>
<tr>
<td>Modbus TCP client / server</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>Number of Modbus clients ModMast in parallel on a CPU master (server)</td>
<td>30</td>
<td>50</td>
<td>120</td>
<td>120</td>
</tr>
<tr>
<td>Number of Modbus server in parallel (e.g. for SCADA access)</td>
<td>15</td>
<td>25</td>
<td>50</td>
<td>50</td>
</tr>
<tr>
<td>IEC 60870-5-104 telecontrol protocol</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>Number of free tags + additional license for extension</td>
<td>1.000</td>
<td>5.000</td>
<td>10.000</td>
<td>10.000</td>
</tr>
<tr>
<td>Control station (number of connections)</td>
<td>5</td>
<td>10</td>
<td>20</td>
<td>20</td>
</tr>
<tr>
<td>Sub-station (number of connections)</td>
<td>5</td>
<td>10</td>
<td>20</td>
<td>20</td>
</tr>
</tbody>
</table>

Remarks:
1) in preparation

### 1.5.1.3 Ethernet protocols and ports for AC500 V3 products

#### Supported as of Automation Builder V 2.1

<table>
<thead>
<tr>
<th>Description</th>
<th>PM5630-2ETH</th>
<th>PM5650-2ETH</th>
<th>PM5670-2ETH</th>
<th>PM5675-2ETH</th>
<th>≥ CPU firmware</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABB netConfig</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>V3.0.0</td>
</tr>
<tr>
<td>Online access with driver 3S UDP BlkDrvUdp</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>V3.0.0</td>
</tr>
<tr>
<td>Online access with driver 3S TCP/IP BlkDrvTcp</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>V3.0.0</td>
</tr>
<tr>
<td>Modbus TCP server</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>V3.0.3</td>
</tr>
<tr>
<td>Modbus TCP client with POU ETHx_MOD_MAST</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>V3.0.1</td>
</tr>
<tr>
<td>UDP out of user program with library netBaseService.lib</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>V3.0.0</td>
</tr>
<tr>
<td>UDP data exchange, Network variables</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>V3.0.0</td>
</tr>
<tr>
<td>Description</td>
<td>PM5630 -2ETH</td>
<td>PM5650 -2ETH</td>
<td>PM5670 -2ETH</td>
<td>PM5675 -2ETH</td>
<td>≥ CPU firmware</td>
</tr>
<tr>
<td>-----------------------------------------------------------------------------</td>
<td>---------------</td>
<td>---------------</td>
<td>---------------</td>
<td>---------------</td>
<td>----------------</td>
</tr>
<tr>
<td>TCP/IP out of user program with library net-BaseService.lib.</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>V3.0.0</td>
</tr>
<tr>
<td>Web server on PLC with web visualization</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>V3.0.0</td>
</tr>
<tr>
<td>SNTP (Simple Network Time Protocol) client with 3S licenced store package</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>V3.0.0</td>
</tr>
<tr>
<td>SNTPService, package. Library container: SNTPService</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>IEC60870-5-104 control station incl. 2nd connection and 2nd port</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>V3.0.0</td>
</tr>
<tr>
<td>IEC60870-5-104 substation incl. 2nd port</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>V3.0.0</td>
</tr>
<tr>
<td>FTP server</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>V3.0.0</td>
</tr>
<tr>
<td>(see FTP server configuration Chapter 1.5.6.3.4.1 “Configuration of FTP server” on page 3428)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CODESYS network variables</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>V3.0.0</td>
</tr>
<tr>
<td>OPC DA server</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>V3.0.0</td>
</tr>
<tr>
<td>OPC UA server</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>V3.0.0</td>
</tr>
<tr>
<td>ICMP – ping out of user project with POU ETHx_ICMP_PING</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>V3.0.0</td>
</tr>
<tr>
<td>DHCP client</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>V3.1.0</td>
</tr>
<tr>
<td>SNTP (Simple Network Time Protocol) client system solution (see SNTP client configuration Chapter 1.5.6.3.3.2.1 “SNTP client configuration” on page 3425)</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>V3.1.0</td>
</tr>
<tr>
<td>SNTP (Simple Network Time Protocol) server system solution (see SNTP server configuration Chapter 1.5.6.3.3.2.2 “SNTP server configuration” on page 3427)</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>V3.1.0</td>
</tr>
<tr>
<td>Maximum number of Input/output allowed variable on Ethernet for the protocol</td>
<td>2 kB /2 kB</td>
<td>4 kB /4 kB</td>
<td>5 kB /5 kB</td>
<td>5 kB /5 kB</td>
<td>V3.4.0</td>
</tr>
<tr>
<td>IEC 61850 (MMS server, GOOSE)</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>V3.1.0</td>
</tr>
<tr>
<td>EthernetIP Adapter 1,2)</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>AB 2.4.1/FW 3.4.1</td>
</tr>
<tr>
<td>KNX - Building communication 2)</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>V3.2.x</td>
</tr>
<tr>
<td>BACnet-BC - Infrastructure communication 2)</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>V3.3.1</td>
</tr>
<tr>
<td>HTTPS – secure web server on PLC with CODESYS web visualization (see secure web server Chapter 1.5.6.3.6.3.2 “Secure web server” on page 3433)</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>V3.1.0</td>
</tr>
<tr>
<td>WebVisu for data visualisation on web-server HTML5</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>V3.0.0</td>
</tr>
</tbody>
</table>
### Default open Ethernet ports of PM56xx-2ETH

After startup without a PLC project the PM56xx-2ETH contains the following Ethernet ports and sockets:

<table>
<thead>
<tr>
<th>Protocol</th>
<th>Port</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABB NetConfig 1)</td>
<td>UDP 24576</td>
</tr>
<tr>
<td>Online access with driver 3S UDP BlkDrvUdp (with scan)</td>
<td>UDP 1740</td>
</tr>
<tr>
<td>Online access with driver 3S Tcp/Ip BlkDrvTcp (no scan)</td>
<td>TCP 11740</td>
</tr>
<tr>
<td>OPC UA server 2)</td>
<td>TCP 4840</td>
</tr>
</tbody>
</table>

Remarks:

1) The port 24576 for ABB NetConfig protocol can be disabled via PLC configuration by deleting the protocol node from configuration tree of Ethernet interfaces ETH1 and ETH2.

2) The port 4840 for OPC UA server is closed by default as of SystemFW V3.1.0.

All other ports are closed by default.

### Overview of protocols, sockets and ports

<table>
<thead>
<tr>
<th>Protocol</th>
<th>Port</th>
<th>Sockets</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABB netConfig</td>
<td>24576</td>
<td>1 permanent socket per interface</td>
</tr>
<tr>
<td>3S gateway client (e.g. CODESYS) to gateway server</td>
<td>1217</td>
<td>1 permanent socket</td>
</tr>
<tr>
<td>Online access with driver 3S UDP BlkDrvUdp (with scan)</td>
<td>1740</td>
<td>1 socket per connection + 4 listen</td>
</tr>
<tr>
<td>Protocol</td>
<td>Port</td>
<td>Sockets</td>
</tr>
<tr>
<td>-------------------------------------------------------------------------</td>
<td>---------------</td>
<td>-------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Online access with driver 3S block driver TCP/IP (no scan)</td>
<td>11740</td>
<td>1 socket per connection + 1 listen</td>
</tr>
<tr>
<td>Modbus TCP server</td>
<td>502 or configurable</td>
<td>1 socket listen + 1 socket per server connection, number of server connections is configurable in AB</td>
</tr>
<tr>
<td>Modbus TCP client with POU ETHx_MOD_MAST</td>
<td>Random</td>
<td>1 socket per connection with POU ETHx_MOD_MAST</td>
</tr>
<tr>
<td>UDP out of user program with library SysLibSockets.lib</td>
<td>1 ... 65535</td>
<td>1 socket per connection</td>
</tr>
<tr>
<td>TCP/IP out of user program with library SysLibSockets.lib</td>
<td>1 ... 65535</td>
<td>1 socket per connection</td>
</tr>
<tr>
<td>Web server on PLC with web visualization</td>
<td>80</td>
<td>1 listen and 1 per connection</td>
</tr>
<tr>
<td>SNTP client</td>
<td>123</td>
<td>1 permanent socket</td>
</tr>
<tr>
<td>IEC60870-5-104 control station</td>
<td>Random</td>
<td>1 per connection</td>
</tr>
<tr>
<td>IEC60870-5-104 substation</td>
<td>2404</td>
<td>1 per connection</td>
</tr>
<tr>
<td>FTP server (see FTP server configuration Table 1.5.6.3.4.1 “Configuration of FTP server” on page 3428)</td>
<td>Command port = 21; Data active mode = 20; Data passive mode = random</td>
<td>1 per session, max. 4 allowed</td>
</tr>
<tr>
<td>CODESYS network variables</td>
<td>1202</td>
<td>(UDP broadcast)</td>
</tr>
<tr>
<td>OPC DA server (default 3S block driver)</td>
<td>UDP = 1740 or TCP/IP = 11740</td>
<td>1 socket per connection</td>
</tr>
<tr>
<td>OPC UA server</td>
<td>4840</td>
<td>1 permanent socket</td>
</tr>
<tr>
<td>ICMP – ping out of user project with POU ETHx_ICMP_PING DHCP</td>
<td>none</td>
<td>No socket</td>
</tr>
<tr>
<td>DHCP</td>
<td>67</td>
<td>1 socket during startup</td>
</tr>
<tr>
<td>SNTP (Simple Network Time Protocol) client system solution (see SNTP client configuration Table 1.5.6.3.3.2.1 “SNTP client configuration” on page 3425)</td>
<td>123</td>
<td>1 permanent socket</td>
</tr>
<tr>
<td>SNTP (Simple Network Time Protocol) server system solution (see SNTP server configuration Table 1.5.6.3.3.2.2 “SNTP server configuration” on page 3427)</td>
<td>123</td>
<td>1 permanent socket</td>
</tr>
<tr>
<td>HTTPS – secure web server on PLC with CODESYS web visualization (see secure web server Table 1.5.6.3.6.3.2 “Secure web server” on page 3433)</td>
<td>443</td>
<td>1 listen and 1 per connection</td>
</tr>
</tbody>
</table>
### Protocol

<table>
<thead>
<tr>
<th>Protocol</th>
<th>Port</th>
<th>Sockets</th>
</tr>
</thead>
<tbody>
<tr>
<td>FTPS – secure FTP (see <a href="#">secure FTP</a> Chapter 1.5.6.3.6.3.3 &quot;Secure FTP” on page 3434)</td>
<td></td>
<td>1 per session, max. 4 allowed</td>
</tr>
<tr>
<td>Secure online access with driver 3S UDP BlkDrvUdp</td>
<td>1740</td>
<td>1 socket per connection + 1 listen</td>
</tr>
<tr>
<td>Secure online access with driver 3S TCP/IP BlkDrvTcp</td>
<td>11740</td>
<td>1 socket per connection + 1 listen</td>
</tr>
<tr>
<td>ICMP – ping out of user project with POU ETHx_ICMP_PING or EthICmpPing (PLCopen style)</td>
<td>None</td>
<td>No socket</td>
</tr>
<tr>
<td>Modbus TCP client (master) with POU ETHx.MOD_MAST or ModTcpMast (PLCopen style)</td>
<td>Random</td>
<td>1 socket per connection with POU ETHx.MOD_MAST or ModTcpMast</td>
</tr>
</tbody>
</table>

### 1.5.1.3.3 Limitation of connections per protocol

<table>
<thead>
<tr>
<th>Protocol</th>
<th>PM5630-2ETH</th>
<th>PM5650-2ETH</th>
<th>PM5670-2ETH</th>
<th>PM5675-2ETH</th>
<th>≥ CPU firmware</th>
</tr>
</thead>
<tbody>
<tr>
<td>Modbus TCP server (e.g. for SCADA access)</td>
<td>30</td>
<td>40</td>
<td>15</td>
<td>10</td>
<td>3.0.3</td>
</tr>
<tr>
<td>Modbus TCP client with POU ETHx.MOD_MAST</td>
<td>40</td>
<td>40</td>
<td>40</td>
<td>40</td>
<td>3.0.1</td>
</tr>
<tr>
<td>Modbus TCP client with POU ETHx.MOD_MAST or ModTcpMast (PLCopen style)</td>
<td>30</td>
<td>30</td>
<td>30</td>
<td>30</td>
<td>3.1.0</td>
</tr>
<tr>
<td>IEC60870-5-104 control station incl. 2nd connection and 2nd port</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>3.1.0</td>
</tr>
<tr>
<td>IEC60870-5-104 substation incl. 2nd port</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>3.1.0</td>
</tr>
<tr>
<td>IEC60870-5-104: No. of free tags + additional license for extension ¹)</td>
<td>1.000</td>
<td>5.000</td>
<td>10.000</td>
<td>10.000</td>
<td>3.4.0</td>
</tr>
<tr>
<td>FTP server</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>3.1.0</td>
</tr>
<tr>
<td>Online access with driver 3S UDP BlkDrvUdp</td>
<td>n/a</td>
<td>4</td>
<td>8</td>
<td>8</td>
<td>3.0.0</td>
</tr>
<tr>
<td>Online access with driver 3S TCP/IP BlkDrvTcp</td>
<td>n/a</td>
<td>4</td>
<td>8</td>
<td>8</td>
<td>3.0.0</td>
</tr>
<tr>
<td>OPC DA server (number of connections)</td>
<td>n/a</td>
<td>4</td>
<td>8</td>
<td>8</td>
<td>3.0.0</td>
</tr>
</tbody>
</table>
### Protocol

<table>
<thead>
<tr>
<th>Protocol</th>
<th>PM5630-2ETH</th>
<th>PM5650-2ETH</th>
<th>PM5670-2ETH</th>
<th>PM5675-2ETH</th>
</tr>
</thead>
<tbody>
<tr>
<td>OPC UA server (number of connections)</td>
<td>50</td>
<td>50</td>
<td>50</td>
<td>50</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>20</td>
<td>50</td>
<td>50</td>
</tr>
<tr>
<td></td>
<td>50</td>
<td>50</td>
<td>50</td>
<td>3.1.0</td>
</tr>
<tr>
<td></td>
<td>50</td>
<td>20</td>
<td>50</td>
<td>3.4.0</td>
</tr>
<tr>
<td>No. of free tags + additional license for extension ¹</td>
<td>1.000</td>
<td>5.000</td>
<td>30.000</td>
<td>30.000</td>
</tr>
<tr>
<td>min sampling rate (limit)</td>
<td>500 ms</td>
<td>100 ms</td>
<td>50 ms</td>
<td>50</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>3.4.0</td>
</tr>
<tr>
<td>Number of Webvisu instances in a project limitation</td>
<td>16</td>
<td>32</td>
<td>64</td>
<td>64</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2.4.0/ FW 3.4.0</td>
</tr>
<tr>
<td>Secure online access with driver 3S UDP BlkDrvUdp</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>3.1.0</td>
</tr>
<tr>
<td>Secure online access with driver 3S TCP/IP BlkDrvTcp</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td>FTPS - secure FTP server</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>RTV (Remote Target Visualization)</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>Remarks:</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>¹): in preparation</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The PLC types PM5630-2ETH, PM5670-2ETH and PM5675-2ETH are available as of SystemFW 3.1.0.

#### 1.5.1.3.4 Ethernet configuration

**Default Ethernet configuration**

<table>
<thead>
<tr>
<th>Module</th>
<th>IP Address</th>
<th>Netmask</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>PM56xx-2ETH</td>
<td>ETH1: 192.168.0.10</td>
<td>255.255.255.0</td>
<td>The Ethernet ports must be configured in different sub networks.</td>
</tr>
<tr>
<td></td>
<td>ETH2: 192.168.1.10</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

For changing the default addresses, see *IP-Configuration tool* Chapter 1.5.6.2.2.4.2 “Configuration of the IP settings with IP configuration tool” on page 3267, or description of function key *®® Chapter 1.5.5.1.4.4 “Description of the function keys” on page 3121.*
1.5.1.3.5 Online access

Preferred driver for online access: 3S UDP block driver BlkDrvUdp. This driver allows to scan and select the connected PLC’s.

Alternative: 3S TCP/IP block driver. This driver requires at least 2 sockets:

- 1x driver “BlkDrvTcp” on port 11740
- 1x listen on port 11740 if PLC has established online connection

Online access can be established from:

- Automation Builder command ‘Login’
- CODESYS OPC DA server
- Panel CP600 series

Each established connection needs one socket. In addition one socket on port 11740 is listening.

1. Startup the PLC.
   ➞ One socket on port 11740 (listen).

2. Login from Automation Builder via driver “BlkDrvTcp”.
   ➞ 2 sockets on port 11740 (1x online, 1x listen)

3. Additional login out of OPC server with the same driver.
   ➞ 3 sockets on port 11740 (2x online, 1x listen)

4. Additional connect CP600 via driver “BlkDrvTcp”.
   ➞ 4 sockets on port 11740 (3x online, 1x listen)

1.5.2 PLC introduction

1.5.2.1 Safety instructions

The examples and diagrams in this manual are included solely for illustrative purposes. Because of the many variants and requirements associated with any particular installation, ABB cannot assume responsibility or liability for actual use based on the examples and diagrams.

No patent liability is assumed by ABB with respect to use of information, circuits, equipment or software described in this manual. No liability is assumed for the direct or indirect consequences of the improper use, improper application or inadequate maintenance of these devices. In no event will ABB be responsible or liable for indirect or consequential damages resulting from the use or application of this equipment.

The product family AC500 control system is designed according to EN 61131-2 IEC 61131-2 standards. Data, different from IEC 61131, are caused by the higher requirements of Maritime Services. Other differences are described in the technical data description of the devices.
NOTICE!

Avoidance of electrostatic charging
PLC devices and equipment is sensitive to electrostatic discharge, which can cause internal damage and affect normal operation. Observe the following rules when handling the system:
- Touch a grounded object to discharge potential static.
- Wear an approved grounding wrist strap.
- Do not touch connectors or pins on component boards.
- Do not touch circuit components inside the equipment.
- If available, use a static-safe workstation.
- When not in use, store the equipment in appropriate static-safe packaging.

NOTICE!

PLC damage due to operation conditions
Protect the devices from dampness, dirt and damage during transport, storage and operation!

NOTICE!

PLC damage due to wrong enclosures
Due to their construction (degree of protection IP 20 according to EN 60529) and their connection technology, the devices are suitable only for operation in enclosed switchgear cabinets.

Cleaning instruction
Do not use cleaning agent for cleaning the device.
Use a damp cloth instead.

Connection plans and user software must be created so that all technical safety aspects, legal regulations and standards are observed. In practice, possible shortcircuits and breakages must not be able to lead to dangerous situations. The extent of resulting errors must be kept to a minimum.

Do not operate devices outside of the specified, technical data!
Trouble-free functioning cannot be guaranteed outside of the specified data.

NOTICE!

PLC damage due to missing earthing
- Ensure to earth the devices.
- The earthing (switch cabinet earthing, PE) is supplied both by the mains connection (or 24 V supply voltage) and via DIN rail. The DIN rail must be connected to the earth before the device is subjected to any power. The earthing may be removed only if it is certain that no more power is being supplied to the control system.
In the description for the devices (operating manual or AC500 system description), reference is made at several points to earthing, electrical isolation and EMC measures. One of the EMC measures consists of discharging interference voltages into the earthing via Y-type capacitors. Capacitor discharge currents must basically be able to flow off to the earthing (in this respect, see also VBG 4 and the relevant VDE regulations).

**CAUTION!**

Do not obstruct the ventilation for cooling!
The ventilation slots on the upper and lower side of the devices must not be covered.

**CAUTION!**

Run signal and power wiring separately!
Signal and supply lines (power cables) must be laid out so that no malfunctions due to capacitive and inductive interference can occur (EMC).

**WARNING!**

Labels on or inside the device alert people that dangerous voltage may be present or that surfaces may have dangerous temperatures.

**WARNING!**

Splaying of strands can cause hazards!
During wiring of terminals with stranded conductors, splaying of strands shall be avoided.
- Ferrules can be used to prevent splaying.

**WARNING!**

Removal/Insertion under power
The devices are not designed for removal or insertion under power. Because of unforeseeable consequences, it is not allowed to plug or unplug devices with the power being ON.

Make sure that all voltage sources (supply and process voltage) are switched off before you
- connect or disconnect any signal or terminal block
- remove, mount or replace a module.

Disconnecting any powered devices while energized in a hazardous location could result in an electric arc, which could create a flammable ignition resulting in fire or explosion.

Make sure that power is removed and that the area has been thoroughly checked to ensure that flammable materials are not present prior to proceeding.

The devices must not be opened when in operation. The same applies to the network interfaces.
Information on batteries

CAUTION!
Use only ABB approved lithium battery modules!
At the end of the battery’s lifetime, always replace it only with a genuine battery module.

CAUTION!
Risk of explosion!
Do not open, re-charge or disassemble a lithium battery. Attempts to charge lithium batteries lead to overheating and possible explosions.

Prevent them from heat and fire and store them in a dry place.

Never short-circuit or operate lithium batteries with the polarities reversed. The batteries are likely to overheat and explode. Avoid chance short circuiting and therefore do not store batteries in metal containers and do not place them on metallic surfaces. Escaping lithium is a health hazard.

Environment considerations
Recycle exhausted batteries. Dispose batteries in an environmentally conscious manner, in accordance to local-authority regulations.

Environment and enclosure information

This equipment is intended for use in a Pollution Degree 2 industrial environment, in overvoltage Category II applications (as defined in IEC publication 60664-1), at altitudes up to 2,000 meters without derating.

This equipment is considered Group 1, Class A industrial equipment according to IEC/CISPR Publication 11. Without appropriate precautions, there may be potential difficulties ensuring electromagnetic compatibility in other environments due to conducted as well as radiated disturbance.

This equipment is supplied as "open type" equipment. It must be mounted within an enclosure that is suitably designed for those specific environmental conditions that will be present and appropriately designed to prevent personal injury resulting from accessibility to live parts. The interior of the enclosure must be accessible only by the use of a tool. Subsequent sections of this publication may contain additional information regarding specific enclosure type ratings that are required to comply with certain product safety certifications.

Refer to NEMA Standards publication 250 and IEC publication 60529, as applicable, for explanations of the degrees of protection provided by different types of enclosure. Also see the appropriate sections in this manual.

1.5.2.2 Cyber security in AC500 V3 products

Cyber security disclaimer
This product is designed to be connected to and to communicate information and data via a network interface. It is your sole responsibility to provide and continuously ensure a secure connection between the product and your network or any other network (as the case may be). You shall establish and maintain any appropriate measures (such as but not limited to the installation of firewalls, application of authentication measures, encryption of data, installation of anti-virus programs, etc.) to protect the product, the network, its system and the interface against any kind of security breaches, unauthorized access, interference, intrusion, leakage and/or theft of data or information. ABB Ltd and its affiliates are not liable for damages and/or losses related to such security breaches, any unauthorized access, interference, intrusion, leakage and/or theft of data or information.
Security details for industrial automation is provided in a whitepaper on ABB website.

Signed firmware updates

The firmware update files for the AC500 V3 PLC are digitally signed releases by ABB. During the update process, these signatures are validated by a hardware security component in the PLC. This way, the AC500 V3 PLC will only update with valid, authentic firmware, signed by ABB.

Open ports and services

The AC500 V3 PLC comes with minimal services opened by default. Only the services needed for initial setup and programming are open before any user application is downloaded. Details Chapter 1.5.1.3 “Ethernet protocols and ports for AC500 V3 products” on page 2224.

Secure shell access for ABB service

The AC500 V3 PLC contains a secure shell service to access core logging data in case of problems which need a deeper analysis. This service is inactive by default, which means that no one can access this privileged shell in the normal operating state.

To activate this service, local access to the PLC is necessary and activation is only valid until the next power cycle of the PLC. Once activated, the service run on TCP port 22. Each PLC also protects the secure shell access by an individual password.

1.5.2.1 Defense in depth

The defense in depth approach implements multi-layer IT security measures. Each layer provides its special security measures. All deployed security mechanisms in the system must be updated regularly. It is also important to follow the system vendor’s recommendations on how to configure and use these mechanisms. As a basis, the components must include security functions such as:

- Virus protection
- Firewall protection
- Strong and regularly changed passwords
- User management
- Using VPN tunnels for connections between networks

Additional security components such as routers and switches with integrated firewalls should be available. A defined user and rights concept managing access to the controllers and their networks is mandatory. Finally, the manufacturer of the components should be able to quickly discover weaknesses and provide patches.

References: CODESYS Security Whitepaper

Security zones

IT resources vary in the extent to which they can be trusted. A common security architecture is therefore based on a layered approach that uses zones of trust to provide increasing levels of security according to increasing security needs. Less-trusted zones contain more-trusted zones and connections between the zones are only possible through secure interconnections such as firewalls. All resources in the same zone must have the same minimum level of trust. The inner layers, where communication interaction needs to flow freely between nodes, must have the highest level of trust. This is the approach described in the IEC 62443 series of standards.

Firewalls, gateways, and proxies are used to control network traffic between zones of different security levels, and to filter out any undesirable or dangerous material. Traffic that is allowed to pass between zones should be limited to what is absolutely necessary because each type of service call or information exchange translates into a possible route that an intruder may be able to exploit. Different types of services represent different risks. Internet access, incoming e-mail and instant messaging, for example, represent very high risks.
Fig. 90: Security zones

Fig. 1 shows three security zones, but the number of zones does not have to be as many as or as few as three. The use of multiple zones allows access between zones of different trust levels to be controlled to protect a trusted resource from attack by a less trusted one.

High-security zones should be kept small and independent. They need to be physically protected, i.e. physical access to computers, network equipment and network cables must be limited by physical means to authorized persons only. A high-security zone should obviously not depend on resources in a less secure zone for its security. Therefore, it should form its own domain that is administered from the inside, and not depend on, e.g., a domain controller in a less secure network.

Even if a network zone is regarded as trusted, an attack is still possible: by a user or compromised resource that is inside the trusted zone, or by an outside user or resource that succeeds to penetrate the secure interconnection. Trust therefore depends also upon the types of measures taken to detect and prevent compromise of resources and violation of the security policy.

References: Security for Industrial Automation and Control Systems

1.5.2.2 Secure operation

The controller must be located in a protected environment in order to avoid accidental or intended access to the controller or the application.

A protected environment can be:

- Locked control cabinets without connection from outside
- No direct internet connection
- Use firewalls and VPN to separate different networks
- Separate different production areas with different access controls

To increase security, physical access protection measures such as fences, turnstiles, cameras or card readers can be added.

Follow these rules for the protected environment:

- Keep the trusted network as small as possible and independent from other networks.
- Protect the cross-communication of controllers and the communication between controllers and field devices via standard communication protocols (fieldbus systems) using appropriate measures.
- Protect such networks from unauthorized physical access.
- Use fieldbus systems only in protected environments. They are not protected by additional measures, such as encryption. Open physical or data access to fieldbus systems and their components is a serious security risk.
Physically protect all equipment, i.e., ensure that physical access to computers, network equipment and cables, controllers, I/O systems, power supplies, etc., is limited to authorized persons.

When connecting a trusted network zone to outer networks, make sure that all connections are through properly configured secure interconnections only, such as a firewall or a system of firewalls, which is configured for “deny by default”, i.e., blocks everything except traffic that is explicitly needed to fulfill operational requirements.

Allow only authorized users to log on to the system, and enforce strong passwords that are changed regularly.

continuously maintain the definitions of authorized users, user groups, and access rights, to properly reflect the current authorities and responsibilities of all individuals at all times. Users should not have more privileges than they need to do their job.

Do not use the system for e-mail, instant messaging, or internet browsing. Use separate computers and networks for these functions if they are needed.

Do not allow installation of any unauthorized software in the system.

Restrict temporary connection of portable computers, USB memory sticks and other removable data carriers. Computers that can be physically accessed by regular users should have ports for removable data carriers disabled.

If portable computers need to be connected, e.g., for service or maintenance purposes, they should be carefully scanned for viruses immediately before connection.

All CDs, DVDs, USB memory sticks and other removable data carriers, and files with software or software updates, should also be checked for viruses before being introduced into the trusted zone.

continuously monitor the system for intrusion attempts.

Define and maintain plans for incident response, including how to recover from potential disasters.

Regularly review the organization as well as technical systems and installations with respect to compliance with security policies, procedures and practices.

A protected local control cabinet could look like in picture 2, page 17. This network is not connected to any external network. Security is primarily a matter of physically protecting the automation system and preventing unauthorized users from accessing the system and from connecting or installing unauthorized hardware and software.

---

**Fig. 91: Isolated automation system**

Servers and workplaces that are not directly involved in the control and supervision of the process should preferably be connected to a subnet that is separated from the automation system network by means of a router/firewall. This makes it possible to better control the network load and to limit access to certain servers on the automation system network. Note that servers and workplaces on this subnet are part of the trusted zone and thus need to be subject to the same security precautions as the nodes on the automation system network.
For the purposes of process control security, a general-purpose information system (IS) network should not be considered a trusted network, not the least since such networks are normally further connected to the Internet or other external networks. The IS network is therefore a different lower-security zone, and it should be separated from the automation system by means of a firewall. The IS and automation system networks should form separate domains.
1.5.2.3 License and third party information

Information on Automation Builder licensing and Third Party software can be found in the "About" window of the Automation Builder Installation Manager.

See further information on PLC runtime licensing Chapter 1.5.6.2.2.2 “PLC runtime licensing” on page 3258.

1.5.2.4 Regulations

Appropriate system setup

The following regulations have to be taken into due consideration:
● DIN VDE 0100: "Regulations for the Setting up of Power Installations"
● DIN VDE 0110 Part 1 and Part 2: "The Rating of Creepage Distances and Clearances"
● DIN VDE 0160 and DIN VDE 0660 Part 500: "The Equipment of Power Installations with Electrical Components"

To ensure project success and proper installation of all systems, customers must be familiar and proficient with the following standards and must comply with their directives:

● DIN VDE 0113 Part 1 & Part 200: "Working & Process Machinery"
● DIN VDE 0106 Part 100: "Close proximity to dangerous voltages"
● DIN VDE 0160, DIN VDE 0110 Part 1: "Protection against direct contact"

The user has to guarantee that the devices and the components are mounted following these regulations. For operating the machines and installations, other national and international relevant regulations, concerning prevention of accidents and using technical working means, also have to be met.

AC500 devices are designed according to IEC 1131 Part 2 under overvoltage category II per DIN VDE 0110 Part 2.

For direct connection of AC Category III overvoltages provide protection measures for overvoltage category II according to IEC-Report 664/1980 and DIN VDE 0110 Part 1.

Equivalent standards:

● DIN VDE 0110 Part 1 ↔ IEC 664
● DIN VDE 0113 Part 1 ↔ EN 60204 Part 1
● DIN VDE 0660 Part 500 ↔ EN 60439-1 ↔ IEC 439-1

All rights reserved to change design, size, weight, etc.

Qualified personnel

Both the control system AC500 and other components in the vicinity are operated with dangerous contact voltages. Touching parts, which are under such voltages, can cause grave damage to health.

In order to avoid such risks and the occurrence of material damage, persons involved with the assembly, starting up and servicing must possess pertinent knowledge of the following:

● Automation technology sector
● Dealing with dangerous voltages
● Using standards and regulations, in particular VDE, accident prevention regulations and regulations concerning special ambient conditions (e.g. areas potentially endangered by explosive materials, heavy pollution or corrosive influences).

1.5.2.5 Standardization

EN 61131-2:2007 compliance statement: AC500/S500 products fully comply with all of the requirements contained in all clauses of this standard, such as indicated by reference to this standard without qualification.

CULus Hazardous Locations approval

Underwriters Laboratories Inc. in accordance with

● UL 508 (Industrial Control Equipment)
● CSA C22.2 No. 142 (Process Control Equipment)
● ANSI/ISA 12.12.01
● CSA C22.2 No. 213 (Hazardous Location)

APPROVED for use in Class I, division 2, Group A, B, C, D Tx;
Installation instructions for cULus Hazardous Locations

- WARNING - Explosion Hazard - Do not disconnect while circuit is live unless area is know to be non-hazardous.
- WARNING - Explosion Hazard - Substitution of components may impair suitability for Class I, Division 2 or Zone 2.
- This equipment is suitable for use in Class I, Division 2, Groups A, B, C, D or non-hazardous locations.

WARNING: EXPOSURE TO SOME CHEMICALS MAY DEGRADE THE SEALING PROPERTIES OF MATERIALS USED IN THE RELAYS.

**Standardization:**

**PROFINET**
- IEC 61131-3: PLC Standardization
- IEC 61158: PROFINET Standardization
- IEC 11801: Wire and connection elements for Ethernet

**CANopen**
- BOSCH CAN specification - version 2.0, part A and part B
- ISO 11898
  - CiA DS 201 V1.1 - CAN Application Layer
  - CiA DS 301 V3.0 - CAL based Communication Profile for Industrial Systems
  - CiA DS 301 V4.02 - CANopen Application Layer and Communication Profile
  - CiA DS 401 V2.1 - CANopen Device Profile Generic I/O modules
  - CiA DS 402 V2.0 - CANopen Device Profile Driver and Motion Control
  - CiA DS 406 V3.0 - CANopen Device Profile Encoder

**PROFIBUS DP**
- IEC61158 / DIN EN series

**EtherCAT**
- IEC 61131-3: PLC Standardization
- IEC 61158: EtherCAT Standardization
- IEC 11801: Wire and connection elements for Ethernet

---

**1.5.2.6 Terms and abbreviations**

### Table 505: PLC

<table>
<thead>
<tr>
<th>AC500</th>
<th>Advant Controller 500</th>
</tr>
</thead>
<tbody>
<tr>
<td>-S</td>
<td>Safety</td>
</tr>
<tr>
<td>-XC</td>
<td>Extreme Conditions</td>
</tr>
</tbody>
</table>

### Table 506: PROFINET

<table>
<thead>
<tr>
<th>CAT 5</th>
<th>Ethernet cable for 100 MHz data size</th>
</tr>
</thead>
<tbody>
<tr>
<td>CAT 5e</td>
<td>Ethernet cable for 100 MHz data size for Gigabit-Ethernet</td>
</tr>
<tr>
<td>CAT 6</td>
<td>Ethernet cable for 250 MHz data size</td>
</tr>
</tbody>
</table>

EtherCAT is registered trademark and patented technology, licensed by Beckhoff Automation GmbH, Germany.
<table>
<thead>
<tr>
<th>CAT 7</th>
<th>Ethernet cable for 600 MHz data size</th>
</tr>
</thead>
<tbody>
<tr>
<td>CAT 8</td>
<td>Ethernet cable for 1 GBit data size</td>
</tr>
<tr>
<td>GSDML</td>
<td>Device data sheet in XML</td>
</tr>
<tr>
<td>IO Controller</td>
<td>Conform to a master in a PROFINET network</td>
</tr>
<tr>
<td>IO Device</td>
<td>Conform to a slave in a PROFINET network</td>
</tr>
<tr>
<td>IO Supervisor</td>
<td>PC-based engineering tool for initial operating or rather diagnosis</td>
</tr>
<tr>
<td>IRT</td>
<td>Isochronous Real-Time</td>
</tr>
<tr>
<td>NRT</td>
<td>Non-Real-Time</td>
</tr>
<tr>
<td>PNO</td>
<td>PROFIBUS Nutzer-Organisation</td>
</tr>
<tr>
<td>PROFINET</td>
<td>Process Fieldbus Network</td>
</tr>
<tr>
<td>PROFINET CBA</td>
<td>PROFINET Component Based Automation for complex assembly modules</td>
</tr>
<tr>
<td>PROFINET IO</td>
<td>PROFINET for decentralized field bus devices, motion control, etc.</td>
</tr>
<tr>
<td>RT</td>
<td>Real-Time</td>
</tr>
<tr>
<td>VLAN-Tag</td>
<td>Virtual Local Area Network Tag, advanced telegram for Ethernet telegram priority</td>
</tr>
</tbody>
</table>

**CANopen**

<table>
<thead>
<tr>
<th>CAL</th>
<th>CAN Application Layer</th>
</tr>
</thead>
<tbody>
<tr>
<td>CAN</td>
<td>Controller Area Network</td>
</tr>
<tr>
<td>CiA</td>
<td>CAN in Automation e.V.</td>
</tr>
<tr>
<td>DLC</td>
<td>Data Length Code</td>
</tr>
<tr>
<td>EDS</td>
<td>Electronic Data Sheet</td>
</tr>
<tr>
<td>ISO</td>
<td>International Standardization Organization</td>
</tr>
<tr>
<td>NMT</td>
<td>Network Management</td>
</tr>
<tr>
<td>OD</td>
<td>Object Directory</td>
</tr>
<tr>
<td>PDO</td>
<td>Process Data Object</td>
</tr>
<tr>
<td>RTR</td>
<td>Remote Transmission Request</td>
</tr>
<tr>
<td>SDO</td>
<td>Service Data Object</td>
</tr>
</tbody>
</table>

**PROFIBUS DP**

<table>
<thead>
<tr>
<th>PROFIBUS DP</th>
<th>Process Fieldbus - Decentral Periphery</th>
</tr>
</thead>
<tbody>
<tr>
<td>DPM1</td>
<td>DP master (class 1), normal bus master</td>
</tr>
<tr>
<td>DPM2</td>
<td>DP master (class 2), commissioning device</td>
</tr>
<tr>
<td>DPS</td>
<td>DP slave, I/O module</td>
</tr>
<tr>
<td>GSD</td>
<td>Modules master data</td>
</tr>
<tr>
<td>DPV1</td>
<td>Guideline for functional expansions of PROFIBUS DP</td>
</tr>
<tr>
<td>PNO</td>
<td>PROFIBUS Nutzer Organisation (PROFIBUS user organization)</td>
</tr>
<tr>
<td>EtherCAT</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>-------------</td>
</tr>
<tr>
<td>CAT 5</td>
<td>Ethernet cable for 100 MHz data size</td>
</tr>
<tr>
<td>CAT 5e</td>
<td>Ethernet cable for 100 MHz data size for Gigabit-Ethernet</td>
</tr>
<tr>
<td>CAT 6</td>
<td>Ethernet cable for 250 MHz data size</td>
</tr>
<tr>
<td>CAT 7</td>
<td>Ethernet cable for 600 MHz data size</td>
</tr>
<tr>
<td>CAT 8</td>
<td>Ethernet cable for 1 GBit data size</td>
</tr>
<tr>
<td>CoE</td>
<td>CAN over EtherCAT</td>
</tr>
<tr>
<td>DC</td>
<td>Distributed Clock</td>
</tr>
<tr>
<td>DDF</td>
<td>Device Description File in XML format</td>
</tr>
<tr>
<td>ETG</td>
<td>EtherCAT Technology Group</td>
</tr>
<tr>
<td>EtherCAT</td>
<td>Ethernet for Control Automation Technology</td>
</tr>
<tr>
<td>FMMU</td>
<td>Fieldbus Memory Management Unit</td>
</tr>
<tr>
<td>NRT</td>
<td>Non Real-Time</td>
</tr>
<tr>
<td>RT</td>
<td>Real-Time</td>
</tr>
<tr>
<td>WKC</td>
<td>Working Counter</td>
</tr>
</tbody>
</table>

1.5.2.7 Definitions: PLC system start-up

**Cold start**
- A cold start is performed by switching power OFF/ON if no battery is connected.
- All RAM memory modules are checked and erased (see Command 'Reset cold' Chapter 1.3.1.25.2.6.9 “Command 'Reset Cold'” on page 931).
- If no user program is stored in the Flash EPROM, the default values (as set on delivery) are applied to the interfaces.
- If there is a user program stored in the Flash EPROM, it is loaded into RAM.
- The default operating modes set by the PLC configuration are applied.

**Warm start**
- A warm start is performed by switching power OFF/ON with a battery connected.
- All RAM memory modules are checked and erased except of the buffered operand areas and the RETAIN variables (see Command 'Reset warm' Chapter 1.3.1.25.2.6.10 “Command 'Reset Warm'” on page 931).
- If there is a user program stored in the Flash EPROM, it is loaded into RAM.
- The default operating modes set by the PLC configuration are applied.

**RUN -> STOP**
- RUN -> STOP means pressing the RUN function key on the PLC while the PLC is in run mode (PLC display "run").
- If a user program is loaded into RAM, execution is stopped.
- All outputs are set to FALSE or 0.
- Variables keep their current values, i.e., they are not initialized.
- The PLC display changes from "run" to "StoP".

**START -> STOP**
- START -> STOP means stopping the execution of the user program in the PLC's RAM using the menu item "Online/Stop" in the programming system.
- All outputs are set to FALSE or 0.
- Variables keep their current values, i.e., they are not initialized.
- The PLC display changes from "run" to "StoP".
Reset
- Performs a START -> STOP process.
- Preparation for program restart, i.e., the variables (VAR) (exception: RETAIN variables) are set to their initialization values.
- Reset is performed using the menu item "Online/Reset" in the programming system or pressing the function key RUN for ≥ 5 s in STOP mode.

Reset (cold)
- Performs a START -> STOP process.
- Preparation for program restart, i.e., the variables (VAR) (also RETAIN variables) are set to their initialization values.
- Reset (cold) is performed using the menu item "Online/Reset (cold)" in the programming system.

Reset (original)
- Resets the controller to its original state (deletion of Flash, SRAM (%M, area, %R area, RETAIN, RETAIN PERSISTENT), Communication Module configurations and user program).
- Reset (original) is performed using the menu item "Online/Reset (original)" in the programming system.

STOP -> RUN
- STOP -> RUN means short pressing the RUN function key on the PLC while the PLC is in STOP mode (PLC display "StoP").
- If a user program is loaded into RAM, execution is continued, i.e., variables will not be set to their initialization values.
- The PLC display changes from "StoP" to "run".

STOP -> START
- STOP -> START means continuing the execution of the user program in the PLC's RAM using the menu item "Online/Start" in the programming system.
- If a user program is loaded into RAM, execution is continued, i.e., variables will not be set to their initialization values.
- The PLC display changes from "StoP" to "run".

Download
- Download means loading the complete user program into the PLC's RAM. This process is started by selecting the menu item "Online/Download" in the programming system or after confirming a corresponding system message when switching to online mode (menu item "Online/Login").
- Execution of the user program is stopped.
- In order to store the user program to the Flash memory, the menu item "Online/Create boot project" must be called after downloading the program.
- Variables are set to their initialization values according to the initialization table.
- RETAIN variables can have wrong values as they can be allocated to other memory addresses in the new project!
- A download is forced by the following:
  - changed PLC configuration
  - changed task configuration
  - changed library management
  - changed compile-specific settings (segment sizes)
  - execution of the commands "Project/Clean all" and "Project/Rebuild All".

Online change
- After a project has changed, only these changes are compiled when pressing the key <F11> or calling the menu item "Project/Build". The changed program parts are marked with a blue arrow in the block list.
- The term Online Change means loading the changes made in the user program into the PLC's RAM using the programming system (after confirming a corresponding system message when switching to online mode, menu item "Online/Login").
● Execution of the user program is not stopped. After downloading the program changes, the program is re-organized. During re-organization, no further online change command is allowed. The storage of the user program to the Flash memory using the command “Online/Create boot project” cannot be initiated until re-organization is completed.

● Online Change is not possible after:
  – changes in the PLC configuration
  – changes in the task configuration
  – changes in the library management
  – changed compile-specific settings (segment sizes)
  – performing the commands "Project/Clean all" and "Project/Rebuild All".

Data buffering

● Data buffering, i.e., maintaining data after power ON/OFF, is only possible, if a battery is connected. The following data can be buffered completely or in parts:
  – Data in the addressable flag area (%M area)
  – RETAIN variable
  – PERSISTENT variable (number is limited, no structured variables)
  – PERSISTENT area (%R area)

● In order to buffer particular data, the data must be excluded from the initialization process (see Initialization of Variables).

1.5.2.8 Device lists

1.5.2.8.1 Device list: Terminal bases

Terminal bases for AC500 (Standard):

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>TB5600-2ETH</strong>&lt;br&gt;☑ Chapter 1.5.3.2.1 “TB56xx for AC500 V3 products” on page 2265</td>
<td>TB5600-2ETH, terminal base AC500, slots: 1 processor module, no communication module, 2 Ethernet RJ45 connector, 1 CAN connector</td>
</tr>
<tr>
<td><strong>TB5600-2ETH-XC</strong>&lt;br&gt;☑ Chapter 1.5.3.2.1 “TB56xx for AC500 V3 products” on page 2265</td>
<td>TB5600-2ETH-XC, terminal base AC500, slots: 1 processor module, no communication module, 2 Ethernet RJ45 connector, 1 CAN connector, XC version</td>
</tr>
<tr>
<td><strong>TB5610-2ETH</strong>&lt;br&gt;☑ Chapter 1.5.3.2.1 “TB56xx for AC500 V3 products” on page 2265</td>
<td>TB5610-2ETH, terminal base AC500, slots: 1 processor module, 1 communication module, 2 Ethernet RJ45 connector, 1 CAN connector</td>
</tr>
<tr>
<td><strong>TB5610-2ETH-XC</strong>&lt;br&gt;☑ Chapter 1.5.3.2.1 “TB56xx for AC500 V3 products” on page 2265</td>
<td>TB5610-2ETH-XC, terminal base AC500, slots: 1 processor module, 1 communication module, 2 Ethernet RJ45 connector, 1 CAN connector, XC version</td>
</tr>
<tr>
<td><strong>TB5620-2ETH</strong>&lt;br&gt;☑ Chapter 1.5.3.2.1 “TB56xx for AC500 V3 products” on page 2265</td>
<td>TB5620-2ETH, terminal base AC500, slots: 1 processor module, 2 communication modules, 2 Ethernet RJ45 connector, 1 CAN connector</td>
</tr>
<tr>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------</td>
<td>--------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>TB5620-2ETH-XC</strong></td>
<td>TB5620-2ETH-XC, terminal base AC500, slots: 1 processor module, 2 communication modules, 2 Ethernet RJ45 connector, 1 CAN connector, XC version</td>
</tr>
<tr>
<td><strong>TB5640-2ETH</strong></td>
<td>TB5640-2ETH, terminal base AC500, slots: 1 processor module, 4 communication modules, 2 Ethernet RJ45 connector, 1 CAN connector</td>
</tr>
<tr>
<td><strong>TB5640-2ETH-XC</strong></td>
<td>TB5640-2ETH-XC, terminal base AC500, slots: 1 processor module, 4 communication modules, 2 Ethernet RJ45 connector, 1 CAN connector, XC version</td>
</tr>
<tr>
<td><strong>TB5660-2ETH</strong></td>
<td>TB5660-2ETH, terminal base AC500, slots: 1 processor module, 6 communication modules, 2 Ethernet RJ45 connector, 1 CAN connector</td>
</tr>
<tr>
<td><strong>TB5660-2ETH-XC</strong></td>
<td>TB5660-2ETH-XC, terminal base AC500, slots: 1 processor module, 6 communication modules, 2 Ethernet RJ45 connector, 1 CAN connector, XC version</td>
</tr>
</tbody>
</table>

### 1.5.2.8.2 Device list: Processor modules (CPUs)

**Processor modules for AC500 (Standard)**

**V3 products**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>PM5630-2ETH</strong></td>
<td>PM5630-2ETH, processor module, memory 8 MB, 24 VDC, memory card slot, interface 1 RS-232/485, display, 2 RJ45 independent onboard Ethernet TCP/IP interfaces with Modbus TCP, web server, IEC60870-5-104 or selectable Ethernet based protocols</td>
</tr>
<tr>
<td><strong>PM5630-2ETH-XC</strong></td>
<td>PM5630-2ETH-XC, processor module, memory 8 MB, 24 VDC, memory card slot, interface 1 RS-232/485, display, 2 RJ45 independent onboard Ethernet TCP/IP interfaces with Modbus TCP, web server, IEC60870-5-104 or selectable Ethernet based protocols, XC version</td>
</tr>
<tr>
<td><strong>PM5650-2ETH</strong></td>
<td>PM5650-2ETH, processor module, memory 80 MB, 24 VDC, memory card slot, interface 1 RS-232/485, display, 2 RJ45 independent onboard Ethernet TCP/IP interfaces with Modbus TCP, web server, IEC60870-5-104 or selectable Ethernet based protocols</td>
</tr>
<tr>
<td><strong>PM5650-2ETH-XC</strong></td>
<td>PM5650-2ETH-XC, processor module, memory 80 MB, 24 VDC, memory card slot, interface 1 RS-232/485, display, 2 RJ45 independent onboard Ethernet TCP/IP interfaces with Modbus TCP, web server, IEC60870-5-104 or selectable Ethernet based protocols, XC version</td>
</tr>
</tbody>
</table>
1.5.2.8.3 Device list: Communication modules

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CM579-ETHCAT</td>
<td>CM579-ETHCAT, EtherCAT communication module</td>
</tr>
<tr>
<td>CM579-PNIO</td>
<td>CM579-PNIO, PROFINET communication module</td>
</tr>
<tr>
<td>CM579-PNIO-XC</td>
<td>CM579-PNIO-XC, PROFINET communication module, XC version</td>
</tr>
<tr>
<td>CM598-CN</td>
<td>CM598-CN, communication module, CANopen master</td>
</tr>
<tr>
<td>CM598-CN-XC</td>
<td>CM598-CN-XC, communication module, CANopen master, XC version</td>
</tr>
</tbody>
</table>
### 1.5.2.8.4 Device list: Terminal units

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>TU507-ETH</strong></td>
<td>Chapter 1.5.3.5.1 &quot;TU507-ETH and TU508-ETH for Ethernet communication interface modules&quot; on page 2307</td>
</tr>
<tr>
<td></td>
<td>TU507-ETH, Ethernet terminal unit, 24 VDC, screw terminals</td>
</tr>
<tr>
<td><strong>TU508-ETH</strong></td>
<td>Chapter 1.5.3.5.1 &quot;TU507-ETH and TU508-ETH for Ethernet communication interface modules&quot; on page 2307</td>
</tr>
<tr>
<td></td>
<td>TU508-ETH, Ethernet terminal unit, 24 VDC, spring terminals</td>
</tr>
<tr>
<td><strong>TU508-ETH-XC</strong></td>
<td>Chapter 1.5.3.5.1 &quot;TU507-ETH and TU508-ETH for Ethernet communication interface modules&quot; on page 2307</td>
</tr>
<tr>
<td></td>
<td>TU508-ETH-XC, Ethernet terminal unit, 24 VDC, spring terminals, XC version</td>
</tr>
<tr>
<td><strong>TU515</strong></td>
<td>Chapter 1.5.3.5.2 &quot;TU515, TU516, TU541 and TU542 for I/O modules&quot; on page 2311</td>
</tr>
<tr>
<td></td>
<td>TU515, I/O terminal unit, 24 VDC, screw terminals</td>
</tr>
<tr>
<td><strong>TU516</strong></td>
<td>Chapter 1.5.3.5.2 &quot;TU515, TU516, TU541 and TU542 for I/O modules&quot; on page 2311</td>
</tr>
<tr>
<td></td>
<td>TU516, I/O terminal unit, 24 VDC, spring terminals</td>
</tr>
<tr>
<td><strong>TU516-XC</strong></td>
<td>Chapter 1.5.3.5.2 &quot;TU515, TU516, TU541 and TU542 for I/O modules&quot; on page 2311</td>
</tr>
<tr>
<td></td>
<td>TU516-XC, I/O terminal unit, 24 VDC, spring terminals, XC version</td>
</tr>
<tr>
<td><strong>TU516-H</strong></td>
<td>Chapter 1.5.3.5.2 &quot;TU515, TU516, TU541 and TU542 for I/O modules&quot; on page 2311</td>
</tr>
<tr>
<td></td>
<td>TU516-H, I/O terminal unit, hot swap, 24 VDC, spring terminals</td>
</tr>
<tr>
<td><strong>TU516-H-XC</strong></td>
<td>Chapter 1.5.3.5.2 &quot;TU515, TU516, TU541 and TU542 for I/O modules&quot; on page 2311</td>
</tr>
<tr>
<td></td>
<td>TU516-H-XC, I/O terminal unit, hot swap, 24 VDC, spring terminals, XC version</td>
</tr>
<tr>
<td><strong>TU517</strong></td>
<td>Chapter 1.5.3.5.3 &quot;TU517 and TU518 for communication interface modules&quot; on page 2317</td>
</tr>
<tr>
<td></td>
<td>TU517, terminal unit for communication interface modules, 24 VDC, screw terminals</td>
</tr>
<tr>
<td><strong>TU518</strong></td>
<td>Chapter 1.5.3.5.3 &quot;TU517 and TU518 for communication interface modules&quot; on page 2317</td>
</tr>
<tr>
<td></td>
<td>TU518, terminal unit for communication interface modules, 24 VDC, spring terminals</td>
</tr>
<tr>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------</td>
<td>--------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>TU518-XC</td>
<td>TU518-XC, terminal unit for communication interface modules, 24 VDC, spring terminals, XC version</td>
</tr>
<tr>
<td>TU531</td>
<td>TU531, I/O terminal unit, 230 VAC, relays, screw terminals</td>
</tr>
<tr>
<td>TU532</td>
<td>TU532, I/O terminal unit, 230 VAC, relays, spring terminals</td>
</tr>
<tr>
<td>TU532-XC</td>
<td>TU532-XC, I/O terminal unit, 230 VAC, relays, spring terminals, XC version</td>
</tr>
<tr>
<td>TU532-H</td>
<td>TU532-H, I/O terminal unit, hot swap, 230 VAC, relays, spring terminals</td>
</tr>
<tr>
<td>TU532-H-XC</td>
<td>TU532-H-XC, I/O terminal unit, hot swap, 230 VAC, relays, spring terminals, XC version</td>
</tr>
<tr>
<td>TU541</td>
<td>TU541, I/O terminal unit, 24 VDC, screw terminals</td>
</tr>
<tr>
<td>TU542</td>
<td>TU542, I/O terminal unit, 24 VDC, spring terminals</td>
</tr>
<tr>
<td>TU542-XC</td>
<td>TU542-XC, I/O terminal unit, 24 VDC, spring terminals, XC version</td>
</tr>
<tr>
<td>TU542-H</td>
<td>TU542-H, I/O terminal unit, hot swap, 24 VDC, spring terminals</td>
</tr>
<tr>
<td>TU542-H-XC</td>
<td>TU542-H-XC, I/O terminal unit, hot swap, 24 VDC, spring terminals, XC version</td>
</tr>
</tbody>
</table>
### 1.5.2.8.5 Device list: S500-eCo I/O modules

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AI561</td>
<td>Chapter 1.5.3.6.2.1.1 “AI561 - Analog input module” on page 2543 AI561, analog input module, 4 AI, U/I</td>
</tr>
<tr>
<td>AI562</td>
<td>Chapter 1.5.3.6.2.1.2 “AI562 - Analog input module” on page 2554 AI562, analog input module, 2 AI, RTD</td>
</tr>
<tr>
<td>AI563</td>
<td>Chapter 1.5.3.6.2.1.3 “AI563 - Analog input module” on page 2565 AI563, analog input module, 4 AI, thermocouple</td>
</tr>
<tr>
<td>AO561</td>
<td>Chapter 1.5.3.6.2.1.4 “AO561 - Analog output module” on page 2576 AO561, analog output module, 2 AO, U/I</td>
</tr>
<tr>
<td>AX561</td>
<td>Chapter 1.5.3.6.2.1.5 “AX561 - Analog input/output module” on page 2586 AX561, analog input/output module, 4 AI, 2AO, U/I</td>
</tr>
<tr>
<td>DC561</td>
<td>Chapter 1.5.3.6.1.1.1 “DC561 - Digital input/output module” on page 2327 DC561, digital input/output module, 16 configurable inputs/outputs, transistor output, interfast connection</td>
</tr>
<tr>
<td>DC562</td>
<td>Chapter 1.5.3.6.1.1.2 “DC562 - Digital input/output module” on page 2335 DC561, digital input/output module, 16 configurable inputs/outputs</td>
</tr>
<tr>
<td>DI561</td>
<td>Chapter 1.5.3.6.1.1.3 “DI561 - Digital input module” on page 2346 DI561, digital input module, 8 DI, 24 VDC</td>
</tr>
<tr>
<td>DI562</td>
<td>Chapter 1.5.3.6.1.1.4 “DI562 - Digital input module” on page 2352 DI562, digital input module, 16 DI, 24 VDC</td>
</tr>
<tr>
<td>DI571</td>
<td>Chapter 1.5.3.6.1.1.5 “DI571 - Digital input module” on page 2360 DI571, digital input module, 8 DI, 120 VAC...240 VAC</td>
</tr>
<tr>
<td>DI572</td>
<td>Chapter 1.5.3.6.1.1.6 “DI572 - Digital input module” on page 2369 DI572, digital input module, 16 DI, 100 VAC...240 VAC</td>
</tr>
<tr>
<td>DO561</td>
<td>Chapter 1.5.3.6.1.1.7 “DO561 - Digital output module” on page 2378 DO561, digital output module, 8 DO, transistor output</td>
</tr>
<tr>
<td>DO562</td>
<td>Chapter 1.5.3.6.1.1.8 “DO562 - Digital output module” on page 2386 DO562, digital output module, 16 DO, transistor output</td>
</tr>
</tbody>
</table>
## 1.5.2.8.6 Device list: S500 I/O modules

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AI523</td>
<td>AI523, analog input module, 16 AI, U/I/Pt100, 12 bits + sign, 2-wires</td>
</tr>
<tr>
<td>AI523-XC</td>
<td>AI523-XC, analog input module, 16 AI, U/I/Pt100, 12 bits + sign, 2-wires, XC version</td>
</tr>
<tr>
<td>AI531</td>
<td>AI531, analog input module, 8 AI, U/I/Pt100, TC, 15 bits + sign, 4-wires</td>
</tr>
<tr>
<td>AI531-XC</td>
<td>AI531-XC, analog input module, 8 AI, U/I/Pt100, TC, 15 bits + sign, 4-wires, XC version</td>
</tr>
<tr>
<td>AO523</td>
<td>AO523, analog output module, 16 AO, U/I, 12 bits + sign, 2-wires</td>
</tr>
<tr>
<td>AO523-XC</td>
<td>AO523-XC, analog output module, 16 AO, U/I, 12 bits + sign, 2-wires, XC version</td>
</tr>
<tr>
<td>AX521</td>
<td>AX521, analog input/output module, 4 AI, 4 AO, U/I/Pt100, 12 bits + sign, 2-wires</td>
</tr>
<tr>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>AX521-XC</td>
<td>AX521-XC, analog input/output module, 4 AI, 4 AO, U/I/Pt100, 12 bits + sign, 2-wires, XC version</td>
</tr>
<tr>
<td>AX522</td>
<td>AX522, analog input/output module, 8 AI, 8 AO, U/I/Pt100, 12 bits + sign, 2-wires</td>
</tr>
<tr>
<td>AX522-XC</td>
<td>AX522-XC, analog input/output module, 8 AI, 8 AO, U/I/Pt100, 12 bits + sign, 2-wires, XC version</td>
</tr>
<tr>
<td>DA501</td>
<td>DA501, digital/analog input/output module, 16 DI, 8 DC, 4 AI, 2 AO</td>
</tr>
<tr>
<td>DA501-XC</td>
<td>DA501-XC, digital/analog input/output module, 16 DI, 8 DC, 4 AI, 2 AO, XC version</td>
</tr>
<tr>
<td>DA502</td>
<td>DA502, digital/analog input/output module, 16 DO, 8 DC, 4 AI, 2 AO</td>
</tr>
<tr>
<td>DA502-XC</td>
<td>DA502-XC, digital/analog input/output module, 16 DO, 8 DC, 4 AI, 2 AO, XC version</td>
</tr>
<tr>
<td>DC522</td>
<td>DC522, digital input/output module, 16 DC, 24 VDC / 0.5 A, 2-wires</td>
</tr>
<tr>
<td>DC522-XC</td>
<td>DC522-XC, digital input/output module, 16 DC, 24 VDC / 0.5 A, 2-wires, XC version</td>
</tr>
<tr>
<td>DC523</td>
<td>DC523, digital input/output module, 24 DC, 24 VDC / 0.5 A, 1-wire</td>
</tr>
<tr>
<td>DC523-XC</td>
<td>DC523-XC, digital input/output module, 24 DC, 24 VDC / 0.5 A, 1-wire, XC version</td>
</tr>
<tr>
<td>DC532</td>
<td>DC532, digital input/output module, 16 DI, 16 DC, 24 VDC / 0.5 A, 1-wire</td>
</tr>
<tr>
<td>DC532-XC</td>
<td>DC532-XC, digital input/output module, 16 DI, 16 DC, 24 VDC / 0.5 A, 1-wire, XC version</td>
</tr>
</tbody>
</table>
### 1.5.2.8.7 Device list: Communication interface modules

#### Table 507: CANopen

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CI581-CN</strong></td>
<td>CANopen communication interface module, 8 DI, 8 DO, 4 AI and 2 AO</td>
</tr>
<tr>
<td><strong>CI581-CN-XC</strong></td>
<td>CANopen communication interface module, 8 DI, 8 DO, 4 AI and 2 AO, XC version</td>
</tr>
</tbody>
</table>
### Table 508: EtherCAT

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CI511-ETHCAT</td>
<td>CI511-ETHCAT, EtherCAT communication interface module, 8 DI, 8 DO, 4 AI and 2 AO</td>
</tr>
<tr>
<td>CI512-ETHCAT</td>
<td>CI512-ETHCAT, EtherCAT communication interface module, 8 DI, 8 DO and 8 DC</td>
</tr>
</tbody>
</table>

### Table 509: Modbus

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CI521-MODTCP</td>
<td>CI521-MODTCP, Modbus TCP communication interface module, 4 AI, 2 AO, 8 DI and 8 DO</td>
</tr>
<tr>
<td>CI521-MODTCP-XC</td>
<td>CI521-MODTCP-XC, Modbus TCP communication interface module, 4 AI, 2 AO, 8 DI and 8 DO, XC version</td>
</tr>
<tr>
<td>CI522-MODTCP</td>
<td>CI522-MODTCP, Modbus TCP communication interface module, 8 DC, 8 DI and 8 DO</td>
</tr>
<tr>
<td>CI522-MODTCP-XC</td>
<td>CI522-MODTCP-XC, Modbus TCP communication interface module, 8 DC, 8 DI and 8 DO, XC version</td>
</tr>
</tbody>
</table>

### Table 510: PROFINET

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CI501-PNIO</td>
<td>CI501-PNIO (V3), PROFINET communication interface module, 8 DI, 8 DO, 4 AI and 2 AO</td>
</tr>
<tr>
<td>CI501-PNIO-XC</td>
<td>CI501-PNIO-XC (V3), PROFINET communication interface module, 8 DI, 8 DO, 4 AI and 2 AO, XC version</td>
</tr>
<tr>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>CI502-PNIO</td>
<td>CI502-PNIO, PROFINET communication interface module, 8 DI, 8 DO and 8 DC</td>
</tr>
<tr>
<td>CI502-PNIO-XC</td>
<td>CI502-PNIO-XC, PROFINET communication interface module, 8 DI, 8 DO and 8 DC, XC version</td>
</tr>
</tbody>
</table>

### 1.5.2.8.8 Device list: Accessories

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Automation Builder</td>
<td>DM-TOOL, Automation Builder software suite, programming software (multilanguage)</td>
</tr>
<tr>
<td>MC502</td>
<td>MC502, SD memory card</td>
</tr>
<tr>
<td>TA521</td>
<td>TA521, lithium battery</td>
</tr>
<tr>
<td>TA523</td>
<td>TA523, pluggable marker holder (10 pcs)</td>
</tr>
<tr>
<td>TA524</td>
<td>TA524, dummy communication module</td>
</tr>
<tr>
<td>TA525</td>
<td>TA525, set of 10 white plastic markers</td>
</tr>
<tr>
<td>TA526</td>
<td>TA526, wall mounting accessory, 10 pcs</td>
</tr>
<tr>
<td>TA535</td>
<td>TA535, protective caps for XC devices</td>
</tr>
</tbody>
</table>

### 1.5.2.9 Your tasks - Documentation from the user's point of view

#### 1.5.2.9.1 Consulting engineers

**Installation** Information on mounting and disassembling (including mechanical dimensions) is provided in chapter System Assembly, connection and construction Chapter 1.5.4 “System assembly, construction and connection” on page 3039.
Hardware

Information on hardware specifics on your devices is provided in the device description of the appropriate device.

1.5.2.9.2 Sales and export people

Installation

Information on mounting and disassembling (including mechanical dimensions) is provided in chapter System Assembly, connection and construction in Chapter 1.5.4 “System assembly, construction and connection” on page 3039.

Hardware

Information on hardware specifics on your devices is provided in the device description of the appropriate device.

1.5.2.9.3 Switch-gear cabinet manufacturers, design engineers

Installation

Information on mounting and disassembling (including mechanical dimensions) is provided in chapter System Assembly, connection and construction in Chapter 1.5.4 “System assembly, construction and connection” on page 3039.

Hardware

Information on hardware specifics on your devices is provided in the device description of the appropriate device.

1.5.2.9.4 Commissioning people

Installation of the software

www.abb.com/automationbuilder

Installation of the devices

Information on mounting and disassembling (including mechanical dimensions) is provided in chapter System Assembly, connection and construction in Chapter 1.5.4 “System assembly, construction and connection” on page 3039.

Hardware

Information on hardware specifics on your devices is provided in the device description of the appropriate device.

1.5.2.9.5 Programmers

Installation of the software

www.abb.com/automationbuilder

Software Description in Chapter 1.1.1 “Introduction” on page 14

Function block libraries

Function Block Libraries

System technology

System Technology for V3 devices in Chapter 1.5.5 “System technology for AC500 V3 products” on page 3104

OPC

OPC V3

Hardware

Information on hardware specifics on your devices is provided in the device description of the appropriate device.
1.5.2.9.6 Operators, service, maintenance

Installation of the software: www.abb.com/automationbuilder

Installation of the devices: Information on mounting and disassembling (including mechanical dimensions) is provided in chapter System Assembly, connection and construction. Chapter 1.5.4 “System assembly, construction and connection” on page 3039.

Hardware: Information on hardware specifics on your devices is provided in the device description of the appropriate device.

1.5.2.9.7 Education, training, plant training

Installation of the software: www.abb.com/automationbuilder

Installation of the devices: Information on mounting and disassembling (including mechanical dimensions) is provided in chapter System Assembly, connection and construction. Chapter 1.5.4 “System assembly, construction and connection” on page 3039.

Hardware: Information on hardware specifics on your devices is provided in the device description of the appropriate device.

1.5.2.9.8 Call center, help line

Installation of the software: www.abb.com/automationbuilder

Installation of the devices: Information on mounting and disassembling (including mechanical dimensions) is provided in chapter System Assembly, connection and construction. Chapter 1.5.4 “System assembly, construction and connection” on page 3039.

Hardware: Information on hardware specifics on your devices is provided in the device description of the appropriate device.

1.5.2.10 PLC system description

1.5.2.10.1 AC500 product family

AC500 programmable logic controllers (PLCs): The AC500 (Standard), AC500-eCo, AC500-S and AC500-XC scalable PLC ranges provide solutions for small, middle and high-end applications. Our AC500 platform offers different performance levels and is the ideal choice for high availability, extreme environments or safety solutions. Our AC500 PLC platform offers interoperability and compatibility in hardware and software from compact PLCs up to high-end and safety PLCs.

Due to the flexible combinations of AC500 devices and components, AC500 PLCs can be used for controlling a wide variety of applications to fulfill your automation needs.

Features of AC500 PLCs:
- Scalable and consistently expandable system
- Different performance classes of Processor Modules (CPUs) available
- Several field busses available
- Parallel connection to several field busses which can be combined arbitrarily
The AC500 product family consists of the product groups:

- **AC500 (standard):**
  AC500 standard PLCs offer a wide range of performance levels and scalability. The PLCs are highly capable of communication and extension for flexible application.

- **AC500-eCo:**
  AC500-eCo PLCs are cost-effective, high-performance compact PLCs that offer total interoperability with the core AC500 range and provide battery-free data buffering. All I/O modules can be freely connected in a simple, stable and reliable manner.

- **AC500-S:**
  AC500-S PLCs are designed for safety applications involved in factory, process or machinery automation area.

- **AC500-XC:**
  AC500 (standard) and AC500-S provide devices with -XC extension as a product variant. These variants operate according to their product group and can, in addition, be operated under extreme conditions. AC500-XC PLCs can be used at high altitudes, extended operating temperature and in humid condition. Further, the devices provide immunity to vibration and hazardous gases. The AC500-XC series is consistent with standard devices in the overall dimensions, control function and software compatibility.

  **AC500-XC system data** · Chapter 1.5.4.6.1 “System data AC500-XC” on page 3099.

The AC500 product family is characterized by functional modularity. As the complete AC500 product family shares the same hardware platform and programming software tool, the devices of the AC500 product groups can be flexibly combined.

S500 devices represent the I/O modules of the product group AC500 (standard), whereas S500-eCo devices represent the I/O Modules of the product group AC500-eCo. Both S500 and S500-eCo devices can be combined with devices of the AC500 product family in a flexible way.

---

**Extensions in the product name**

AC500 devices support different protocols and technologies (e.g. Ethernet, PROFIBUS etc.) in variable number. AC500 devices with onboard interfaces for support of a certain protocol or technology can be identified easily by the extension in the product name of the AC500 device. For example the AC500 Communication Module PM592-DP provides onboard support for PROFIBUS DP, the AC500 Processor Module PM595-4ETH provides onboard support for four provided Ethernet interfaces.

Further extensions in AC500 device names:

- -ETH: Ethernet
- -ARC: ARCNET
- -DP: PROFIBUS DP
- -CAN: CAN/CANopen
- -ETHCAT: EtherCAT
- -PNIO: PROFINET
- -RCOM: RCOM/RCOM+
- -RS: Serial interface

---

**1.5.2.10.2 AC500/S500 system structure**

The AC500 product family provides a variety of modules and pluggable components for expanding the capabilities of the CPU with additional I/Os or other communication protocols. Depending on the features and functions of the processor module (CPU) compatible components can be added to a complete AC500 PLC system.

Example of an AC500 PLC system:
1 Plug-in communication module
2 Processor module
3 Plug-in I/O module
4 Plug-in function module (AC500-eCo)
   - Plug-in communication module (AC500-S), not displayed
   - Plug-in I/O module (AC500-S), not displayed

Centralized I/O extension

Fig. 94: S500 I/O modules directly connected to a processor module
1.5.2.10.3 AC500/S500: Short description hardware

Processor modules

AC500 processor modules contain the CPU with the core component of the PLC. The CPU is connected with the user memory, input and output module, communication port and other units via system bus and performs tasks by means of system programs preset in the system memory. The CPU adopts the function preset by the system program to command the PLC for operation.

Its functions include:

- To receive user program and data entered
- To diagnose work faults of the power supply and PLC circuit as well as syntax error in programming
- To receive the state or data of the site via the input interface and save it into the shadow register or data register
- To read the user program in the memory one by one and execute it after interpretation
- To update the state of related flag bits and output shadow register contents according to execution results and realize output control by means of output unit.

Processor modules are available in different performance classes. Only one processor module is required for a valid system architecture.

There are different types of processor module available that differ in the features and functions they provide, e.g. performance, LED display etc.

If required, processor modules are also available with an integrated Ethernet communication module (TCP/IP).
AC500 communication modules are required for
- a connection to standard field bus systems and
- for integration into existing networks.

AC500 communication modules
- enable communication on different field buses.
- are mounted on the left side of the processor module on the same terminal base.
- are directly powered via the internal communication module bus of the terminal base. A separate voltage source is not required.

I/O modules
The I/O modules are the input / output unit which connects the PLC with the industrial production site. The PLC can detect controlled object data via the input interface and the data is taken as the basis for PLC control on the controlled object. In addition, the PLC sends processing results via the output interface to the controlled object to realize the control purpose.

External input equipment and output equipment need various signal levels while the information processed by the CPU in the PLC only can be the standard level. In order to realize such conversion, the I/O interface generally shall perform optical isolation and filtering to improve interference immunity of the PLC. In addition, the I/O interface generally can indicate the working state to facilitate maintenance.

The PLC provides multiple I/O interface for operation level and drive capability to users for selection such as digital input, digital output, analog input, analog output, etc. I/O interfaces of the PLC have the number of input / output signals taken as the number of PLC I/O points. The number of I/O points is an important basis for PLC selection. If the system is insufficient in the I/O points, it can be expanded via the I/O extension interface of the PLC.

The I/O modules for digital and/or analog inputs and outputs are available in different versions and allow flexible use thanks to configurable channels.

The modules can be simply plugged onto a terminal unit for a centralized I/O extension or for a decentralized I/O extension via communication interface modules.

Function modules extend the PLC system to perform special task control. Those modules often provide independent components such as a CPU, system programs, memory and interfaces connected with the PLC system bus.

It is connected with the PLC via the I/O bus to exchange data and independently work under cooperative management of the PLC.

Communication interface modules enable a decentralized I/O station. It contains embedded digital I/Os and a fieldbus interface.

Communication interface modules act as I/O slave devices within a master-slave-arrangement.
Terminal bases

On a terminal base the processor module and the communication modules are plugged.

For AC500-eCo processor modules and special AC500 (Standard) processor modules the terminal base cannot be removed.

Terminal units

On a terminal unit the I/O modules are plugged.
Terminal units enable simple prewiring without electronics and are available for 24 VDC and 230 VAC, optionally for spring or screw-type terminals.

Memory

In the PLC, the memory is mainly used for saving system programs, user programs and work data. The following memory types can be distinguished:

- Volatile memory:
  All saved data will be lost after power failure of the memory but the memory can provide high access rate and unlimited programming cycles. Common volatile memories mainly include SRAM and DRAM (including common memories such as SDRAM).

- Nonvolatile memory:
  All saved data will not be lost after power failure of the memory, but the memory is subject to low read-write rate and limited rewrite cycles. Common nonvolatile memories mainly include NORflash, NANDflash, EEPROM, SD card, etc.

AC500 PLCs store all user programs in the nonvolatile memory to get protected from power failure. The programs are exported to the volatile memory under operation of the PLC to ensure high-speed and efficient operation. If user program debugging is finished, the programs can be fixed in the nonvolatile memory when they need no change. The work data is subject to frequent change and access in the PLC operation. It is saved in the volatile memory to meet the requirements for random access.

The work data memory of PLC has the memory area for input and output relay, auxiliary relay, timer, counter and other logic devices. The state of these devices depends on initial setting and operation of the user programs. Some data maintains existing state by using built-in supercapacitors or backup batteries in case of power failure. The memory area for data saving in case of power failure is called the data retention area.
The PLC is equipped with a switch power supply for internal circuit. Compared with ordinary power supply, the PLC power supply has the higher stability and interference immunity. A number of PLC products provide 24 VDC stabilized voltage supply to meet external sensors.

1.5.2.10.4 Short description software

Configuration and programming of all AC500 control systems (CPUs) is done by using Automation Builder software.

Features:
- Standardized programming according to IEC 61131-3 - five programming languages (Structured Text (ST), Function Block Diagram (FBD), Instruction List (IL), Ladder Diagram (LD), Sequential Function Chart (SFC)), free graphical function chart (CFC), debugging functions for program test
- Application programming in C/C++
- Online diagnosis
- Debugging functions for the program test: Single step, Single cycle, Breakpoint
- Offline simulation - simulate commands without PLC being connected
- Sampling trace - timing diagrams for process variables
- Recipe management and watch lists
- Visualization
- Configuration of the communication interface modules (for PROFINET, EtherCAT, CANopen, Ethernet, Modbus)
- Programming - serial or via Ethernet networks
- Comprehensive libraries
- Export and import interfaces for devices, signals, applications, visualization, etc.
- Multi-user support and project compare
- Project scripting

IEC 61131-3 commands can be simulated without a PLC being connected, including the relevant malfunctions. After the program test, the application can be downloaded to the control system.

Sampling trace Timing diagrams for process variables and storage of data in a ring buffer with event trigger.
Recipe management and watch lists
Values of selected variables are displayed. Pre-defined values can be assigned to variables which can then be downloaded to the control system all at once ("Write recipe"). Actual values from the control system can also be pre-assigned for reading into the Watch and Recipe Manager, and stored in memory there ("Read recipe"). These functions are also helpful, for example, for setting and entering control parameters.

Visualization
Includes color change, moving elements, bitmaps, text display, allows input of setpoint values and display of process variables read from the PLC, dynamic bar diagrams, alarm and event management, function keys and ActiveX elements.

Programming
Serial or via Ethernet networks.

Engineering interface
Provides access from the programming system to an external project database in which the program source code of one or several automation projects is managed. Optionally, a version control system, such as Visual Source Safe, can be used in order to ensure data consistency of the program code for several different users and projects.

1.5.2.10.5 Control panels (HMI)
ABB control panels offer a wide range of features and functionalities for maximum operability. The panels are distinguished by their robustness and easy usability, providing all the relevant information from production plants and machines at a single touch.

Human Machine Interfaces - human control and operation of machines and processes.
Individual solutions for each application - this enables an operator at any time to have an overview on a profitable production and intervene manually if necessary.
Control panels with TFT graphical display and touch screen.
Available in various resolutions.

1.5.2.11 AC500-S
The AC500-S safety user manual must be read and understood before using safety configuration and programming tools of Automation Builder / PS501 Control Builder Plus. Only qualified personnel shall be allowed to work with AC500-S safety PLCs.
In order to have always the latest version and due to a different lifecycle compared to Automation Builder help, the AC500-S safety user manual is only available on our website.

1.5.2.12 Convert an AC500 V2 project to an AC500 V3 project
Instructions on how to convert a V2 project to a V3 project and differences between V2 and V3.
Keywords: Migration, conversion, V2, V3, differences between V2 and V3, application example, checkbox “Change to AC500 V3 PLC”

1.5.3 Device specifications

1.5.3.1 Status LEDs, display and control elements

Depending on the device type, various operating elements provided on the front panel can be used to control the devices of the PLC system and/or to change the operating mode.

Operating elements:

- Status LEDs:
  Indicates the availability of devices/components such as communication modules, communication interface modules or function modules. Functionality of the status LEDs depends on the specific module and is described in the device description of the appropriate module. Possible status: on/off/blinking

- I/O LEDs:
  Displays the status of the the inputs and outputs.

- LED display:
  Available for some processor modules. It can be used for simple configurations.

- Buttons and switches:
  Allows to change the current operating modes/status manually.

1.5.3.2 Terminal bases (AC500 standard)

1.5.3.2.1 TB56xx for AC500 V3 products

- TB5600-2ETH: 1 processor module, with network interface 2 Ethernet RJ45, 1 CAN and 1 COM1
- TB5610-2ETH: 1 processor module, 1 communication module, with network interface 2 Ethernet RJ45, 1 CAN and 1 COM1
- TB5620-2ETH: 1 processor module, 2 communication modules, with network interface 2 Ethernet RJ45, 1 CAN and 1 COM1
- TB5640-2ETH: 1 processor module, 4 communication modules, with network interface 2 Ethernet RJ45, 1 CAN and 1 COM1
- TB5660-2ETH: 1 processor module, 6 communication modules, with network interface 2 Ethernet RJ45, 1 CAN and 1 COM1
- XC version for use in extreme ambient conditions available

Table 511: Combination of TB56xx-2ETH(-XC) and PM56xx(-XC)

<table>
<thead>
<tr>
<th>Processor module</th>
<th>PM5630</th>
<th>PM5650</th>
<th>PM5670</th>
<th>PM5675</th>
</tr>
</thead>
<tbody>
<tr>
<td>TB5600-2ETH</td>
<td>0 slot</td>
<td>0 slot</td>
<td>0 slot</td>
<td>0 slot</td>
</tr>
<tr>
<td>TB5610-2ETH</td>
<td>1 slot</td>
<td>1 slot</td>
<td>1 slot</td>
<td>1 slot</td>
</tr>
<tr>
<td>TB5620-2ETH</td>
<td>2 slots</td>
<td>2 slots</td>
<td>2 slots</td>
<td>2 slots</td>
</tr>
<tr>
<td>TB5640-2ETH</td>
<td>-</td>
<td>4 slots</td>
<td>4 slots</td>
<td>4 slots</td>
</tr>
</tbody>
</table>

Terminal bases TB56xx-2ETH can only be used with processor modules PM56xx-2ETH.
### Processor module

<table>
<thead>
<tr>
<th>Processor module</th>
<th>PM5630</th>
<th>PM5650</th>
<th>PM5670</th>
<th>PM5675</th>
</tr>
</thead>
<tbody>
<tr>
<td>TB5660-2ETH</td>
<td>-</td>
<td>-</td>
<td>6 slots</td>
<td>6 slots</td>
</tr>
</tbody>
</table>

#### Remarks:

The slots can be used for connecting communication modules or AC500-S modules. Note that only one AC500-S module can be connected at one terminal base.

1) PM567x must have an index \( \geq C0 \).

The following figure shows the TB5620-2ETH as example.

1. I/O bus (10-pin, female) to electrically connect the I/O terminal units
2. One available slot for the processor module
3. Slots for communication modules
4. Interface for CAN (5-pin terminal block, removable)
5. Power supply (5-pin terminal block, removable)
6. Serial interface COM1 (9-pin terminal block, removable)
7. RJ45 female connector for ETHERNET1 connection
8. RJ45 female connector for ETHERNET2 connection
9. Holes for screw mounting
10. DIN rail

#### XC version

**XC = eXtreme Conditions**

*Extreme conditions*

Terminal bases for use in extreme ambient conditions have no \( \times \) sign for XC version.

*The figure 3 in the Part no. 1SAP3... (label) identifies the XC version.*
Short description

Terminal bases TB56xx are used as sockets for processor modules PM56xx and communication modules.

Up to 10 I/O terminal units for I/O expansion modules can be added to these terminal bases.

The terminal bases have slots for one processor module and for communication modules as well as terminals and interfaces for power supply, expansion and networking.

Table 512: Combination of TB56xx-2ETH(-XC) and PM56xx(-XC)

<table>
<thead>
<tr>
<th>Processor module</th>
<th>PM5630</th>
<th>PM5650</th>
<th>PM5670</th>
<th>PM5675</th>
</tr>
</thead>
<tbody>
<tr>
<td>TB5600-2ETH</td>
<td>0 slot</td>
<td>0 slot</td>
<td>0 slot</td>
<td>0 slot</td>
</tr>
<tr>
<td>TB5610-2ETH</td>
<td>1 slot</td>
<td>1 slot</td>
<td>1 slot</td>
<td>1 slot</td>
</tr>
<tr>
<td>TB5620-2ETH</td>
<td>2 slots</td>
<td>2 slots</td>
<td>2 slots</td>
<td>2 slots</td>
</tr>
<tr>
<td>TB5640-2ETH</td>
<td>-</td>
<td>4 slots</td>
<td>4 slots</td>
<td>4 slots</td>
</tr>
<tr>
<td>TB5660-2ETH</td>
<td>-</td>
<td>-</td>
<td>6 slots</td>
<td>6 slots</td>
</tr>
</tbody>
</table>

Remarks:
The slots can be used for connecting communication modules or AC500-S modules. Note that only one AC500-S module can be connected at one terminal base.

1) PM567x must have an index ≥ C0.

NOTICE!
Risk of malfunctions!

- Unused slots for communication modules must be covered with dummy communication modules (TA524 ☞ Chapter 1.5.3.8.1.3 “TA524 - Dummy communication module” on page 3034) to achieve IP20 rating.
- I/O bus connectors must not be touched during operation.

Connections

I/O bus

The I/O bus is the I/O data bus for the I/O modules. Through this bus, I/O and diagnosis data are transferred between the processor module and the I/O modules. Up to 10 I/O modules can be added (see description for I/O bus in the system assembly chapter ☞ Chapter 1.5.4.4.1 “Serial I/O bus” on page 3044).

Power supply

The supply voltage of 24 VDC is connected to a removable 5-pin terminal block. L+/M exist twice. It is therefore possible to feed e.g. external sensors (up to 8 A max. with 1.5 mm² conductor) via these terminals, when the ambient temperature never exceeds 60 °C.
### Pin assignment

<table>
<thead>
<tr>
<th>Pin Assignment</th>
<th>Label</th>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Terminal block removed</td>
<td>L+</td>
<td>+24 VDC</td>
<td>Positive pin of the power supply voltage</td>
</tr>
<tr>
<td>Terminal block inserted</td>
<td>L+</td>
<td>+24 VDC</td>
<td>Positive pin of the power supply voltage</td>
</tr>
<tr>
<td></td>
<td>M</td>
<td>0 V</td>
<td>Negative pin of the power supply voltage</td>
</tr>
<tr>
<td></td>
<td>M</td>
<td>0 V</td>
<td>Negative pin of the power supply voltage</td>
</tr>
<tr>
<td></td>
<td>↓</td>
<td>FE</td>
<td>Functional earth</td>
</tr>
</tbody>
</table>

### Faulty wiring on power supply terminals

**NOTICE!**
**Risk of damaging the processor module and terminal base!**
Exceeding the maximum voltage could lead to unrecoverable damage to the system.

The system could be destroyed.

**NOTICE!**
**Risk of malfunction!**
To ensure reliability and proper functionality of processor modules below index C0, the supply voltage must ramp-up from 0 V to 24 V within max. 2.5 s.

**NOTICE!**
**Risk of damaging the terminal base and power supply!**
Short circuits might damage the terminal base and power supply.

Make sure that the four clamps L+ and M (two of each) are not wrongly connected (e. g. +/- of power supply is connected to both L+/L+ or both M/M).

**NOTICE!**
**Risk of damaging the terminal base!**
Terminal base can be damaged by connecting the power supply terminal block (L+/M) to COM1.

Make sure that the COM1 terminal block is always connected to the terminal base even if you do not use COM1 to prevent this.

**NOTICE!**
**Risk of damaging the terminal base!**
Excessive current might damage the clamp and terminal base.

Make sure that the current flowing through the removable clamps never exceeds 8 A (with 1.5 mm² conductor).
NOTICE!
For applications using XC versions!
To ensure reliability and proper function, make sure the ambient temperature never exceeds 60 °C when the current flowing through the removable clamps is 8 A (with 1.5 mm² conductor).

Serial interface COM1
The serial interface COM1 is connected to a removable 9-pin terminal block. From firmware version V3.1 it is configurable for RS-232 or RS-485 (V3.0 RS-232 only).

### Pin assignment

<table>
<thead>
<tr>
<th>Pin</th>
<th>Signal</th>
<th>Interface</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Terminator P</td>
<td>RS-485</td>
<td>Terminator P</td>
</tr>
<tr>
<td>2</td>
<td>RxD/TxD-P</td>
<td>RS-485</td>
<td>Receive/Transmit, positive</td>
</tr>
<tr>
<td>3</td>
<td>RxD/TxD-N</td>
<td>RS-485</td>
<td>Receive/Transmit, negative</td>
</tr>
<tr>
<td>4</td>
<td>Terminator N</td>
<td>RS-485</td>
<td>Terminator N</td>
</tr>
<tr>
<td>5</td>
<td>RTS</td>
<td>RS-232</td>
<td>Request to send (output)</td>
</tr>
<tr>
<td>6</td>
<td>TxD</td>
<td>RS-232</td>
<td>Transmit data (output)</td>
</tr>
<tr>
<td>7</td>
<td>SGND</td>
<td>Signal Ground</td>
<td>Signal Ground</td>
</tr>
<tr>
<td>8</td>
<td>RxD</td>
<td>RS-232</td>
<td>Receive data (input)</td>
</tr>
<tr>
<td>9</td>
<td>CTS</td>
<td>RS-232</td>
<td>Clear to send (input)</td>
</tr>
</tbody>
</table>

**NOTICE!**
**Unused connector!**
Make sure that the terminal block is always connected to the terminal base, even if you do not use the interface.

For further information on connection and wiring please refer to Serial Interface COM1 of the Terminal Bases.

---

Ethernet interface
This interface is the connection to a processor module with onboard Ethernet e.g. PM56xx-2ETH.
TB56xx-2ETH for processor modules PM56xx-2ETH provide 2 independent Ethernet interfaces.

The two Ethernet interfaces can be configured as independent interfaces or with switch functionality.

In case of two independent interfaces they must be configured to different subnets.

For structured Ethernet cabling only use cables according to TIA/EIA-568-A, ISO/IEC 11801 or EN 50173.

<table>
<thead>
<tr>
<th>Interface</th>
<th>Pin</th>
<th>Signal</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>TxD+</td>
<td>Transmit Data +</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>TxD-</td>
<td>Transmit Data -</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>RxD+</td>
<td>Receive Data +</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>NU</td>
<td>Not used</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>NU</td>
<td>Not used</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>RxD-</td>
<td>Receive Data -</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>NU</td>
<td>Not used</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>NU</td>
<td>Not used</td>
<td></td>
</tr>
<tr>
<td>Shield</td>
<td></td>
<td>Cable shield</td>
<td>Functional earth</td>
</tr>
</tbody>
</table>

See supported protocols and used Ethernet ports for AC500 V3 products: Ethernet Protocols and Ports Chapter 1.5.1.3 “Ethernet protocols and ports for AC500 V3 products” on page 2224.

See communication via Modbus for AC500 V3 products: Modbus TCP/IP Chapter 1.5.5.1.9 “Communication with Modbus TCP/IP” on page 3182.

See communication via Modbus for AC500 V3 products: Modbus RTU Chapter 1.5.5.1.8 “Communication with Modbus RTU” on page 3166.

**CAN interface**

This interface is the connection to a processor module with onboard CAN e.g. PM56xx-2ETH.

<table>
<thead>
<tr>
<th>Interface socket</th>
<th>COMBICON, 5-pin, female, removable plug with spring terminals</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transmission standard</td>
<td>ISO 11898, potential-free</td>
</tr>
<tr>
<td>Transmission protocol</td>
<td>CANopen (CAN), 1 Mbaud max.</td>
</tr>
<tr>
<td>Transfer rate (transmission rate)</td>
<td>50 kbit/s, 100 kbit/s, 125 kbit/s, 250 kbit/s, 500 kbit/s, 800 kbit/s and 1 Mbit/s,</td>
</tr>
</tbody>
</table>
## Pin assignment

<table>
<thead>
<tr>
<th>Interface</th>
<th>Pin</th>
<th>Signal</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CAN</td>
<td>1</td>
<td>CAN_GND</td>
<td>CAN reference potential</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>CAN_L</td>
<td>Bus line, receive/transmit line, LOW</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>CAN_SHLD</td>
<td>Shield of the bus line</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>CAN_H</td>
<td>Bus line, receive/transmit line, HIGH</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>NC</td>
<td>Not connected</td>
</tr>
</tbody>
</table>

**NOTICE!**

Unused connector!

Make sure that the terminal block is always connected to the terminal base, even if you do not use the interface.

## Bus length

The maximum possible bus length of a CAN network depends on bit rate (transmission rate) and cable type. The sum of all bus segments must not exceed the maximum bus length.

<table>
<thead>
<tr>
<th>Bit Rate (speed)</th>
<th>Bus Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Mbit/s</td>
<td>40 m</td>
</tr>
<tr>
<td>800 kbit/s</td>
<td>50 m</td>
</tr>
<tr>
<td>500 kbit/s</td>
<td>100 m</td>
</tr>
<tr>
<td>250 kbit/s</td>
<td>250 m</td>
</tr>
<tr>
<td>125 kbit/s</td>
<td>500 m</td>
</tr>
<tr>
<td>50 kbit/s</td>
<td>1000 m</td>
</tr>
</tbody>
</table>

## Types of bus cables

Only bus cables with characteristics as recommended in ISO 11898 are to be used. The requirements for the bus cables depend on the length of the bus segment. See CANopen fieldbus Chapter 1.5.4.5.4.6 “CANopen field bus” on page 3082.

## Bus terminating resistors

Both ends of the CAN bus have to be terminated with a 120 Ω (≥ 1/4 W, ≤ 5 %) bus terminating resistor, to minimize signal reflection. The bus terminating resistor should be connected directly at the bus connector between the CAN signals (CAN_H and CAN_L). See CANopen fieldbus Chapter 1.5.4.5.4.6 “CANopen field bus” on page 3082.

## Technical data

The system data of AC500 and S500 are valid for standard version Chapter 1.5.4.5.1 “System data AC500” on page 3058.

The system data of AC500-XC are valid for the XC version Chapter 1.5.4.6.1 “System data AC500-XC” on page 3099.

Only additional details are therefore documented below.

The technical data are also valid for the XC version.
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Connection of the supply voltage 24 VDC at the terminal base of the processor module</td>
<td>Removable 5-pin terminal block spring type</td>
</tr>
</tbody>
</table>
| Max. current consumption from 24 VDC                                    | TB5600: 0.25 A ¹)  
TB5610: 0.35 A ¹)  
TB5620: 0.4 A ¹)  
TB5640: 0.6 A ¹)  
TB5660: 0.8 A ¹)  |
| Fuse melting integral at 24 VDC                                         | Min. 1 A's ²)                                                          |
| Peak inrush current from 24 VDC                                         | 55 A ²)                                                               |
| Number of slots for processor modules                                   | 1 (on all terminal bases)                                              |
| Processor module interfaces at TB56xx                                   | I/O bus, ETH1, ETH2, CAN, COM1                                         |
| Net weight (terminal base without processor module)                     | TB5600: 155 g  
TB5610: 180 g  
TB5620: 210 g  
TB5640: 260 g  
TB5660: 310 g  |
| Mounting position                                                       | Horizontal or vertical                                                |

¹) Including processor module, communication modules and I/O bus modules
²) The inrush current and the melting integral depends on the internal power supply of the processor module and the number and type of communication modules and I/O modules connected to the I/O bus.

Table 513: Combination of TB56xx-2ETH(-XC) and PM56xx(-XC)

<table>
<thead>
<tr>
<th>Processor module</th>
<th>PM5630</th>
<th>PM5650</th>
<th>PM5670</th>
<th>PM5675</th>
</tr>
</thead>
<tbody>
<tr>
<td>TB5600-2ETH</td>
<td>0 slot</td>
<td>0 slot</td>
<td>0 slot</td>
<td>0 slot</td>
</tr>
<tr>
<td>TB5610-2ETH</td>
<td>1 slot</td>
<td>1 slot</td>
<td>1 slot</td>
<td>1 slot</td>
</tr>
<tr>
<td>TB5620-2ETH</td>
<td>2 slots</td>
<td>2 slots</td>
<td>2 slots</td>
<td>2 slots</td>
</tr>
<tr>
<td>TB5640-2ETH</td>
<td>-</td>
<td>4 slots</td>
<td>4 slots</td>
<td>4 slots</td>
</tr>
<tr>
<td>TB5660-2ETH</td>
<td>-</td>
<td>-</td>
<td>6 slots ¹)</td>
<td>6 slots ¹)</td>
</tr>
</tbody>
</table>

Remarks:
The slots can be used for connecting communication modules or AC500-S modules. Note that only one AC500-S module can be connected at one terminal base.

¹) PM567x must have an index ≥ C0.
## Ordering data

<table>
<thead>
<tr>
<th>Part no.</th>
<th>Description</th>
<th>Product life cycle phase *)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1SAP 110 300 R0278</td>
<td>TB5600-2ETH, terminal base AC500, slots: 1 processor module, 2 Ethernet RJ45, 1 CAN connector</td>
<td>Active</td>
</tr>
<tr>
<td>1SAP 310 300 R0278</td>
<td>TB5600-2ETH-XC, terminal base AC500, slots: 1 processor module, 2 Ethernet RJ45, 1 CAN connector, XC version</td>
<td>Active</td>
</tr>
<tr>
<td>1SAP 111 300 R0278</td>
<td>TB5610-2ETH, terminal base AC500, slots: 1 processor module, 1 communication module, 2 Ethernet RJ45, 1 CAN connector</td>
<td>Active</td>
</tr>
<tr>
<td>1SAP 311 300 R0278</td>
<td>TB5610-2ETH-XC, terminal base AC500, slots: 1 processor module, 1 communication module, 2 Ethernet RJ45, 1 CAN connector, XC version</td>
<td>Active</td>
</tr>
<tr>
<td>1SAP 112 300 R0278</td>
<td>TB5620-2ETH, terminal base AC500, slots: 1 processor module, 2 communication modules, 2 Ethernet RJ45, 1 CAN connector</td>
<td>Active</td>
</tr>
<tr>
<td>1SAP 312 300 R0278</td>
<td>TB5620-2ETH-XC, terminal base AC500, slots: 1 processor module, 2 communication modules, 2 Ethernet RJ45, 1 CAN connector, XC version</td>
<td>Active</td>
</tr>
<tr>
<td>1SAP 114 300 R0278</td>
<td>TB5640-2ETH, terminal base AC500, slots: 1 processor module, 4 communication modules, 2 Ethernet RJ45, 1 CAN connector</td>
<td>Active</td>
</tr>
<tr>
<td>1SAP 314 300 R0278</td>
<td>TB5640-2ETH-XC, terminal base AC500, slots: 1 processor module, 4 communication modules, 2 Ethernet RJ45, 1 CAN connector, XC version</td>
<td>Active</td>
</tr>
<tr>
<td>1SAP 116 300 R0278</td>
<td>TB5660-2ETH, terminal base AC500, slots: 1 processor module, 6 communication modules, 2 Ethernet RJ45, 1 CAN connector</td>
<td>Active</td>
</tr>
<tr>
<td>1SAP 316 300 R0278</td>
<td>TB5660-2ETH-XC, terminal base AC500, slots: 1 processor module, 6 communication modules, 2 Ethernet RJ45, 1 CAN connector, XC version</td>
<td>Active</td>
</tr>
</tbody>
</table>

*) For planning and commissioning of new installations use modules in Active status only.

### Table 514: Combination of TB56xx-2ETH(-XC) and PM56xx(-XC)

<table>
<thead>
<tr>
<th>Processor module</th>
<th>PM5630</th>
<th>PM5650</th>
<th>PM5670</th>
<th>PM5675</th>
</tr>
</thead>
<tbody>
<tr>
<td>TB5600-2ETH</td>
<td>0 slot</td>
<td>0 slot</td>
<td>0 slot</td>
<td>0 slot</td>
</tr>
<tr>
<td>TB5610-2ETH</td>
<td>1 slot</td>
<td>1 slot</td>
<td>1 slot</td>
<td>1 slot</td>
</tr>
<tr>
<td>TB5620-2ETH</td>
<td>2 slots</td>
<td>2 slots</td>
<td>2 slots</td>
<td>2 slots</td>
</tr>
<tr>
<td>TB5640-2ETH</td>
<td>-</td>
<td>4 slots</td>
<td>4 slots</td>
<td>4 slots</td>
</tr>
</tbody>
</table>
1.5.3.3 Processor modules

The AC500 product family consists of the product groups:

- **AC500 (standard):**
  AC500 standard PLCs offer a wide range of performance levels and scalability. The PLCs are highly capable of communication and extension for flexible application.

- **AC500-eCo:**
  AC500-eCo PLCs are cost-effective, high-performance compact PLCs that offer total interoperability with the core AC500 range and provide battery-free data buffering. All I/O modules can be freely connected in a simple, stable and reliable manner.

- **AC500-S:**
  AC500-S PLCs are designed for safety applications involved in factory, process or machinery automation area.

- **AC500-XC:**
  AC500 (standard) and AC500-S provide devices with -XC extension as a product variant. These variants operate according to their product group and can, in addition, be operated under extreme conditions. AC500-XC PLCs can be used at high altitudes, extended operating temperature and in humid condition. Further, the devices provide immunity to vibration and hazardous gases. The AC500-XC series is consistent with standard devices in the overall dimensions, control function and software compatibility. 

The AC500 product family is characterized by functional modularity. As the complete AC500 product family shares the same hardware platform and programming software tool, the devices of the AC500 product groups can be flexibly combined.

S500 devices represent the I/O modules of the product group AC500 (standard), whereas S500-eCo devices represent the I/O Modules of the product group AC500-eCo. Both S500 and S500-eCo devices can be combined with devices of the AC500 product family in a flexible way.

1.5.3.3.1 AC500 (standard)

**PM56xx-2ETH for AC500 V3 products**

Processor modules with onboard interfaces:

- **PM5630-2ETH:** processor module, memory 8 MB, with Ethernet support (onboard Ethernet) – 2 network interfaces RJ45, CAN and COM1 on the terminal base.
- **PM5650-2ETH:** processor module, memory 80 MB, with Ethernet support (onboard Ethernet) – 2 network interfaces RJ45, CAN and COM1 on the terminal base.

---

### Table 515: Accessories

<table>
<thead>
<tr>
<th>Part no.</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1SAP 180 800 R0001</td>
<td>TA526, wall mounting accessory</td>
</tr>
</tbody>
</table>
PM5670-2ETH: processor module, memory 160 MB, with Ethernet support (onboard Ethernet) – 2 network interfaces RJ45, CAN and COM1 on the terminal base.

PM5675-2ETH: processor module, 160 MB, 8 GB flash disk, with Ethernet support (onboard Ethernet) – 2 network interfaces RJ45, CAN and COM1 on the terminal base.

XC version for use in extreme ambient conditions available

1. 6 7-segment state displays with background lighting
2. "Triangle" displays for "item"
3. "Square" displays for "state"
4. 3 state LEDs
5. 8 function keys
6. Slot for memory card
7. Label
8. Compartment for lithium battery TA521
9. Lithium battery TA521
10. Memory card (MC502)
11. I/O bus for connection of I/O modules
12. Slot for processor module (processor module mounted on terminal base)
13. Slots for communication modules (multiple, depending on terminal base; unused slots must be covered with TA524)
14. Interface for CAN (5-pin terminal block, removable)
15. Power supply (5-pin terminal block, removable)
16. Serial interface COM1 (9-pin terminal block, removable)
17. RJ45 female connector for ETHERNET1 connection
18. RJ45 female connector for ETHERNET2 connection
19. DIN rail

* * * Sign for XC version

2020/12/10
**Short description**

The processor modules are the central units of the control system AC500. The types differ in their performance (memory size, speed etc.). Each processor module must be mounted on a suitable terminal base.

The terminal base type (TB56xx) depends on the number of communication modules which are used together with the processor module.

**Table 516: Comparison: TB56xx**

<table>
<thead>
<tr>
<th>Processor module</th>
<th>PM5630</th>
<th>PM5650</th>
<th>PM5670</th>
<th>PM5675</th>
</tr>
</thead>
<tbody>
<tr>
<td>Max. number of variables allowed for each communication module supported</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Input variables</td>
<td>4 kB</td>
<td>4 kB</td>
<td>5 kB</td>
<td>5 kB</td>
</tr>
<tr>
<td>Output variables</td>
<td>4 kB</td>
<td>4 kB</td>
<td>5 kB</td>
<td>5 kB</td>
</tr>
<tr>
<td>Type of communication module supported</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CM574-RS/RCOM - serial interface</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>CM582-DP - PROFIBUS DP V0/V1 slave</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>CM592-DP - PROFIBUS DP V0/V1 master</td>
<td>1)</td>
<td>1)</td>
<td>1)</td>
<td>1)</td>
</tr>
<tr>
<td>CM579-ETHCAT - EtherCAT master</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>CM579-PNIO - PROFINET IO RT controller</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>CM589-PNIO - PROFINET IO RT device</td>
<td>1)</td>
<td>1)</td>
<td>1)</td>
<td>1)</td>
</tr>
<tr>
<td>CM589-PNIO-4 - PROFINET IO RT with 4 devices</td>
<td>1)</td>
<td>1)</td>
<td>1)</td>
<td>1)</td>
</tr>
<tr>
<td>CM597-ETH - Ethernet interface</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>CM588-CN - CAN, CANopen slave</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>CM598-CN - CAN, CANopen master</td>
<td>only CAN 2A/2B</td>
<td>only CAN 2A/2B</td>
<td>only CAN 2A/2B</td>
<td>only CAN 2A/2B</td>
</tr>
<tr>
<td>Type of AC500-S module supported</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SM560-S - safety module</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>SM560-S-FD-1 - safety module with F-Device functionality for 1 PROFIsafe network</td>
<td>1)</td>
<td>1)</td>
<td>1)</td>
<td>1)</td>
</tr>
<tr>
<td>SM560-S-FD-4 - safety module with F-Device functionality for 1 PROFIsafe network</td>
<td>1)</td>
<td>1)</td>
<td>1)</td>
<td>1)</td>
</tr>
</tbody>
</table>

**Remarks:**

1) in preparation

All terminal bases (TB56xx) provide the same communication interfaces (ETH1, ETH2, CAN and COM1). Technical Data Chapter 1.5.3.2.1.3 “Technical data” on page 2271

All other V3 processor modules can operate multiple communication modules via their communication module interface.

The communication modules are mounted on the left side of the processor module on the same terminal base.

On the right side of the processor module, up to 10 digital or analog I/O expansion modules can be connected to the I/O bus. Each I/O module requires a suitable terminal unit depending on the module type.

Terminal bases, terminal units, I/O modules, communication modules and accessories have their own technical descriptions.
Each processor module can be used as:

- Stand-alone processor module
- Stand-alone processor module with local I/Os
- Remote IO server
- Remote IO client

The processor modules are powered with 24 VDC.

**WARNING!**

Removal/Insertion under power
The devices are not designed for removal or insertion under power. Because of unforeseeable consequences, it is not allowed to plug or unplug devices with the power being ON.

Make sure that all voltage sources (supply and process voltage) are switched off before you
- connect or disconnect any signal or terminal block
- remove, mount or replace a module.

Disconnecting any powered devices while energized in a hazardous location could result in an electric arc, which could create a flammable ignition resulting in fire or explosion.

Make sure that power is removed and that the area has been thoroughly checked to ensure that flammable materials are not present prior to proceeding.

The devices must not be opened when in operation. The same applies to the network interfaces.

**Connections**

All terminals for electrical connection are available on the terminal base. For information on connection and available interfaces see the descriptions for

- **TB56xx** % Chapter 1.5.3.2.1 “TB56xx for AC500 V3 products” on page 2265.

---

**Processor modules PM56xx-2ETH can only be used with TB56xx-2ETH terminal bases.**

---

**Table 517: Combination of TB56xx-2ETH(-XC) and PM56xx(-XC)**

<table>
<thead>
<tr>
<th>Processor module</th>
<th>PM5630</th>
<th>PM5650</th>
<th>PM5670</th>
<th>PM5675</th>
</tr>
</thead>
<tbody>
<tr>
<td>TB5600-2ETH</td>
<td>0 slot</td>
<td>0 slot</td>
<td>0 slot</td>
<td>0 slot</td>
</tr>
<tr>
<td>TB5610-2ETH</td>
<td>1 slot</td>
<td>1 slot</td>
<td>1 slot</td>
<td>1 slot</td>
</tr>
<tr>
<td>TB5620-2ETH</td>
<td>2 slots</td>
<td>2 slots</td>
<td>2 slots</td>
<td>2 slots</td>
</tr>
<tr>
<td>TB5640-2ETH</td>
<td>-</td>
<td>4 slots</td>
<td>4 slots</td>
<td>4 slots</td>
</tr>
<tr>
<td>TB5660-2ETH</td>
<td>-</td>
<td>-</td>
<td>6 slots ¹</td>
<td>6 slots ¹</td>
</tr>
</tbody>
</table>

Remarks:
The slots can be used for connecting communication modules or AC500-S modules. Note that only one AC500-S module can be connected at one terminal base.

¹) PM567x must have an index ≥ C0.
Storage elements

Lithium battery

The processor modules are supplied without lithium battery. It must be ordered separately. The TA521 lithium battery is used for data (SRAM) and RTC buffering while the processor module is not powered.

See system technology - AC500 battery Chapter 1.5.5.1.3.2 “AC500 battery” on page 3116.

The CPU monitors the discharge degree of the battery. A warning is issued before the battery condition becomes critical (about 2 weeks before). Once the warning message appears, the battery should be replaced as soon as possible.

The technical data, handling instructions and the insertion/replacement of the battery is described in detail in the chapter TA521 lithium battery Chapter 1.5.3.8.1.2 “TA521 - Lithium battery” on page 3030.

Memory card

AC500 processor modules are supplied without memory card. It must be ordered separately.

The memory card can be used
• to read and write user files
• to download a user program
• for firmware updates

Detailed information can be found in the System Technology chapter Chapter 1.5.5.1.2 “System processing” on page 3110.

AC500 processor modules can be operated with and without memory cards. The processor module uses a standard file system (FAT). This allows standard card readers to read and write the memory cards (MC502).

Only genuine MC502 memory cards are supported.

For more information on the technical data, handling instructions and the insertion/replacement of the memory card, please refer to the chapter memory card MC502 Chapter 1.5.3.8.1.1 “MC502 - SD memory card” on page 3028.
LEDs, display and function keys on the front panel

Detailed information on using the LEDs, display and the function keys such as startup procedure and error coding is described in the system technology section Display § Chapter 1.5.5.1.4 “LEDs, display and function keys on the front panel” on page 3118.

Technical data

The system data of AC500 and S500 are valid for standard version § Chapter 1.5.4.5.1 “System data AC500” on page 3058.

The system data of AC500-XC are valid for the XC version § Chapter 1.5.4.6.1 “System data AC500-XC” on page 3099.

Only additional details are therefore documented below.

The technical data are also valid for the XC version.

Processor module and terminal base

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Connection of the supply voltage 24 VDC at the terminal base of the processor module</td>
<td>Removable 5-pin terminal block with spring connection</td>
</tr>
<tr>
<td>Current consumption on 24 VDC</td>
<td></td>
</tr>
<tr>
<td>Min. typ. (module alone)</td>
<td>PM5630-2ETH: 110 mA</td>
</tr>
<tr>
<td></td>
<td>PM5650-2ETH: 120 mA</td>
</tr>
<tr>
<td></td>
<td>PM5670-2ETH: 130 mA</td>
</tr>
<tr>
<td></td>
<td>PM5675-2ETH: 140 mA</td>
</tr>
<tr>
<td>Parameter</td>
<td>Value</td>
</tr>
<tr>
<td>-------------------------------------------------------------------------</td>
<td>----------------------------------------------------------------------</td>
</tr>
<tr>
<td>Max. typ. (all couplers and I/Os)</td>
<td>PM5630-2ETH: 850 mA</td>
</tr>
<tr>
<td></td>
<td>PM5650-2ETH: 900 mA</td>
</tr>
<tr>
<td></td>
<td>PM5670-2ETH: 950 mA</td>
</tr>
<tr>
<td></td>
<td>PM5675-2ETH: 950 mA</td>
</tr>
<tr>
<td>Number of slots for processor modules</td>
<td>1 (on all terminal bases)</td>
</tr>
<tr>
<td>Processor module interfaces at the terminal bases TB56xx</td>
<td>I/O bus, ETH1, ETH2, CAN, COM1</td>
</tr>
<tr>
<td>Connection system</td>
<td>see System Assembly, Construction and Connection</td>
</tr>
<tr>
<td></td>
<td>Chapter 1.5.4.5.4 “Connection and wiring” on page 3076</td>
</tr>
<tr>
<td>Weight (processor module without terminal base)</td>
<td>135 g</td>
</tr>
<tr>
<td>Mounting position</td>
<td>Horizontal or vertical</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Table 518: Comparison: PM56xx</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Parameter</strong></td>
</tr>
<tr>
<td>Program and data memory (FLASH and DDR3 SDRAM)</td>
</tr>
<tr>
<td>Thereof user program and data (dynamically allocated)</td>
</tr>
<tr>
<td>Thereof user webservice data</td>
</tr>
<tr>
<td>Number of webvisu instances in a project limitation (web pages)</td>
</tr>
<tr>
<td>Remaining for all other usage (project save, infrastructure...)</td>
</tr>
<tr>
<td>Buffered (SRAM)</td>
</tr>
<tr>
<td>Thereof VAR retain persistent</td>
</tr>
<tr>
<td>Thereof %M memory (e.g. Modbus register)</td>
</tr>
<tr>
<td>Expandable memory</td>
</tr>
<tr>
<td>Integrated mass storage memory (FLASH)</td>
</tr>
<tr>
<td>Slot for pluggable memory card</td>
</tr>
<tr>
<td>Processor speed</td>
</tr>
<tr>
<td>Cycle time for 1 instruction (minimum):</td>
</tr>
<tr>
<td>Binary</td>
</tr>
<tr>
<td>Word</td>
</tr>
<tr>
<td>Floating point</td>
</tr>
<tr>
<td>Mathematic co-processor</td>
</tr>
<tr>
<td>Motion capability</td>
</tr>
<tr>
<td>No. synchronized axis per 1 ms on EtherCAT CM typically</td>
</tr>
</tbody>
</table>
### Processor module

<table>
<thead>
<tr>
<th>Processor module</th>
<th>PM5630</th>
<th>PM5650</th>
<th>PM5670</th>
<th>PM5675</th>
</tr>
</thead>
<tbody>
<tr>
<td>No. synchronized axis per 2 ms on EtherCAT CM typically</td>
<td>-</td>
<td>8</td>
<td>32</td>
<td>32</td>
</tr>
<tr>
<td>No. synchronized axis per 4 ms on EtherCAT CM or CANopen onboard typically</td>
<td>2</td>
<td>8</td>
<td>&gt;32</td>
<td>&gt;32</td>
</tr>
<tr>
<td>Min. bus cycle time for EtherCAT using external CM579</td>
<td>2 ms</td>
<td>1 ms</td>
<td>0,5 ms</td>
<td>0,5 ms</td>
</tr>
</tbody>
</table>

Max. number of central inputs and outputs (10 exp. modules):

| Digital inputs | 320 |
| Digital outputs | 320 |
| Analog inputs | 160 |
| Analog outputs | 160 |

Number of decentralized inputs and outputs: Depends on the used fieldbus

**Data backup:** Battery

**Data buffering time at 25 °C:** Typ. 3 years

**Battery low indication:** via application program

**Real-time clock:**
- With battery back-up: x
- Accuracy: Typ. ±2 s / day at 25 °C

**Program execution:**
- Cyclic: x
- Time-controlled: x
- Multitasking: x
- Minimum cycle time configurable for cyclical task: 1 ms, 1 ms, 0.5 ms, 0.5 ms

**User program protection by password:** x (user management)

**Internal interfaces for communication:**

**Serial interface COM1:**
- Physical link: Configurable for RS-232 or RS-485 (9.6 kb/s, 19.2 kb/s, 38.4 kb/s, 57.6 kb/s and 115.2 kb/s)
- Connection: Pluggable terminal block, spring connection
- Usage: Serial ASCII communication, Modbus RTU

**CAN interface:**
- Physical link: CAN 2A/2B (from 50 kb/s to 1 Mb/s)
- Connection: Pluggable terminal block, spring connection
- Usage: CANopen master communication, CAN 2A/2B, J1939 protocol, CAN sync
- Max. number of variables allowed:
  - Input variables: 2 kB, 4 kB, 5 kB, 5 kB
  - Output variables: 2 kB, 4 kB, 5 kB, 5 kB

**Network interface ETH1, ETH2:**
- Usage: Ethernet
### Processor module

<table>
<thead>
<tr>
<th>Processor module</th>
<th>PM5630</th>
<th>PM5650</th>
<th>PM5670</th>
<th>PM5675</th>
</tr>
</thead>
<tbody>
<tr>
<td>Physical link</td>
<td>10/100 base-TX, configurable as internal switch or independent Interfaces</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Connection</td>
<td>2x RJ45 socket, provided on TB56xx-2ETH</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LEDs, LCD display, function keys</td>
<td>RUN / STOP, status, diagnosis, settings</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Number of timers</td>
<td>Unlimited</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Number of counters</td>
<td>Unlimited</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Programming languages:</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Structured text ST</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Instruction list IL</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Function block diagram FBD</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ladder diagram LD</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sequential function chart SFC</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Continuous function chart (CFC)</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Table 519: Combination of TB56xx-2ETH(-XC) and PM56xx(-XC)**

<table>
<thead>
<tr>
<th>Processor module</th>
<th>PM5630</th>
<th>PM5650</th>
<th>PM5670</th>
<th>PM5675</th>
</tr>
</thead>
<tbody>
<tr>
<td>TB5600-2ETH</td>
<td>0 slot</td>
<td>0 slot</td>
<td>0 slot</td>
<td>0 slot</td>
</tr>
<tr>
<td>TB5610-2ETH</td>
<td>1 slot</td>
<td>1 slot</td>
<td>1 slot</td>
<td>1 slot</td>
</tr>
<tr>
<td>TB5620-2ETH</td>
<td>2 slots</td>
<td>2 slots</td>
<td>2 slots</td>
<td>2 slots</td>
</tr>
<tr>
<td>TB5640-2ETH</td>
<td>-</td>
<td>4 slots</td>
<td>4 slots</td>
<td>4 slots</td>
</tr>
<tr>
<td>TB5660-2ETH</td>
<td>-</td>
<td>-</td>
<td>6 slots</td>
<td>6 slots</td>
</tr>
</tbody>
</table>

Remarks:

The slots can be used for connecting communication modules or AC500-S modules. Note that only one AC500-S module can be connected at one terminal base.

1) PM567x must have an index $\geq C0$.

**Table 520: Comparison: TB56xx**

<table>
<thead>
<tr>
<th>Processor module</th>
<th>PM5630</th>
<th>PM5650</th>
<th>PM5670</th>
<th>PM5675</th>
</tr>
</thead>
<tbody>
<tr>
<td>Max. number of variables allowed for each communication module supported</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Input variables</td>
<td>4 kB</td>
<td>4 kB</td>
<td>5 kB</td>
<td>5 kB</td>
</tr>
<tr>
<td>Output variables</td>
<td>4 kB</td>
<td>4 kB</td>
<td>5 kB</td>
<td>5 kB</td>
</tr>
<tr>
<td>Type of communication module supported</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CM574-RS/RCOM - serial interface</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>CM582-DP - PROFIBUS DP V0/V1 slave</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>CM592-DP - PROFIBUS DP V0/V1 master</td>
<td>1)</td>
<td>1)</td>
<td>1)</td>
<td>1)</td>
</tr>
<tr>
<td>CM579-ETHCAT - EtherCAT master</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>CM579-PNIO - PROFINET IO RT controller</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>CM589-PNIO - PROFINET IO RT device</td>
<td>1)</td>
<td>1)</td>
<td>1)</td>
<td>1)</td>
</tr>
<tr>
<td>CM589-PNIO-4 - PROFINET IO RT with 4 devices</td>
<td>1)</td>
<td>1)</td>
<td>1)</td>
<td>1)</td>
</tr>
<tr>
<td>CM597-ETH - Ethernet interface</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>CM588-CN - CAN, CANopen slave</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>
### Processor module

<table>
<thead>
<tr>
<th>Type of AC500-S module supported</th>
<th>PM5630</th>
<th>PM5650</th>
<th>PM5670</th>
<th>PM5675</th>
</tr>
</thead>
<tbody>
<tr>
<td>CM598-CN - CAN, CANopen master</td>
<td>only CAN 2A/2B</td>
<td>only CAN 2A/2B</td>
<td>only CAN 2A/2B</td>
<td>only CAN 2A/2B</td>
</tr>
<tr>
<td>SM560-S - safety module</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>SM560-S-FD-1 - safety module with F-Device functionality for 1 PROFIsafe network</td>
<td>1) 1) 1) 1)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SM560-S-FD-4 - safety module with F-Device functionality for 1 PROFIsafe network</td>
<td>1) 1) 1) 1)</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Remarks:
1) in preparation

---

### Communication and onboard protocols

#### Table 521: OPC UA server / OPC DA server

<table>
<thead>
<tr>
<th>Processor module</th>
<th>PM5630</th>
<th>PM5650</th>
<th>PM5670</th>
<th>PM5675</th>
</tr>
</thead>
<tbody>
<tr>
<td>OPC UA server</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>Number of free tags + additional license for extension 1)</td>
<td>1.000 5.000</td>
<td>30.000 30.000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Number of connections</td>
<td>10 20</td>
<td>50 50</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Min. sampling rate (limit)</td>
<td>500 ms 100 ms</td>
<td>50 ms 50 ms</td>
<td></td>
<td></td>
</tr>
<tr>
<td>OPC DA server AE</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>Number of connections</td>
<td>8 8</td>
<td>8 8</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Remarks:
1) in preparation

#### Table 522: Modbus, Telecontrol

<table>
<thead>
<tr>
<th>Processor module</th>
<th>PM5630</th>
<th>PM5650</th>
<th>PM5670</th>
<th>PM5675</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of Modbus clients ModMast in parallel on a CPU master (server)</td>
<td>30 50</td>
<td>120 120</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Number of Modbus server in parallel (e.g. for SCADA access)</td>
<td>15 25</td>
<td>50 50</td>
<td></td>
<td></td>
</tr>
<tr>
<td>IEC 60870-5-104 telecontrol protocol</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>Number of free tags + additional license for extension 1)</td>
<td>1.000 5.000</td>
<td>10.000 10.000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Control station (number of connections)</td>
<td>5 10</td>
<td>20 20</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sub-station (number of connections)</td>
<td>5 10</td>
<td>20 20</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Remarks:
1) in preparation
<table>
<thead>
<tr>
<th>Part no.</th>
<th>Description</th>
<th>Product life cycle phase *)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1SAP 131 000 R0278</td>
<td>PM5630-2ETH, processor module, memory 8 MB, 24 VDC, memory card slot, interface 1 RS-232/485, display, 2 RJ45 independent onboard Ethernet TCP/IP interfaces with Modbus TCP, web server, IEC60870-5-104 or selectable Ethernet based protocols</td>
<td>Active</td>
</tr>
<tr>
<td>1SAP 331 000 R0278</td>
<td>PM5630-2ETH-XC, processor module, memory 8 MB, 24 VDC, memory card slot, interface 1 RS-232/485, display, 2 RJ45 independent onboard Ethernet TCP/IP interfaces with Modbus TCP, web server, IEC60870-5-104 or selectable Ethernet based protocols, XC version</td>
<td>Active</td>
</tr>
<tr>
<td>1SAP 141 000 R0278</td>
<td>PM5650-2ETH, processor module, memory 80 MB, 24 VDC, memory card slot, interface 1 RS-232/485, display, 2 RJ45 independent onboard Ethernet TCP/IP interfaces with Modbus TCP, web server, IEC60870-5-104 or selectable Ethernet based protocols</td>
<td>Active</td>
</tr>
<tr>
<td>1SAP 341 000 R0278</td>
<td>PM5650-2ETH-XC, processor module, memory 80 MB, 24 VDC, memory card slot, interface 1 RS-232/485, display, 2 RJ45 independent onboard Ethernet TCP/IP interfaces with Modbus TCP, web server, IEC60870-5-104 or selectable Ethernet based protocols, XC version</td>
<td>Active</td>
</tr>
<tr>
<td>1SAP 151 000 R0278</td>
<td>PM5670-2ETH, processor module, memory 160 MB, 24 VDC, memory card slot, interface 1 RS-232/485, display, 2 RJ45 independent onboard Ethernet TCP/IP interfaces with Modbus TCP, web server, IEC60870-5-104 or selectable Ethernet based protocols</td>
<td>Active</td>
</tr>
<tr>
<td>1SAP 351 000 R0278</td>
<td>PM5670-2ETH-XC, processor module, memory 160 MB, 24 VDC, memory card slot, interface 1 RS-232/485, display, 2 RJ45 independent onboard Ethernet TCP/IP interfaces with Modbus TCP, web server, IEC60870-5-104 or selectable Ethernet based protocols, XC version</td>
<td>Active</td>
</tr>
<tr>
<td>Part no.</td>
<td>Description</td>
<td>Product life cycle phase *)</td>
</tr>
<tr>
<td>------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>-----------------------------</td>
</tr>
<tr>
<td>1SAP 151 500 R0278</td>
<td>PM5675-2ETH, processor module, memory 160 MB, 8 GB flash disk, 24 VDC, memory card slot, interface 1 RS-232/485, display, 2 RJ45 independent onboard Ethernet TCP/IP interfaces with Modbus TCP, web server, IEC60870-5-104 or selectable Ethernet based protocols</td>
<td>Active</td>
</tr>
<tr>
<td>1SAP 351 500 R0278</td>
<td>PM5675-2ETH-XC, processor module, memory 160 MB, 8 GB flash disk, 24 VDC, memory card slot, interface 1 RS-232/485, display, 2 RJ45 independent onboard Ethernet TCP/IP interfaces with Modbus TCP, web server, IEC60870-5-104 or selectable Ethernet based protocols, XC version</td>
<td>Active</td>
</tr>
</tbody>
</table>

*) For planning and commissioning of new installations use modules in Active status only.

Table 523: Accessories

<table>
<thead>
<tr>
<th>Part no.</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1SAP 180 300 R0001</td>
<td>TA521, lithium battery</td>
</tr>
<tr>
<td>1SAP 180 100 R0001</td>
<td>MC502, memory card</td>
</tr>
</tbody>
</table>
1.5.3.4 Communication modules (AC500 standard)

1.5.3.4.1 Overview

AC500 communication modules are required for

- a connection to standard field bus systems and
- for integration into existing networks.

AC500 communication modules

- enable communication on different field buses.
- are mounted on the left side of the processor module on the same terminal base.
- are directly powered via the internal communication module bus of the terminal base.

A separate voltage source is not required.
**WARNING!**

**Removal/Insertion under power**

The devices are not designed for removal or insertion under power. Because of unforeseeable consequences, it is not allowed to plug or unplug devices with the power being ON.

Make sure that all voltage sources (supply and process voltage) are switched off before you
- connect or disconnect any signal or terminal block
- remove, mount or replace a module.

Disconnecting any powered devices while energized in a hazardous location could result in an electric arc, which could create a flammable ignition resulting in fire or explosion.

Make sure that power is removed and that the area has been thoroughly checked to ensure that flammable materials are not present prior to proceeding.

The devices must not be opened when in operation. The same applies to the network interfaces.

---

*For information on mounting and demounting, please refer to the chapter mounting and demounting the communication modules Chapter 1.5.4.5.3.5 “Mounting and demounting the communication modules” on page 3075.*

The communication between the processor module and the communication modules takes place via the communication module bus, which is integrated in the terminal base. Depending on the used terminal base up to 6 communication modules can be connected.

- **TB56xx** Chapter 1.5.3.2.1 “TB56xx for AC500 V3 products” on page 2265

There are no restrictions concerning which communication modules can be arranged for a processor module.

Within the AC500 control system, the communication modules can be used as
- bus master or
- slave.

It depends on the
- selected protocol,
- the functionality of the communication module and
- the several field buses and networks.

The following name extensions of the device names describe the supported field bus/protocol:
- CMxyz-ETH: Ethernet
- CMxyz-DP: PROFIBUS
- CMxyz-PNIO: PROFINET
- CMxyz-ETHCAT: EtherCAT
- CMxyz-CN: CANopen
- CMxyz-RCOM: RCOM/RCOM+ protocol (and 2 serial interfaces)
- CMxyz-RS: 2 serial interfaces (COM1/COM2)

If a XC version of the device is available, for use in extreme ambient conditions (e.g. wider temperature and humidity range), this is indicated with a snowflake sign.
Compatibility of communication modules and communication interface modules

### Table 524: Modbus TCP

<table>
<thead>
<tr>
<th>Communication module</th>
<th>Communication interface module</th>
<th>I/O expansion module S500</th>
<th>I/O expansion module S500-eCo</th>
<th>I/O expansion module S500-S</th>
<th>Applications</th>
</tr>
</thead>
<tbody>
<tr>
<td>Onboard Ethernet inter-face</td>
<td>CI521-MODTCP CI522-MODTCP</td>
<td>x</td>
<td>x</td>
<td>--</td>
<td>high availability, remote I/O</td>
</tr>
</tbody>
</table>

### Table 525: PROFINET IO RT

<table>
<thead>
<tr>
<th>Communication module</th>
<th>Communication interface module</th>
<th>I/O expansion module S500</th>
<th>I/O expansion module S500-eCo</th>
<th>I/O expansion module S500-S</th>
<th>Applications</th>
</tr>
</thead>
<tbody>
<tr>
<td>CM579-PNIO controller</td>
<td>CI501-PNIO CI502-PNIO</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>remote I/O, safety I/O</td>
</tr>
<tr>
<td>CM579-PNIO controller</td>
<td>CI501-PNIO CI502-PNIO</td>
<td>x</td>
<td>--</td>
<td>--</td>
<td>hot swap I/O</td>
</tr>
</tbody>
</table>

### Table 526: CANopen

<table>
<thead>
<tr>
<th>Communication module</th>
<th>Communication interface module</th>
<th>I/O expansion module S500</th>
<th>I/O expansion module S500-eCo</th>
<th>I/O expansion module S500-S</th>
<th>Applications</th>
</tr>
</thead>
<tbody>
<tr>
<td>Onboard CAN interface</td>
<td>CI581-CN CI582-CN</td>
<td>--</td>
<td>--</td>
<td>--</td>
<td>remote I/O</td>
</tr>
</tbody>
</table>

### Table 527: EtherCAT

<table>
<thead>
<tr>
<th>Communication module</th>
<th>Communication interface module</th>
<th>I/O expansion module S500</th>
<th>I/O expansion module S500-eCo</th>
<th>I/O expansion module S500-S</th>
<th>Applications</th>
</tr>
</thead>
<tbody>
<tr>
<td>CM579-ETHCAT master</td>
<td>CI511-ETHCAT CI512-ETHCAT</td>
<td>x</td>
<td>x</td>
<td>--</td>
<td>remote I/O</td>
</tr>
</tbody>
</table>
### Technical data (Overview)

<table>
<thead>
<tr>
<th>Communication module</th>
<th>Field bus</th>
<th>Transmission rate</th>
<th>Field bus connector</th>
<th>Processor</th>
<th>Communication module interface</th>
<th>Current consumption from 24 V DC power supply at the terminal base of the CPU</th>
<th>Internal RAM memory</th>
<th>External RAM memory</th>
<th>External flash memory</th>
</tr>
</thead>
<tbody>
<tr>
<td>CM579-ETHCAT</td>
<td>EtherCAT</td>
<td>10 or 100 MBit/s</td>
<td>2 x RJ45</td>
<td>Hilscher netX100</td>
<td>Dual-port memory, 16 kByte</td>
<td>Typ. 85 mA</td>
<td>128 kByte</td>
<td>8 MByte</td>
<td>4 or 8 MByte</td>
</tr>
<tr>
<td>CM598-CN</td>
<td>CANopen</td>
<td>10 ... 1 MBit/s</td>
<td>2x 5-pin, bended</td>
<td>Hilscher netX100</td>
<td>Dual-port memory, 16 kByte</td>
<td>Typ. 65 mA</td>
<td>128 kByte</td>
<td>8 MByte</td>
<td>8 MByte</td>
</tr>
<tr>
<td>CM579-PNIO</td>
<td>PROFINET</td>
<td>100 MBit/s</td>
<td>2 x RJ45</td>
<td>Hilscher netX100</td>
<td>Dual-port memory, 16 kByte</td>
<td>Typ. 85 mA</td>
<td>128 kByte</td>
<td>8 MByte</td>
<td>4 or 8 MByte</td>
</tr>
</tbody>
</table>

#### 1.5.3.4.2 Compatibility of communication modules and communication interface modules

**Table 528: Modbus TCP**

<table>
<thead>
<tr>
<th>Communication module</th>
<th>Communication interface module</th>
<th>I/O expansion module S500</th>
<th>I/O expansion module S500-eCo</th>
<th>I/O expansion module S500-S</th>
<th>Applications</th>
</tr>
</thead>
<tbody>
<tr>
<td>Onboard Ethernet interface</td>
<td>CI521-MODTCP CI522-MODTCP</td>
<td>x</td>
<td>x</td>
<td>--</td>
<td>high availability, remote I/O</td>
</tr>
</tbody>
</table>

**Table 529: PROFINET IO RT**

<table>
<thead>
<tr>
<th>Communication module</th>
<th>Communication interface module</th>
<th>I/O expansion module S500</th>
<th>I/O expansion module S500-eCo</th>
<th>I/O expansion module S500-S</th>
<th>Applications</th>
</tr>
</thead>
<tbody>
<tr>
<td>CM579-PNIO controller</td>
<td>CI501-PNIO CI502-PNIO</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>remote I/O, safety I/O</td>
</tr>
<tr>
<td>CM579-PNIO controller</td>
<td>CI501-PNIO CI502-PNIO</td>
<td>x</td>
<td>--</td>
<td>--</td>
<td>hot swap I/O</td>
</tr>
</tbody>
</table>
### Table 530: CANopen

<table>
<thead>
<tr>
<th>Communication module</th>
<th>Communication interface module</th>
<th>I/O expansion module S500</th>
<th>I/O expansion module S500-eCo</th>
<th>I/O expansion module S500-S</th>
<th>Applications</th>
</tr>
</thead>
<tbody>
<tr>
<td>Onboard CAN interface</td>
<td>CI581-CN CI582-CN</td>
<td>--</td>
<td>--</td>
<td>--</td>
<td>remote I/O</td>
</tr>
</tbody>
</table>

### Table 531: EtherCAT

<table>
<thead>
<tr>
<th>Communication module</th>
<th>Communication interface module</th>
<th>I/O expansion module S500</th>
<th>I/O expansion module S500-eCo</th>
<th>I/O expansion module S500-S</th>
<th>Applications</th>
</tr>
</thead>
<tbody>
<tr>
<td>CM579-ETHCAT master</td>
<td>CI511-ETHCAT CI512-ETHCAT</td>
<td>x</td>
<td>x</td>
<td>--</td>
<td>remote I/O</td>
</tr>
</tbody>
</table>

### 1.5.3.4.3 CANopen

**CM598-CN - CANopen master**

- CANopen master 1 Mbit/s
- XC version for use in extreme ambient conditions available
1 5 LEDs for state display  
2 Label  
3 Communication interface, 5-pin, Combicon, male, removable plug with spring terminals  
Sign for XC version

Purpose

Communication Module CM598-CN enables communication over the CANopen field bus. For use in extreme ambient conditions (e.g. wider temperature and humidity range), a special XC version of the device is available.

The AC500 V3 CPUs only support CAN 2A/2B protocol on the communication module CM598-CAN. Chapter 1.5.6.2.7.1.1.2 "Configuration of the protocols CAN 2.0 A / CAN 2.0 B" on page 3288. Support of CANopen protocol is in preparation.
Electrical connection

Field bus interface

<table>
<thead>
<tr>
<th>Interface socket</th>
<th>5-pin COMBICON</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transmission standard</td>
<td>ISO 11898, potential-free</td>
</tr>
<tr>
<td>Transmission protocol</td>
<td>CANopen (CAN), 1 Mbaud max.</td>
</tr>
<tr>
<td>Transfer rate (transmission rate)</td>
<td>10 kbit/s, 20 kbit/s, 50 kbit/s, 100 kbit/s, 125 kbit/s, 250 kbit/s, 500 kbit/s, 800 kbit/s and 1 Mbit/s,</td>
</tr>
</tbody>
</table>

The CANopen connector has the following pin assignment:

<table>
<thead>
<tr>
<th>Interface</th>
<th>Pin</th>
<th>Signal</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Terminal block removed</td>
<td>1</td>
<td>CAN_GND</td>
<td>CAN reference potential</td>
</tr>
<tr>
<td>Terminal block inserted</td>
<td>2</td>
<td>CAN_L</td>
<td>Bus line, receive/transmit line, LOW</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>CAN_SHLD</td>
<td>Shield of the bus line</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>CAN_H</td>
<td>Bus line, receive/transmit line, HIGH</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>NC</td>
<td>Not connected</td>
</tr>
</tbody>
</table>

NOTICE!

Unused connector!

Make sure that the terminal block is always connected to the terminal base, even if you do not use the interface.

Bus length

The maximum possible bus length of a CAN network depends on bit rate (transmission rate) and cable type. The sum of all bus segments must not exceed the maximum bus length.

<table>
<thead>
<tr>
<th>Bit Rate (speed)</th>
<th>Bus Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Mbit/s</td>
<td>40 m</td>
</tr>
<tr>
<td>800 kbit/s</td>
<td>50 m</td>
</tr>
<tr>
<td>500 kbit/s</td>
<td>100 m</td>
</tr>
<tr>
<td>250 kbit/s</td>
<td>250 m</td>
</tr>
<tr>
<td>125 kbit/s</td>
<td>500 m</td>
</tr>
<tr>
<td>50 kbit/s</td>
<td>1000 m</td>
</tr>
</tbody>
</table>

Types of bus cables

For CANopen, only bus cables with characteristics as recommended in ISO 11898 are to be used. The requirements for the bus cables depend on the length of the bus segment. Regarding this, the following recommendations are given by ISO 11898:
<table>
<thead>
<tr>
<th>Length of segment [m]</th>
<th>Bus cable (shielded, twisted pair)</th>
<th>Max. transmission rate [kbit/s]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Conductor cross section [mm²]</td>
<td>Line resistance [Ω/km]</td>
</tr>
<tr>
<td>0...40</td>
<td>0.25...0.34 / AWG23, AWG22</td>
<td>70</td>
</tr>
<tr>
<td>40...300</td>
<td>0.34...0.60 / AWG22, AWG20</td>
<td>&lt; 60</td>
</tr>
<tr>
<td>300...600</td>
<td>0.50...0.60 / AWG20</td>
<td>&lt; 40</td>
</tr>
<tr>
<td>600...1000</td>
<td>0.75...0.80 / AWG18</td>
<td>&lt; 26</td>
</tr>
</tbody>
</table>

**Bus terminating resistors**

The ends of the data lines have to be terminated with a 120 Ω bus terminating resistor. The bus terminating resistor is usually installed directly at the bus connector.

---

**Fig. 96: CANopen interface, bus terminating resistors connected to the line ends**

<table>
<thead>
<tr>
<th></th>
<th>CANopen interface, bus terminating resistors connected to the line ends</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>CAN_GND</td>
</tr>
<tr>
<td>2</td>
<td>CAN_L</td>
</tr>
<tr>
<td>3</td>
<td>Shield</td>
</tr>
<tr>
<td>4</td>
<td>CAN_H</td>
</tr>
<tr>
<td>5</td>
<td>Data line, shielded twisted pair</td>
</tr>
<tr>
<td>6</td>
<td>COMBICON connection, CANopen interface</td>
</tr>
</tbody>
</table>
The earthing of the shield should take place at the switch-gear. Please refer to Chapter 1.5.4.5.1 “System data AC500” on page 3058.
## State LEDs

### Table 532: Meaning of the diagnosis LEDs

<table>
<thead>
<tr>
<th>LED</th>
<th>Color</th>
<th>State</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PWR</td>
<td>Green</td>
<td>ON (light)</td>
<td>Power supply available</td>
</tr>
<tr>
<td></td>
<td></td>
<td>OFF (dark)</td>
<td>Power supply not available or defective hardware</td>
</tr>
<tr>
<td>RDY</td>
<td>Yellow</td>
<td>ON</td>
<td>Boot procedure</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Blinking</td>
<td>Boot failure</td>
</tr>
<tr>
<td></td>
<td></td>
<td>OFF</td>
<td>---</td>
</tr>
<tr>
<td>RUN</td>
<td>Green</td>
<td>ON</td>
<td>Communication module is operational</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Blinking</td>
<td>---</td>
</tr>
<tr>
<td></td>
<td></td>
<td>OFF</td>
<td>Communication module is not operational</td>
</tr>
<tr>
<td>CAN-RUN</td>
<td>Green</td>
<td>ON</td>
<td>Operational: Device is in the OPERATIONAL state</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Single Flash</td>
<td>Stopped: Device is in STOPPED state</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Blinking</td>
<td>Pre-operational: Device is in the PREOPERATIONAL state</td>
</tr>
<tr>
<td></td>
<td></td>
<td>OFF</td>
<td>No communication or no power supply</td>
</tr>
<tr>
<td>CAN-ERR</td>
<td>Red</td>
<td>ON</td>
<td>CANopen bus is off</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Single flash</td>
<td>Warning limit reached: At least one of the error counters of the CAN controller has reached or exceeded the warning level (too many error frames)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Double flash</td>
<td>Error control event: A guard event (NMT Slave or NMTmaster) or a heartbeat event (Heartbeat consumer) has occurred</td>
</tr>
<tr>
<td></td>
<td></td>
<td>OFF</td>
<td>No Error: Device is in working condition</td>
</tr>
<tr>
<td>CAN-RUN</td>
<td>Yellow</td>
<td>Blinking</td>
<td>(synchronously) No production data available, no bus communication possible.</td>
</tr>
<tr>
<td>CAN-ERR</td>
<td>Yellow</td>
<td>Blinking</td>
<td>(synchronously) Firmware file transfers during communication module firmware update.</td>
</tr>
<tr>
<td>CAN-RUN</td>
<td>Green</td>
<td>Blinking</td>
<td>(alternately) Communication module writes the firmware file to the internal flash. Do not power off the PLC!</td>
</tr>
</tbody>
</table>

### Technical data

The system data of AC500 and S500 are valid for standard version.

The system data of AC500-XC are valid for the XC version.

Only additional details are therefore documented below.

The technical data are also valid for the XC version.
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Protocol</td>
<td>CANopen master (in preparation), CAN2A, CAN2B</td>
</tr>
<tr>
<td>Transmission rate</td>
<td>10 kbit/s to 1 Mbit/s</td>
</tr>
<tr>
<td>Ambient temperature</td>
<td>see: System data AC500 $\S$ Chapter 1.5.4.5.1 “System data AC500” on page 3058 System Data AC500 XC $\S$ Chapter 1.5.4.6.1 “System data AC500-XC” on page 3099</td>
</tr>
<tr>
<td>Usable terminal bases</td>
<td>All TB5xx</td>
</tr>
<tr>
<td>Field bus connector</td>
<td>Pluggable connector COMBICON, 5-pin</td>
</tr>
<tr>
<td>Technology</td>
<td>Hilscher netX100</td>
</tr>
<tr>
<td>Indicators</td>
<td>5 LEDs</td>
</tr>
<tr>
<td>Internal power supply</td>
<td>Via the communication module interface of the terminal base</td>
</tr>
<tr>
<td>Current consumption from 24 VDC power supply at the Terminal Base of the CPU</td>
<td>Typ. 65 mA</td>
</tr>
<tr>
<td>Number of Slaves</td>
<td>Max. 126</td>
</tr>
<tr>
<td>Number of receive/transmit PDOs</td>
<td>Max. 512 (respectively for receive and transmit)</td>
</tr>
<tr>
<td>Total quantity of input and output data</td>
<td>Max. 3584 byte (respectively for input and output)</td>
</tr>
<tr>
<td>Weight</td>
<td>Ca. 150 g</td>
</tr>
</tbody>
</table>

**Ordering data**

<table>
<thead>
<tr>
<th>Part no.</th>
<th>Description</th>
<th>Product life cycle phase *)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1SAP 173 800 R0001</td>
<td>CM598-CN, communication module CANopen master</td>
<td>Active</td>
</tr>
<tr>
<td>1SAP 373 800 R0001</td>
<td>CM598-CN-XC, communication module CANopen master, XC version</td>
<td>Active</td>
</tr>
</tbody>
</table>

*) For planning and commissioning of new installations use modules in Active status only.
1.5.3.4.4 EtherCAT

CM579-ETHCAT - EtherCAT master

1. 5 LEDs for state display
2. 2 rotary switches for address setting (not used)
3. Label
4. 2 communication interfaces RJ45 (ETHCAT1 and ETHCAT2)

**Intended purpose**

Communication module CM579-ETHCAT is for EtherCAT communication. The communication module is configured via the dual-port memory by means of a system configurator. The configuration is saved in a non-volatile way on a flash EPROM.

*Error codes*

**Electrical connection**

**Field bus interfaces**

The EtherCAT communication module provides 2 RJ45 interfaces with the following pin assignment. The pin assignment is used for the EtherCAT slaves (communication interface modules CI5xy-ETHCAT) as well.
### Table 533: Pin assignment RJ45 jack:

<table>
<thead>
<tr>
<th>Interface</th>
<th>Pin</th>
<th>Signal</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
<td>TxD+</td>
<td>Transmit data +</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>TxD-</td>
<td>Transmit data -</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>RxD+</td>
<td>Receive data +</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>NC</td>
<td>not used</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>NC</td>
<td>not used</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>RxD-</td>
<td>Receive data -</td>
</tr>
<tr>
<td></td>
<td>7</td>
<td>NC</td>
<td>not used</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>NC</td>
<td>not used</td>
</tr>
<tr>
<td></td>
<td>Shield</td>
<td>Cable shield</td>
<td>Functional earth</td>
</tr>
</tbody>
</table>

For further information regarding wiring and cable types see chapter Ethernet & Chapter 1.5.4.7 “Ethernet connection details” on page 3084.

The EtherCAT network differentiates between input-connectors (IN) and output-connectors (OUT):

At the EtherCAT slaves (communication interface modules), the ETH1-con- nector is IN and the ETH2-connector is OUT.

At the EtherCAT master (communication module), the ETHCAT1 connector has to be used. The ETHCAT2 connector is reserved for future extensions.
State LEDs

The EtherCAT state is shown by the EtherCAT communication module's LEDs. Some LEDs are two-colored.

**Table 534: Meaning of the diagnosis LEDs**

<table>
<thead>
<tr>
<th>LED</th>
<th>Color</th>
<th>State</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PWR</td>
<td>Green</td>
<td>On</td>
<td>Power supply available</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Blanking</td>
<td>---</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Off</td>
<td>Power supply not available or defective</td>
</tr>
<tr>
<td></td>
<td></td>
<td>hardware</td>
<td></td>
</tr>
<tr>
<td>RDY</td>
<td>Yellow</td>
<td>On</td>
<td>Boot procedure</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Blanking</td>
<td>Boot failure</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Off</td>
<td>---</td>
</tr>
<tr>
<td>RUN</td>
<td>Green</td>
<td>On</td>
<td>Communication module is operational</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Blanking</td>
<td>---</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Off</td>
<td>Communication module is not operational</td>
</tr>
<tr>
<td>STA1</td>
<td>Green</td>
<td>On</td>
<td>No bus error, communication running</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Blanking</td>
<td>Establishing communication</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Off</td>
<td>System error</td>
</tr>
<tr>
<td>STA2</td>
<td>Red</td>
<td>On</td>
<td>Configuration error</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Blanking</td>
<td>---</td>
</tr>
<tr>
<td>STA1</td>
<td>Yellow</td>
<td>Blanking</td>
<td>No production data available, no bus</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(synchronously)</td>
<td>communication possible.</td>
</tr>
<tr>
<td>STA2</td>
<td>Yellow</td>
<td>Off</td>
<td>No error</td>
</tr>
<tr>
<td>ETHCAT1 LED &quot;Link&quot;</td>
<td>Green</td>
<td>On</td>
<td>Ethernet connection established</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Off</td>
<td>No Ethernet connection</td>
</tr>
<tr>
<td>ETHCAT1 LED &quot;RX/TX&quot;</td>
<td>Yellow</td>
<td>On</td>
<td>Device sends/receives frames</td>
</tr>
<tr>
<td>ETHCAT2 LED &quot;Link&quot;</td>
<td>Green</td>
<td>Off</td>
<td>Connector ETHCAT2 is not used</td>
</tr>
<tr>
<td>ETHCAT2 LED &quot;RX/TX&quot;</td>
<td>Yellow</td>
<td>Off</td>
<td>No Ethernet connection</td>
</tr>
</tbody>
</table>

The RJ45 Ethernet connector contains two LEDs showing the current Ethernet port connection state.

**Table 535: Meaning of the diagnosis LEDs**

<table>
<thead>
<tr>
<th>LED</th>
<th>Color</th>
<th>State</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ETHCAT1 LED &quot;Link&quot;</td>
<td>Green</td>
<td>On</td>
<td>Ethernet connection established</td>
</tr>
<tr>
<td>ETHCAT1 LED &quot;RX/TX&quot;</td>
<td>Yellow</td>
<td>On</td>
<td>Device sends/receives frames</td>
</tr>
<tr>
<td>ETHCAT2 LED &quot;Link&quot;</td>
<td>Green</td>
<td>Off</td>
<td>Connector ETHCAT2 is not used</td>
</tr>
<tr>
<td>ETHCAT2 LED &quot;RX/TX&quot;</td>
<td>Yellow</td>
<td>Off</td>
<td>No Ethernet connection</td>
</tr>
</tbody>
</table>
## Technical data

The system data of AC500 and S500 \(\text{\textcopyright Chapter 1.5.4.5.1 "System data AC500" on page 3058}\) are valid for standard version.

The system data of AC500-XC \(\text{\textcopyright Chapter 1.5.4.6.1 "System data AC500-XC" on page 3099}\) are valid for the XC version.

Only additional details are therefore documented below.

The technical data are also valid for the XC version.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Internal Supply</td>
<td>Via the communication module interface of the terminal base</td>
</tr>
<tr>
<td>Protocol</td>
<td>EtherCAT</td>
</tr>
<tr>
<td>Field bus connector</td>
<td>2 x RJ45 (ETHCAT1 and ETHCAT2)</td>
</tr>
<tr>
<td>Technology</td>
<td>Hilscher netX100</td>
</tr>
<tr>
<td>Transfer rate</td>
<td>10/100 Mbit/s (full-duplex)</td>
</tr>
<tr>
<td>Transfer method</td>
<td>According to Ethernet II, IEEE802.3</td>
</tr>
<tr>
<td>Ethernet</td>
<td>100 base-TX, internal switch, 2x RJ45 socket</td>
</tr>
<tr>
<td>Bus length (segment length max.)</td>
<td>100 m at 100 Mbit/s</td>
</tr>
<tr>
<td>Indicators</td>
<td>5 LEDs</td>
</tr>
<tr>
<td>Usable CPUs</td>
<td>PM57x, PM58x, PM59x</td>
</tr>
<tr>
<td></td>
<td>(\text{\textcopyright Chapter 1.5.3.3.1.1 &quot;PM56xx-2ETH for AC500 V3 products&quot; on page 2274})</td>
</tr>
<tr>
<td>Usable terminal bases</td>
<td>All TB5xx</td>
</tr>
<tr>
<td></td>
<td>All (\text{\textcopyright Chapter 1.5.3.2.1 &quot;TB56xx for AC500 V3 products&quot; on page 2265}) (not TB5600)</td>
</tr>
<tr>
<td>Ambient temperature</td>
<td>System data AC500 (\text{\textcopyright Chapter 1.5.4.5.1 &quot;System data AC500&quot; on page 3058})</td>
</tr>
<tr>
<td></td>
<td>System Data AC500 XC (\text{\textcopyright Chapter 1.5.4.6.1 &quot;System data AC500-XC&quot; on page 3099})</td>
</tr>
<tr>
<td>Current consumption from 24 VDC power supply at the terminal base of the CPU</td>
<td>Typ. 85 mA</td>
</tr>
<tr>
<td>Internal supply</td>
<td>Via the communication module interface of the terminal base</td>
</tr>
<tr>
<td>Number of slaves</td>
<td>Limited to 200</td>
</tr>
<tr>
<td>Quantity of input and output data for a single slave</td>
<td>Max. 5760 byte (respectively for input and output)</td>
</tr>
<tr>
<td>Total quantity of input and output data</td>
<td>Max. 5760 byte (respectively for input and output)</td>
</tr>
<tr>
<td>Supported protocols</td>
<td>RTC - Real-time cyclic protocol, class 1</td>
</tr>
<tr>
<td></td>
<td>RTA - Real-time acyclic protocol</td>
</tr>
<tr>
<td>Acyclic services</td>
<td>● CoE upload</td>
</tr>
<tr>
<td></td>
<td>● CoE download (1500 bytes max.)</td>
</tr>
<tr>
<td></td>
<td>● Emergency</td>
</tr>
<tr>
<td>Min. bus cycle</td>
<td>1 ms</td>
</tr>
</tbody>
</table>
### Parameter

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Max. size of the bus configuration file</td>
<td>2 MB</td>
</tr>
<tr>
<td>Weight</td>
<td>Ca. 170 g</td>
</tr>
</tbody>
</table>

### Ordering data

<table>
<thead>
<tr>
<th>Part no.</th>
<th>Description</th>
<th>Product life cycle phase *)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1SAP 170 902 R0101</td>
<td>CM579-ETHCAT, EtherCAT communication module</td>
<td>Active</td>
</tr>
</tbody>
</table>

*) *For planning and commissioning of new installations use modules in Active status only.*

### 1.5.3.4.5 PROFINET

**CM579-PNIO**

- PROFINET I/O controller
- Integrated 2-port switch
- XC version for use in extreme ambient conditions available
1 5 LEDs for state display
2 2 rotary switches for address setting (not used)
3 Label
4 2 communication interfaces RJ45 (PNIO1 and PNIO2)
   Sign for XC version

**Intended purpose**

The communication module is for PROFINET RT communication.

The PROFINET communication module includes an internal Ethernet switch. The connection to the Ethernet can be established directly to the communication module. An additional switch is not necessary.

The communication module is configured via the dual-port memory by means of a system configurator. The configuration is saved in a non-volatile way on a flash EPROM.

For use in extreme ambient conditions (e.g. wider temperature and humidity range), a special XC version of the device is available.
**Functionality**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Protocol</td>
<td>PROFINET I/O RT</td>
</tr>
<tr>
<td>Usable CPUs</td>
<td>PM57x, PM58x, PM59x</td>
</tr>
<tr>
<td></td>
<td><strong>PM56xx</strong> [(Chapter 1.5.3.3.1.1] “PM56xx-2ETH for AC500 V3 products” on page 2274</td>
</tr>
<tr>
<td>Usable terminal bases</td>
<td>All TB5xx</td>
</tr>
<tr>
<td></td>
<td><strong>All TB56xx</strong> [(Chapter 1.5.3.2.1] “TB56xx for AC500 V3 products” on page 2265 (not TB5600)</td>
</tr>
<tr>
<td>Field bus connector</td>
<td>2 RJ45 (PNIO1 and PNIO2), with integrated 2-port switch</td>
</tr>
<tr>
<td>Internal supply</td>
<td>Via the communication module interface of the terminal base</td>
</tr>
</tbody>
</table>

**Electrical connection**

**Field bus interfaces**

The communication module provides 2 RJ45 interfaces.

**Table 536: Pin assignment RJ45 jack:**

<table>
<thead>
<tr>
<th>Interface</th>
<th>Pin</th>
<th>Signal</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ethernet RJ45</td>
<td>1</td>
<td>TxD+</td>
<td>Transmit data +</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>TxD-</td>
<td>Transmit data -</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>RxD+</td>
<td>Receive data +</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>NC</td>
<td>not used</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>NC</td>
<td>not used</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>RxD-</td>
<td>Receive data -</td>
</tr>
<tr>
<td></td>
<td>7</td>
<td>NC</td>
<td>not used</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>NC</td>
<td>not used</td>
</tr>
<tr>
<td></td>
<td>Shield</td>
<td>Cable shield</td>
<td>Functional earth</td>
</tr>
</tbody>
</table>

For further information regarding wiring and cable types see chapter Ethernet [\(Chapter 1.5.4.5.4.7\] “Ethernet connection details” on page 3084.
State LEDs

The PROFINET state is shown by the state LEDs.

Table 537: Meaning of the diagnosis LEDs

<table>
<thead>
<tr>
<th>LED</th>
<th>Color</th>
<th>State</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PWR</td>
<td>Green</td>
<td>On</td>
<td>Power supply available</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Blinking</td>
<td>---</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Off</td>
<td>Power supply not available or defective hardware</td>
</tr>
<tr>
<td>RDY</td>
<td>Yellow</td>
<td>On</td>
<td>Boot procedure</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Blinking</td>
<td>Boot failure</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Off</td>
<td>---</td>
</tr>
<tr>
<td>RUN</td>
<td>Green</td>
<td>On</td>
<td>Communication module is operational</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Blinking</td>
<td>---</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Off</td>
<td>Communication module is not operational</td>
</tr>
<tr>
<td>STA1</td>
<td>Red</td>
<td>On</td>
<td>In incorporation with STA2 PNIO: License fault</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Blinking</td>
<td>System error</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Off</td>
<td>No system error</td>
</tr>
<tr>
<td>STA2</td>
<td>Red</td>
<td>On</td>
<td>No connection; in incorporation with STA1 PNIO: license fault</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Blinking</td>
<td>Configuration fault: some configured IO devices are not connected</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Off</td>
<td>No bus error, communication is running</td>
</tr>
<tr>
<td>STA1</td>
<td>Yellow</td>
<td>Blinking (synchronously)</td>
<td>No production data available, no bus communication possible.</td>
</tr>
<tr>
<td>STA2</td>
<td>Yellow</td>
<td>Blinking (alternately)</td>
<td>Communication module writes the firmware file to the internal flash. Do not power off the PLC!</td>
</tr>
</tbody>
</table>

The RJ45 Ethernet connector contains two LEDs showing the current Ethernet port connection state.

Table 538: Meaning of the diagnosis LEDs

<table>
<thead>
<tr>
<th>LED</th>
<th>Color</th>
<th>State</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PNIO1 LED &quot;Link&quot;</td>
<td>Green</td>
<td>On</td>
<td>Ethernet connection established</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Off</td>
<td>No Ethernet connection</td>
</tr>
<tr>
<td>PNIO1 LED &quot;RX/TX&quot;</td>
<td>Yellow</td>
<td>On</td>
<td>---</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Blinking</td>
<td>Device sends/receives frames</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Off</td>
<td>---</td>
</tr>
<tr>
<td>PNIO2 LED &quot;Link&quot;</td>
<td>Green</td>
<td>On</td>
<td>Ethernet connection established</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Off</td>
<td>No Ethernet connection</td>
</tr>
</tbody>
</table>
### Technical data

The system data of AC500 and S500 [Chapter 1.5.4.5.1 “System data AC500” on page 3058](#) are valid for standard version.

The system data of AC500-XC [Chapter 1.5.4.6.1 “System data AC500-XC” on page 3099](#) are valid for the XC version.

Only additional details are therefore documented below.

The technical data are also valid for the XC version.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Protocol</td>
<td>PROFINET I/O RT</td>
</tr>
<tr>
<td>Bus connection</td>
<td>2 RJ45 (PNIO1 and PNIO2), with integrated 2-port switch</td>
</tr>
<tr>
<td>Switch</td>
<td>Integrated</td>
</tr>
<tr>
<td>Technology</td>
<td>Hilscher netX100</td>
</tr>
<tr>
<td>Transfer rate</td>
<td>100 Mbit/s (full-duplex)</td>
</tr>
<tr>
<td>Transfer method</td>
<td>According to Ethernet II, IEE802.3</td>
</tr>
<tr>
<td>Ethernet</td>
<td>100 base-TX, internal switch, 2x RJ45 socket</td>
</tr>
<tr>
<td>Bus length (segment length max.)</td>
<td>100 m</td>
</tr>
<tr>
<td>Indicators</td>
<td>5 LEDs</td>
</tr>
<tr>
<td>Usable terminal bases</td>
<td>All TB5xx</td>
</tr>
<tr>
<td></td>
<td>All TB56xx <a href="#">Chapter 1.5.3.2.1 “TB56xx for AC500 V3 products” on page 2265</a> (not TB5600)</td>
</tr>
<tr>
<td>Supported alarm types</td>
<td>Process alarm, diagnostic alarm, return of Sub-Module, plug alarm, pull alarm</td>
</tr>
<tr>
<td>Alarm processing</td>
<td>Requires handling in application program</td>
</tr>
<tr>
<td>Current consumption from 24 VDC power supply at the terminal base of the CPU</td>
<td>Typ. 85 mA</td>
</tr>
<tr>
<td>Internal supply</td>
<td>Via the communication module interface of the terminal base</td>
</tr>
<tr>
<td>Weight</td>
<td>Ca. 170 g</td>
</tr>
<tr>
<td>Supported protocols</td>
<td>RTC - real-time cyclic protocol, class 1</td>
</tr>
<tr>
<td></td>
<td>RTA - real-time acyclic protocol</td>
</tr>
<tr>
<td></td>
<td>DCP - discovery and configuration protocol *)</td>
</tr>
<tr>
<td></td>
<td>CL-RPC - connectionless remote procedure call</td>
</tr>
<tr>
<td></td>
<td>Since revision FW 2.4.8.0 additionally</td>
</tr>
<tr>
<td></td>
<td>LLDP - link layer discovery protocol</td>
</tr>
<tr>
<td></td>
<td>SNMP - simply network management protocol (SNMP v1)</td>
</tr>
<tr>
<td>Parameter</td>
<td>Value</td>
</tr>
<tr>
<td>-----------------------------------------------</td>
<td>----------------------------------------------------------------------</td>
</tr>
<tr>
<td>Acyclic services</td>
<td>PNIO read / write (max. 1392 bytes per telegram, max. 4096 bytes per service request)</td>
</tr>
<tr>
<td>Total quantity of input and output data</td>
<td></td>
</tr>
<tr>
<td>CM579-PNIO &lt; FW 2.4.8.0</td>
<td>1024 bytes per IO device</td>
</tr>
<tr>
<td></td>
<td>3072 bytes in total</td>
</tr>
<tr>
<td>CM579-PNIO = FW 2.4.8.0</td>
<td>1024 bytes per IO device</td>
</tr>
<tr>
<td></td>
<td>4096 bytes in total</td>
</tr>
<tr>
<td>CM579-PNIO &gt; FW 2.4.8.0</td>
<td>1440 bytes per IO device</td>
</tr>
<tr>
<td></td>
<td>PM5630, PM5650: 4096 bytes in total</td>
</tr>
<tr>
<td></td>
<td>PM567x: 5120 bytes in total</td>
</tr>
<tr>
<td>Min. bus cycle</td>
<td>1 ms</td>
</tr>
<tr>
<td>Conformance class</td>
<td>CC A</td>
</tr>
</tbody>
</table>

*) CM579-PNIO does not allow setting "Station name" by using PROFINET service "DCP SET NameOfStation".

## Ordering data

<table>
<thead>
<tr>
<th>Part no.</th>
<th>Description</th>
<th>Product life cycle phase *)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1SAP 170 901 R0101</td>
<td>CM579-PNIO, PROFINET communication module</td>
<td>Active</td>
</tr>
<tr>
<td>1SAP 370 901 R0101</td>
<td>CM579-PNIO-XC, PROFINET communication module, XC version</td>
<td>Active</td>
</tr>
</tbody>
</table>

*) For planning and commissioning of new installations use modules in Active status only.
1.5.3.5 Terminal units (AC500 standard)

**Hot swap**

System requirements for hot swapping of I/O modules:
- Types of terminal units that support hot swapping of I/O modules have the appendix TU5xx-H.
- I/O modules as of index F0.

The following I/O bus masters support hot swapping of attached I/O modules:
- Communication interface modules CI5xx as of index F0.
- Processor modules PM56xx-2ETH with firmware version as of V3.2.0.

**NOTICE!**

Risk of damage to I/O modules!
Hot swapping is only allowed for I/O modules.
Processor modules and communication interface modules must not be removed or inserted during operation.

**Conditions for hot swapping**
- Digital outputs are not under load.
- Input/output voltages above safety extra low voltage/protective extra low voltages (SELV/PELV) are switched off.
- Modules are completely plugged on the terminal unit with both snap fit engaged before switching on loads or input/output voltage.

Further information about hot swap: **System technology** Chapter 1.5.5.1.6 “Hot swap” on page 3148.

1.5.3.5.1 TU507-ETH and TU508-ETH for Ethernet communication interface modules

- TU507-ETH, Ethernet terminal unit, 24 VDC, screw terminals
- TU508-ETH, Ethernet terminal unit, 24 VDC, spring terminals
- TU508-ETH-XC, Ethernet terminal unit, 24 VDC, spring terminals, XC version
1 I/O bus (10 pins, female) to electrically connect the first terminal unit
2a Plug (2x 25 pins) to electrically connect the inserted Ethernet communication interface module
2b Plug (3x 19 pins) to electrically connect the inserted Ethernet communication interface module
3 With a screwdriver, inserted in this place, the terminal unit and the adjacent terminal unit can be shoved from each other
4 2 holes for wall mounting
5 2 RJ45 interfaces with indication LEDs for connection with the Ethernet network
6 30 terminals for signals and process supply voltages (UP and UP3)
7 DIN rail

The Ethernet communication interface modules plug into the Ethernet terminal unit. When properly seated, they are secured with two mechanical locks. All the electrical connections are made through the Ethernet terminal unit, which allows removal and replacement of the Ethernet communication interface modules without disturbing the wiring at the Ethernet terminal unit.

The Ethernet terminal units TU507-ETH and TU508-ETH are specifically designed for use with AC500/S500 Ethernet communication interface modules (e.g. CI501-PNIO).
**XC version**

**XC = eXtreme Conditions**

![Image](image_url)

**Extreme conditions**

Terminal units for use in extreme ambient conditions have no sign for XC version. The figure 4 in the Part no. 1SAP4... (label) identifies the XC version.

**Terminals**

<table>
<thead>
<tr>
<th>Screw terminals</th>
<th>Conductor</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.5</td>
<td>1.6</td>
</tr>
<tr>
<td>1.6</td>
<td>1.7</td>
</tr>
<tr>
<td>1.7</td>
<td>1.8</td>
</tr>
<tr>
<td>1.8</td>
<td>1.9</td>
</tr>
</tbody>
</table>

**Screwdriver**

<table>
<thead>
<tr>
<th>1.5</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.6</td>
</tr>
<tr>
<td>1.7</td>
</tr>
<tr>
<td>1.8</td>
</tr>
<tr>
<td>1.9</td>
</tr>
</tbody>
</table>

**Spring terminals**

<table>
<thead>
<tr>
<th>Conductor</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.5</td>
</tr>
<tr>
<td>1.6</td>
</tr>
<tr>
<td>1.7</td>
</tr>
<tr>
<td>1.8</td>
</tr>
<tr>
<td>1.9</td>
</tr>
</tbody>
</table>

**Screwdriver (opens terminal)**

For information about wiring specifications see the description of the Terminal Units % Chapter 1.5.4.5.4.3 “Terminals at the terminal unit” on page 3078.

For a detailed description of the mounting, disassembly and electrical connection of the terminal units and the I/O modules, please refer to the “System Assembly, Construction and Connection” chapter % Chapter 1.5.4.5.3 “Mounting and demounting” on page 3068.

The terminals 1.8 and 2.8 as well as 1.9, 2.9 and 3.9 are electrically interconnected within the terminal unit and have always the same assignment, independent of the inserted module:

- Terminals 1.8 and 2.8: Process supply voltage UP = +24 VDC
- Terminal 3.8: Process supply voltage UP3 = +24 VDC
- Terminals 1.9, 2.9 and 3.9: Process supply voltage ZP = 0 V

The assignment of the other terminals is dependent on the inserted communication interface module.
NOTICE!
Risk of corrosion!
Unused connectors and slots may corrode if XC devices are used in salt-mist environments.
Protect unused connectors and slots with TA535 protective caps for XC devices TA535 Chapter 1.5.3.8.2.4 “TA535 - Protective caps for XC devices” on page 3039.

Technical data

The system data of AC500 and S500 Chapter 1.5.4.5.1 “System data AC500” on page 3058 are valid for standard version.

The system data of AC500-XC Chapter 1.5.4.6.1 “System data AC500-XC” on page 3099 are valid for the XC version.

Only additional details are therefore documented below.
The technical data are also valid for the XC version.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of I/O channels per module</td>
<td>Max. 24 (depending on the inserted bus module)</td>
</tr>
<tr>
<td>Distribution of the channels into groups</td>
<td>3 groups of max. 8 channels each (1.0...1.7, 2.0...2.7, 3.0...3.7), the allocation of the channels is given by the inserted Ethernet bus module</td>
</tr>
<tr>
<td>Network interface connector</td>
<td>2 RJ45, 8-pole</td>
</tr>
<tr>
<td>Rated voltage</td>
<td>24 VDC</td>
</tr>
<tr>
<td>Max. permitted total current</td>
<td>10 A via the supply terminals (UP, UP3 and ZP)</td>
</tr>
<tr>
<td>Ethernet</td>
<td>10/100 base-TX or 100 base-TX (depending on CI5xx module plugged in), 2 RJ45 socket</td>
</tr>
<tr>
<td>Earthing</td>
<td>Direct connection to the earthed DIN rail or via the screws with wall mounting</td>
</tr>
<tr>
<td>Screw terminals</td>
<td>Front terminal, conductor connection vertically with respect to the printed circuit board</td>
</tr>
<tr>
<td>Spring-type terminals</td>
<td>Front terminal, conductor connection vertically with respect to the printed circuit board</td>
</tr>
<tr>
<td>Weight</td>
<td>200 g</td>
</tr>
<tr>
<td>Mounting position</td>
<td>Horizontal or vertical</td>
</tr>
</tbody>
</table>
Ordering data

<table>
<thead>
<tr>
<th>Part no.</th>
<th>Description</th>
<th>Product life cycle phase *)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1SAP 214 200 R0001</td>
<td>TU507-ETH, Ethernet terminal unit, 24 VDC, screw terminals</td>
<td>Active</td>
</tr>
<tr>
<td>1SAP 214 000 R0001</td>
<td>TU508-ETH, Ethernet terminal unit, 24 VDC, spring terminals</td>
<td>Active</td>
</tr>
<tr>
<td>1SAP 414 000 R0001</td>
<td>TU508-ETH-XC, Ethernet terminal unit, 24 VDC, spring terminals, XC version</td>
<td>Active</td>
</tr>
</tbody>
</table>

*) For planning and commissioning of new installations use modules in Active status only.

1.5.3.5.2 TU515, TU516, TU541 and TU542 for I/O modules

- TU515, I/O terminal unit, 24 VDC, screw terminals
- TU516, I/O terminal unit, 24 VDC, spring terminals
- TU516-XC, I/O terminal unit, 24 VDC, spring terminals, XC version
- TU516-H, I/O terminal unit, hot swap, 24 VDC, spring terminals
- TU516-H-XC, I/O terminal unit, hot swap, 24 VDC, spring terminals, XC version
- TU541, I/O terminal unit, 24 VDC, screw terminals
- TU542, I/O terminal unit, 24 VDC, spring terminals
- TU542-XC, I/O terminal unit, 24 VDC, spring terminals, XC version
- TU542-H, I/O terminal unit, hot swap, 24 VDC, spring terminals
- TU542-H-XC, I/O terminal unit, hot swap, 24 VDC, spring terminals, XC version

The input/output modules plug into the I/O terminal unit. When properly seated, they are secured with two mechanical locks. All the electrical connections are established via the terminal unit, which allows removal and replacement of the I/O modules without disturbing the wiring at the terminal unit.
1. I/O bus (10 pins, male) to electrically connect the previous terminal unit, the CPU terminal base or the communication interface module to the terminal unit.
2. I/O bus (10 pins, female) to electrically connect other terminal units.
3a. Plug (2x25 pins) to electrically connect the inserted I/O modules.
3b. Plug (2x19 pins) to electrically connect the inserted I/O modules.
4. With a screwdriver inserted in this place, the terminal unit and the adjacent terminal unit can be shoved from each other.
5. Holes for wall mounting.
6. 40 terminals for signals and process supply voltage.
7. DIN rail.
8. White border signifies hot swap capability of the terminal unit.

**WARNING!**
Risk of explosion or fire in hazardous environments during hot swapping!
Hot swap must not be performed in flammable environments to avoid life-threatening injury and property damage resulting from fire or explosion.
WARNING!
Electric shock due to negligent behavior during hot swapping!
To avoid electric shock
– make sure the following conditions apply:
  – Digital outputs are not under load.
  – Input/output voltages above safety extra low voltage/
    protective extra low voltage (SELV/PELV) are switched off.
  – Modules are fully interlocked with the terminal unit with both snap-fits
    engaged before switching on loads or input/output voltage.
  – Never touch exposed contacts (dangerous voltages).
  – Stay away from electrical contacts to avoid arc discharge.
  – Do not operate a mechanical installation improperly.

NOTICE!
Risk of damage to I/O modules!
Hot swapping is only allowed for I/O modules.
Processor modules and communication interface modules must not be removed
or inserted during operation.

H = Hot swap

Hot swap
System requirements for hot swapping of I/O modules:
– Types of terminal units that support hot swapping of I/O modules have the
  appendix TU5xx-H.
– I/O modules as of index F0.
The following I/O bus masters support hot swapping of attached I/O modules:
– Communication interface modules CI5xx as of index F0.
– Processor modules PM56xx-2ETH with firmware version as of V3.2.0.

The index of the module is in the right corner of the label.
NOTICE!
Risk of damage to I/O modules!
Modules with index below F0 can be damaged when inserted or removed from the terminal unit in a powered system.

NOTICE!
Risk of damage to I/O modules!
Do not perform hot swapping if any I/O module with firmware version lower than 3.0.14 is part of the I/O configuration.
For min. required device index see table below.

<table>
<thead>
<tr>
<th>Device</th>
<th>Min. required device index for I/O module as of FW Version 3.0.14</th>
</tr>
</thead>
<tbody>
<tr>
<td>AI523 (-XC)</td>
<td>D2</td>
</tr>
<tr>
<td>AI531</td>
<td>D4</td>
</tr>
<tr>
<td>AI531-XC</td>
<td>D2</td>
</tr>
<tr>
<td>AI561</td>
<td>B2</td>
</tr>
<tr>
<td>AI562</td>
<td>B2</td>
</tr>
<tr>
<td>AI563</td>
<td>B3</td>
</tr>
<tr>
<td>AO523 (-XC)</td>
<td>D2</td>
</tr>
<tr>
<td>AO561</td>
<td>B2</td>
</tr>
<tr>
<td>AX521 (-XC)</td>
<td>D2</td>
</tr>
<tr>
<td>AX522 (-XC)</td>
<td>D2</td>
</tr>
<tr>
<td>AX561</td>
<td>B2</td>
</tr>
<tr>
<td>CD522 (-XC)</td>
<td>D1</td>
</tr>
<tr>
<td>DA501 (-XC)</td>
<td>D2</td>
</tr>
<tr>
<td>DC522 (-XC)</td>
<td>D2</td>
</tr>
<tr>
<td>DC523 (-XC)</td>
<td>D2</td>
</tr>
<tr>
<td>DC532 (-XC)</td>
<td>D2</td>
</tr>
<tr>
<td>DC561</td>
<td>B2</td>
</tr>
<tr>
<td>DC562</td>
<td>A2</td>
</tr>
<tr>
<td>DI524 (-XC)</td>
<td>D2</td>
</tr>
<tr>
<td>DI561</td>
<td>B2</td>
</tr>
<tr>
<td>DI562</td>
<td>B2</td>
</tr>
<tr>
<td>DI571</td>
<td>B2</td>
</tr>
<tr>
<td>DI572</td>
<td>A1</td>
</tr>
<tr>
<td>DO524 (-XC)</td>
<td>A3</td>
</tr>
<tr>
<td>DO526</td>
<td>A2</td>
</tr>
<tr>
<td>DO526-XC</td>
<td>A0</td>
</tr>
<tr>
<td>DO561</td>
<td>B2</td>
</tr>
<tr>
<td>DO562</td>
<td>A2</td>
</tr>
</tbody>
</table>
### Device

<table>
<thead>
<tr>
<th>Device</th>
<th>Min. required device index for I/O module as of FW Version 3.0.14</th>
</tr>
</thead>
<tbody>
<tr>
<td>DO571</td>
<td>B3</td>
</tr>
<tr>
<td>DO572</td>
<td>B2</td>
</tr>
<tr>
<td>DO573</td>
<td>A1</td>
</tr>
<tr>
<td>DX522 (-XC)</td>
<td>D2</td>
</tr>
<tr>
<td>DX531</td>
<td>D2</td>
</tr>
<tr>
<td>DX561</td>
<td>B2</td>
</tr>
<tr>
<td>DX571</td>
<td>B3</td>
</tr>
<tr>
<td>FM562</td>
<td>A1</td>
</tr>
</tbody>
</table>

#### XC version

**XC = eXtreme Conditions**

### Extreme conditions

Terminal units for use in extreme ambient conditions have no sign for XC version.

The figure 4 in the Part no. 1SAP... (lable) identifies the XC version.

### Terminals

#### Screw terminals

<table>
<thead>
<tr>
<th>Conductor</th>
<th>Screwdriver</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.5</td>
<td></td>
</tr>
<tr>
<td>1.6</td>
<td></td>
</tr>
<tr>
<td>1.7</td>
<td></td>
</tr>
<tr>
<td>1.8</td>
<td></td>
</tr>
<tr>
<td>1.9</td>
<td></td>
</tr>
</tbody>
</table>

#### Spring terminals

<table>
<thead>
<tr>
<th>Conductor</th>
<th>Screwdriver (opens terminal)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.5</td>
<td></td>
</tr>
<tr>
<td>1.6</td>
<td></td>
</tr>
<tr>
<td>1.7</td>
<td></td>
</tr>
<tr>
<td>1.8</td>
<td></td>
</tr>
<tr>
<td>1.9</td>
<td></td>
</tr>
</tbody>
</table>

For information about wiring specifications see the description of the Terminal Units (Chapter 1.5.4.5.4.3 “Terminals at the terminal unit” on page 3078).

For a detailed description of the mounting, disassembly and electrical connection of the terminal units and the I/O modules, please refer to the “System Assembly, Construction and Connection” chapter (Chapter 1.5.4.5.3 “Mounting and demounting” on page 3068).
The following terminals are used for connection of the process supply voltage.

<table>
<thead>
<tr>
<th>Type</th>
<th>1.8</th>
<th>2.8</th>
<th>3.8</th>
<th>4.8</th>
<th>1.9</th>
<th>2.9</th>
<th>3.9</th>
<th>4.9</th>
</tr>
</thead>
<tbody>
<tr>
<td>TU515, TU516 and TU516-H</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>These terminals are internally connected with assignment: process supply voltage UP = +24 VDC</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TU541, TU542 and TU542-H</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>These terminals are internally connected with assignment: process voltage UP = +24 VDC</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Separate process supply voltage UP3 = +24 VDC</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Separate process supply voltage UP4 = +24 VDC</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>These terminals are internally connected with assignment: process supply voltage ZP = 0 V</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Separate process supply voltage ZP = 0 V</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Separate process supply voltage ZP = 0 V</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The assignment of the other terminals depends on the inserted decentralized communication interface module (see the description of the respective module used).

### Technical data

The system data of AC500 and S500  Chapter 1.5.4.5.1 “System data AC500” on page 3058 are valid for standard version.

The system data of AC500-XC  Chapter 1.5.4.6.1 “System data AC500-XC” on page 3099 are valid for the XC version.

Only additional details are therefore documented below.

The technical data are also valid for the XC version.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of channels per module</td>
<td>Max. 32</td>
</tr>
<tr>
<td>Distribution of the channels into groups</td>
<td>4 groups of 8 channels each (1.0...1.7, 2.0...2.7, 3.0...3.7, 4.0...4.7), the allocation of the channels is given by the inserted I/O module</td>
</tr>
<tr>
<td>Rated voltage</td>
<td>24 VDC</td>
</tr>
<tr>
<td>Max. permitted total current</td>
<td>10 A, per separated process voltage terminal or for internal connection of process voltages</td>
</tr>
<tr>
<td>Earthing</td>
<td>Direct connection to the earthed DIN rail or via the screws with wall mounting</td>
</tr>
<tr>
<td>Screw terminals</td>
<td>Front terminal, conductor connection vertically with respect to the printed circuit board</td>
</tr>
<tr>
<td>Spring terminals</td>
<td>Front terminal, conductor connection vertically with respect to the printed circuit board</td>
</tr>
<tr>
<td>Weight</td>
<td>200 g</td>
</tr>
<tr>
<td>Mounting position</td>
<td>Horizontal or vertical</td>
</tr>
</tbody>
</table>
## Ordering data

<table>
<thead>
<tr>
<th>Part no.</th>
<th>Description</th>
<th>Product life cycle phase *)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1SAP 212 200 R0001</td>
<td>TU515, I/O terminal unit, 24 VDC, screw terminals</td>
<td>Active</td>
</tr>
<tr>
<td>1SAP 212 000 R0001</td>
<td>TU516, I/O terminal unit, 24 VDC, spring terminals</td>
<td>Active</td>
</tr>
<tr>
<td>1SAP 412 000 R0001</td>
<td>TU516-XC, I/O terminal unit, 24 VDC, spring terminals, XC version</td>
<td>Active</td>
</tr>
<tr>
<td>1SAP 215 000 R0001</td>
<td>TU516-H, I/O terminal unit, hot swap, 24 VDC, spring terminals, XC version</td>
<td>Active</td>
</tr>
<tr>
<td>1SAP 415 000 R0001</td>
<td>TU516-H-XC, I/O terminal unit, hot swap, 24 VDC, spring terminals, XC version</td>
<td>Active</td>
</tr>
<tr>
<td>1SAP 213 000 R0001</td>
<td>TU541, I/O terminal unit, 24 VDC, screw terminals</td>
<td>Active</td>
</tr>
<tr>
<td>1SAP 213 200 R0001</td>
<td>TU542, I/O terminal unit, 24 VDC, spring terminals</td>
<td>Active</td>
</tr>
<tr>
<td>1SAP 413 200 R0001</td>
<td>TU542-XC, I/O terminal unit, 24 VDC, spring terminals, XC version</td>
<td>Active</td>
</tr>
<tr>
<td>1SAP 215 200 R0001</td>
<td>TU542-H, I/O terminal unit, hot swap, 24 VDC, spring terminals</td>
<td>Active</td>
</tr>
<tr>
<td>1SAP 415 200 R0001</td>
<td>TU542-H-XC, I/O terminal unit, hot swap, 24 VDC, spring terminals, XC version</td>
<td>Active</td>
</tr>
</tbody>
</table>

*) For planning and commissioning of new installations use modules in Active status only.

### 1.5.3.3 TU517 and TU518 for communication interface modules
- TU517, terminal unit, 24 VDC, screw terminals
- TU518, terminal unit, 24 VDC, spring terminals
- TU518-XC, terminal unit, 24 VDC, spring terminals, XC version
1 I/O bus (10 pins, female) to electrically connect the first terminal unit
2a Plug (2 25 pins) to electrically connect the inserted communication interface module
2b Plug (2 19 pins) to electrically connect the inserted communication interface module
3 With a screwdriver, inserted in this place, the terminal unit and the adjacent I/O terminal unit can be shoved from each other
4 2 holes for wall mounting
5 10 terminals for connection with the bus system
6 30 terminals for signals and process supply voltages (UP and UP3)
7 DIN rail

The communication interface modules plug into the terminal unit. When properly plugged-in, they are secured with two mechanical locks. All the electrical connections are established via the terminal unit, which allows removal and replacement of the communication interface modules without disturbing the wiring at the terminal unit.

The terminal units TU517 and TU518 are specifically designed for use with AC500/S500 communication interface modules (e.g. CI581-CN, CI541-DP):

- CANopen communication interface modules
- DeviceNet modules
- PROFIBUS DP communication interface modules
Extreme conditions
Terminal units for use in extreme ambient conditions have no sign for XC version.
The figure 4 in the Part no. 1SAP4... (label) identifies the XC version.

## Terminals

<table>
<thead>
<tr>
<th>Screw terminals</th>
<th>Spring terminals</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Conductor</strong></td>
<td><strong>Screwdriver</strong></td>
</tr>
<tr>
<td>1.5</td>
<td></td>
</tr>
<tr>
<td>1.6</td>
<td></td>
</tr>
<tr>
<td>1.7</td>
<td></td>
</tr>
<tr>
<td>1.8</td>
<td></td>
</tr>
<tr>
<td>1.9</td>
<td></td>
</tr>
<tr>
<td><strong>Conductor</strong></td>
<td><strong>Screwdriver</strong></td>
</tr>
<tr>
<td></td>
<td>(opens termi-</td>
</tr>
<tr>
<td></td>
<td>nal)</td>
</tr>
</tbody>
</table>

For information about wiring specifications see the description of the Terminal Units % Chapter 1.5.4.5.4.3 “Terminals at the terminal unit” on page 3078.

For a detailed description of the mounting, disassembly and electrical connection of the terminal units and the I/O modules, please refer to the “System Assembly, Construction and Connection” chapter % Chapter 1.5.4.5.3 “Mounting and demounting” on page 3068.

The terminals 2.8, 3.8, 2.9, 3.9 and 4.9 are electrically interconnected within the terminal unit and always have the same assignment, irrespective of the inserted communication interface module:

- Terminals 2.8 and 3.8: process supply voltage UP = +24 VDC
- Terminal 4.8: process supply voltage UP3 = +24 VDC
- Terminals 2.9, 3.9 and 4.9: process supply voltage ZP = 0 V

The assignment of the other terminals depends on the inserted communication interface module (see communication interface modules for CANopen and PROFIBUS).
Technical data

The system data of AC500 and S500 (Chapter 1.5.4.5.1 “System data AC500” on page 3058) are valid for the standard version.

The system data of AC500-XC (Chapter 1.5.4.6.1 “System data AC500-XC” on page 3099) are valid for the XC version.

Only additional details are therefore documented below.

The technical data are also valid for the XC version.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of I/O channels per module</td>
<td>Max. 24 (depending on the inserted bus module)</td>
</tr>
<tr>
<td>Distribution of the channels into groups</td>
<td>3 groups of max. 8 channels each (2.0…2.7, 3.0…3.7, 4.0…4.7), the allocation of the channels is given by the inserted bus module</td>
</tr>
<tr>
<td>Network interface connector</td>
<td>10 screw or spring terminals (1.0…1.9)</td>
</tr>
<tr>
<td>Rated voltage</td>
<td>24 VDC</td>
</tr>
<tr>
<td>Max. permitted total current</td>
<td>10 A via the supply terminals (UP, UP3 and ZP)</td>
</tr>
<tr>
<td>Earthing</td>
<td>Direct connection to the earthed DIN rail or via the screws with wall mounting</td>
</tr>
<tr>
<td>Screw terminals</td>
<td>Front terminal, conductor connection vertically with respect to the printed circuit board</td>
</tr>
<tr>
<td>Spring terminals</td>
<td>Front terminal, conductor connection vertically with respect to the printed circuit board</td>
</tr>
<tr>
<td>Weight</td>
<td>200 g</td>
</tr>
<tr>
<td>Mounting position</td>
<td>Horizontal or vertical</td>
</tr>
</tbody>
</table>

Ordering data

<table>
<thead>
<tr>
<th>Part no.</th>
<th>Description</th>
<th>Product life cycle phase *)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1SAP 211 400 R0001</td>
<td>TU517, terminal unit, 24 VDC, screw terminals</td>
<td>Active</td>
</tr>
<tr>
<td>1SAP 211 200 R0001</td>
<td>TU518, terminal unit, 24 VDC, spring terminals</td>
<td>Active</td>
</tr>
<tr>
<td>1SAP 411 200 R0001</td>
<td>TU518-XC, terminal unit, 24 VDC, spring terminals, XC version</td>
<td>Active</td>
</tr>
</tbody>
</table>

*) For planning and commissioning of new installations use modules in Active status only.

1.5.3.5.4 TU531 and TU532 for I/O modules

- TU531, I/O terminal unit, 230 VAC, screw terminals
- TU532, I/O terminal unit, 230 VAC, spring terminals
- TU532-XC, I/O terminal unit, 230 VAC, spring terminals, XC version
- TU532-H, I/O terminal unit, hot swap, 230 VAC, spring terminals
- TU532-H-XC, I/O terminal unit, hot swap, 230 VAC, spring terminals, XC version

1  I/O bus (10 pins, male) to electrically connect the previous terminal unit, the CPU terminal base or the communication interface module to the terminal unit
2  I/O bus (10 pins, female) to electrically connect other terminal units
3a Plug (2 25 pins) to electrically connect the inserted I/O modules
3b Plug (3 19 pins) to electrically connect the inserted I/O modules
4  With a screwdriver inserted in this place, the terminal unit and the adjacent I/O terminal unit can be shoved from each other
5  2 holes for wall mounting
6  40 terminals for signals and process supply voltage
7  DIN rail
8  White boarder signifies hot swap capability of the terminal unit

The input/output modules (I/O modules) plug into the I/O terminal unit. When properly plugged-in, they are secured with two mechanical locks. All the electrical connections are established via the terminal unit, which allows removal and replacement of the I/O modules without disturbing the wiring at the terminal unit.

The terminal units TU531 and TU532 are specifically designed for use with AC500/S500 I/O modules that incorporate 115-230 VAC inputs and/or 230 VAC relay outputs.

**WARNING!**
Risk of explosion or fire in hazardous environments during hot swapping!
Hot swap must not be performed in flammable environments to avoid life-threatening injury and property damage resulting from fire or explosion.
WARNING!
Electric shock due to negligent behavior during hot swapping!
To avoid electric shock
– make sure the following conditions apply:
  – Digital outputs are not under load.
  – Input/output voltages above safety extra low voltage/protective extra low voltage (SELV/PELV) are switched off.
  – Modules are fully interlocked with the terminal unit with both snap-fits engaged before switching on loads or input/output voltage.
  – Never touch exposed contacts (dangerous voltages).
  – Stay away from electrical contacts to avoid arc discharge.
  – Do not operate a mechanical installation improperly.

NOTICE!
Risk of damage to I/O modules!
Hot swapping is only allowed for I/O modules.
Processor modules and communication interface modules must not be removed or inserted during operation.

H = Hot swap

Hot swap
System requirements for hot swapping of I/O modules:
– Types of terminal units that support hot swapping of I/O modules have the appendix TU5xx-H.
– I/O modules as of index F0.
The following I/O bus masters support hot swapping of attached I/O modules:
– Communication interface modules CI5xx as of index F0.
– Processor modules PM56xx-2ETH with firmware version as of V3.2.0.

The index of the module is in the right corner of the label.
**NOTICE!**
Risk of damage to I/O modules!
Modules with index below F0 can be damaged when inserted or removed from the terminal unit in a powered system.

**NOTICE!**
Risk of damage to I/O modules!
Do not perform hot swapping if any I/O module with firmware version lower than 3.0.14 is part of the I/O configuration.
For min. required device index see table below.

<table>
<thead>
<tr>
<th>Device</th>
<th>Min. required device index for I/O module as of FW Version 3.0.14</th>
</tr>
</thead>
<tbody>
<tr>
<td>AI523 (-XC)</td>
<td>D2</td>
</tr>
<tr>
<td>AI531</td>
<td>D4</td>
</tr>
<tr>
<td>AI531-XC</td>
<td>D2</td>
</tr>
<tr>
<td>AI561</td>
<td>B2</td>
</tr>
<tr>
<td>AI562</td>
<td>B2</td>
</tr>
<tr>
<td>AI563</td>
<td>B3</td>
</tr>
<tr>
<td>AO523 (-XC)</td>
<td>D2</td>
</tr>
<tr>
<td>AO561</td>
<td>B2</td>
</tr>
<tr>
<td>AX521 (-XC)</td>
<td>D2</td>
</tr>
<tr>
<td>AX522 (-XC)</td>
<td>D2</td>
</tr>
<tr>
<td>AX561</td>
<td>B2</td>
</tr>
<tr>
<td>CD522 (-XC)</td>
<td>D1</td>
</tr>
<tr>
<td>DA501 (-XC)</td>
<td>D2</td>
</tr>
<tr>
<td>DC522 (-XC)</td>
<td>D2</td>
</tr>
<tr>
<td>DC523 (-XC)</td>
<td>D2</td>
</tr>
<tr>
<td>DC532 (-XC)</td>
<td>D2</td>
</tr>
<tr>
<td>DC561</td>
<td>B2</td>
</tr>
<tr>
<td>DC562</td>
<td>A2</td>
</tr>
<tr>
<td>DI524 (-XC)</td>
<td>D2</td>
</tr>
<tr>
<td>DI561</td>
<td>B2</td>
</tr>
<tr>
<td>DI562</td>
<td>B2</td>
</tr>
<tr>
<td>DI571</td>
<td>B2</td>
</tr>
<tr>
<td>DI572</td>
<td>A1</td>
</tr>
<tr>
<td>DO524 (-XC)</td>
<td>A3</td>
</tr>
<tr>
<td>DO526</td>
<td>A2</td>
</tr>
<tr>
<td>DO526-XC</td>
<td>A0</td>
</tr>
<tr>
<td>DO561</td>
<td>B2</td>
</tr>
<tr>
<td>DO562</td>
<td>A2</td>
</tr>
</tbody>
</table>
### Device Specifications

<table>
<thead>
<tr>
<th>Device</th>
<th>Min. required device index for I/O module as of FW Version 3.0.14</th>
</tr>
</thead>
<tbody>
<tr>
<td>DO571</td>
<td>B3</td>
</tr>
<tr>
<td>DO572</td>
<td>B2</td>
</tr>
<tr>
<td>DO573</td>
<td>A1</td>
</tr>
<tr>
<td>DX522 (-XC)</td>
<td>D2</td>
</tr>
<tr>
<td>DX531</td>
<td>D2</td>
</tr>
<tr>
<td>DX561</td>
<td>B2</td>
</tr>
<tr>
<td>DX571</td>
<td>B3</td>
</tr>
<tr>
<td>FM562</td>
<td>A1</td>
</tr>
</tbody>
</table>

### Extreme Conditions

Terminal units for use in extreme ambient conditions have no \( \text{XC} \) sign for XC version.

The figure 4 in the Part no. 1SAP4... (label) identifies the XC version.

### Terminals

<table>
<thead>
<tr>
<th>Screw terminals</th>
<th>Spring terminals</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Conductor</strong></td>
<td><strong>Screwdriver</strong></td>
</tr>
<tr>
<td>1.5</td>
<td></td>
</tr>
<tr>
<td>1.6</td>
<td></td>
</tr>
<tr>
<td>1.7</td>
<td></td>
</tr>
<tr>
<td>1.8</td>
<td></td>
</tr>
<tr>
<td>1.9</td>
<td></td>
</tr>
</tbody>
</table>

For information about wiring specifications see the description of the Terminal Units \( \text{\textsection} \) Chapter 1.5.4.5.4.3 "Terminals at the terminal unit" on page 3078.

For a detailed description of the mounting, disassembly and electrical connection of the terminal units and the I/O modules, please refer to the "System Assembly, Construction and Connection" chapter \( \text{\textsection} \) Chapter 1.5.4.5.3 "Mounting and demounting" on page 3068.
The terminals 1.8 to 4.8 and 1.9 to 4.9 are electrically interconnected within the terminal unit and always have the same assignment, independent of the inserted module:

- Terminals 1.8 to 4.8: process supply voltage $UP = +24\ \text{VDC}$
- Terminals 1.9 to 4.9: process supply voltage $ZP = 0\ \text{V}$

The assignment of the other terminals depends on the inserted decentralized communication interface module (see the description of the respective module used).

The supply voltage of 24 VDC for the module's circuitry comes from the I/O expansion bus (I/O bus).

### Technical data

The system data of AC500 and S500 ☞ Chapter 1.5.4.5.1 “System data AC500” on page 3058 are valid for standard version.

The system data of AC500-XC ☞ Chapter 1.5.4.6.1 “System data AC500-XC” on page 3099 are valid for the XC version.

Only additional details are therefore documented below.

The technical data are also valid for the XC version.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of channels per module</td>
<td>32</td>
</tr>
<tr>
<td>Distribution of the channels into groups</td>
<td>4 groups of 8 channels each (1.0…1.7, 2.0…2.7, 3.0…3.7, 4.0…4.7), the allocation of the channels is given by the inserted I/O module</td>
</tr>
<tr>
<td>Terminals 1.8…4.8 and 1.9…4.9</td>
<td>Max. voltage 30 VDC</td>
</tr>
<tr>
<td></td>
<td>Max. permitted total current 10 A</td>
</tr>
<tr>
<td>Terminals 1.0…1.7, 2.0…2.7, 3.0…3.7, 4.0…4.7</td>
<td>Max. voltage 300 VAC 1)</td>
</tr>
<tr>
<td></td>
<td>Max. permitted current 3 A 2)</td>
</tr>
<tr>
<td>Earthing</td>
<td>Direct connection to the earthed DIN rail or via the screws with wall mounting</td>
</tr>
<tr>
<td>Screw terminals</td>
<td>Front terminal, conductor connection vertically with respect to the printed circuit board</td>
</tr>
<tr>
<td>Spring terminals</td>
<td>Front terminal, conductor connection vertically with respect to the printed circuit board</td>
</tr>
<tr>
<td>Weight</td>
<td>200 g</td>
</tr>
<tr>
<td>Mounting position</td>
<td>Horizontal or vertical</td>
</tr>
</tbody>
</table>

1) Only when the voltage is not limited by the specification of the I/O channel or the supply input which is internally connected to the terminal.

2) The terminals are connected to the electronic module via internal connectors (X22 (or 3b), X23 (or 3b), X32, X33 and X34). The current per terminal is limited by the permitted current of these connectors.
## Ordering data

<table>
<thead>
<tr>
<th>Part no.</th>
<th>Description</th>
<th>Product life cycle phase *)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1SAP 217 200 R0001</td>
<td>TU531, terminal unit, 230 VAC, relays, screw terminals</td>
<td>Active</td>
</tr>
<tr>
<td>1SAP 217 000 R0001</td>
<td>TU532, terminal unit, 230 VAC, relays, spring terminals</td>
<td>Active</td>
</tr>
<tr>
<td>1SAP 417 000 R0001</td>
<td>TU532-XC, terminal unit, 230 VAC, relays, spring terminals, XC version</td>
<td>Active</td>
</tr>
<tr>
<td>1SAP 215 100 R0001</td>
<td>TU532-H, terminal unit, hot swap, 230 VAC, relays, spring terminals</td>
<td>Active</td>
</tr>
<tr>
<td>1SAP 415 100 R0001</td>
<td>TU532-H-XC, terminal unit, hot swap, 230 VAC, relays, spring terminals, XC version</td>
<td>Active</td>
</tr>
</tbody>
</table>

*) For planning and commissioning of new installations use modules in Active status only.
1.5.3.6 I/O modules

**Hot swap**

System requirements for hot swapping of I/O modules:

- Types of terminal units that support hot swapping of I/O modules have the appendix TU5xx-H.
- I/O modules as of index F0.

The following I/O bus masters support hot swapping of attached I/O modules:

- Communication interface modules CI5xx as of index F0.
- Processor modules PM56xx-2ETH with firmware version as of V3.2.0.

**NOTICE!**

Risk of damage to I/O modules!

Hot swapping is only allowed for I/O modules. Processor modules and communication interface modules must not be removed or inserted during operation.

**Conditions for hot swapping**

- Digital outputs are not under load.
- Input/output voltages above safety extra low voltage/protective extra low voltages (SELV/PELV) are switched off.
- Modules are completely plugged on the terminal unit with both snap fit engaged before switching on loads or input/output voltage.

Further information about hot swap: System technology > Chapter 1.5.5.1.6 “Hot swap” on page 3148.

---

1.5.3.6.1 Digital I/O modules

**S500-eCo**

**DC561** - Digital input/output module

- 16 configurable digital inputs/outputs 24 VDC,
- Connection via Interfast
- Module-wise electrically isolated
Intended purpose

The digital input/output module DC561 can be connected to the following devices via the I/O bus connector:

- S500 bus modules (e.g. CI501-PNIO, CI541-DP, CI581-CN)
- AC500 CPUs (PM5xx)
- other AC500 I/O modules

The I/O module must not be used as a decentralized I/O module with CI590-CS31-HA bus modules.

The module contains 16 digital channels in 1 group, each channel can be used as a digital 24 VDC input or 24 VDC output.
The inputs/outputs are group-wise electrically isolated from each other. All other circuitry of the module is electrically isolated from the inputs/outputs.

### Functionality

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Digital inputs</td>
<td>Max. 16 (24 VDC), can be used as sink inputs</td>
</tr>
<tr>
<td>Digital outputs</td>
<td>Max. 16 (transistor outputs 24 VDC, max. 0.1 A)</td>
</tr>
<tr>
<td>LED displays</td>
<td>For signal states</td>
</tr>
<tr>
<td>Internal power supply</td>
<td>Via I/O bus</td>
</tr>
<tr>
<td>External power supply</td>
<td>Via the terminals ZP and UP (process voltage 24 VDC)</td>
</tr>
</tbody>
</table>

### Electrical connection

*For a detailed description of the mounting, disassembly and electrical connection of the module, please refer to the System Assembly chapter.*

The electrical connection is established out by using the 20-pin Interfast connector. For further information, refer to the Interfast documentation.

The assignment of the terminals:

<table>
<thead>
<tr>
<th>Terminal</th>
<th>Signal</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>C0</td>
<td>Input/output signal C0</td>
</tr>
<tr>
<td>2</td>
<td>C1</td>
<td>Input/output signal C1</td>
</tr>
<tr>
<td>3</td>
<td>C2</td>
<td>Input/output signal C2</td>
</tr>
<tr>
<td>4</td>
<td>C3</td>
<td>Input/output signal C3</td>
</tr>
<tr>
<td>5</td>
<td>C4</td>
<td>Input/output signal C4</td>
</tr>
<tr>
<td>6</td>
<td>C5</td>
<td>Input/output signal C5</td>
</tr>
<tr>
<td>7</td>
<td>C6</td>
<td>Input/output signal C6</td>
</tr>
<tr>
<td>8</td>
<td>C7</td>
<td>Input/output signal C7</td>
</tr>
<tr>
<td>9</td>
<td>C8</td>
<td>Input/output signal C8</td>
</tr>
<tr>
<td>10</td>
<td>C9</td>
<td>Input/output signal C9</td>
</tr>
<tr>
<td>11</td>
<td>C10</td>
<td>Input/output signal C10</td>
</tr>
<tr>
<td>12</td>
<td>C11</td>
<td>Input/output signal C11</td>
</tr>
<tr>
<td>13</td>
<td>C12</td>
<td>Input/output signal C12</td>
</tr>
<tr>
<td>14</td>
<td>C13</td>
<td>Input/output signal C13</td>
</tr>
<tr>
<td>15</td>
<td>C14</td>
<td>Input/output signal C14</td>
</tr>
<tr>
<td>16</td>
<td>C15</td>
<td>Input/output signal C15</td>
</tr>
<tr>
<td>17</td>
<td>UP</td>
<td>Process voltage UP +24 VDC</td>
</tr>
<tr>
<td>18</td>
<td>ZP</td>
<td>Process voltage ZP 0 VDC</td>
</tr>
</tbody>
</table>
The arrow located next to the Interfast connector marks terminal 1.

The internal power supply voltage for the module's circuitry is carried out via the I/O bus (provided by a bus module or a CPU). Thus, the current consumption from 24 VDC power supply at the terminals L+/UP and M/ZP of the CPU/bus module increases by 10 mA per DC561.

The external power supply connection is carried out via the UP (+24 VDC) and ZP (0 VDC) terminals.

**WARNING!**

Removal/Insertion under power

The devices are not designed for removal or insertion under power. Because of unforeseeable consequences, it is not allowed to plug or unplug devices with the power being ON.

Make sure that all voltage sources (supply and process voltage) are switched off before you

- connect or disconnect any signal or terminal block
- remove, mount or replace a module.

Disconnecting any powered devices while energized in a hazardous location could result in an electric arc, which could create a flammable ignition resulting in fire or explosion.

Make sure that power is removed and that the area has been thoroughly checked to ensure that flammable materials are not present prior to proceeding.

The devices must not be opened when in operation. The same applies to the network interfaces.

**NOTICE!**

Risk of damaging the PLC modules!

Overvoltages and short circuits might damage the PLC modules.

- Make sure that all voltage sources (supply voltage and process supply voltage) are switched off before you begin with operations at the system.
- Never connect any voltages or signals to reserved terminals (marked with ---). Reserved terminals may carry internal voltages.

Process supply voltage must be connected to UP/ZP of the module. The inputs and UP/ZP must use the same power supply.

If DC561 with index A0 is used, the process supply voltage must stem from the same source as the power supply voltage of the CPU. The index consists of 1 letter, followed by 1 digit, and can be found on the type plate of the module next to the type designator "DC561".
The module provides several diagnosis functions \( \text{\textsuperscript{\textcopyright} Chapter 1.5.3.6.1.1.6 \textit{Diagnosis} on page 2332.} \)

The meaning of the LEDs is described in the section State LEDs \( \text{\textsuperscript{\textcopyright} Chapter 1.5.3.6.1.1.7 \textit{State LEDs} on page 2332.} \)

**I/O Configuration**

The module itself does not store configuration data. It receives its parameterization data from the master device of the I/O bus (CPU or bus module) during power-up of the system.

Hence, replacing I/O modules is possible without any re-parameterization via software.

*If the external power supply voltage via UP/ZP terminals fails, the I/O module loses its configuration data. The whole station has to be switched off and on again to re-configure the module.*

**Parameterization**

The arrangement of the parameter data is performed with Automation Builder software.

The parameter data directly influences the functionality of modules.

For non-standard applications, it is necessary to adapt the parameters to your system configuration.

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>Internal Value</th>
<th>Internal Value, Type</th>
<th>Default</th>
<th>Min.</th>
<th>Max.</th>
<th>EDS Slot Index</th>
</tr>
</thead>
<tbody>
<tr>
<td>Module ID</td>
<td>Internal</td>
<td>6100 (^1))</td>
<td>WORD</td>
<td>6100 0x17D4</td>
<td>0</td>
<td>65535</td>
<td>xx01</td>
</tr>
<tr>
<td>Ignore module</td>
<td>No</td>
<td>0</td>
<td>BYTE</td>
<td>No (0x00)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Parameter length (^2))</td>
<td>Internal</td>
<td>1 - CPU</td>
<td>BYTE</td>
<td>0</td>
<td>0</td>
<td>255</td>
<td>xx02 (^3))</td>
</tr>
</tbody>
</table>

Remarks:

\(^1\) With CS31 and addresses smaller than 70, the value is increased by 1
\(^2\) The module has no additional user-configurable parameters
\(^3\) Value is hexadecimal: HighByte is slot (xx: 0...7), LowByte is index (1...n)

GSD file:

```plaintext
Ext_User_Prm_Data_Len = 0x03
Ext_User_Prm_Data_Const(0) = 0x25, 0x17, 0x00;
```
Diagnosis

<table>
<thead>
<tr>
<th>E1...E4</th>
<th>d1</th>
<th>d2</th>
<th>d3</th>
<th>d4</th>
<th>Identifier 000...063</th>
<th>AC500-Display</th>
<th>&lt;- Display in</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class</td>
<td>Comp</td>
<td>Dev</td>
<td>Mod</td>
<td>Ch</td>
<td>Err</td>
<td>PS501 PLC Browser</td>
<td></td>
</tr>
<tr>
<td>Byte 6</td>
<td>Byte 3</td>
<td>Byte 4</td>
<td>Byte 5</td>
<td>Byte 6 Bit 0...5</td>
<td>PNIO diagnosis block</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bit 6...7</td>
<td>Interface</td>
<td>Device</td>
<td>Module</td>
<td>Channel</td>
<td>Error Identifier</td>
<td>Error message</td>
<td>Remedy</td>
</tr>
<tr>
<td>Class</td>
<td>Interface</td>
<td>Device</td>
<td>Module</td>
<td>Channel</td>
<td>Error Identifier</td>
<td>Error message</td>
<td>Remedy</td>
</tr>
<tr>
<td>1)</td>
<td>2)</td>
<td>3)</td>
<td>4)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Module error DI571

<table>
<thead>
<tr>
<th>Class</th>
<th>Interface</th>
<th>Device</th>
<th>Module</th>
<th>Channel</th>
<th>Error Identifier</th>
<th>Error message</th>
<th>Remedy</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>14</td>
<td>1...10</td>
<td>31</td>
<td>31</td>
<td>19</td>
<td>Checksum error in the I/O module</td>
<td>Replace I/O module</td>
</tr>
<tr>
<td></td>
<td>11 / 12</td>
<td>ADR</td>
<td>1...10</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>14</td>
<td>1...10</td>
<td>31</td>
<td>31</td>
<td>43</td>
<td>Internal error in the module</td>
<td>Replace I/O module</td>
</tr>
<tr>
<td></td>
<td>11 / 12</td>
<td>ADR</td>
<td>1...10</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>14</td>
<td>1...10</td>
<td>31</td>
<td>31</td>
<td>9</td>
<td>Overflow diagnosis buffer</td>
<td>Restart</td>
</tr>
<tr>
<td></td>
<td>11 / 12</td>
<td>ADR</td>
<td>1...10</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>14</td>
<td>1...10</td>
<td>31</td>
<td>31</td>
<td>26</td>
<td>Parameter error</td>
<td>Check master</td>
</tr>
<tr>
<td></td>
<td>11 / 12</td>
<td>ADR</td>
<td>1...10</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Remarks:

1) In AC500 the following interface identifier applies:
14 = I/O bus, 11 = COM1 (e.g. CS31 bus), 12 = COM2.
The PNIO diagnosis block does not contain this identifier.

2) With "Device" the following allocation applies:
31 = module itself, 1...10 = decentralized communication interface module 1...10, ADR = hardware address (e.g. of the DC551-CS31)

3) With "Module" the following allocation applies depending on the master:
Module error: I/O bus or PNIO: 31 = module itself; COM1/COM2: 1...10 = expansion 1...10

4) In case of module errors, with channel "31 = Module itself" is output.

State LEDs

<table>
<thead>
<tr>
<th>LED</th>
<th>Inputs/outputs C0...C15</th>
<th>State</th>
<th>Color</th>
<th>LED = OFF</th>
<th>LED = ON</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Digital input or digital output</td>
<td>Yellow</td>
<td>Input/output is OFF</td>
<td>Input/output is ON (the LEDs are only operating if the module's circuitry is supplied via the I/O bus)</td>
<td></td>
</tr>
</tbody>
</table>
Technical data

The System Data of AC500-eCo apply.

Only additional details are therefore documented below.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Process voltage UP</td>
<td></td>
</tr>
<tr>
<td>Connections</td>
<td>Terminals 17 and 19 for UP (+24 VDC); terminals 18 and 20 for ZP (0 V)</td>
</tr>
<tr>
<td>Rated value</td>
<td>24 VDC</td>
</tr>
<tr>
<td>Current consumption via UP terminal</td>
<td>10 mA + 0.1 A per output (max.)</td>
</tr>
<tr>
<td>Max. ripple</td>
<td>5 %</td>
</tr>
<tr>
<td>Inrush current</td>
<td>0.000001 A²s</td>
</tr>
<tr>
<td>Protection against reversed voltage</td>
<td>Yes</td>
</tr>
<tr>
<td>Protection fuse on UP</td>
<td>Recommended; the outputs must be protected by an 1 A fast fuse</td>
</tr>
<tr>
<td>Current consumption from 24 VDC power supply</td>
<td>Ca. 10 mA</td>
</tr>
<tr>
<td>supply at the L+/UP and M/ZP terminals of</td>
<td></td>
</tr>
<tr>
<td>the CPU/bus module</td>
<td></td>
</tr>
<tr>
<td>Galvanic isolation</td>
<td>Yes, between the input/output group and the rest of the module</td>
</tr>
<tr>
<td>Isolated groups</td>
<td>1 group for 16 channels</td>
</tr>
<tr>
<td>Surge voltage (max.)</td>
<td>35 VDC for 0.5 s</td>
</tr>
<tr>
<td>Max. power dissipation within the module</td>
<td>On request</td>
</tr>
<tr>
<td>Input data length</td>
<td>2 bytes</td>
</tr>
<tr>
<td>Output data length</td>
<td>2 bytes</td>
</tr>
<tr>
<td>Weight</td>
<td>Ca. 115 g</td>
</tr>
<tr>
<td>Mounting position</td>
<td>Horizontal or vertical</td>
</tr>
<tr>
<td>Cooling</td>
<td>The natural convection cooling must not be hindered by cable ducts or other parts in the switch-gear cabinet.</td>
</tr>
</tbody>
</table>

**No effects of multiple overloads**

No effects of multiple overloads on isolated multi-channel modules occur, as every channel is protected individually by an external fuse.

Technical data of the digital inputs/outputs if used as inputs

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of channels per module</td>
<td>16 configurable inputs (24 VDC)</td>
</tr>
<tr>
<td>Distribution of the channels into groups</td>
<td>1 (16 channels per group)</td>
</tr>
<tr>
<td>Connections of the channels C0 to C15</td>
<td>Terminals 1 to 16</td>
</tr>
<tr>
<td>Reference potential for the channels C0 to C15</td>
<td>Terminals 18 and 20 (negative pole of the process voltage, name ZP)</td>
</tr>
<tr>
<td>Indication of the input signals</td>
<td>1 yellow LED per channel; the LED is ON when the input signal is high (signal 1). The module is powered via the I/O bus.</td>
</tr>
</tbody>
</table>
### Parameter Value

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input type according to EN 61131-2</td>
<td>Type 1 sink</td>
</tr>
<tr>
<td>Input signal range</td>
<td>+24 VDC</td>
</tr>
<tr>
<td>Signal 0</td>
<td>-3 V...+5 V</td>
</tr>
<tr>
<td>Undefined signal</td>
<td>+5 V...+15 V</td>
</tr>
<tr>
<td>Signal 1</td>
<td>+15 V...+30 V</td>
</tr>
<tr>
<td>Ripple with signal 0</td>
<td>-3 V...+5 V</td>
</tr>
<tr>
<td>Ripple with signal 1</td>
<td>+15 V...+30 V</td>
</tr>
<tr>
<td>Input current per channel</td>
<td></td>
</tr>
<tr>
<td>Input voltage +24 V</td>
<td>Typ. 5 mA</td>
</tr>
<tr>
<td>Input voltage +5 V</td>
<td>Typ. 1 mA</td>
</tr>
<tr>
<td>Input voltage +15 V</td>
<td>&gt; 2.5 mA</td>
</tr>
<tr>
<td>Input voltage +30 V</td>
<td>&lt; 8 mA</td>
</tr>
<tr>
<td>Max. permissible leakage current (at 2-wire proximity switches)</td>
<td>1 mA</td>
</tr>
<tr>
<td>Input delay (0-&gt;1 or 1-&gt;0)</td>
<td>Typ. 8 ms</td>
</tr>
<tr>
<td>Max. cable length</td>
<td></td>
</tr>
<tr>
<td>Shielded</td>
<td>500 m</td>
</tr>
<tr>
<td>Unshielded</td>
<td>300 m</td>
</tr>
</tbody>
</table>

### Technical data of the digital inputs/outputs if used as outputs

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of channels per module</td>
<td>16 configurable transistor outputs</td>
</tr>
<tr>
<td>Distribution of the channels into groups</td>
<td>1 (16 channels per group)</td>
</tr>
<tr>
<td>Connections of the channels C0 to C15</td>
<td>Terminals 1 to 16</td>
</tr>
<tr>
<td>Reference potential for the channels C0 to C15</td>
<td>Terminals 18 and 20 (negative pole of the process voltage, signal name ZP)</td>
</tr>
<tr>
<td>Common power supply voltage</td>
<td>Terminals 17 and 19 (positive pole of the process voltage, signal name UP)</td>
</tr>
<tr>
<td>Indication of the input signals</td>
<td>1 yellow LED per channel; the LED is ON when the input signal is high (signal 1). The module is powered via the I/O bus.</td>
</tr>
<tr>
<td>Way of operation</td>
<td>Non-latching type</td>
</tr>
<tr>
<td>Output voltage at signal 1</td>
<td>UP -0.3 V at max. current</td>
</tr>
<tr>
<td>Output delay (max. at rated load)</td>
<td></td>
</tr>
<tr>
<td>0 to 1</td>
<td>50 µs</td>
</tr>
<tr>
<td>1 to 0</td>
<td>200 µs</td>
</tr>
<tr>
<td>Output current</td>
<td></td>
</tr>
<tr>
<td>Rated current per channel (max.)</td>
<td>0.1 A at UP 24 VDC</td>
</tr>
<tr>
<td>Rated current per group (max.)</td>
<td>1.6 A</td>
</tr>
<tr>
<td>Rated current (all channels together, max.)</td>
<td>1.6 A</td>
</tr>
<tr>
<td>Parameter</td>
<td>Value</td>
</tr>
<tr>
<td>-----------------------------------------------</td>
<td>-------------------------------------------------</td>
</tr>
<tr>
<td>Lamp load (max.)</td>
<td>Not applicable</td>
</tr>
<tr>
<td>Max. leakage current with signal 0</td>
<td>&lt; 0.5 mA</td>
</tr>
<tr>
<td>Output type</td>
<td>Non-protected</td>
</tr>
<tr>
<td>Protection type</td>
<td>External fuse on each channel</td>
</tr>
<tr>
<td>Rated protection fuse (for each channel)</td>
<td>1 A fast</td>
</tr>
<tr>
<td>Demagnetization when inductive loads are</td>
<td>Must be performed externally according to load</td>
</tr>
<tr>
<td>switched off</td>
<td>specification</td>
</tr>
<tr>
<td>Switching frequency</td>
<td></td>
</tr>
<tr>
<td>With inductive loads</td>
<td>Max. 0.5 Hz</td>
</tr>
<tr>
<td>Short-circuit-proof / overload-proof</td>
<td>No</td>
</tr>
<tr>
<td>Overload message</td>
<td>No</td>
</tr>
<tr>
<td>Output current limitation</td>
<td>No</td>
</tr>
<tr>
<td>Resistance to feedback against 24 VDC signals</td>
<td>Yes</td>
</tr>
<tr>
<td>Connection of 2 outputs in parallel</td>
<td>Not possible</td>
</tr>
<tr>
<td>Max. cable length</td>
<td></td>
</tr>
<tr>
<td>Shielded</td>
<td>500 m</td>
</tr>
<tr>
<td>Unshielded</td>
<td>150 m</td>
</tr>
</tbody>
</table>

### Ordering data

<table>
<thead>
<tr>
<th>Part no.</th>
<th>Description</th>
<th>Product life cycle phase *)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1TNE 968 902 R2001</td>
<td>DC561, digital input/output module, 16 configurable inputs/outputs,</td>
<td>Active</td>
</tr>
<tr>
<td></td>
<td>transistor output, interfast connector</td>
<td></td>
</tr>
</tbody>
</table>

*) For planning and commissioning of new installations use modules in Active status only.

**DC562 - Digital input/output module**

- 16 configurable digital inputs/outputs in 1 group, 24 VDC
- Module-wise electrically isolated
1. I/O bus
2. 16 yellow LEDs to display the states of the inputs/outputs C0 to C15
3. Terminal number
4. Allocation of signal name
5. Terminal block for input and output signals (9-pin)
6. Terminal block for input and output signals (11-pin)
7. 2 holes for wall-mounting with screws
8. DIN rail

**Intended purpose**

The device can be used as a decentralized I/O extension module for S500 Communication Interface Modules (e.g. CI592-CS31, CI501-PNIO, CI541-DP, CI581-CN) or as a centralized extension module for AC500 CPUs (PM5xx).

The inputs/outputs are group-wise electrically isolated from each other.
All other circuitry of the module is electrically isolated from the inputs/outputs.

*The I/O module must not be used as a decentralized I/O module with CI590-CS31-HA bus modules.*
**Functionality**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>LED displays</td>
<td>For signal states</td>
</tr>
<tr>
<td>Internal power supply</td>
<td>Via I/O bus</td>
</tr>
<tr>
<td>External power supply</td>
<td>Via the terminals ZP and UP (process voltage 24 VDC)</td>
</tr>
</tbody>
</table>

**Electrical connection**

For a detailed description of the mounting, disassembly and electrical connection of the module, please refer to the System Assembly chapter.

The electrical connection is carried out by using a removable 9-pin and 11-pin terminal block. These terminal blocks differ in their connection system (spring terminals or screw terminals, cable mounting from the front or from the side). The terminal blocks are not included in the module's scope of delivery and must be ordered separately.

The following block diagram shows the internal construction of the digital inputs and outputs:
Table 539: Assignment of the terminals:

<table>
<thead>
<tr>
<th>Terminal</th>
<th>Signal</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>---</td>
<td>Reserved</td>
</tr>
<tr>
<td>2</td>
<td>C0</td>
<td>Input/output signal C0</td>
</tr>
<tr>
<td>3</td>
<td>C1</td>
<td>Input/output signal C1</td>
</tr>
<tr>
<td>4</td>
<td>C2</td>
<td>Input/output signal C2</td>
</tr>
<tr>
<td>5</td>
<td>C3</td>
<td>Input/output signal C3</td>
</tr>
<tr>
<td>6</td>
<td>C4</td>
<td>Input/output signal C4</td>
</tr>
<tr>
<td>7</td>
<td>C5</td>
<td>Input/output signal C5</td>
</tr>
<tr>
<td>Terminal</td>
<td>Signal</td>
<td>Description</td>
</tr>
<tr>
<td>---------</td>
<td>--------</td>
<td>------------------------------</td>
</tr>
<tr>
<td>8</td>
<td>C6</td>
<td>Input/output signal C6</td>
</tr>
<tr>
<td>9</td>
<td>C7</td>
<td>Input/output signal C7</td>
</tr>
<tr>
<td>10</td>
<td>---</td>
<td>Reserved</td>
</tr>
<tr>
<td>11</td>
<td>C8</td>
<td>Input/output signal C8</td>
</tr>
<tr>
<td>12</td>
<td>C9</td>
<td>Input/output signal C9</td>
</tr>
<tr>
<td>13</td>
<td>C10</td>
<td>Input/output signal C10</td>
</tr>
<tr>
<td>14</td>
<td>C11</td>
<td>Input/output signal C11</td>
</tr>
<tr>
<td>15</td>
<td>C12</td>
<td>Input/output signal C12</td>
</tr>
<tr>
<td>16</td>
<td>C13</td>
<td>Input/output signal C13</td>
</tr>
<tr>
<td>17</td>
<td>C14</td>
<td>Input/output signal C14</td>
</tr>
<tr>
<td>18</td>
<td>C15</td>
<td>Input/output signal C15</td>
</tr>
<tr>
<td>19</td>
<td>UP</td>
<td>Process voltage UP +24 VDC</td>
</tr>
<tr>
<td>20</td>
<td>ZP</td>
<td>Process voltage ZP 0 VDC</td>
</tr>
</tbody>
</table>

The internal power supply voltage for the module’s circuitry is carried out via the I/O bus (provided by a bus module or a CPU). Thus, the current consumption from 24 VDC power supply at the terminals L+/UP and M/ZP of the CPU/bus module increases by 10 mA per DC562.

The external power supply connection is carried out via the UP (+24 VDC) and ZP (0 VDC) terminals.

**WARNING!**

**Removal/Insertion under power**

The devices are not designed for removal or insertion under power. Because of unforeseeable consequences, it is not allowed to plug or unplug devices with the power being ON.

Make sure that all voltage sources (supply and process voltage) are switched off before you

– connect or disconnect any signal or terminal block
– remove, mount or replace a module.

Disconnecting any powered devices while energized in a hazardous location could result in an electric arc, which could create a flammable ignition resulting in fire or explosion.

Make sure that power is removed and that the area has been thoroughly checked to ensure that flammable materials are not present prior to proceeding.

The devices must not be opened when in operation. The same applies to the network interfaces.

**NOTICE!**

**Risk of damaging the PLC modules!**

Overvoltages and short circuits might damage the PLC modules.

– Make sure that all voltage sources (supply voltage and process supply voltage) are switched off before you begin with operations at the system.
– Never connect any voltages or signals to reserved terminals (marked with ---). Reserved terminals may carry internal voltages.
Process supply voltage must be connected to UP/ZP of the module. The inputs and UP/ZP must use the same power supply.

The following figure shows the electrical connection of the digital input/output module DC562:

In this connection example, the inputs/outputs C0...C7 are connected as inputs and the inputs/outputs C8...C15 are connected as outputs.

The module provides several diagnosis functions "DC562 - Digital input/output module" on page 2335.

The meaning of the LEDs is described in the section "State LEDs" on page 2342.
I/O configuration

The module itself does not store configuration data. It receives its parameterization data from the master device of the I/O bus (CPU or bus module) during power-up of the system. Hence, replacing I/O modules is possible without any re-parameterization via software.

If the external power supply voltage via UP/ZP terminals fails, the I/O module loses its configuration data. The whole station has to be switched off and on again to re-configure the module.

Parameterization

The arrangement of the parameter data is performed with Automation Builder software. The parameter data directly influences the functionality of modules. For non-standard applications, it is necessary to adapt the parameters to your system configuration.

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>Internal Value</th>
<th>Internal Value, Type</th>
<th>Default</th>
<th>Min.</th>
<th>Max.</th>
<th>EDS Slot Index</th>
</tr>
</thead>
<tbody>
<tr>
<td>Module ID</td>
<td>Internal</td>
<td>6155 1)</td>
<td>WORD</td>
<td>6155</td>
<td>0</td>
<td>65535</td>
<td>xx01</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0x180B</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ignore module</td>
<td>No</td>
<td>0</td>
<td>BYTE</td>
<td>No</td>
<td>0</td>
<td>255</td>
<td>xx02 3)</td>
</tr>
<tr>
<td></td>
<td>Yes</td>
<td>1</td>
<td></td>
<td>(0x00)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Parameter length 2)</td>
<td>Internal</td>
<td>1 - CPU</td>
<td>BYTE</td>
<td>0</td>
<td>0</td>
<td>255</td>
<td>xx02 3)</td>
</tr>
</tbody>
</table>

1) with CS31 and addresses less than 70, the value is increased by 1
2) the module has no additional user-configurable parameters
3) Value is hexadecimal: HighByte is slot (xx: 0...7), LowByte is index (1...n)

GSD file:

```
Ext_User_Prm_Data_Len = 0x06
Ext_User_Prm_Data_Const(0) = 0x18, 0x0C, 0x00, 0x02, 0x00, 0x00;
```
### Diagnosis

<table>
<thead>
<tr>
<th>E1...E4</th>
<th>d1</th>
<th>d2</th>
<th>d3</th>
<th>d4</th>
<th>Identifier</th>
<th>AC500-Display</th>
<th>&lt;- Display in</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class</td>
<td>Comp</td>
<td>Dev</td>
<td>Mod</td>
<td>Ch</td>
<td>Err</td>
<td>PS501 PLC Browser</td>
<td></td>
</tr>
<tr>
<td>Byte 6</td>
<td>Byte 3</td>
<td>Byte 4</td>
<td>Byte 5</td>
<td>Byte 6 Bit 0...5</td>
<td>PNIO diagnosis block</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bit 6...7</td>
<td>Bit 6...7</td>
<td>Bit 6...7</td>
<td>Bit 6...7</td>
<td>Bit 6...7</td>
<td>Bit 6...7</td>
<td>Bit 6...7</td>
<td>Bit 6...7</td>
</tr>
</tbody>
</table>

#### Module error DC562

<table>
<thead>
<tr>
<th>Byte</th>
<th>Bit</th>
<th>Device</th>
<th>Module</th>
<th>Channel</th>
<th>Error-Identifier</th>
<th>Error message</th>
<th>Remedy</th>
</tr>
</thead>
<tbody>
<tr>
<td>14</td>
<td>1...10</td>
<td>31</td>
<td>31</td>
<td>19</td>
<td>Checksum error in the I/O module</td>
<td>Replace I/O module</td>
<td></td>
</tr>
<tr>
<td>11 / 12</td>
<td>ADR</td>
<td>1...10</td>
<td>11 / 12</td>
<td>ADR</td>
<td>1...10</td>
<td>Internal error in the module</td>
<td>Replace I/O module</td>
</tr>
<tr>
<td>14</td>
<td>1...10</td>
<td>31</td>
<td>31</td>
<td>43</td>
<td>Internal error in the module</td>
<td>Replace I/O module</td>
<td></td>
</tr>
<tr>
<td>11 / 12</td>
<td>ADR</td>
<td>1...10</td>
<td>11 / 12</td>
<td>ADR</td>
<td>1...10</td>
<td>Internal error in the module</td>
<td>Replace I/O module</td>
</tr>
</tbody>
</table>

#### Remarks:

1) In AC500 the following interface identifier applies:
   - 14 = I/O bus, 11 = COM1 (e.g. CS31-Bus), 12 = COM2.
   - The PNIO diagnosis block does not contain this identifier.

2) With "Device" the following allocation applies:
   - 31 = Module itself, 1...10 = expansion module 1...10, ADR = hardware address (e.g. of the DC551-CS31)

3) With "Module" the following allocation applies depending on the master:
   - Module error: I/O bus or PNIO: 31 = module itself; COM1/COM2: 1...10 = expansion 1...10
   - Channel error: I/O bus or PNIO = module type (4 = DC); COM1/COM2: 1...10 = expansion 1...10

4) In case of module errors, with channel "31 = Module itself" is output.

### State LEDs

<table>
<thead>
<tr>
<th>LED</th>
<th>Inputs/outputs C0...C15</th>
<th>State</th>
<th>Color</th>
<th>LED = OFF</th>
<th>LED = ON</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Digital input or digital output</td>
<td>Yellow</td>
<td>Input/output is OFF</td>
<td>Input/output is ON (the LEDs are only operating if the module's circuitry is supplied via the I/O bus)</td>
<td></td>
</tr>
</tbody>
</table>
## Technical data

The System Data of AC500-eCo apply

Only additional details are therefore documented below.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Process voltage UP</td>
<td></td>
</tr>
<tr>
<td>Connections</td>
<td>Terminal 19 for UP (+24 VDC) and terminal 20 for ZP (0 V)</td>
</tr>
<tr>
<td>Rated value</td>
<td>24 VDC</td>
</tr>
<tr>
<td>Current consumption via UP terminal</td>
<td>90 mA + 0.5 A per output (max.)</td>
</tr>
<tr>
<td>Max. ripple</td>
<td>5 %</td>
</tr>
<tr>
<td>Inrush current</td>
<td>0.0000001 A²s</td>
</tr>
<tr>
<td>Protection against reversed voltage</td>
<td>Yes</td>
</tr>
<tr>
<td>Current consumption from 24 VDC power supply at the L+/UP and M/ZP terminals of the CPU/bus module</td>
<td>Ca. 10 mA</td>
</tr>
<tr>
<td>Galvanic isolation</td>
<td>Yes, between the input/output group and the rest of the module</td>
</tr>
<tr>
<td>Isolated groups</td>
<td>1 group for 16 channels</td>
</tr>
<tr>
<td>Surge voltage (max.)</td>
<td>35 VDC for 0.5 s</td>
</tr>
<tr>
<td>Max. power dissipation within the module</td>
<td>4.8 W</td>
</tr>
<tr>
<td>Input data length</td>
<td>2 bytes</td>
</tr>
<tr>
<td>Output data length</td>
<td>2 bytes</td>
</tr>
<tr>
<td>Weight</td>
<td>Ca. 125 g</td>
</tr>
<tr>
<td>Mounting position</td>
<td>Horizontal or vertical</td>
</tr>
<tr>
<td>Cooling</td>
<td>The natural convection cooling must not be hindered by cable ducts or other parts in the switch-gear cabinet.</td>
</tr>
</tbody>
</table>

### No effects of multiple overloads

No effects of multiple overloads on isolated multi-channel modules occur, as every channel is protected individually by an external fuse.

### Technical data of the digital inputs/outputs if used as inputs

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of channels per module</td>
<td>16 configurable inputs (24 VDC)</td>
</tr>
<tr>
<td>Distribution of the channels into groups</td>
<td>1 (16 channels per group)</td>
</tr>
<tr>
<td>Connections of the channels C0 to C15</td>
<td>Terminals 1 to 16</td>
</tr>
<tr>
<td>Reference potential for the channels C0 to C15</td>
<td>Terminal 20 (minus pole of the process voltage, name ZP)</td>
</tr>
<tr>
<td>Indication of the input signals</td>
<td>1 yellow LED per channel; the LED is ON when the input signal is high (signal 1). The module is powered through the I/O-Bus.</td>
</tr>
<tr>
<td>Input type according to EN 61131-2</td>
<td>Type 1 sink</td>
</tr>
<tr>
<td>Input signal range</td>
<td>+24 VDC</td>
</tr>
<tr>
<td>Parameter</td>
<td>Value</td>
</tr>
<tr>
<td>-----------------------------------</td>
<td>----------------------------</td>
</tr>
<tr>
<td>Signal 0</td>
<td>-3 V...+5 V</td>
</tr>
<tr>
<td>Undefined signal</td>
<td>+5 V...+15 V</td>
</tr>
<tr>
<td>Signal 1</td>
<td>+15 V...+30 V</td>
</tr>
<tr>
<td>Ripple with signal 0</td>
<td>-3 V...+5 V</td>
</tr>
<tr>
<td>Ripple with signal 1</td>
<td>+15 V...+30 V</td>
</tr>
<tr>
<td>Input current per channel</td>
<td></td>
</tr>
<tr>
<td>Input voltage +24 V</td>
<td>Typ. 5 mA</td>
</tr>
<tr>
<td>Input voltage +5 V</td>
<td>Typ. 1 mA</td>
</tr>
<tr>
<td>Input voltage +15 V</td>
<td>&gt; 2.5 mA</td>
</tr>
<tr>
<td>Input voltage +30 V</td>
<td>&lt; 8 mA</td>
</tr>
<tr>
<td>Max. permissible leakage current (at 2-wire proximity switches)</td>
<td>1 mA</td>
</tr>
<tr>
<td>Input delay (0-&gt;1 or 1-&gt;0)</td>
<td>Typ. 8 ms</td>
</tr>
<tr>
<td>Max. cable length</td>
<td></td>
</tr>
<tr>
<td>Shielded</td>
<td>500 m</td>
</tr>
<tr>
<td>Unshielded</td>
<td>300 m</td>
</tr>
</tbody>
</table>

**Technical data of the digital inputs/outputs if used as outputs**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of channels per module</td>
<td>16 configurable transistor outputs</td>
</tr>
<tr>
<td>Distribution of the channels into groups</td>
<td>1 (16 channels per group)</td>
</tr>
<tr>
<td>Connections of the channels C0 to C15</td>
<td>Terminals 1 to 16</td>
</tr>
<tr>
<td>Reference potential for the channels C0 to C15</td>
<td>Terminal 20 (negative pole of the process voltage, signal name ZP)</td>
</tr>
<tr>
<td>Common power supply voltage</td>
<td>Terminal 19 (positive pole of the process voltage, signal name UP)</td>
</tr>
<tr>
<td>Indication of the input signals</td>
<td>1 yellow LED per channel; the LED is ON when the input signal is high (signal 1). The module is powered through the I/O bus.</td>
</tr>
<tr>
<td>Way of operation</td>
<td>Non-latching type</td>
</tr>
<tr>
<td>Output voltage at signal 1</td>
<td>UP -0.3 V at max. current</td>
</tr>
<tr>
<td>Output delay (max. at rated load)</td>
<td></td>
</tr>
<tr>
<td>0 to 1</td>
<td>50 µs</td>
</tr>
<tr>
<td>1 to 0</td>
<td>200 µs</td>
</tr>
<tr>
<td>Output current</td>
<td></td>
</tr>
<tr>
<td>Rated current per channel (max.)</td>
<td>0.5 A at UP 24 VDC</td>
</tr>
<tr>
<td>Rated current per group (max.)</td>
<td>8 A</td>
</tr>
<tr>
<td>Rated current (all channels together, max.)</td>
<td>8 A</td>
</tr>
<tr>
<td>Lamp load (max.)</td>
<td>5 W</td>
</tr>
<tr>
<td>Max. leakage current with signal 0</td>
<td>&lt; 0.5 mA</td>
</tr>
<tr>
<td>Parameter</td>
<td>Value</td>
</tr>
<tr>
<td>---------------------------------------------------------------</td>
<td>----------------------------------------------------------------------</td>
</tr>
<tr>
<td>Output type</td>
<td>Non-protected</td>
</tr>
<tr>
<td>Protection type</td>
<td>External fuse on each channel</td>
</tr>
<tr>
<td>Rated protection fuse (for each channel)</td>
<td>3 A fast</td>
</tr>
<tr>
<td>Demagnetization when inductive loads are switched off</td>
<td>Must be performed externally according to driven load specification</td>
</tr>
<tr>
<td>Switching frequency</td>
<td></td>
</tr>
<tr>
<td>With inductive loads</td>
<td>Max. 0.5 Hz</td>
</tr>
<tr>
<td>With lamp loads</td>
<td>Max. 11 Hz at max. 5 W</td>
</tr>
<tr>
<td>Short-circuit-proof / Overload-proof</td>
<td>No</td>
</tr>
<tr>
<td>Overload message</td>
<td>No</td>
</tr>
<tr>
<td>Output current limitation</td>
<td>No</td>
</tr>
<tr>
<td>Resistance to feedback against 24 VDC signals</td>
<td>Yes</td>
</tr>
<tr>
<td>Connection of 2 outputs in parallel</td>
<td>Not possible</td>
</tr>
<tr>
<td>Max. cable length</td>
<td></td>
</tr>
<tr>
<td>Shielded</td>
<td>500 m</td>
</tr>
<tr>
<td>Unshielded</td>
<td>150 m</td>
</tr>
</tbody>
</table>

**Ordering data**

<table>
<thead>
<tr>
<th>Part no.</th>
<th>Description</th>
<th>Product life cycle phase *)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1SAP 231 900 R0000</td>
<td>DC562, digital input/output module, 16 configurable inputs/outputs, transistor output</td>
<td>Active</td>
</tr>
<tr>
<td>1TNE 968 901 R3101</td>
<td>Terminal block TA563-9, 9 pins, screw front, cable side, 6 pieces per unit</td>
<td>Active</td>
</tr>
<tr>
<td>1TNE 968 901 R3102</td>
<td>Terminal block TA563-11, 11 pins, screw front, cable side, 6 pieces per unit</td>
<td>Active</td>
</tr>
<tr>
<td>1TNE 968 901 R3103</td>
<td>Terminal block TA564-9, 9 pins, screw front, cable front, 6 pieces per unit</td>
<td>Active</td>
</tr>
<tr>
<td>1TNE 968 901 R3104</td>
<td>Terminal block TA564-11, 11 pins, screw front, cable front, 6 pieces per unit</td>
<td>Active</td>
</tr>
<tr>
<td>1TNE 968 901 R3105</td>
<td>Terminal block TA565-9, 9 pins, spring front, cable front, 6 pieces per unit</td>
<td>Active</td>
</tr>
<tr>
<td>1TNE 968 901 R3106</td>
<td>Terminal block TA565-11, 11 pins, spring front, cable front, 6 pieces per unit</td>
<td>Active</td>
</tr>
</tbody>
</table>

*) For planning and commissioning of new installations use modules in Active status only.
**DI561 - Digital input module**

- 8 digital inputs 24 VDC / 24 VAC (I0 to I7) in 1 group
- Module-wise electrically isolated

---

**Intended purpose**

The device can be used as a decentralized I/O extension module for S500 Communication Interface Modules (e.g. CI592-CS31, CI501-PNIO, CI541-DP, CI581-CN) or as a centralized extension module for AC500 CPUs (PM5xx).

The inputs are group-wise electrically isolated from each other.

All other circuitry of the module is electrically isolated from the inputs.
The I/O module must not be used as a decentralized I/O module with Ci590-CS31-HA bus modules.

**Functionality**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>LED displays</td>
<td>For signal states</td>
</tr>
<tr>
<td>Internal power supply</td>
<td>Via I/O bus</td>
</tr>
<tr>
<td>External power supply</td>
<td>Not necessary</td>
</tr>
</tbody>
</table>

**Electrical connection**

For a detailed description of the mounting, disassembly and electrical connection of the module, please refer to the System Assembly chapter.

The electrical connection is carried out by using a removable 9-pin terminal block. These terminal blocks differ in their connection system (spring terminals or screw terminals, cable mounting from the front or from the side). For more information, please refer to the chapter terminal blocks for S500-eCo I/O modules. The terminal blocks are not included in the module's scope of delivery and must be ordered separately.

The following block diagram shows the internal construction of the digital inputs:

![Block diagram of digital inputs]

**Table 540: Assignment of the terminals:**

<table>
<thead>
<tr>
<th>Terminal</th>
<th>Signal</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>C0..7</td>
<td>Input common for signals I0 to I7</td>
</tr>
<tr>
<td>2</td>
<td>I0</td>
<td>Input signal I0</td>
</tr>
<tr>
<td>Terminal</td>
<td>Signal</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>--------</td>
<td>-------------------</td>
</tr>
<tr>
<td>3</td>
<td>I1</td>
<td>Input signal I1</td>
</tr>
<tr>
<td>4</td>
<td>I2</td>
<td>Input signal I2</td>
</tr>
<tr>
<td>5</td>
<td>I3</td>
<td>Input signal I3</td>
</tr>
<tr>
<td>6</td>
<td>I4</td>
<td>Input signal I4</td>
</tr>
<tr>
<td>7</td>
<td>I5</td>
<td>Input signal I5</td>
</tr>
<tr>
<td>8</td>
<td>I6</td>
<td>Input signal I6</td>
</tr>
<tr>
<td>9</td>
<td>I7</td>
<td>Input signal I7</td>
</tr>
</tbody>
</table>

The internal power supply voltage for the module’s circuitry is carried out via the I/O bus (provided by a bus module or a CPU). Thus, the current consumption from 24 VDC power supply at the terminals L+/UP and M/ZP of the CPU/bus module increases by 10 mA per DI561.

An external power supply connection is not needed.

**WARNING!**

**Removal/Insertion under power**

The devices are not designed for removal or insertion under power. Because of unforeseeable consequences, it is not allowed to plug or unplug devices with the power being ON.

Make sure that all voltage sources (supply and process voltage) are switched off before you
- connect or disconnect any signal or terminal block
- remove, mount or replace a module.

Disconnecting any powered devices while energized in a hazardous location could result in an electric arc, which could create a flammable ignition resulting in fire or explosion.

Make sure that power is removed and that the area has been thoroughly checked to ensure that flammable materials are not present prior to proceeding.

The devices must not be opened when in operation. The same applies to the network interfaces.

**NOTICE!**

**Risk of damaging the PLC modules!**

Overvoltages and short circuits might damage the PLC modules.
- Make sure that all voltage sources (supply voltage and process supply voltage) are switched off before you begin with operations at the system.
- Never connect any voltages or signals to reserved terminals (marked with ---). Reserved terminals may carry internal voltages.

The digital inputs can be used as source inputs or as sink inputs.

**NOTICE!**

**Risk of malfunctions in the plant!**

A ground closure, e. g. caused by a damaged cable insulation, can bridge switches accidentally.

Use sink inputs when possible or make sure that, in case of error, there will be no risks to persons or plant.

The following figure shows the electrical connection of the digital input module DI561:
I/O Configuration

The module itself does not store configuration data. It receives its parameterization data from the master device of the I/O bus (CPU or bus module) during power-up of the system. Hence, replacing I/O modules is possible without any re-parameterization via software.

Parameterization

The arrangement of the parameter data is performed with Automation Builder software. The parameter data directly influences the functionality of modules. For non-standard applications, it is necessary to adapt the parameters to your system configuration.

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>Internal Value</th>
<th>Internal Value, Type</th>
<th>Default</th>
<th>Min.</th>
<th>Max.</th>
<th>EDS Slot Index</th>
</tr>
</thead>
<tbody>
<tr>
<td>Module ID</td>
<td>Internal</td>
<td>6105</td>
<td>WORD</td>
<td>6105</td>
<td>0</td>
<td>65535</td>
<td>xx01</td>
</tr>
<tr>
<td>Ignore module</td>
<td>No</td>
<td>0</td>
<td>BYTE</td>
<td>No (0x00)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Yes</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Parameter length</td>
<td>Internal</td>
<td>1 - CPU</td>
<td>BYTE</td>
<td>0</td>
<td>0</td>
<td>255</td>
<td>xx02 3)</td>
</tr>
</tbody>
</table>

1) with CS31 and addresses smaller than 70, the value is increased by 1
2) the module has no additional user-configurable parameters
3) Value is hexadecimal: HighByte is slot (xx: 0...7), LowByte is index (1...n)

GSD file:

<table>
<thead>
<tr>
<th>Ext_User_Prm_Data.Len</th>
<th>Ext_User_Prm_Data.Const(0)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x03</td>
<td>0xDA, 0x17, 0x00;</td>
</tr>
</tbody>
</table>

### Diagnosis

<table>
<thead>
<tr>
<th>E1...E4</th>
<th>d1</th>
<th>d2</th>
<th>d3</th>
<th>d4</th>
<th>Identifier 000...063</th>
<th>AC500-Display</th>
<th>&lt;-- Display in</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class</td>
<td>Comp</td>
<td>Dev</td>
<td>Mod</td>
<td>Ch</td>
<td>Err</td>
<td>PS501 PLC Browser</td>
<td></td>
</tr>
<tr>
<td>Byte 6</td>
<td>Bit 6...7</td>
<td>Byte 3</td>
<td>Byte 4</td>
<td>Byte 5</td>
<td>Byte 6 Bit 0...5</td>
<td>PNIO diagnosis block</td>
<td></td>
</tr>
</tbody>
</table>

#### Module error

<table>
<thead>
<tr>
<th>Class</th>
<th>Interface</th>
<th>Device</th>
<th>Module</th>
<th>Channel</th>
<th>Error Identifier</th>
<th>Error message</th>
<th>Remedy</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>14</td>
<td>1...10</td>
<td>31</td>
<td>31</td>
<td>19</td>
<td>Checksum error in the I/O module</td>
<td>Replace I/O module</td>
</tr>
<tr>
<td></td>
<td>11/12</td>
<td>ADR</td>
<td>1...10</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>14</td>
<td>1...10</td>
<td>31</td>
<td>31</td>
<td>43</td>
<td>Internal error in the module</td>
<td>Replace I/O module</td>
</tr>
<tr>
<td></td>
<td>11/12</td>
<td>ADR</td>
<td>1...10</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>14</td>
<td>1...10</td>
<td>31</td>
<td>31</td>
<td>9</td>
<td>Overflow diagnosis buffer</td>
<td>Restart</td>
</tr>
<tr>
<td></td>
<td>11/12</td>
<td>ADR</td>
<td>1...10</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>14</td>
<td>1...10</td>
<td>31</td>
<td>31</td>
<td>26</td>
<td>Parameter error</td>
<td>Check master</td>
</tr>
<tr>
<td></td>
<td>11/12</td>
<td>ADR</td>
<td>1...10</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Remarks:

1) In AC500 the following interface identifier applies:
   14 = I/O bus, 11 = COM1 (e.g. CS31 bus), 12 = COM2.
   The PNIO diagnosis block does not contain this identifier.

2) With "Device" the following allocation applies:
   31 = module itself,
   1...10 = decentralized communication interface module 1...10,
   ADR = hardware address (e.g. of the DC551-CS31)

3) With "Module" the following allocation applies depending on the master:
   Module error: I/O bus or PNIO: 31 = module itself; COM1/COM2: 1...10 = expansion 1...10

4) In case of module errors, with channel "31 = module itself" is output.
State LEDs

<table>
<thead>
<tr>
<th>LED</th>
<th>State</th>
<th>Color</th>
<th>LED = OFF</th>
<th>LED = ON</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Inputs I0...I7</td>
<td>Digital input</td>
<td>Yellow</td>
<td>Input is OFF</td>
</tr>
</tbody>
</table>

In the undefined signal range, the state LED for the inputs can be ON although the input state detected by the module is OFF.

Technical data

The System Data of AC500-eCo apply

Only additional details are therefore documented below.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Galvanic isolation</td>
<td>Yes, between the input group and the rest of the module</td>
</tr>
<tr>
<td>Isolated groups</td>
<td>1 (8 channels per group)</td>
</tr>
<tr>
<td>Current consumption from 24 VDC power supply at the L+/UP and M/ZP terminals of the CPU/bus module</td>
<td>Ca. 10 mA</td>
</tr>
<tr>
<td>Max. power dissipation within the module</td>
<td>1.6 W</td>
</tr>
<tr>
<td>Weight</td>
<td>Ca. 110 g</td>
</tr>
<tr>
<td>Mounting position</td>
<td>Horizontal or vertical</td>
</tr>
<tr>
<td>Cooling</td>
<td>The natural convection cooling must not be hindered by cable ducts or other parts in the switch-gear cabinet.</td>
</tr>
</tbody>
</table>

Technical data of the digital inputs

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of channels per module</td>
<td>8 inputs (24 VDC / 24 VAC)</td>
</tr>
<tr>
<td>Distribution of the channels into groups</td>
<td>1 (8 channels per group)</td>
</tr>
<tr>
<td>Connections of the channels I0 to I7</td>
<td>Terminals 2 to 9</td>
</tr>
<tr>
<td>Reference potential for the channels I0 to I7</td>
<td>Terminal 1 (plus or negative pole of the process supply voltage, signal name C0..7)</td>
</tr>
<tr>
<td>Indication of the input signals</td>
<td>1 yellow LED per channel; the LED is ON when the input signal is high (signal 1). The module is powered through the I/O bus.</td>
</tr>
<tr>
<td>Monitoring point of input indicator</td>
<td>LED is part of the input circuitry</td>
</tr>
<tr>
<td>Input type according to EN 61131-2</td>
<td>Type 1 source Type 1 sink Type 1 AC ¹)</td>
</tr>
<tr>
<td>Parameter</td>
<td>Value</td>
</tr>
<tr>
<td>-----------------------------------</td>
<td>--------------------------------------------</td>
</tr>
<tr>
<td>Input signal range</td>
<td>-24 VDC</td>
</tr>
<tr>
<td>Signal 0</td>
<td>-5 V...+3 V</td>
</tr>
<tr>
<td>Undefined signal</td>
<td>-15 V...-5 V</td>
</tr>
<tr>
<td>Signal 1</td>
<td>-30 V...-15 V</td>
</tr>
<tr>
<td>Input current per channel</td>
<td></td>
</tr>
<tr>
<td>Input voltage 24 V</td>
<td>Typ. 5 mA</td>
</tr>
<tr>
<td>Input voltage 5 V</td>
<td>Typ. 1 mA</td>
</tr>
<tr>
<td>Input voltage 14 V</td>
<td></td>
</tr>
<tr>
<td>Input voltage 15 V</td>
<td>&gt; 2.5 mA</td>
</tr>
<tr>
<td>Input voltage 27 V</td>
<td></td>
</tr>
<tr>
<td>Input voltage 30 V</td>
<td>&lt; 8 mA</td>
</tr>
<tr>
<td>Max. permissible leakage current</td>
<td>1 mA</td>
</tr>
<tr>
<td>(at 2-wire proximity switches)</td>
<td></td>
</tr>
<tr>
<td>Input delay (0-&gt;1 or 1-&gt;0)</td>
<td>Typ. 8 ms</td>
</tr>
<tr>
<td>Input data length</td>
<td>1 byte</td>
</tr>
<tr>
<td>Max. cable length</td>
<td></td>
</tr>
<tr>
<td>Shielded</td>
<td>500 m</td>
</tr>
<tr>
<td>Unshielded</td>
<td>300 m</td>
</tr>
</tbody>
</table>

1) When inputs are used with 24 VAC, external surge limiting filters are required.

**Ordering data**

<table>
<thead>
<tr>
<th>Part no.</th>
<th>Description</th>
<th>Product life cycle phase *)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1TNE 968 902 R2101</td>
<td>DI561, digital input module, 8 DI, 24 VDC / 24 VAC</td>
<td>Active</td>
</tr>
<tr>
<td>1TNE 968 901 R3101</td>
<td>Terminal block TA563-9, 9 pins, screw front, cable side, 6 pieces per unit</td>
<td>Active</td>
</tr>
<tr>
<td>1TNE 968 901 R3103</td>
<td>Terminal block TA564-9, 9 pins, screw front, cable front, 6 pieces per unit</td>
<td>Active</td>
</tr>
<tr>
<td>1TNE 968 901 R3105</td>
<td>Terminal block TA565-9, 9 pins, spring front, cable front, 6 pieces per unit</td>
<td>Active</td>
</tr>
</tbody>
</table>

*) For planning and commissioning of new installations use modules in Active status only.

**DI562 - Digital input module**

- 16 digital inputs 24 VDC / 24 VAC (I0 to I15) in 2 groups
- Group-wise electrically isolated
1. I/O bus
2. 16 yellow LEDs to display the signal states of the inputs I0 to I15
3. Terminal number
4. Allocation of signal name
5. Terminal block for input signals (9-pin)
6. Terminal block for input signals (11-pin)
7. 2 holes for wall-mounting with screws
8. DIN rail

Intended purpose

The device can be used as a decentralized I/O extension module for S500 Communication Interface Modules (e.g., CI592-CS31, CI501-PNIO, CI541-DP, CI581-CN) or as a centralized extension module for AC500 CPUs (PM5xx).

The inputs are group-wise electrically isolated from each other.
The other electronic circuitry of the module is electrically isolated from the inputs.

The I/O module must not be used as a decentralized I/O module with CI590-CS31-HA bus modules.
Functionality

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>LED displays</td>
<td>For signal states</td>
</tr>
<tr>
<td>Internal power supply</td>
<td>Via I/O bus</td>
</tr>
<tr>
<td>External power supply</td>
<td>Not necessary</td>
</tr>
</tbody>
</table>

Electrical connection

For a detailed description of the mounting, disassembly and electrical connection of the module, please refer to the System Assembly chapter.

The electrical connection is carried out by using removable 9-pin and 11-pin terminal blocks. These terminal blocks differ in their connection system (spring terminals or screw-type terminals, cable mounting from the front or from the side). For more information, please refer to the chapter Terminal Blocks for S500-eCo I/O Modules. The terminal blocks are not included in the module's scope of delivery and must be ordered separately.

The following block diagram shows the internal construction of the digital inputs:

The assignment of the terminals:
<table>
<thead>
<tr>
<th>Terminal</th>
<th>Signal</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>C0...7</td>
<td>Input common for signals I0 to I7</td>
</tr>
<tr>
<td>2</td>
<td>I0</td>
<td>Input signal I0</td>
</tr>
<tr>
<td>3</td>
<td>I1</td>
<td>Input signal I1</td>
</tr>
<tr>
<td>4</td>
<td>I2</td>
<td>Input signal I2</td>
</tr>
<tr>
<td>5</td>
<td>I3</td>
<td>Input signal I3</td>
</tr>
<tr>
<td>6</td>
<td>I4</td>
<td>Input signal I4</td>
</tr>
<tr>
<td>7</td>
<td>I5</td>
<td>Input signal I5</td>
</tr>
<tr>
<td>8</td>
<td>I6</td>
<td>Input signal I6</td>
</tr>
<tr>
<td>9</td>
<td>I7</td>
<td>Input signal I7</td>
</tr>
<tr>
<td>10</td>
<td>C8...15</td>
<td>Input common for signals I8 to I15</td>
</tr>
<tr>
<td>11</td>
<td>I8</td>
<td>Input signal I8</td>
</tr>
<tr>
<td>12</td>
<td>I9</td>
<td>Input signal I9</td>
</tr>
<tr>
<td>13</td>
<td>I10</td>
<td>Input signal I10</td>
</tr>
<tr>
<td>14</td>
<td>I11</td>
<td>Input signal I11</td>
</tr>
<tr>
<td>15</td>
<td>I12</td>
<td>Input signal I12</td>
</tr>
<tr>
<td>16</td>
<td>I13</td>
<td>Input signal I13</td>
</tr>
<tr>
<td>17</td>
<td>I14</td>
<td>Input signal I14</td>
</tr>
<tr>
<td>18</td>
<td>I15</td>
<td>Input signal I15</td>
</tr>
<tr>
<td>19</td>
<td>---</td>
<td>Reserved</td>
</tr>
<tr>
<td>20</td>
<td>---</td>
<td>Reserved</td>
</tr>
</tbody>
</table>

The internal power supply voltage for the module's circuitry is carried out via the I/O bus (provided by a bus module or a CPU). Thus, the current consumption from 24 VDC power supply at the terminals L+/UP and M/ZP of the CPU/bus module increases by 10 mA per DI562. An external power supply connection is not needed.

**WARNING!**

**Removal/Insertion under power**

The devices are not designed for removal or insertion under power. Because of unforeseeable consequences, it is not allowed to plug or unplug devices with the power being ON.

Make sure that all voltage sources (supply and process voltage) are switched off before you

- connect or disconnect any signal or terminal block
- remove, mount or replace a module.

Disconnecting any powered devices while energized in a hazardous location could result in an electric arc, which could create a flammable ignition resulting in fire or explosion.

Make sure that power is removed and that the area has been thoroughly checked to ensure that flammable materials are not present prior to proceeding.

The devices must not be opened when in operation. The same applies to the network interfaces.


**NOTICE!**

**Risk of damaging the PLC modules!**

Overvoltages and short circuits might damage the PLC modules.
- Make sure that all voltage sources (supply voltage and process supply voltage) are switched off before you begin with operations at the system.
- Never connect any voltages or signals to reserved terminals (marked with ---). Reserved terminals may carry internal voltages.

The module provides several diagnosis functions \(\Downarrow\) *Chapter 1.5.3.6.1.4.6 “Diagnosis” on page 2358.*

The digital inputs can be used as source inputs or as sink inputs.

**NOTICE!**

**Risk of malfunctions in the plant!**

A ground closure, e. g. caused by a damaged cable insulation, can bridge switches accidentally.

Use sink inputs when possible or make sure that, in case of error, there will be no risks to persons or plant.

The following figure shows the electrical connection of the digital input module DI562:

![Electrical connection of DI562](image-url)
The meaning of the LEDs is described in section State LEDs \( \cong \) Chapter 1.5.3.6.1.4.7 “State LEDs” on page 2358.

**I/O configuration**

The module itself does not store configuration data. It receives its parameterization data from the master device of the I/O bus (CPU or bus module) during power-up of the system.

Hence, replacing I/O modules is possible without any re-parameterization via software.

*If the external power supply voltage via UP/ZP terminals fails, the I/O module loses its configuration data. The whole station has to be switched off and on again to re-configure the module.*

**Parameterization**

The arrangement of the parameter data is performed with Automation Builder software.

The parameter data directly influences the functionality of modules.

For non-standard applications, it is necessary to adapt the parameters to your system configuration.

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>Internal Value</th>
<th>Internal Value, Type</th>
<th>Default</th>
<th>Min.</th>
<th>Max.</th>
<th>EDS Slot Index</th>
</tr>
</thead>
<tbody>
<tr>
<td>Module ID</td>
<td>Internal</td>
<td>6110 1)</td>
<td>WORD</td>
<td>6110</td>
<td>0</td>
<td>65535</td>
<td>xx01</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0x17DE</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ignore module</td>
<td>No</td>
<td>0</td>
<td>BYTE</td>
<td>No (0x00)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Yes</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Parameter length 2)</td>
<td>Internal</td>
<td>1 - CPU</td>
<td>BYTE</td>
<td>0</td>
<td>0</td>
<td>255</td>
<td>xx02 3)</td>
</tr>
</tbody>
</table>

Remarks:

1) With CS31 and addresses less than 70, the value is increased by 1

2) The module has no additional user-configurable parameters

3) Value is hexadecimal: HighByte is slot (xx: 0...7), LowByte is index (1...n)

**GSD file:**

```plaintext
Ext_User_Prm_Data_Len = 0x03
Ext_User_Prm_Data_Const(0) = 0xDF, 0x17, 0x00;
```
Diagnosis

<table>
<thead>
<tr>
<th>E1...E4</th>
<th>d1</th>
<th>d2</th>
<th>d3</th>
<th>d4</th>
<th>Identifier 000...063</th>
<th>AC500-Display</th>
<th>&lt;- Display in</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class</td>
<td>Comp</td>
<td>Dev</td>
<td>Mod</td>
<td>Ch</td>
<td>Err</td>
<td>PS501 PLC Browser</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Byte 6 Bit 6...7</th>
<th>-</th>
<th>Byte 3</th>
<th>Byte 4</th>
<th>Byte 5</th>
<th>Byte 6 Bit 0...5</th>
<th>PNIO diagnosis block</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class</td>
<td>Interface</td>
<td>Device</td>
<td>Module</td>
<td>Channel</td>
<td>Error Identifier</td>
<td>Error message</td>
</tr>
<tr>
<td></td>
<td>1)</td>
<td>2)</td>
<td>3)</td>
<td>4)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Module error DI562

<table>
<thead>
<tr>
<th>3</th>
<th>14</th>
<th>1...10</th>
<th>31</th>
<th>31</th>
<th>19</th>
<th>Checksum error in the I/O module</th>
<th>Replace I/O module</th>
</tr>
</thead>
<tbody>
<tr>
<td>11 / 12</td>
<td>ADR</td>
<td>1...10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>3</th>
<th>14</th>
<th>1...10</th>
<th>31</th>
<th>31</th>
<th>43</th>
<th>Internal error in the module</th>
<th>Replace I/O module</th>
</tr>
</thead>
<tbody>
<tr>
<td>11 / 12</td>
<td>ADR</td>
<td>1...10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>3</th>
<th>14</th>
<th>1...10</th>
<th>31</th>
<th>31</th>
<th>9</th>
<th>Overflow diagnosis buffer</th>
<th>Restart</th>
</tr>
</thead>
<tbody>
<tr>
<td>11 / 12</td>
<td>ADR</td>
<td>1...10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>3</th>
<th>14</th>
<th>1...10</th>
<th>31</th>
<th>31</th>
<th>26</th>
<th>Parameter error</th>
<th>Check master</th>
</tr>
</thead>
<tbody>
<tr>
<td>11 / 12</td>
<td>ADR</td>
<td>1...10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Remarks:

1) In AC500 the following interface identifier applies:
14 = I/O bus, 11 = COM1 (e.g. CS31 bus), 12 = COM2.
The PNIO diagnosis block does not contain this identifier.

2) With "Device" the following allocation applies:
31 = module itself,
1...10 = decentralized communication interface module 1...10,
ADR = hardware address (e.g. of the DC551-CS31)

3) With "Module" the following allocation applies depending on the master:
Module error: I/O bus or PNIO: 31 = module itself; COM1/COM2: 1...10 = expansion 1...10

4) In case of module errors, with channel "31 = module itself" is output.

State LEDs

<table>
<thead>
<tr>
<th>LED DI562</th>
<th>Inputs I0...I15</th>
<th>Digital input</th>
<th>Yellow</th>
<th>LED = OFF</th>
<th>LED = ON</th>
</tr>
</thead>
</table>

Remarks:

2358 3ADR010583, 1, en_US 2020/12/10
In the undefined signal range, the state LED for the inputs can be ON although the input state detected by the module is OFF.

Technical data

The System Data of AC500-eCo apply. Only additional details are therefore documented below.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Galvanic isolation</td>
<td>Yes, between the input groups and the rest of the module</td>
</tr>
<tr>
<td>Isolated groups</td>
<td>2 (8 channels per group)</td>
</tr>
<tr>
<td>Current consumption from 24 VDC power supply at the L+/UP and M/ZP terminals of the CPU/bus module</td>
<td>Ca. 10 mA</td>
</tr>
<tr>
<td>Max. power dissipation within the module</td>
<td>3.2 W</td>
</tr>
<tr>
<td>Weight</td>
<td>Ca. 115 g</td>
</tr>
<tr>
<td>Mounting position</td>
<td>Horizontal or vertical</td>
</tr>
<tr>
<td>Cooling</td>
<td>The natural convection cooling must not be hindered by cable ducts or other parts in the switch-gear cabinet.</td>
</tr>
</tbody>
</table>

Technical data of the digital inputs

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of channels per module</td>
<td>16 inputs (24 VDC / 24 VAC)</td>
</tr>
<tr>
<td>Distribution of the channels into groups</td>
<td>2 (8 channels per group)</td>
</tr>
<tr>
<td>Connections of the channels I0 to I7</td>
<td>Terminals 2 to 9</td>
</tr>
<tr>
<td>Connections of the channels I8 to I15</td>
<td>Terminals 11 to 18</td>
</tr>
<tr>
<td>Reference potential for the channels I0 to I7</td>
<td>Terminal 1 (positive or negative pole of the process supply voltage, signal name C0..7)</td>
</tr>
<tr>
<td>Reference potential for the channels I8 to I15</td>
<td>Terminal 10 (positive or negative pole of the process supply voltage, signal name C8..15)</td>
</tr>
<tr>
<td>Indication of the input signals</td>
<td>1 yellow LED per channel; the LED is ON when the input signal is high (signal 1). The module is powered through the I/O bus.</td>
</tr>
<tr>
<td>Monitoring point of input indicator</td>
<td>LED is part of the input circuitry</td>
</tr>
<tr>
<td>Input type according to EN 61131-2</td>
<td>Type 1 source, Type 1 sink, Type 1 AC ¹)</td>
</tr>
<tr>
<td>Input signal range</td>
<td>-24 VDC, +24 VDC, 24 VAC 50/60 Hz</td>
</tr>
<tr>
<td>Signal 0</td>
<td>-5 V...+3 V, -3 V...+5 V, 0 VAC...5 VAC</td>
</tr>
<tr>
<td>Undefined signal</td>
<td>-15 V...-5 V, +5 V...+15 V, 5 VAC...14 VAC</td>
</tr>
<tr>
<td>Signal 1</td>
<td>-30 V...-15 V, +15 V...+30 V, 14 VAC...27 VAC</td>
</tr>
<tr>
<td>Input current per channel</td>
<td></td>
</tr>
</tbody>
</table>
## Parameter

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input voltage 24 V</td>
<td>Typ. 5 mA</td>
</tr>
<tr>
<td>Input voltage 5 V</td>
<td>Typ. 1 mA</td>
</tr>
<tr>
<td>Input voltage 14 V</td>
<td>Typ. 2.7 mA r.m.s.</td>
</tr>
<tr>
<td>Input voltage 15 V</td>
<td>&gt; 2.5 mA</td>
</tr>
<tr>
<td>Input voltage 27 V</td>
<td>Typ. 5.5 mA r.m.s.</td>
</tr>
<tr>
<td>Input voltage 30 V</td>
<td>&lt; 8 mA</td>
</tr>
<tr>
<td>Max. permissible leakage current (at 2-wire proximity switches)</td>
<td>1 mA Typ. 1 mA r.m.s.</td>
</tr>
<tr>
<td>Input delay (0-&gt;1 or 1-&gt;0)</td>
<td>Typ. 8 ms</td>
</tr>
<tr>
<td>Input data length</td>
<td>2 bytes</td>
</tr>
<tr>
<td>Max. cable length</td>
<td></td>
</tr>
<tr>
<td>Shielded</td>
<td>500 m</td>
</tr>
<tr>
<td>Unshielded</td>
<td>300 m</td>
</tr>
</tbody>
</table>

1) When inputs are used with 24 VAC, external surge limiting filters are required.

### Ordering data

<table>
<thead>
<tr>
<th>Part no.</th>
<th>Description</th>
<th>Product life cycle phase *)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1TNE 968 902 R2102</td>
<td>DI562, digital input module, 16 DI, 24 VDC / 24 VAC</td>
<td>Active</td>
</tr>
<tr>
<td>1TNE 968 901 R3101</td>
<td>Terminal block TA563-9, 9 pins, screw front, cable side, 6 pieces per unit</td>
<td>Active</td>
</tr>
<tr>
<td>1TNE 968 901 R3102</td>
<td>Terminal block TA563-11, 11 pins, screw front, cable side, 6 pieces per unit</td>
<td>Active</td>
</tr>
<tr>
<td>1TNE 968 901 R3103</td>
<td>Terminal block TA564-9, 9 pins, screw front, cable front, 6 pieces per unit</td>
<td>Active</td>
</tr>
<tr>
<td>1TNE 968 901 R3104</td>
<td>Terminal block TA564-11, 11 pins, screw front, cable front, 6 pieces per unit</td>
<td>Active</td>
</tr>
<tr>
<td>1TNE 968 901 R3105</td>
<td>Terminal block TA565-9, 9 pins, spring front, cable front, 6 pieces per unit</td>
<td>Active</td>
</tr>
<tr>
<td>1TNE 968 901 R3106</td>
<td>Terminal block TA565-11, 11 pins, spring front, cable front, 6 pieces per unit</td>
<td>Active</td>
</tr>
</tbody>
</table>

*) For planning and commissioning of new installations use modules in Active status only.

### DI571 - Digital input module

- 8 digital inputs 100-240 VAC (I0 to I7) in 8 groups
- Module-wise electrically isolated
Intended purpose

The device can be used as a decentralized I/O extension module for S500 Communication Interface Modules (e.g., CI592-CS31, CI501-PNIO, CI541-DP, CI581-CN) or as a centralized extension module for AC500 CPUs (PM5xx).

The inputs are group-wise electrically isolated from each other.
All other circuitry of the module is electrically isolated from the inputs.

The I/O module must not be used as a decentralized I/O module with CI590-CS31-HA bus modules.
Functionality

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>LED displays</td>
<td>For signal states</td>
</tr>
<tr>
<td>Internal power supply</td>
<td>Via I/O bus</td>
</tr>
<tr>
<td>External power supply</td>
<td>Not necessary</td>
</tr>
</tbody>
</table>

Electrical connection

For a detailed description of the mounting, disassembly and electrical connection of the module, please refer to the System Assembly chapter.

The electrical connection is carried out by using removable 9-pin and 11-pin terminal blocks. These terminal blocks differ in their connection system (spring terminals or screw terminals, cable mounting from the front or from the side). For more information, please refer to the chapter Terminal Blocks for S500-eCo I/O Modules. The terminal blocks are not included in the module's scope of delivery and must be ordered separately.

The following block diagram shows the internal construction of the digital inputs:
Table 541: Assignment of the terminals:

<table>
<thead>
<tr>
<th>Terminal</th>
<th>Signal</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>I0</td>
<td>Input signal I0</td>
</tr>
<tr>
<td>2</td>
<td>N0</td>
<td>Neutral conductor for the input signal I0</td>
</tr>
<tr>
<td>3</td>
<td>I1</td>
<td>Input signal I1</td>
</tr>
<tr>
<td>4</td>
<td>N1</td>
<td>Neutral conductor for the input signal I1</td>
</tr>
<tr>
<td>5</td>
<td>I2</td>
<td>Input signal I2</td>
</tr>
<tr>
<td>6</td>
<td>N2</td>
<td>Neutral conductor for the input signal I2</td>
</tr>
<tr>
<td>7</td>
<td>I3</td>
<td>Input signal I3</td>
</tr>
<tr>
<td>8</td>
<td>N3</td>
<td>Neutral conductor for the input signal I3</td>
</tr>
<tr>
<td>9</td>
<td>---</td>
<td>Reserved</td>
</tr>
<tr>
<td>10</td>
<td>I4</td>
<td>Input signal I4</td>
</tr>
<tr>
<td>11</td>
<td>N4</td>
<td>Neutral conductor for the input signal I4</td>
</tr>
<tr>
<td>12</td>
<td>I5</td>
<td>Input signal I5</td>
</tr>
<tr>
<td>13</td>
<td>N5</td>
<td>Neutral conductor for the input signal I5</td>
</tr>
<tr>
<td>14</td>
<td>I6</td>
<td>Input signal I6</td>
</tr>
<tr>
<td>Terminal</td>
<td>Signal</td>
<td>Description</td>
</tr>
<tr>
<td>---------</td>
<td>--------</td>
<td>--------------------------------------------------</td>
</tr>
<tr>
<td>15</td>
<td>N6</td>
<td>Neutral conductor for the input signal I6</td>
</tr>
<tr>
<td>16</td>
<td>I7</td>
<td>Input signal I7</td>
</tr>
<tr>
<td>17</td>
<td>N7</td>
<td>Neutral conductor for the input signal I7</td>
</tr>
<tr>
<td>18</td>
<td>---</td>
<td>Reserved</td>
</tr>
<tr>
<td>19</td>
<td>---</td>
<td>Reserved</td>
</tr>
<tr>
<td>20</td>
<td>---</td>
<td>Reserved</td>
</tr>
</tbody>
</table>

The internal power supply voltage for the module’s circuitry is carried out via the I/O bus (provided by a bus module or a CPU). Thus, the current consumption from 24 VDC power supply at the terminals L+/UP and M/ZP of the CPU/bus module increases by 10 mA per DI571.

An external power supply connection is not needed.

**WARNING!**
Risk of death by electric shock!
Hazardous voltages can be present at the terminals of the module.
Make sure that all voltage sources (supply voltage and process supply voltage) are switched off before you begin with operations at the system.

**WARNING!**
Removal/Insertion under power
The devices are not designed for removal or insertion under power. Because of unforeseeable consequences, it is not allowed to plug or unplug devices with the power being ON.
Make sure that all voltage sources (supply and process voltage) are switched off before you
- connect or disconnect any signal or terminal block
- remove, mount or replace a module.
Disconnecting any powered devices while energized in a hazardous location could result in an electric arc, which could create a flammable ignition resulting in fire or explosion.
Make sure that power is removed and that the area has been thoroughly checked to ensure that flammable materials are not present prior to proceeding.
The devices must not be opened when in operation. The same applies to the network interfaces.

**NOTICE!**
Risk of damaging the PLC modules!
Overvoltages and short circuits might damage the PLC modules.
- Make sure that all voltage sources (supply voltage and process supply voltage) are switched off before you begin with operations at the system.
- Never connect any voltages or signals to reserved terminals (marked with ---). Reserved terminals may carry internal voltages.

The following figure shows the electrical connection of the digital input module DI571:
**NOTICE!**

Risk of damaging the PLC modules!

The PLC modules will be irreparably damaged if a voltage > 240 V is connected.

Make sure that all inputs are fed from the same phase. The module must not be connected to a 400 V voltage.

The module provides several diagnosis functions [Chapter 1.5.3.6.1.1.5.7 “Diagnosis” on page 2367](#).

The meaning of the LEDs is described in the section State LEDs [Chapter 1.5.3.6.1.1.5.8 “State LEDs” on page 2367](#).
Internal data exchange

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Digital inputs (bytes)</td>
<td>1</td>
</tr>
<tr>
<td>Digital outputs (bytes)</td>
<td>0</td>
</tr>
</tbody>
</table>

I/O configuration

The module itself does not store configuration data. It receives its parameterization data from the master device of the I/O bus (CPU or bus module) during power-up of the system. Hence, replacing I/O modules is possible without any re-parameterization via software.

If the external power supply voltage via UP/ZP terminals fails, the I/O module loses its configuration data. The whole station has to be switched off and on again to re-configure the module.

Parameterization

The arrangement of the parameter data is performed with Automation Builder software. The parameter data directly influences the functionality of the modules.

For non-standard applications, it is necessary to adapt the parameters to your system configuration.

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>Internal Value</th>
<th>Internal Value, Type</th>
<th>Default</th>
<th>Min.</th>
<th>Max.</th>
<th>EDS Slot Index</th>
</tr>
</thead>
<tbody>
<tr>
<td>Module ID</td>
<td>Internal</td>
<td>6115 1)</td>
<td>WORD</td>
<td>6115 0x17E3</td>
<td>0</td>
<td>65535</td>
<td>xx01</td>
</tr>
<tr>
<td>Ignore module</td>
<td>No Yes</td>
<td>0 1</td>
<td>BYTE</td>
<td>No (0x00)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Parameter length</td>
<td>Internal</td>
<td>1 - CPU</td>
<td>BYTE</td>
<td>0</td>
<td>0</td>
<td>255</td>
<td>xx02 3)</td>
</tr>
</tbody>
</table>

1) with CS31 and addresses less than 70, the value is increased by 1
2) the module has no additional user-configurable parameters
3) Value is hexadecimal: HighByte is slot (xx: 0...7), LowByte is index (1...n)

GSD file:

```plaintext
Ext_User_Prm_Data_Len = 0x03
Ext_User_Prm_Data_Const(0) = 0xDF, 0x17, 0x00;
```
**Diagnosis**

<table>
<thead>
<tr>
<th>E1...E4</th>
<th>d1</th>
<th>d2</th>
<th>d3</th>
<th>d4</th>
<th>Identifier 000...063</th>
<th>AC500-Display</th>
<th>← Display in</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class</td>
<td>Comp</td>
<td>Dev</td>
<td>Mod</td>
<td>Ch</td>
<td>Err</td>
<td>PS501 PLC Browser</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Byte 6 Bit 6...7</th>
<th>- Byte 3 Byte 4 Byte 5 Byte 6 Bit 0...5</th>
<th>PNIO diagnosis block</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Class</th>
<th>Interface</th>
<th>Device</th>
<th>Module</th>
<th>Channel</th>
<th>Error Identifier</th>
<th>Error message</th>
<th>Remedy</th>
</tr>
</thead>
<tbody>
<tr>
<td>1)</td>
<td>2)</td>
<td>3)</td>
<td>4)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Module error

<table>
<thead>
<tr>
<th>3</th>
<th>14</th>
<th>1...10</th>
<th>31</th>
<th>31</th>
<th>19</th>
<th>Checksum error in the I/O module</th>
<th>Replace I/O module</th>
</tr>
</thead>
<tbody>
<tr>
<td>11 / 12</td>
<td>ADR</td>
<td>1...10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>3</th>
<th>14</th>
<th>1...10</th>
<th>31</th>
<th>31</th>
<th>43</th>
<th>Internal error in the module</th>
<th>Replace I/O module</th>
</tr>
</thead>
<tbody>
<tr>
<td>11 / 12</td>
<td>ADR</td>
<td>1...10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>3</th>
<th>14</th>
<th>1...10</th>
<th>31</th>
<th>31</th>
<th>9</th>
<th>Overflow diagnosis buffer</th>
<th>Restart</th>
</tr>
</thead>
<tbody>
<tr>
<td>11 / 12</td>
<td>ADR</td>
<td>1...10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>3</th>
<th>14</th>
<th>1...10</th>
<th>31</th>
<th>31</th>
<th>26</th>
<th>Parameter error</th>
<th>Check master</th>
</tr>
</thead>
<tbody>
<tr>
<td>11 / 12</td>
<td>ADR</td>
<td>1...10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Remarks:**

1) In AC500 the following interface identifier applies:
14 = I/O bus, 11 = COM1 (e.g. CS31 bus), 12 = COM2.
The PNIO diagnosis block does not contain this identifier.

2) With "Device" the following allocation applies:
31 = module itself, 1...10 = decentralized communication interface module 1...10, ADR = hardware address (e.g. of the DC551-CS31)

3) With "Module" the following allocation applies depending on the master:
Module error: I/O bus or PNIO: 31 = module itself; COM1/COM2: 1...10 = expansion 1...10

4) In case of module errors, with channel "31 = Module itself" is output.

### State LEDs

<table>
<thead>
<tr>
<th>LED</th>
<th>State</th>
<th>Color</th>
<th>LED = OFF</th>
<th>LED = ON</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Inputs I0...I7</td>
<td>Digital input</td>
<td>Yellow</td>
<td>Input is OFF (the input voltage is only displayed if the supply voltage of the module is ON)</td>
</tr>
</tbody>
</table>
Technical data

The System Data of AC500-eCo apply

Only additional details are therefore documented below.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Galvanic isolation</td>
<td>Yes, between the channels and the rest of the module</td>
</tr>
<tr>
<td>Isolated groups</td>
<td>8 (1 channel per group)</td>
</tr>
<tr>
<td>Current consumption from 24 VDC power supply at the L+/UP and M/ZP terminals of the CPU/bus module</td>
<td>Ca. 10 mA</td>
</tr>
<tr>
<td>Max. power dissipation within the module</td>
<td>On request</td>
</tr>
<tr>
<td>Weight</td>
<td>Ca. 135 g</td>
</tr>
<tr>
<td>Mounting position</td>
<td>Horizontal or vertical</td>
</tr>
<tr>
<td>Cooling</td>
<td>The natural convection cooling must not be hindered by cable ducts or other parts in the switch-gear cabinet.</td>
</tr>
</tbody>
</table>

Technical data of the digital inputs

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of channels per module</td>
<td>8 AC inputs (100-240 VAC)</td>
</tr>
<tr>
<td>Distribution of the channels into groups</td>
<td>8 (1 channel per group)</td>
</tr>
<tr>
<td>Input voltage range</td>
<td>0 VAC..264 VAC (47 Hz...63 Hz)</td>
</tr>
<tr>
<td>Input current per channel (typically at 25 °C)</td>
<td>&lt;5 mA (at 40 VAC)</td>
</tr>
<tr>
<td></td>
<td>&gt;6 mA (at 159 VAC, 50 Hz)</td>
</tr>
<tr>
<td></td>
<td>&gt;7 mA (at 159 VAC, 60 Hz)</td>
</tr>
<tr>
<td>Connections of the channels I0 to I7</td>
<td>Terminals 1, 3, 5, 7, 10, 12, 14, 16</td>
</tr>
<tr>
<td>Reference potential for the channels I0 to I7</td>
<td>Terminals 2, 4, 6, 8, 11, 13, 15, 17</td>
</tr>
<tr>
<td>Indication of the input signals</td>
<td>1 yellow LED per channel; the LED is ON when the input signal is high (signal 1)</td>
</tr>
<tr>
<td>Input type according to EN 61131-2</td>
<td>Type 1</td>
</tr>
<tr>
<td>Input signal range</td>
<td></td>
</tr>
<tr>
<td>Signal 0 (max.)</td>
<td>20 VAC</td>
</tr>
<tr>
<td>Undefined signal</td>
<td>20 VAC &lt; U &lt; 79 VAC</td>
</tr>
<tr>
<td>Signal 1 (min.)</td>
<td>79 VAC</td>
</tr>
<tr>
<td>Input delay</td>
<td></td>
</tr>
<tr>
<td>Signal 0 -&gt; 1</td>
<td>Typ. 15 ms</td>
</tr>
<tr>
<td>Signal 1 -&gt; 0</td>
<td>Typ. 30 ms</td>
</tr>
<tr>
<td>Input data length</td>
<td>1 byte</td>
</tr>
<tr>
<td>Max. permissible leakage current (at 2-wire proximity switches)</td>
<td>1 mA</td>
</tr>
<tr>
<td>Max. cable length</td>
<td></td>
</tr>
<tr>
<td>Parameter</td>
<td>Value</td>
</tr>
<tr>
<td>----------------</td>
<td>-------</td>
</tr>
<tr>
<td>Shielded</td>
<td>500 m</td>
</tr>
<tr>
<td>Unshielded</td>
<td>300 m</td>
</tr>
</tbody>
</table>

## Ordering data

<table>
<thead>
<tr>
<th>Part no.</th>
<th>Description</th>
<th>Product life cycle phase *)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1TNE 968 902 R2103</td>
<td>DI571, digital input module, 8 DI, 100 VAC...240 VAC</td>
<td>Active</td>
</tr>
<tr>
<td>1TNE 968 901 R3101</td>
<td>Terminal block TA563-9, 9 pins, screw front, cable side, 6 pieces per unit</td>
<td>Active</td>
</tr>
<tr>
<td>1TNE 968 901 R3102</td>
<td>Terminal block TA563-11, 11 pins, screw front, cable side, 6 pieces per unit</td>
<td>Active</td>
</tr>
<tr>
<td>1TNE 968 901 R3103</td>
<td>Terminal block TA564-9, 9 pins, screw front, cable front, 6 pieces per unit</td>
<td>Active</td>
</tr>
<tr>
<td>1TNE 968 901 R3104</td>
<td>Terminal block TA564-11, 11 pins, screw front, cable front, 6 pieces per unit</td>
<td>Active</td>
</tr>
<tr>
<td>1TNE 968 901 R3105</td>
<td>Terminal block TA565-9, 9 pins, spring front, cable front, 6 pieces per unit</td>
<td>Active</td>
</tr>
<tr>
<td>1TNE 968 901 R3106</td>
<td>Terminal block TA565-11, 11 pins, spring front, cable front, 6 pieces per unit</td>
<td>Active</td>
</tr>
</tbody>
</table>

*) For planning and commissioning of new installations use modules in Active status only.

**DI572 - Digital input module**

- 16 digital inputs 100-240 VAC (I0 to I15) in 2 groups
- Module-wise electrically isolated
1 I/O bus
2 16 yellow LEDs to display the signal states of the inputs I0 to I15
3 Terminal number
4 Allocation of signal name
5 Terminal block for input signals (9-pin)
6 Terminal block for input signals (11-pin)
7 2 holes for wall-mounting with screws
8 DIN rail

**Intended purpose**

The device can be used as a decentralized I/O extension module for S500 Communication Interface Modules (e.g. CI592-CS31, CI501-PNIO, CI541-DP, CI581-CN) or as a centralized extension module for AC500 CPUs (PM5xx).

The inputs are group-wise electrically isolated from each other.

All other circuitry of the module is electrically isolated from the inputs.

*The I/O module must not be used as a decentralized I/O module with CI590-CS31-HA bus modules.*
### Functionality

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>LED displays</td>
<td>For signal states</td>
</tr>
<tr>
<td>Internal power supply</td>
<td>Via I/O bus</td>
</tr>
<tr>
<td>External power supply</td>
<td>Not necessary</td>
</tr>
</tbody>
</table>

### Electrical connection

For a detailed description of the mounting, disassembly and electrical connection of the module, please refer to the System Assembly chapter.

The electrical connection is carried out by using removable 9-pin and 11-pin terminal blocks. These terminal blocks differ in their connection system (spring terminals or screw terminals, cable mounting from the front or from the side). For more information, please refer to the chapter Terminal Blocks for S500-eCo I/O Modules. The terminal blocks are not included in the module's scope of delivery and must be ordered separately.

![Block diagram for the internal construction of the digital inputs.](image)

*Fig. 98: Block diagram for the internal construction of the digital inputs.*
### Table 542: Assignment of the terminals

<table>
<thead>
<tr>
<th>Terminal</th>
<th>Signal</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>I0</td>
<td>Input signal I0</td>
</tr>
<tr>
<td>2</td>
<td>I1</td>
<td>Input signal I1</td>
</tr>
<tr>
<td>3</td>
<td>I2</td>
<td>Input signal I2</td>
</tr>
<tr>
<td>4</td>
<td>I3</td>
<td>Input signal I3</td>
</tr>
<tr>
<td>5</td>
<td>I4</td>
<td>Input signal I4</td>
</tr>
<tr>
<td>6</td>
<td>I5</td>
<td>Input signal I5</td>
</tr>
<tr>
<td>7</td>
<td>I6</td>
<td>Input signal I6</td>
</tr>
<tr>
<td>8</td>
<td>I7</td>
<td>Input signal I7</td>
</tr>
<tr>
<td>9</td>
<td>N0...7</td>
<td>Neutral conductor for the input signals I0...I7</td>
</tr>
<tr>
<td>10</td>
<td>I8</td>
<td>Input signal I8</td>
</tr>
<tr>
<td>11</td>
<td>I9</td>
<td>Input signal I9</td>
</tr>
<tr>
<td>12</td>
<td>I10</td>
<td>Input signal I10</td>
</tr>
<tr>
<td>13</td>
<td>I11</td>
<td>Input signal I11</td>
</tr>
<tr>
<td>14</td>
<td>I12</td>
<td>Input signal I12</td>
</tr>
<tr>
<td>15</td>
<td>I13</td>
<td>Input signal I13</td>
</tr>
<tr>
<td>16</td>
<td>I14</td>
<td>Input signal I14</td>
</tr>
<tr>
<td>17</td>
<td>I15</td>
<td>Input signal I15</td>
</tr>
<tr>
<td>18</td>
<td>N8...15</td>
<td>Neutral conductor for the input signals I8...I15</td>
</tr>
<tr>
<td>19</td>
<td>---</td>
<td>Reserved</td>
</tr>
<tr>
<td>20</td>
<td>---</td>
<td>Reserved</td>
</tr>
</tbody>
</table>

The internal power supply voltage for the module's circuitry is carried out via the I/O bus (provided by a bus module or a CPU). Thus, the current consumption from 24 VDC power supply at the terminals L+/UP and M/ZP of the CPU/bus module increases by 10 mA per DI572.

An external power supply connection is not needed.

**WARNING!**

**Risk of death by electric shock!**

Hazardous voltages can be present at the terminals of the module.

Make sure that all voltage sources (supply voltage and process supply voltage) are switched off before you begin with operations at the system.
WARNING!
Removal/Insertion under power
The devices are not designed for removal or insertion under power. Because of unforeseeable consequences, it is not allowed to plug or unplug devices with the power being ON.

Make sure that all voltage sources (supply and process voltage) are switched off before you
- connect or disconnect any signal or terminal block
- remove, mount or replace a module.

Disconnecting any powered devices while energized in a hazardous location could result in an electric arc, which could create a flammable ignition resulting in fire or explosion.

Make sure that power is removed and that the area has been thoroughly checked to ensure that flammable materials are not present prior to proceeding.

The devices must not be opened when in operation. The same applies to the network interfaces.

NOTICE!
Risk of damaging the PLC modules!
- Overvoltages and short circuits might damage the PLC modules.
  - Make sure that all voltage sources (supply voltage and process supply voltage) are switched off before you begin with operations at the system.
  - Never connect any voltages or signals to reserved terminals (marked with ---). Reserved terminals may carry internal voltages.
**NOTICE!**

**Risk of damaging the PLC modules!**

The PLC modules will be irreparably damaged if a voltage > 240 V is connected.

Make sure that all inputs are fed from the same phase. The module must not be connected to a 400 V voltage.

The module provides several diagnosis functions - Chapter 1.5.3.6.1.1.6.6 “Diagnosis” on page 2376.

**I/O configuration**

The module itself does not store configuration data. It receives its parameterization data from the master device of the I/O bus (CPU or bus module) during power-up of the system.
Hence, replacing I/O modules is possible without any re-parameterization via software.

If the external power supply voltage via UP/ZP terminals fails, the I/O module loses its configuration data. The whole station has to be switched off and on again to re-configure the module.

Parameterization

The arrangement of the parameter data is performed with Automation Builder software. The parameter data directly influences the functionality of modules. For non-standard applications, it is necessary to adapt the parameters to your system configuration.

<table>
<thead>
<tr>
<th>Parameter name</th>
<th>Value</th>
<th>Internal value</th>
<th>Data type of internal value</th>
<th>Default value</th>
<th>Min.</th>
<th>Max.</th>
<th>EDS Slot Index</th>
</tr>
</thead>
<tbody>
<tr>
<td>Module ID</td>
<td>Internal</td>
<td>6160 1)</td>
<td>WORD</td>
<td>6160 0x1810</td>
<td>0</td>
<td>65535</td>
<td>xx01 2)</td>
</tr>
<tr>
<td>Ignore module</td>
<td>No</td>
<td>0</td>
<td>BYTE</td>
<td>No 0x00</td>
<td>-</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Yes</td>
<td>1</td>
<td></td>
<td></td>
<td>-</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>Parameter length</td>
<td>Internal</td>
<td>3</td>
<td>BYTE</td>
<td>3</td>
<td>0</td>
<td>255</td>
<td>xx02 2)</td>
</tr>
<tr>
<td>Input delay</td>
<td>20 ms</td>
<td>0</td>
<td>BYTE</td>
<td>20 ms 0x00</td>
<td>0</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>100 ms</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td>-</td>
<td></td>
</tr>
</tbody>
</table>

1) With CS31 and addresses less than 70, the value is increased by 1.
2) Value is hexadecimal: HighByte is slot (xx: 0...7), LowByte is index (1...n).

GSD file:

```
Ext_Module_Prm_Data_Len = 7
Ext_User_Prm_Data_Const(0) = 0x18, 0x11, 0x00, 0x03, 0x00, 0x00, 0x00;
```
Diagnosis

<table>
<thead>
<tr>
<th>E1...E4</th>
<th>d1</th>
<th>d2</th>
<th>d3</th>
<th>d4</th>
<th>Identifier 000...063</th>
<th>AC500-Display</th>
<th>← Display in</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class</td>
<td>Comp</td>
<td>Dev</td>
<td>Mod</td>
<td>Ch</td>
<td>Err</td>
<td>PS501 PLC Browser</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Byte 6 Bit 6...7</th>
<th>-</th>
<th>Byte 3</th>
<th>Byte 4</th>
<th>Byte 5</th>
<th>Byte 6 Bit 0...5</th>
<th>PNIO diagnosis block</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class</td>
<td>Interface</td>
<td>Device</td>
<td>Module</td>
<td>Channel</td>
<td>Error Identifier</td>
<td>Error message</td>
</tr>
<tr>
<td>1)</td>
<td>2)</td>
<td>3)</td>
<td>4)</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

| 3 | 14 | 1...10 | 31 | 31 | 19 | Checksum error in the I/O module | Replace I/O module |
| 11 / 12 | ADR | 1...10 |
| 3 | 14 | 1...10 | 31 | 31 | 43 | Internal error in the module | Replace I/O module |
| 11 / 12 | ADR | 1...10 |
| 3 | 14 | 1...10 | 31 | 31 | 9 | Overflow diagnosis buffer | Restart |
| 11 / 12 | ADR | 1...10 |
| 3 | 14 | 1...10 | 31 | 31 | 26 | Parameter error | Check master |
| 11 / 12 | ADR | 1...10 |

Remarks:

Parameter | Remark
---|---
1) | In AC500 the following interface identifier applies:
14 = I/O bus, 11 = COM1 (e.g. CS31-Bus), 12 = COM2.
The PNIO diagnosis block does not contain this identifier.

2) | With "Device" the following allocation applies:
31 = module itself, 1...10 = decentralized communication interface module 1...10,
ADR = hardware address (e.g. of the DC551-CS31)

3) | With "Module" the following allocation applies depending on the master:
module error: I/O bus or PNIO: 31 = module itself; COM1/COM2: 1...10 = expansion 1...10

4) | In case of module errors, with channel "31 = module itself" is output.

State LEDs

<table>
<thead>
<tr>
<th>LED</th>
<th>State</th>
<th>Color</th>
<th>LED = OFF</th>
<th>LED = ON</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inputs I0...I15</td>
<td>Digital input</td>
<td>Yellow</td>
<td>Input is OFF</td>
<td>Input is ON</td>
</tr>
</tbody>
</table>

(The input voltage is only displayed if the supply voltage of the module is ON)
Technical data

The System Data of AC500-eCo apply
Only additional details are therefore documented below.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Galvanic isolation</td>
<td>Yes, between the input groups and the rest of the module</td>
</tr>
<tr>
<td>Isolated groups</td>
<td>2 (8 channels per group)</td>
</tr>
<tr>
<td>Current consumption</td>
<td>Ca. 10 mA</td>
</tr>
<tr>
<td>Max. power dissipation</td>
<td>6 W</td>
</tr>
<tr>
<td>Weight</td>
<td>Ca. 222 g</td>
</tr>
<tr>
<td>Mounting position</td>
<td>Horizontal or vertical</td>
</tr>
<tr>
<td>Cooling</td>
<td>The natural convection cooling must not be hindered by cable ducts or other parts in the switch-gear cabinet.</td>
</tr>
</tbody>
</table>

Technical data of the digital inputs

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of channels per module</td>
<td>16 AC inputs (100-240 VAC)</td>
</tr>
<tr>
<td>Distribution of the channels into groups</td>
<td>2 (8 channels per group)</td>
</tr>
<tr>
<td>Input voltage range</td>
<td>0 VAC...264 VAC (47 Hz...63 Hz)</td>
</tr>
<tr>
<td>Input current per channel</td>
<td>&lt; 3 mA (at 40 VAC)</td>
</tr>
<tr>
<td>Connections of the channels I0..I7</td>
<td>Terminals 1...8</td>
</tr>
<tr>
<td>Connections of the channels I8...I15</td>
<td>Terminals 10...17</td>
</tr>
<tr>
<td>Reference potential for the channels I0..I7</td>
<td>Terminal 9</td>
</tr>
<tr>
<td>Reference potential for the channels I8...I15</td>
<td>Terminal 18</td>
</tr>
<tr>
<td>Indication of the input signals</td>
<td>1 yellow LED per channel. The LED is on when the input signal is high (signal 1).</td>
</tr>
<tr>
<td>Input type according to EN 61131-2</td>
<td>Type 1</td>
</tr>
<tr>
<td>Input signal range</td>
<td></td>
</tr>
<tr>
<td>Signal 0 (max.)</td>
<td>40 VAC</td>
</tr>
<tr>
<td>Undefined signal</td>
<td>40 VAC &lt; U &lt; 79 VAC</td>
</tr>
<tr>
<td>Signal 1 (min.)</td>
<td>79 VAC</td>
</tr>
<tr>
<td>Input delay</td>
<td></td>
</tr>
<tr>
<td>Signal 0 -&gt; 1</td>
<td>Typ. 24 ms</td>
</tr>
<tr>
<td>Signal 1 -&gt; 0</td>
<td>Typ. 24 ms</td>
</tr>
<tr>
<td>Input data length</td>
<td>2 bytes</td>
</tr>
<tr>
<td>Max. permissible leakage current</td>
<td>1 mA</td>
</tr>
</tbody>
</table>
### Parameter Table

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Max. cable length</td>
<td></td>
</tr>
<tr>
<td>Shielded</td>
<td>1000 m</td>
</tr>
<tr>
<td>Unshielded</td>
<td>600 m</td>
</tr>
</tbody>
</table>

### Ordering Data

<table>
<thead>
<tr>
<th>Part no.</th>
<th>Description</th>
<th>Product life cycle phase *)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1SAP 230 500 R0000</td>
<td>DI572, digital input module, 16 DI, 100 VAC...240 VAC</td>
<td>Active</td>
</tr>
<tr>
<td>1TNE 968 901 R3101</td>
<td>Terminal block TA563-9, 9 pins, screw front, cable side, 6 pieces per unit</td>
<td>Active</td>
</tr>
<tr>
<td>1TNE 968 901 R3102</td>
<td>Terminal block TA563-11, 11 pins, screw front, cable side, 6 pieces per unit</td>
<td>Active</td>
</tr>
<tr>
<td>1TNE 968 901 R3103</td>
<td>Terminal block TA564-9, 9 pins, screw front, cable front, 6 pieces per unit</td>
<td>Active</td>
</tr>
<tr>
<td>1TNE 968 901 R3104</td>
<td>Terminal block TA564-11, 11 pins, screw front, cable front, 6 pieces per unit</td>
<td>Active</td>
</tr>
<tr>
<td>1TNE 968 901 R3105</td>
<td>Terminal block TA565-9, 9 pins, spring front, cable front, 6 pieces per unit</td>
<td>Active</td>
</tr>
<tr>
<td>1TNE 968 901 R3106</td>
<td>Terminal block TA565-11, 11 pins, spring front, cable front, 6 pieces per unit</td>
<td>Active</td>
</tr>
</tbody>
</table>

*) For planning and commissioning of new installations use modules in Active status only.

### DO561 - Digital output module

- 8 digital outputs 24 VDC (O0 to O7) in 1 group
- Module-wise electrically isolated
1 I/O bus
2 8 yellow LEDs to display the signal states of the outputs O0 to O7
3 Terminal number
4 Allocation of signal name
5 Terminal block for output signals (11-pin)
6 2 holes for wall-mounting with screws
7 DIN rail

**Intended purpose**

The device can be used as a decentralized I/O extension module for S500 Communication Interface Modules (e.g. CI592-CS31, CI501-PNIO, CI541-DP, CI581-CN) or as a centralized extension module for AC500 CPUs (PM5xx).

The outputs are group-wise electrically isolated from each other.
All other circuitry of the module is electrically isolated from the outputs.

*The I/O module must not be used as a decentralized I/O module with CI590-CS31-HA bus modules.*
Functionality

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>LED displays</td>
<td>For signal states</td>
</tr>
<tr>
<td>Internal power supply</td>
<td>Via I/O bus</td>
</tr>
<tr>
<td>External power supply</td>
<td>Via the terminals ZP and UP (process supply voltage 24 VDC)</td>
</tr>
</tbody>
</table>

Electrical connection

For a detailed description of the mounting, disassembly and electrical connection of the module, please refer to the System Assembly chapter.

The electrical connection is carried out by using removable 9-pin and 11-pin terminal blocks. These terminal blocks differ in their connection system (spring terminals or screw terminals, cable mounting from the front or from the side). For more information, please refer to the chapter Terminal Blocks for S500-eCo I/O Modules. The terminal blocks are not included in the module’s scope of delivery and must be ordered separately.

The following block diagram shows the internal construction of the digital outputs:

![Block diagram of digital outputs]

Table 543: Assignment of the terminals:

<table>
<thead>
<tr>
<th>Terminals</th>
<th>Signal</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>---</td>
<td>Reserved</td>
</tr>
<tr>
<td>11</td>
<td>O0</td>
<td>Output signal O0</td>
</tr>
<tr>
<td>12</td>
<td>O1</td>
<td>Output signal O1</td>
</tr>
<tr>
<td>13</td>
<td>O2</td>
<td>Output signal O2</td>
</tr>
<tr>
<td>14</td>
<td>O3</td>
<td>Output signal O3</td>
</tr>
<tr>
<td>15</td>
<td>O4</td>
<td>Output signal O4</td>
</tr>
<tr>
<td>Terminals</td>
<td>Signal</td>
<td>Description</td>
</tr>
<tr>
<td>-----------</td>
<td>--------</td>
<td>--------------------------------------------</td>
</tr>
<tr>
<td>16</td>
<td>O5</td>
<td>Output signal O5</td>
</tr>
<tr>
<td>17</td>
<td>O6</td>
<td>Output signal O6</td>
</tr>
<tr>
<td>18</td>
<td>O7</td>
<td>Output signal O7</td>
</tr>
<tr>
<td>19</td>
<td>UP</td>
<td>Process supply voltage UP +24 VDC</td>
</tr>
<tr>
<td>20</td>
<td>ZP</td>
<td>Process supply voltage ZP 0 V</td>
</tr>
</tbody>
</table>

The internal power supply voltage for the module's circuitry is carried out via the I/O bus (provided by a bus module or a CPU). Thus, the current consumption from 24 VDC power supply at the terminals L+/UP and M/ZP of the CPU/bus module increases by 10 mA per DO561.

The external power supply connection is carried out via the UP (+24 VDC) and ZP (0 VDC) terminals.

**WARNING!**

**Removal/Insertion under power**

The devices are not designed for removal or insertion under power. Because of unforeseeable consequences, it is not allowed to plug or unplug devices with the power being ON.

Make sure that all voltage sources (supply and process voltage) are switched off before you

- connect or disconnect any signal or terminal block
- remove, mount or replace a module.

Disconnecting any powered devices while energized in a hazardous location could result in an electric arc, which could create a flammable ignition resulting in fire or explosion.

Make sure that power is removed and that the area has been thoroughly checked to ensure that flammable materials are not present prior to proceeding.

The devices must not be opened when in operation. The same applies to the network interfaces.

**NOTICE!**

**Risk of damaging the PLC modules!**

Overvoltages and short circuits might damage the PLC modules.

- Make sure that all voltage sources (supply voltage and process supply voltage) are switched off before you begin with operations at the system.
- Never connect any voltages or signals to reserved terminals (marked with ---). Reserved terminals may carry internal voltages.

The following figure shows the electrical connection of the digital output module DO561:
NOTICE!
Risk of malfunctions in the plant!
The outputs may switch on for a period of 10 to 50 μs if the process supply voltage UP/ZP is switched on. This must be considered in the planning of the application.

NOTICE!
Risk of damaging the I/O module!
The outputs are not protected against short circuits and overload.
- Never short-circuit or overload the outputs.
- Never connect the outputs to other voltages.
- Use an external 3 A fast-protection fuse for the outputs.

The module provides several diagnosis functions (see Diagnosis "Chapter 1.5.3.6.1.7.6 "Diagnosis" on page 2383). The meaning of the LEDs is described in the section State LEDs "Chapter 1.5.3.6.1.7.7 "State LEDs" on page 2384.

I/O configuration
The module itself does not store configuration data. It receives its parameterization data from the master device of the I/O bus (CPU or bus module) during power-up of the system. Hence, replacing I/O modules is possible without any re-parameterization via software.

If the external power supply voltage via UP/ZP terminals fails, the I/O module loses its configuration data. The whole station has to be switched off and on again to re-configure the module.
Parameterization

The arrangement of the parameter data is performed with Automation Builder software. The parameter data directly influences the functionality of modules. For non-standard applications, it is necessary to adapt the parameters to your system configuration.

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>Internal Value</th>
<th>Internal Value, Type</th>
<th>Default</th>
<th>Min.</th>
<th>Max.</th>
<th>EDS Slot Index</th>
</tr>
</thead>
<tbody>
<tr>
<td>Module ID</td>
<td>Internal</td>
<td>6120 1)</td>
<td>WORD</td>
<td>6120</td>
<td>0</td>
<td>65535</td>
<td>xx01</td>
</tr>
<tr>
<td>Ignore module</td>
<td>No</td>
<td>0</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td>xx02 2)</td>
</tr>
<tr>
<td>Parameter length</td>
<td>Internal</td>
<td>1</td>
<td>BYTE</td>
<td>0</td>
<td>0</td>
<td>255</td>
<td>xx02</td>
</tr>
</tbody>
</table>

1) with CS31 and addresses smaller than 70, the value is increased by 1
2) Value is hexadecimal: HighByte is slot (xx: 0...7), LowByte is index (1...n)

GSD file:

```
Ext_User_Prm_Data_Len = 0x03
Ext_User_Prm_Data_Const(0) = 0x03 0xE9, 0x17, 0x00;
```

Diagnosis

```
<table>
<thead>
<tr>
<th>E1...E4</th>
<th>d1</th>
<th>d2</th>
<th>d3</th>
<th>d4</th>
<th>Identifier</th>
<th>AC500-Display</th>
<th>&lt;-- Display in</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>PS501 PLC</td>
<td>Browser</td>
</tr>
<tr>
<td>Byte 6</td>
<td>Byte 3</td>
<td>Byte 4</td>
<td>Byte 5</td>
<td>Byte 6</td>
<td>Bit 0...5</td>
<td>PNIO diagnosis block</td>
<td></td>
</tr>
<tr>
<td>Bit 6...7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Class</td>
<td>Interface</td>
<td>Device</td>
<td>Module</td>
<td>Channel</td>
<td>Error Identifier</td>
<td>Error message</td>
<td>Remedy</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1)</td>
<td>2)</td>
<td>3)</td>
<td>4)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Module error DO561

| 3 | 14 | 1...10 | 31 | 31 | 19 | Checksum error in the I/O module | Replace I/O module |
| 11 / 12 | ADR | 1...10 |
| 3 | 14 | 1...10 | 31 | 31 | 43 | Internal error in the module | Replace I/O module |
| 11 / 12 | ADR | 1...10 |
| 3 | 14 | 1...10 | 31 | 31 | 9  | Overflow diagnosis buffer | Restart |
| 11 / 12 | ADR | 1...10 |
| 3 | 14 | 1...10 | 31 | 31 | 26 | Parameter error | Check master |
| 11 / 12 | ADR | 1...10 |
Remarks:

1) In AC500 the following interface identifier applies:
   14 = I/O bus, 11 = COM1 (e.g. CS31 bus), 12 = COM2.
   The PNIO diagnosis block does not contain this identifier.

2) With "Device" the following allocation applies:
   31 = module itself, 1...10 = decentralized communication interface module 1...10, ADR = hardware address (e.g. of the DC551-CS31)

3) With "Module" the following allocation applies depending on the master:
   Module error: I/O bus or PNIO: 31 = module itself; COM1/COM2: 1...10 = expansion 1...10
   Channel error: I/O bus or PNIO = module type (2 = DO); COM1/COM2: 1...10 = expansion 1...10

4) In case of module errors, with channel "31 = Module itself" is output.

State LEDs

<table>
<thead>
<tr>
<th>LED</th>
<th>State</th>
<th>Color</th>
<th>LED = OFF</th>
<th>LED = ON</th>
</tr>
</thead>
<tbody>
<tr>
<td>DO561</td>
<td>Outputs O0...O7</td>
<td>Digital output</td>
<td>Yellow</td>
<td>Output is OFF</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(the output voltage is only displayed if the supply voltage of the module is ON)</td>
<td></td>
</tr>
</tbody>
</table>

Technical data

The System Data of AC500-eCo apply
Only additional details are therefore documented below.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Process supply voltage UP</td>
<td>Terminal 19 for UP (+24 VDC) and terminal 20 for ZP (0 VDC)</td>
</tr>
<tr>
<td>Connections</td>
<td></td>
</tr>
<tr>
<td>Rated value</td>
<td>24 VDC</td>
</tr>
<tr>
<td>Current consumption via UP terminal</td>
<td>5 mA + max. 0.5 A per output</td>
</tr>
<tr>
<td>Max. ripple</td>
<td>5 %</td>
</tr>
<tr>
<td>Inrush current</td>
<td>0.000002 A²s</td>
</tr>
<tr>
<td>Protection against reversed voltage</td>
<td>Yes</td>
</tr>
<tr>
<td>Rated protection fuse for UP</td>
<td>Recommended; the outputs must be protected by an 3 A fast fuse</td>
</tr>
<tr>
<td>Current consumption from 24 VDC power supply at the</td>
<td>Ca. 10 mA</td>
</tr>
<tr>
<td>L+/UP and M/ZP terminals of the CPU/bus module</td>
<td></td>
</tr>
<tr>
<td>Galvanic isolation</td>
<td>Yes, between the output group and the rest of the module</td>
</tr>
</tbody>
</table>
### Technical data of the digital outputs

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of channels per module</td>
<td>8 transistor outputs (24 VDC, 0.5 A max.)</td>
</tr>
<tr>
<td>Distribution of the channels into groups</td>
<td>1 (8 channels per group)</td>
</tr>
<tr>
<td>Connection of the channels O0 to O7</td>
<td>Terminals 11 to 18</td>
</tr>
<tr>
<td>Common power supply voltage</td>
<td>Terminal 19 (plus pole of the process voltage, signal name UP)</td>
</tr>
<tr>
<td>Reference potential for the channels O0 to O7</td>
<td>Terminal 20 (minus pole of the process voltage, signal name ZP)</td>
</tr>
<tr>
<td>Indication of the output signals</td>
<td>1 yellow LED per channel; the LED is on when the output signal is high (signal 1) and the module is powered via the I/O bus</td>
</tr>
<tr>
<td>Way of operation</td>
<td>Non-latching type</td>
</tr>
<tr>
<td>Min. output voltage at signal 1</td>
<td>20 VDC at max. current consumption</td>
</tr>
<tr>
<td>Output delay (max. at rated load)</td>
<td></td>
</tr>
<tr>
<td>0 to 1</td>
<td>50 (\mu)s</td>
</tr>
<tr>
<td>1 to 0</td>
<td>200 (\mu)s</td>
</tr>
<tr>
<td>Output data length</td>
<td>1 byte</td>
</tr>
<tr>
<td>Output current</td>
<td></td>
</tr>
<tr>
<td>Rated current per channel (max.)</td>
<td>0.5 A at UP 24 VDC</td>
</tr>
<tr>
<td>Rated current per group (max.)</td>
<td>4 A</td>
</tr>
<tr>
<td>Lamp load (max.)</td>
<td>5 W</td>
</tr>
<tr>
<td>Max. leakage current with signal 0</td>
<td>0.5 mA</td>
</tr>
<tr>
<td>Output type</td>
<td>Non-protected</td>
</tr>
<tr>
<td>Protection type</td>
<td>External fuse on each channel</td>
</tr>
<tr>
<td>Rated protection fuse (for each channel)</td>
<td>3 A fast</td>
</tr>
<tr>
<td>Demagnetization when inductive loads are switched off</td>
<td>Must be performed externally according to driven load specification</td>
</tr>
<tr>
<td>Switching Frequencies</td>
<td></td>
</tr>
<tr>
<td>With inductive loads</td>
<td>Max. 0.5 Hz</td>
</tr>
</tbody>
</table>
### Parameter Table

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>With lamp loads</td>
<td>Max. 11 Hz at max. 5 W</td>
</tr>
<tr>
<td>Short-circuit-proof / Overload-proof</td>
<td>No</td>
</tr>
<tr>
<td>Overload message</td>
<td>No</td>
</tr>
<tr>
<td>Output current limitation</td>
<td>No</td>
</tr>
<tr>
<td>Resistance to feedback against 24 VDC</td>
<td>No</td>
</tr>
<tr>
<td>Connection of 2 outputs in parallel</td>
<td>Not possible</td>
</tr>
<tr>
<td>Max. cable length</td>
<td></td>
</tr>
<tr>
<td>Shielded</td>
<td>500 m</td>
</tr>
<tr>
<td>Unshielded</td>
<td>150 m</td>
</tr>
</tbody>
</table>

#### Ordering data

<table>
<thead>
<tr>
<th>Part no.</th>
<th>Description</th>
<th>Product life cycle phase *)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1TNE 968 902 R2201</td>
<td>DO561, digital output module, 8 DO, transistor output</td>
<td>Active</td>
</tr>
<tr>
<td>1TNE 968 901 R3102</td>
<td>Terminal block TA563-11, 11 pins, screw front, cable side, 6 pieces per unit</td>
<td>Active</td>
</tr>
<tr>
<td>1TNE 968 901 R3104</td>
<td>Terminal block TA564-11, 11 pins, screw front, cable front, 6 pieces per unit</td>
<td>Active</td>
</tr>
<tr>
<td>1TNE 968 901 R3106</td>
<td>Terminal block TA565-11, 11 pins, spring front, cable front, 6 pieces per unit</td>
<td>Active</td>
</tr>
</tbody>
</table>

*) For planning and commissioning of new installations use modules in Active status only.

**DO562 - Digital output module**

- 16 digital outputs 24 VDC (O0 to O15) in 1 group
- Module-wise electrically isolated
1 I/O bus
2 16 yellow LEDs to display the signal states of the outputs O0 to O15
3 Terminal number
4 Allocation of signal name
5 Terminal block for output signals (9-pin)
6 Terminal block for output signals (11-pin)
7 2 holes for wall-mounting with screws
8 DIN rail

Intended purpose

The device can be used as a decentralized I/O extension module for S500 Communication Interface Modules (e.g., CI592-CS31, CI501-PNIO, CI541-DP, CI581-CN) or as a centralized extension module for AC500 CPUs (PM5xx).

The outputs are group-wise electrically isolated from each other. All other circuitry of the module is electrically isolated from the outputs.

The I/O module must not be used as a decentralized I/O module with CI590-CS31-HA bus modules.
## Functionality

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>LED displays</td>
<td>For signal states</td>
</tr>
<tr>
<td>Internal power supply</td>
<td>Via I/O bus</td>
</tr>
<tr>
<td>External power supply</td>
<td>Via the terminals ZP and UP (process supply voltage 24 VDC)</td>
</tr>
</tbody>
</table>

## Electrical connection

For a detailed description of the mounting, disassembly and electrical connection of the module, please refer to the System Assembly chapter.

The electrical connection is carried out by using removable 9-pin and 11-pin terminal blocks. These terminal blocks differ in their connection system (spring terminals or screw terminals, cable mounting from the front or from the side). For more information, please refer to the chapter Terminal Blocks for S500-eCo I/O Modules. The terminal blocks are not included in the module's scope of delivery and must be ordered separately.

The following block diagram shows the internal construction of the digital outputs:
Table 544: Assignment of the terminals:

<table>
<thead>
<tr>
<th>Terminal</th>
<th>Signal</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>---</td>
<td>Reserved</td>
</tr>
<tr>
<td>2</td>
<td>O0</td>
<td>Output signal O0</td>
</tr>
<tr>
<td>3</td>
<td>O1</td>
<td>Output signal O1</td>
</tr>
<tr>
<td>4</td>
<td>O2</td>
<td>Output signal O2</td>
</tr>
<tr>
<td>5</td>
<td>O3</td>
<td>Output signal O3</td>
</tr>
<tr>
<td>6</td>
<td>O4</td>
<td>Output signal O4</td>
</tr>
<tr>
<td>7</td>
<td>O5</td>
<td>Output signal O5</td>
</tr>
<tr>
<td>8</td>
<td>O6</td>
<td>Output signal O6</td>
</tr>
<tr>
<td>9</td>
<td>O7</td>
<td>Output signal O7</td>
</tr>
<tr>
<td>10</td>
<td>---</td>
<td>Reserved</td>
</tr>
<tr>
<td>11</td>
<td>O8</td>
<td>Output signal O8</td>
</tr>
<tr>
<td>12</td>
<td>O9</td>
<td>Output signal O9</td>
</tr>
<tr>
<td>13</td>
<td>O10</td>
<td>Output signal O10</td>
</tr>
<tr>
<td>14</td>
<td>O11</td>
<td>Output signal O11</td>
</tr>
<tr>
<td>Terminal</td>
<td>Signal</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>--------</td>
<td>------------------------------------</td>
</tr>
<tr>
<td>15</td>
<td>O12</td>
<td>Output signal O12</td>
</tr>
<tr>
<td>16</td>
<td>O13</td>
<td>Output signal O13</td>
</tr>
<tr>
<td>17</td>
<td>O14</td>
<td>Output signal O14</td>
</tr>
<tr>
<td>18</td>
<td>O15</td>
<td>Output signal O15</td>
</tr>
<tr>
<td>19</td>
<td>UP</td>
<td>Process voltage UP (24 VDC)</td>
</tr>
<tr>
<td>20</td>
<td>ZP</td>
<td>Process voltage ZP (0 VDC)</td>
</tr>
</tbody>
</table>

The internal power supply voltage for the module's circuitry is carried out via the I/O bus (provided by a bus module or a CPU). Thus, the current consumption from 24 VDC power supply at the terminals L+/UP and M/ZP of the CPU/bus module increases by 10 mA per DO562.

The external power supply connection is carried out via the UP (+24 VDC) and ZP (0 VDC) terminals.

**WARNING!**
Removal/Insertion under power
The devices are not designed for removal or insertion under power. Because of unforeseeable consequences, it is not allowed to plug or unplug devices with the power being ON.

Make sure that all voltage sources (supply and process voltage) are switched off before you
- connect or disconnect any signal or terminal block
- remove, mount or replace a module.

Disconnecting any powered devices while energized in a hazardous location could result in an electric arc, which could create a flammable ignition resulting in fire or explosion.

Make sure that power is removed and that the area has been thoroughly checked to ensure that flammable materials are not present prior to proceeding.

The devices must not be opened when in operation. The same applies to the network interfaces.

**NOTICE!**
Risk of damaging the PLC modules!
- Overvoltages and short circuits might damage the PLC modules.
  - Make sure that all voltage sources (supply voltage and process supply voltage) are switched off before you begin with operations at the system.
  - Never connect any voltages or signals to reserved terminals (marked with ---). Reserved terminals may carry internal voltages.

The following figure shows the electrical connection of the digital output module DO562:
**NOTICE!**

**Risk of malfunctions in the plant!**

The outputs may switch on for a period of 10 to 50 µs if the process supply voltage UP/ZP is switched on.

This must be considered in the planning of the application.
NOTICE!
Risk of damaging the I/O module!
The outputs are not protected against short circuits and overload.
– Never short-circuit or overload the outputs.
– Never connect the outputs to other voltages.
– Use an external 3 A fast-protection fuse for the outputs.

The module provides several diagnosis functions (see Diagnosis § Chapter 1.5.3.6.1.1.8.6 “Diagnosis” on page 2393).
The meaning of the LEDs is described in the section Status LEDs § Chapter 1.5.3.6.1.1.8.7 “State LEDs” on page 2393.

I/O configuration

The module itself does not store configuration data. It receives its parameterization data from
the master device of the I/O bus (CPU or bus module) during power-up of the system.
Hence, replacing I/O modules is possible without any re-parameterization via software.

If the external power supply voltage via UP/ZP terminals fails, the I/O module
loses its configuration data. The whole station has to be switched off and on
again to re-configure the module.

Parameterization

The arrangement of the parameter data is performed with Automation Builder software.
The parameter data directly influences the functionality of modules.
For non-standard applications, it is necessary to adapt the parameters to your system configu-
ration.

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>Internal Value</th>
<th>Internal Value, Type</th>
<th>Default</th>
<th>Min.</th>
<th>Max.</th>
<th>EDS Slot Index</th>
</tr>
</thead>
<tbody>
<tr>
<td>Module ID</td>
<td>Internal</td>
<td>6145 1)</td>
<td>WORD</td>
<td>6145</td>
<td>0</td>
<td>65535</td>
<td>xx01</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0x1801</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ignore module</td>
<td>No</td>
<td>0</td>
<td>BYTE</td>
<td>No</td>
<td>0</td>
<td>255</td>
<td>xx02 2)</td>
</tr>
<tr>
<td></td>
<td>Yes</td>
<td>1</td>
<td></td>
<td>(0x00)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Parameter length</td>
<td>Internal</td>
<td>1</td>
<td>BYTE</td>
<td>0</td>
<td>0</td>
<td>255</td>
<td>xx02 2)</td>
</tr>
</tbody>
</table>

1) with CS31 and addresses less than 70, the value is increased by 1
2) Value is hexadecimal: HighByte is slot (xx: 0...7), LowByte is index (1...n)

GSD file:

```
Ext_User_Prm_Data_Len = 0x06
Ext_User_Prm_Data_Const(0) = 0x18, 0x02, 0x00, 0x02, 0x00, 0x00;
```
Diagnosis

<table>
<thead>
<tr>
<th>Byte 6</th>
<th>d1</th>
<th>d2</th>
<th>d3</th>
<th>d4</th>
<th>Identifier 000...063</th>
<th>AC500-Display</th>
<th>← Display in</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class</td>
<td>Comp</td>
<td>Dev</td>
<td>Mod</td>
<td>Ch</td>
<td>Err</td>
<td>PS501 PLC Browser</td>
<td></td>
</tr>
<tr>
<td>Bit 6...7</td>
<td>-</td>
<td>Byte 3</td>
<td>Byte 4</td>
<td>Byte 5</td>
<td>Byte 6 Bit 0...5</td>
<td>PNIO diagnosis block</td>
<td></td>
</tr>
</tbody>
</table>

Class | Interface | Device | Module | Channel | Error-Identifier | Error message | Remedy |
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1)</td>
<td>2)</td>
<td>3)</td>
<td>4)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Module error**

<table>
<thead>
<tr>
<th>Class</th>
<th>d1</th>
<th>d2</th>
<th>d3</th>
<th>d4</th>
<th>Identifier 000...063</th>
<th>AC500-Display</th>
<th>← Display in</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>14</td>
<td>1...10</td>
<td>31</td>
<td>31</td>
<td>19</td>
<td>Checksum error in the I/O module</td>
<td>Replace I/O module</td>
</tr>
<tr>
<td></td>
<td>11 / 12</td>
<td>ADR</td>
<td>1...10</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>14</td>
<td>1...10</td>
<td>31</td>
<td>31</td>
<td>43</td>
<td>Internal error in the module</td>
<td>Replace I/O module</td>
</tr>
<tr>
<td></td>
<td>11 / 12</td>
<td>ADR</td>
<td>1...10</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>14</td>
<td>1...10</td>
<td>31</td>
<td>31</td>
<td>9</td>
<td>Overflow diagnosis buffer</td>
<td>Restart</td>
</tr>
<tr>
<td></td>
<td>11 / 12</td>
<td>ADR</td>
<td>1...10</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>14</td>
<td>1...10</td>
<td>31</td>
<td>31</td>
<td>26</td>
<td>Parameter error</td>
<td>Check master</td>
</tr>
<tr>
<td></td>
<td>11 / 12</td>
<td>ADR</td>
<td>1...10</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Remarks:**

1) In AC500 the following interface identifier applies:
   14 = I/O bus, 11 = COM1 (e.g. CS31-Bus), 12 = COM2.
   The PNIO diagnosis block does not contain this identifier.

2) With "Device" the following allocation applies:
   31 = module itself, 1...10 = decentralized communication interface module 1...10,
   ADR = hardware address (e.g. of the DC551-CS31)

3) With "Module" the following allocation depends on the master:
   Module error: I/O bus or PNIO: 31 = Module itself; COM1/COM2: 1...10 = expansion 1...10
   Channel error: I/O bus or PNIO = module type (2 = DO); COM1/COM2: 1...10 = expansion 1...10

4) In case of module errors, with channel "31 = Module itself" is output.

**State LEDs**

<table>
<thead>
<tr>
<th>LED</th>
<th>State</th>
<th>Color</th>
<th>LED = OFF</th>
<th>LED = ON</th>
</tr>
</thead>
<tbody>
<tr>
<td>DO562</td>
<td>Outputs O0...O15</td>
<td>Digital output</td>
<td>Yellow</td>
<td>Output is OFF (the output voltage is only displayed if the supply voltage of the module is ON)</td>
</tr>
</tbody>
</table>

2020/12/10
3ADR010583, 1, en_US 2393
Technical data

The System Data of AC500-eCo apply

Only additional details are therefore documented below.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Process supply voltage UP</td>
<td></td>
</tr>
<tr>
<td>Connections</td>
<td>Terminal 19 for UP (+24 VDC) and terminal 20 for ZP (0 VDC)</td>
</tr>
<tr>
<td>Rated value</td>
<td>24 VDC</td>
</tr>
<tr>
<td>Current consumption via UP terminal</td>
<td>20 mA + max. 0.5 A per output</td>
</tr>
<tr>
<td>Max. ripple</td>
<td>5 %</td>
</tr>
<tr>
<td>Inrush current</td>
<td>0.000002 A²s</td>
</tr>
<tr>
<td>Protection against reversed voltage</td>
<td>Yes</td>
</tr>
<tr>
<td>Rated protection fuse for UP</td>
<td>Recommended; the outputs must be protected by an 3 A fast fuse</td>
</tr>
<tr>
<td>Current consumption from 24 VDC power supply at the L+/UP and M/ZP terminals of the CPU/bus module</td>
<td>Ca. 10 mA</td>
</tr>
<tr>
<td>Galvanic isolation</td>
<td>Yes, between the output group and the rest of the module</td>
</tr>
<tr>
<td>Isolated groups</td>
<td>1 (16 channels per group)</td>
</tr>
<tr>
<td>Surge-voltage (max.)</td>
<td>35 VDC for 0.5 s</td>
</tr>
<tr>
<td>Max. power dissipation within the module</td>
<td>1.4 W</td>
</tr>
<tr>
<td>Weight</td>
<td>Ca. 125 g</td>
</tr>
<tr>
<td>Mounting position</td>
<td>Horizontal or vertical</td>
</tr>
<tr>
<td>Cooling</td>
<td>The natural convection cooling must not be hindered by cable ducts or other parts in the switch-gear cabinet.</td>
</tr>
</tbody>
</table>

No effects of multiple overloads

No effects of multiple overloads on isolated multi-channel modules occur, as every channel is protected individually by an external fuse.

Technical data of the digital outputs

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of channels per module</td>
<td>16 transistor outputs (24 VDC, 0.5 A max.)</td>
</tr>
<tr>
<td>Distribution of the channels into groups</td>
<td>1 (16 channels per group)</td>
</tr>
<tr>
<td>Connection of the channels O0 to O7</td>
<td>Terminals 1 to 9</td>
</tr>
<tr>
<td>Connection of the channels O8 to O15</td>
<td>Terminals 11 to 18</td>
</tr>
<tr>
<td>Common power supply voltage</td>
<td>Terminal 19 (positive pole of the process voltage, signal name UP)</td>
</tr>
<tr>
<td>Reference potential for the channels O0 to O15</td>
<td>Terminal 20 (negative pole of the process voltage, signal name ZP)</td>
</tr>
<tr>
<td>Parameter</td>
<td>Value</td>
</tr>
<tr>
<td>------------------------------------------------</td>
<td>----------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Indication of the output signals</td>
<td>1 yellow LED per channel; the LED is on when the output signal is high (signal 1) and the module is powered via the I/O bus</td>
</tr>
<tr>
<td>Way of operation</td>
<td>Non-latching type</td>
</tr>
<tr>
<td>Min. output voltage at signal 1</td>
<td>UP -0.3 V at max. current consumption</td>
</tr>
<tr>
<td>Output delay (max. at rated load)</td>
<td></td>
</tr>
<tr>
<td>0 to 1</td>
<td>50 µs</td>
</tr>
<tr>
<td>1 to 0</td>
<td>200 µs</td>
</tr>
<tr>
<td>Output data length</td>
<td>2 bytes</td>
</tr>
<tr>
<td>Output current</td>
<td></td>
</tr>
<tr>
<td>Rated current per channel (max.)</td>
<td>0.5 A at UP 24 VDC</td>
</tr>
<tr>
<td>Rated current per group (max.)</td>
<td>8 A</td>
</tr>
<tr>
<td>Lamp load (max.)</td>
<td>5 W</td>
</tr>
<tr>
<td>Max. leakage current with signal 0</td>
<td>0.5 mA</td>
</tr>
<tr>
<td>Output type</td>
<td>Non-protected</td>
</tr>
<tr>
<td>Protection type</td>
<td>External fuse on each channel</td>
</tr>
<tr>
<td>Rated protection fuse (for each channel)</td>
<td>3 A fast</td>
</tr>
<tr>
<td>Demagnetization when inductive loads are</td>
<td>Must be performed externally according to driven load specification</td>
</tr>
<tr>
<td>switched off</td>
<td></td>
</tr>
<tr>
<td>Switching Frequencies</td>
<td></td>
</tr>
<tr>
<td>With inductive loads</td>
<td>Max. 0.5 Hz</td>
</tr>
<tr>
<td>With lamp loads</td>
<td>Max. 11 Hz at max. 5 W</td>
</tr>
<tr>
<td>Short-circuit-proof / Overload-proof</td>
<td>No</td>
</tr>
<tr>
<td>Overload message</td>
<td>No</td>
</tr>
<tr>
<td>Output current limitation</td>
<td>No</td>
</tr>
<tr>
<td>Resistance to feedback against 24 VDC</td>
<td>No</td>
</tr>
<tr>
<td>Connection of 2 outputs in parallel</td>
<td>Not possible</td>
</tr>
<tr>
<td>Max. cable length</td>
<td></td>
</tr>
<tr>
<td>Shielded</td>
<td>500 m</td>
</tr>
<tr>
<td>Unshielded</td>
<td>150 m</td>
</tr>
</tbody>
</table>

**Ordering data**

<table>
<thead>
<tr>
<th>Part no.</th>
<th>Description</th>
<th>Product life cycle phase *)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1SAP 230 900 R0000</td>
<td>DO562, digital output module, 16 DO, transistor output</td>
<td>Active</td>
</tr>
<tr>
<td>1TNE 968 901 R3101</td>
<td>Terminal block TA563-9, 9 pins, screw front, cable side, 6 pieces per unit</td>
<td>Active</td>
</tr>
<tr>
<td>1TNE 968 901 R3102</td>
<td>Terminal block TA563-11, 11 pins, screw front, cable side, 6 pieces per unit</td>
<td>Active</td>
</tr>
<tr>
<td>Part no.</td>
<td>Description</td>
<td>Product life cycle phase *)</td>
</tr>
<tr>
<td>--------------------</td>
<td>--------------------------------------------------</td>
<td>-----------------------------</td>
</tr>
<tr>
<td>1TNE 968 901 R3103</td>
<td>Terminal block TA564-9, 9 pins, screw front, cable front, 6 pieces per unit</td>
<td>Active</td>
</tr>
<tr>
<td>1TNE 968 901 R3104</td>
<td>Terminal block TA564-11, 11 pins, screw front, cable front, 6 pieces per unit</td>
<td>Active</td>
</tr>
<tr>
<td>1TNE 968 901 R3105</td>
<td>Terminal block TA565-9, 9 pins, spring front, cable front, 6 pieces per unit</td>
<td>Active</td>
</tr>
<tr>
<td>1TNE 968 901 R3106</td>
<td>Terminal block TA565-11, 11 pins, spring front, cable front, 6 pieces per unit</td>
<td>Active</td>
</tr>
</tbody>
</table>

*) For planning and commissioning of new installations use modules in Active status only.

**DO571 - Digital output module**
- 8 digital normally open relay outputs 24 VDC / 24 VAC or 100-240 VAC, 2 A max. (NO0 to NO7) in 2 groups
- Group-wise electrically isolated
1. I/O bus
2. 8 yellow LEDs to display the signal states of the outputs O0 to O7
3. Terminal number
4. Allocation of signal name
5. Terminal block for output signals (11-pin)
6. 2 holes for wall-mounting with screws
7. DIN rail

### Intended purpose

The device can be used as a decentralized I/O extension module for S500 Communication Interface Modules (e.g., CI592-CS31, CI501-PNIO, CI541-DP, CI581-CN) or as a centralized extension module for AC500 CPUs (PM5xx).

The outputs are group-wise electrically isolated from each other. All other circuitry of the module is electrically isolated from the outputs.

The I/O module must not be used as a decentralized I/O module with CI590-CS31-HA bus modules.
### Functionality

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>LED displays</td>
<td>For signal states</td>
</tr>
<tr>
<td>Internal power supply</td>
<td>Via I/O bus</td>
</tr>
<tr>
<td>External power supply</td>
<td>Via the terminal L+ (process voltage 24 VDC). The negative pole is provided by the I/O bus.</td>
</tr>
</tbody>
</table>

### Electrical connection

For a detailed description of the mounting, disassembly and electrical connection of the module, please refer to the System Assembly chapter.

The electrical connection is carried out by using removable 9-pin and 11-pin terminal blocks. These terminal blocks differ in their connection system (spring terminals or screw terminals, cable mounting from the front or from the side). For more information, please refer to the chapter Terminal Blocks for S500-eCo I/O Modules. The terminal blocks are not included in the module’s scope of delivery and must be ordered separately.

The following block diagram shows the internal construction of the digital outputs:

![Block Diagram]

Table 545: Assignment of the terminals:

<table>
<thead>
<tr>
<th>Terminal</th>
<th>Signal</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>NO0</td>
<td>Normally-open contact of the output NO0</td>
</tr>
<tr>
<td>11</td>
<td>NO1</td>
<td>Normally-open contact of the output NO1</td>
</tr>
<tr>
<td>12</td>
<td>NO2</td>
<td>Normally-open contact of the output NO2</td>
</tr>
<tr>
<td>13</td>
<td>NO3</td>
<td>Normally-open contact of the output NO3</td>
</tr>
<tr>
<td>14</td>
<td>R0..3</td>
<td>Output common for signals NO0 to NO3</td>
</tr>
<tr>
<td>15</td>
<td>NO4</td>
<td>Normally-open contact of the output NO4</td>
</tr>
<tr>
<td></td>
<td>L+</td>
<td></td>
</tr>
<tr>
<td>Terminal</td>
<td>Signal</td>
<td>Description</td>
</tr>
<tr>
<td>---------</td>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td>16</td>
<td>NO5</td>
<td>Normally-open contact of the output NO5</td>
</tr>
<tr>
<td>17</td>
<td>NO6</td>
<td>Normally-open contact of the output NO6</td>
</tr>
<tr>
<td>18</td>
<td>NO7</td>
<td>Normally-open contact of the output NO7</td>
</tr>
<tr>
<td>19</td>
<td>R4..7</td>
<td>Output common for signals NO4 to NO7</td>
</tr>
<tr>
<td>20</td>
<td>L+</td>
<td>Process voltage L+ +24 VDC</td>
</tr>
</tbody>
</table>

The internal power supply voltage for the module's circuitry is carried out via the I/O bus (provided by a bus module or a CPU). Thus, the current consumption from 24 VDC power supply at the terminals L+/UP and M/ZP of the CPU/bus module increases by 5 mA per DO571.

The external power supply connection is carried out via the L+ (+24 VDC) terminal. The negative pole of the external power supply is realized via the I/O bus. Therefore, the CPU/bus module and the DO571 must have a common power supply.

**WARNING!**
**Risk of death by electric shock!**
Hazardous voltages can be present at the terminals of the module.
Make sure that all voltage sources (supply voltage and process supply voltage) are switched off before you begin with operations at the system.

For screw-type terminals only:

**WARNING!**
**For screw terminals only: Danger of death by electric shock!**
The IP 20 protection degree is only provided if all terminal screws are tightened.
Tighten all screws of unused load terminals of relay outputs if voltages > 24 V are connected to the relay group.

**WARNING!**
**Removal/Insertion under power**
The devices are not designed for removal or insertion under power. Because of unforeseeable consequences, it is not allowed to plug or unplug devices with the power being ON.
Make sure that all voltage sources (supply and process voltage) are switched off before you
- connect or disconnect any signal or terminal block
- remove, mount or replace a module.
Disconnecting any powered devices while energized in a hazardous location could result in an electric arc, which could create a flammable ignition resulting in fire or explosion.
Make sure that power is removed and that the area has been thoroughly checked to ensure that flammable materials are not present prior to proceeding.
The devices must not be opened when in operation. The same applies to the network interfaces.
NOTICE!
Risk of damaging the PLC modules!
Overvoltages and short circuits might damage the PLC modules.
– Make sure that all voltage sources (supply voltage and process supply voltage) are switched off before you begin with operations at the system.
– Never connect any voltages or signals to reserved terminals (marked with ---). Reserved terminals may carry internal voltages.

The following figure shows the electrical connection of the module:

![Diagram showing electrical connection of PLC module](image)

Fig. 99: Connection of 24 VDC actuators
**Fig. 100: Connection of 24 VAC or 100-240 VAC actuators**

**NOTICE!**

*Risk of damaging the I/O module!*

The outputs are not protected against short circuit and overload.

- Never short-circuit or overload the outputs.
- Never connect inductive loads without an external suppression against voltage peaks due to inductive kickback.
- Never connect voltages > 240 V. All outputs must be supplied from the same phase.
- Use an external 5 A fast protection fuse for the outputs.

**Fig. 101: Power supply - the negative connection is realized via the I/O bus**
The L+ connection of the DO571 and the 24 V supply of the CPU/bus module must be connected to the same 24 V power supply.

The module provides several diagnosis functions (see Diagnosis Chapter 1.5.3.6.1.1.9.6 “Diagnosis” on page 2403).

The meaning of the LEDs is described in the section Status LEDs Chapter 1.5.3.6.1.1.9.7 “State LEDs” on page 2404.

I/O configuration

The module itself does not store configuration data. It receives its parameterization data from the master device of the I/O bus (CPU or bus module) during power-up of the system. Hence, replacing I/O modules is possible without any re-parameterization via software.

If the external power supply voltage via UP/ZP terminals fails, the I/O module loses its configuration data. The whole station has to be switched off and on again to re-configure the module.

Parameterization

The arrangement of the parameter data is performed with Automation Builder software. The parameter data directly influences the functionality of modules.

For non-standard applications, it is necessary to adapt the parameters to your system configuration.

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>Internal Value</th>
<th>Internal Value, Type</th>
<th>Default</th>
<th>Min.</th>
<th>Max.</th>
<th>EDS Slot Index</th>
</tr>
</thead>
<tbody>
<tr>
<td>Module ID</td>
<td>Internal</td>
<td>6125 1)</td>
<td>WORD</td>
<td>6125</td>
<td>0x17ED</td>
<td>0</td>
<td>65535</td>
</tr>
<tr>
<td>Ignore module</td>
<td>No Yes</td>
<td>0</td>
<td>1</td>
<td>No</td>
<td>(0x00)</td>
<td>0</td>
<td>255</td>
</tr>
<tr>
<td>Parameter length</td>
<td>Internal</td>
<td>1</td>
<td>BYTE</td>
<td>0</td>
<td>0</td>
<td>255</td>
<td>xx02 2)</td>
</tr>
<tr>
<td>Check supply</td>
<td>Off On</td>
<td>0</td>
<td>1</td>
<td>On</td>
<td>0x01</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

1) with CS31 and addresses smaller than 70, the value is increased by 1
2) Value is hexadecimal: HighByte is slot (xx: 0...7), LowByte is index (1...n)

GSD file:

```
Ext_User_Prm_Data_Len = 0x04
Ext_User_Prm_Data_Const(0) = 0xEF, 0x17, 0x00,
0x01;
```
Diagnosis

<table>
<thead>
<tr>
<th>E1...E4</th>
<th>d1</th>
<th>d2</th>
<th>d3</th>
<th>d4</th>
<th>Identifer 000...063</th>
<th>AC500-Display</th>
<th>&lt;− Display in</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class</td>
<td>Comp</td>
<td>Dev</td>
<td>Mod</td>
<td>Ch</td>
<td>Err</td>
<td>PS501 PLC Browser</td>
<td></td>
</tr>
<tr>
<td>Byte 6</td>
<td>Bit 6...7</td>
<td>-</td>
<td>Byte 3</td>
<td>Byte 4</td>
<td>Byte 5</td>
<td>Byte 6 Bit 0...5</td>
<td>PNIO diagnosis block</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Class</th>
<th>Interface</th>
<th>Device</th>
<th>Module</th>
<th>Channel</th>
<th>Error Identifier</th>
<th>Error message</th>
<th>Remedy</th>
</tr>
</thead>
<tbody>
<tr>
<td>1)</td>
<td>2)</td>
<td>3)</td>
<td>4)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Module error

<table>
<thead>
<tr>
<th>3</th>
<th>14</th>
<th>1...10</th>
<th>31</th>
<th>31</th>
<th>19</th>
<th>Checksum error in the I/O module</th>
<th>Replace I/O module</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>11 / 12</td>
<td>ADR</td>
<td>1...10</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>14</td>
<td>1...10</td>
<td>31</td>
<td>31</td>
<td>43</td>
<td>Internal error in the module</td>
<td>Replace I/O module</td>
</tr>
<tr>
<td></td>
<td>11 / 12</td>
<td>ADR</td>
<td>1...10</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>14</td>
<td>1...10</td>
<td>31</td>
<td>31</td>
<td>9</td>
<td>Overflow diagnosis buffer</td>
<td>Restart</td>
</tr>
<tr>
<td></td>
<td>11 / 12</td>
<td>ADR</td>
<td>1...10</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>14</td>
<td>1...10</td>
<td>31</td>
<td>31</td>
<td>26</td>
<td>Parameter error</td>
<td>Check master</td>
</tr>
<tr>
<td></td>
<td>11 / 12</td>
<td>ADR</td>
<td>1...10</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>14</td>
<td>1...10</td>
<td>31</td>
<td>31</td>
<td>11</td>
<td>Process voltage too low</td>
<td>Check process voltage</td>
</tr>
<tr>
<td></td>
<td>11 / 12</td>
<td>ADR</td>
<td>1...10</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Remarks:

1) In AC500 the following interface identifier applies: 14 = I/O bus, 11 = COM1 (e.g. CS31-Bus), 12 = COM2. The PNIO diagnosis block does not contain this identifier.

2) With "Device" the following allocation applies: 31 = module itself, 1...10 = decentralized communication interface module 1...10, ADR = Hardware address (e.g. of the DC551-CS31)

3) With "Module" the following allocation applies depending on the master: Module error: I/O bus or PNIO: 31 = module itself; COM1/COM2: 1...10 = expansion 1...10 Channel error: I/O bus or PNIO = module type (2 = DO); COM1/COM2: 1...10 = expansion 1..10

4) In case of module errors, with channel "31 = Module itself" is output.
State LEDs

<table>
<thead>
<tr>
<th>LED</th>
<th>Outputs O0...O7</th>
<th>State</th>
<th>Color</th>
<th>LED = OFF</th>
<th>LED = ON</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Digital output</td>
<td></td>
<td>Yellow</td>
<td>Output is OFF</td>
<td>Output is ON</td>
</tr>
</tbody>
</table>

(Outputs O0...O7 are digital outputs. LED color yellow indicates the output is OFF (the output voltage is only displayed if the supply voltage of the module is ON).)

Technical data

The System Data of AC500-eCo apply

Only additional details are therefore documented below.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Process supply voltage L+</td>
<td></td>
</tr>
<tr>
<td>Connections</td>
<td>Terminal 20 for L+ (+24 VDC). The negative pole is provided by the I/O bus.</td>
</tr>
<tr>
<td>Rated value</td>
<td>24 VDC</td>
</tr>
<tr>
<td>Current consumption via L+</td>
<td>50 mA</td>
</tr>
<tr>
<td>Inrush current (at power-up)</td>
<td>0.0035 A's</td>
</tr>
<tr>
<td>Max. ripple</td>
<td>5 %</td>
</tr>
<tr>
<td>Protection against reversed voltage</td>
<td>Yes</td>
</tr>
<tr>
<td>Rated protection fuse for UP</td>
<td>Recommended; the outputs must be protected by a 3 A fast fuse</td>
</tr>
<tr>
<td>Current consumption from 24 VDC power supply at the L+/UP and M/ZP terminals of the CPU/bus module</td>
<td>Ca. 5 mA</td>
</tr>
<tr>
<td>Galvanic isolation</td>
<td>Yes, between the output group and the rest of the module</td>
</tr>
<tr>
<td>Isolated groups</td>
<td>2 (4 channels per group)</td>
</tr>
<tr>
<td>Surge-voltage (max.)</td>
<td>35 VDC for 0.5 s</td>
</tr>
<tr>
<td>Max. power dissipation within the module</td>
<td>2.0 W</td>
</tr>
<tr>
<td>Weight</td>
<td>Ca. 150 g</td>
</tr>
<tr>
<td>Mounting position</td>
<td>Horizontal or vertical</td>
</tr>
<tr>
<td>Cooling</td>
<td>The natural convection cooling must not be hindered by cable ducts or other parts in the switch-gear cabinet.</td>
</tr>
</tbody>
</table>

No effects of multiple overloads

No effects of multiple overloads on isolated multi-channel modules occur, as every channel is protected individually by an external fuse.
## Technical data of the digital outputs

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of channels per module</td>
<td>8 normally-open relay outputs</td>
</tr>
<tr>
<td>Distribution of the channels into groups</td>
<td>2 (4 channels per group)</td>
</tr>
<tr>
<td>Connection of the channels O0 to O3</td>
<td>Terminals 10 to 13</td>
</tr>
<tr>
<td>Connection of the channels O4 to O7</td>
<td>Terminals 15 to 18</td>
</tr>
<tr>
<td>Reference potential for the channels O0 to O3</td>
<td>Terminal 14 (signal name R0..3)</td>
</tr>
<tr>
<td>Reference potential for the channels O4 to O7</td>
<td>Terminal 19 (signal name R4..7)</td>
</tr>
<tr>
<td>Relay coil power supply</td>
<td>Terminal 20 (plus pole of the process supply voltage, signal name L+). The minus pole is provided by the I/O bus.</td>
</tr>
<tr>
<td>Indication of the output signals</td>
<td>1 yellow LED per channel; the LED is on when the output signal is high (signal 1) and the module is powered via the I/O bus.</td>
</tr>
<tr>
<td>Way of operation</td>
<td>Non-latching type</td>
</tr>
<tr>
<td>Relay output voltage</td>
<td></td>
</tr>
<tr>
<td>Rated value</td>
<td>24 VDC / 24 VAC or 120/240 VAC</td>
</tr>
<tr>
<td>Output delay</td>
<td></td>
</tr>
<tr>
<td>Switching 0 to 1 (max.)</td>
<td>Typ. 10 ms</td>
</tr>
<tr>
<td>Switching 1 to 0 (max.)</td>
<td>Typ. 10 ms</td>
</tr>
<tr>
<td>Output data length</td>
<td>1 byte</td>
</tr>
<tr>
<td>Output current</td>
<td></td>
</tr>
<tr>
<td>Rated current per channel (max.)</td>
<td>2.0 A (24 VDC / 24 VAC / 48 VAC / 120 VAC / 240 VAC, only resistive loads)</td>
</tr>
<tr>
<td></td>
<td>2.0 A (24 VAC / 48 VAC / 120 VAC, only pilot duty)</td>
</tr>
<tr>
<td></td>
<td>1.5 A (240 VAC, only pilot duty)</td>
</tr>
<tr>
<td>Rated current per group (max.)</td>
<td>8 A</td>
</tr>
<tr>
<td>Lamp load (max.)</td>
<td>200 W (230 VAC), 30 W (24 VDC)</td>
</tr>
<tr>
<td>Spark suppression with inductive AC loads</td>
<td>Must be performed externally according to driven load specification</td>
</tr>
<tr>
<td>Switching Frequencies</td>
<td></td>
</tr>
<tr>
<td>With resistive loads</td>
<td>Max. 1 Hz</td>
</tr>
<tr>
<td>With inductive loads</td>
<td>On Request</td>
</tr>
<tr>
<td>With lamp loads</td>
<td>Max. 1 Hz</td>
</tr>
<tr>
<td>Output type</td>
<td>Non-protected</td>
</tr>
<tr>
<td>Protection type</td>
<td>External fuse on each channel</td>
</tr>
<tr>
<td>Rated protection fuse (for each channel)</td>
<td>5 A fast</td>
</tr>
<tr>
<td>Short-circuit-proof / Overload-proof</td>
<td>No, should be provided by an external fuse or circuit breaker</td>
</tr>
<tr>
<td>Overload message</td>
<td>No</td>
</tr>
<tr>
<td>Output current limitation</td>
<td>No</td>
</tr>
<tr>
<td>Connection of 2 outputs in parallel</td>
<td>Not possible</td>
</tr>
<tr>
<td>Life time of relay contacts (cycles)</td>
<td>100.000 at rated load</td>
</tr>
<tr>
<td>Parameter</td>
<td>Value</td>
</tr>
<tr>
<td>--------------------</td>
<td>------------</td>
</tr>
<tr>
<td>Max. cable length</td>
<td></td>
</tr>
<tr>
<td>Shielded</td>
<td>500 m</td>
</tr>
<tr>
<td>Unshielded</td>
<td>150 m</td>
</tr>
</tbody>
</table>

**Ordering data**

<table>
<thead>
<tr>
<th>Part no.</th>
<th>Description</th>
<th>Product life cycle phase *)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1TNE 968 902 R2202</td>
<td>DO571, digital output module, 8 DO, relay output</td>
<td>Active</td>
</tr>
<tr>
<td>1TNE 968 901 R3102</td>
<td>Terminal block TA563-11, 11 pins, screw front, cable side, 6 pieces per unit</td>
<td>Active</td>
</tr>
<tr>
<td>1TNE 968 901 R3104</td>
<td>Terminal block TA564-11, 11 pins, screw front, cable front, 6 pieces per unit</td>
<td>Active</td>
</tr>
<tr>
<td>1TNE 968 901 R3106</td>
<td>Terminal block TA565-11, 11 pins, spring front, cable front, 6 pieces per unit</td>
<td>Active</td>
</tr>
</tbody>
</table>

*) For planning and commissioning of new installations use modules in Active status only.

**DO572 - Digital output module**
- 8 digital triac outputs (O0 to O7) in 8 groups
- 240 VAC
- Module-wise electrically isolated
Intended purpose

The device can be used as a decentralized I/O extension module for S500 Communication Interface Modules (e.g., CI592-CS31, CI501-PNIO, CI541-DP, CI581-CN) or as a centralized extension module for AC500 CPUs (PM5xx).

The outputs are group-wise electrically isolated from each other.

All other circuitry of the module is electrically isolated from the outputs.

The I/O module must not be used as a decentralized I/O module with CI590-CS31-HA bus modules.
### Functionality

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>LED displays</td>
<td>For signal states</td>
</tr>
<tr>
<td>Internal power supply</td>
<td>Via I/O bus</td>
</tr>
<tr>
<td>External power supply</td>
<td>Not necessary</td>
</tr>
</tbody>
</table>

### Electrical connection

For a detailed description of the mounting, disassembly and electrical connection of the module, please refer to the System Assembly chapter.

The electrical connection is carried out by using removable 9-pin and 11-pin terminal blocks. These terminal blocks differ in their connection system (spring terminals or screw terminals, cable mounting from the front or from the side). For more information, please refer to the chapter Terminal Blocks for S500-eCo I/O Modules. The terminal blocks are not included in the module's scope of delivery and must be ordered separately.

The following block diagram shows the internal construction of the digital outputs:
### Table 546: Assignment of the terminals:

<table>
<thead>
<tr>
<th>Terminal</th>
<th>Signal</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>O0</td>
<td>Output signal O0</td>
</tr>
<tr>
<td>2</td>
<td>N0</td>
<td>Neutral conductor for the output signal O0</td>
</tr>
<tr>
<td>3</td>
<td>O1</td>
<td>Output signal O1</td>
</tr>
<tr>
<td>4</td>
<td>N1</td>
<td>Neutral conductor for the output signal O1</td>
</tr>
<tr>
<td>5</td>
<td>---</td>
<td>Reserved</td>
</tr>
<tr>
<td>6</td>
<td>O2</td>
<td>Output signal O2</td>
</tr>
<tr>
<td>7</td>
<td>N2</td>
<td>Neutral conductor for the output signal O2</td>
</tr>
<tr>
<td>8</td>
<td>O3</td>
<td>Output signal O3</td>
</tr>
<tr>
<td>9</td>
<td>N3</td>
<td>Neutral conductor for the output signal O3</td>
</tr>
<tr>
<td>10</td>
<td>---</td>
<td>Reserved</td>
</tr>
<tr>
<td>11</td>
<td>O4</td>
<td>Output signal O4</td>
</tr>
</tbody>
</table>
### Terminal Signal Description

<table>
<thead>
<tr>
<th>Terminal</th>
<th>Signal</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>12</td>
<td>N4</td>
<td>Neutral conductor for the output signal O4</td>
</tr>
<tr>
<td>13</td>
<td>O5</td>
<td>Output signal O5</td>
</tr>
<tr>
<td>14</td>
<td>N5</td>
<td>Neutral conductor for the output signal O5</td>
</tr>
<tr>
<td>15</td>
<td>---</td>
<td>Reserved</td>
</tr>
<tr>
<td>16</td>
<td>O6</td>
<td>Output signal O6</td>
</tr>
<tr>
<td>17</td>
<td>N6</td>
<td>Neutral conductor for the output signal O6</td>
</tr>
<tr>
<td>18</td>
<td>O7</td>
<td>Output signal O7</td>
</tr>
<tr>
<td>19</td>
<td>N7</td>
<td>Neutral conductor for the output signal O7</td>
</tr>
<tr>
<td>20</td>
<td>---</td>
<td>Reserved</td>
</tr>
</tbody>
</table>

The internal power supply voltage for the module's circuitry is carried out via the I/O bus (provided by a bus module or a CPU). Thus, the current consumption from 24 VDC power supply at the terminals L+/UP and M/ZP of the CPU/bus module increases by 10 mA per DO572.

An external power supply connection is not needed.

---

**WARNING!**

**Risk of death by electric shock!**

Hazardous voltages can be present at the terminals of the module.

Make sure that all voltage sources (supply voltage and process supply voltage) are switched off before you begin with operations at the system.

---

**WARNING!**

**Removal/Insertion under power**

The devices are not designed for removal or insertion under power. Because of unforeseeable consequences, it is not allowed to plug or unplug devices with the power being ON.

Make sure that all voltage sources (supply and process voltage) are switched off before you

- connect or disconnect any signal or terminal block
- remove, mount or replace a module.

Disconnecting any powered devices while energized in a hazardous location could result in an electric arc, which could create a flammable ignition resulting in fire or explosion.

Make sure that power is removed and that the area has been thoroughly checked to ensure that flammable materials are not present prior to proceeding.

The devices must not be opened when in operation. The same applies to the network interfaces.
NOTICE!
Risk of damaging the PLC modules!
Overvoltages and short circuits might damage the PLC modules.
- Make sure that all voltage sources (supply voltage and process supply voltage) are switched off before you begin with operations at the system.
- Never connect any voltages or signals to reserved terminals (marked with ---). Reserved terminals may carry internal voltages.

The following figure shows the electrical connection of the module:
**NOTICE!**

Risk of damaging the PLC modules!

The PLC modules will be irreparably damaged if a voltage > 240 V is connected.

Make sure that all inputs are fed from the same phase. The module must not be connected to a 400 V voltage.

The module provides several diagnosis functions (see chapter Diagnosis ⇨ Chapter 1.5.3.6.1.1.10.6 “Diagnosis” on page 2413).

The meaning of the LEDs is described in the section State LEDs ⇨ Chapter 1.5.3.6.1.1.10.7 “State LEDs” on page 2414.

**I/O configuration**

The module itself does not store configuration data. It receives its parameterization data from the master device of the I/O bus (CPU or bus module) during power-up of the system.

Hence, replacing I/O modules is possible without any re-parameterization via software.

*If the external power supply voltage via UP/ZP terminals fails, the I/O module loses its configuration data. The whole station has to be switched off and on again to re-configure the module.*

**Parameterization**

The arrangement of the parameter data is performed with Automation Builder software.

The parameter data directly influences the functionality of modules.

For non-standard applications, it is necessary to adapt the parameters to your system configuration.

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>Internal Value</th>
<th>Internal Value, Type</th>
<th>Default</th>
<th>Min.</th>
<th>Max.</th>
<th>EDS Slot Index</th>
</tr>
</thead>
<tbody>
<tr>
<td>Module ID</td>
<td>Internal</td>
<td>6130 ¹)</td>
<td>WORD</td>
<td>6130 0x17F2</td>
<td>0</td>
<td>65535</td>
<td>xx01</td>
</tr>
<tr>
<td>Ignore module</td>
<td>No Yes</td>
<td>0 1</td>
<td>BYTE</td>
<td>No (0x00)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Parameter length ²)</td>
<td>Internal</td>
<td>1 - CPU</td>
<td>BYTE</td>
<td>0</td>
<td>0</td>
<td>255</td>
<td>xx02 ³)</td>
</tr>
</tbody>
</table>

¹) With CS31 and addresses smaller than 70, the value is increased by 1

²) The module has no additional user-configurable parameters

³) Value is hexadecimal: HighByte is slot (xx: 0...7), LowByte is index (1...n)

GSD file:
Diagnosis

<table>
<thead>
<tr>
<th>E1...E4</th>
<th>d1</th>
<th>d2</th>
<th>d3</th>
<th>d4</th>
<th>Identifier 000...063</th>
<th>AC500-Display</th>
<th>&lt;-- Display in</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class</td>
<td>Comp</td>
<td>Dev</td>
<td>Mod</td>
<td>Ch</td>
<td>Err</td>
<td>PS501 PLC Browser</td>
<td></td>
</tr>
<tr>
<td>Byte 6</td>
<td>Interface</td>
<td>Device</td>
<td>Module</td>
<td>Channel</td>
<td>Error Identifier</td>
<td>Error message</td>
<td>Remedy</td>
</tr>
<tr>
<td>Bit 6...7</td>
<td>Class</td>
<td>Interface</td>
<td>Device</td>
<td>Module</td>
<td>Channel</td>
<td>Error Identifier</td>
<td>Error message</td>
</tr>
</tbody>
</table>

### Module error

<table>
<thead>
<tr>
<th>Class</th>
<th>Interface</th>
<th>Device</th>
<th>Module</th>
<th>Channel</th>
<th>Error Identifier</th>
<th>Error message</th>
<th>Remedy</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>14</td>
<td>1...10</td>
<td>31</td>
<td>31</td>
<td>19</td>
<td>Checksum error in the I/O module</td>
<td>Replace I/O module</td>
</tr>
<tr>
<td></td>
<td>11 / 12</td>
<td>ADR</td>
<td>1...10</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>14</td>
<td>1...10</td>
<td>31</td>
<td>31</td>
<td>43</td>
<td>Internal error in the module</td>
<td>Replace I/O module</td>
</tr>
<tr>
<td></td>
<td>11 / 12</td>
<td>ADR</td>
<td>1...10</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>14</td>
<td>1...10</td>
<td>31</td>
<td>31</td>
<td>9</td>
<td>Overflow diagnosis buffer</td>
<td>Restart</td>
</tr>
<tr>
<td></td>
<td>11 / 12</td>
<td>ADR</td>
<td>1...10</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>14</td>
<td>1...10</td>
<td>31</td>
<td>31</td>
<td>26</td>
<td>Parameter error</td>
<td>Check master</td>
</tr>
<tr>
<td></td>
<td>11 / 12</td>
<td>ADR</td>
<td>1...10</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Remarks:

1) In AC500 the following interface identifier applies:
   - 14 = I/O bus,
   - 11 = COM1 (e.g. CS31-Bus),
   - 12 = COM2.
   The PNIO diagnosis block does not contain this identifier.

2) With "Device" the following allocation applies:
   - 31 = module itself,
   - 1...10 = decentralized communication interface module 1...10,
   - ADR = hardware address (e.g. of the DC551-CS31)

3) With "Module" the following allocation applies depending on the master:
   - Module error: I/O bus or PNIO: 31 = module itself;
   - COM1/COM2: 1...10 = expansion 1...10
   - Channel error: I/O bus or PNIO = module type (2 = DO);
   - COM1/COM2: 1...10 = expansion 1...10

4) In case of module errors, with channel "31 = Module itself" is output.
State LEDs

<table>
<thead>
<tr>
<th>LED</th>
<th>State</th>
<th>Color</th>
<th>LED = OFF</th>
<th>LED = ON</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Outputs O0...O7</td>
<td>Digital output</td>
<td>Yellow</td>
<td>Output is OFF</td>
</tr>
</tbody>
</table>

Technical data

The System Data of AC500-eCo apply
Only additional details are therefore documented below.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Galvanic isolation</td>
<td>Yes, between the channels and the rest of the module</td>
</tr>
<tr>
<td>Isolated groups</td>
<td>8 (1 channel per group)</td>
</tr>
<tr>
<td>Current consumption from 24 VDC power supply at the L+/UP and M/ZP terminals of the CPU/Bus Module</td>
<td>Ca. 10 mA</td>
</tr>
<tr>
<td>Max. power dissipation within the module</td>
<td>On Request</td>
</tr>
<tr>
<td>Weight</td>
<td>ca. 120 g</td>
</tr>
<tr>
<td>Mounting position</td>
<td>Horizontal or vertical</td>
</tr>
<tr>
<td>Cooling</td>
<td>The natural convection cooling must not be hindered by cable ducts or other parts in the switch-gear cabinet.</td>
</tr>
</tbody>
</table>

No effects of multiple overloads

No effects of multiple overloads on isolated multi-channel modules occur, as every channel is protected individually by an external fuse.

Technical data of the digital outputs

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of channels per module</td>
<td>8 triac outputs</td>
</tr>
<tr>
<td>Distribution of the channels into groups</td>
<td>8 groups (1 channel per group)</td>
</tr>
<tr>
<td>Connection of the channels O0 to O7</td>
<td>Terminals 1, 3, 5, 7, 10, 12, 14, 16</td>
</tr>
<tr>
<td>Reference potential for the channels O0 to O7</td>
<td>Terminals 2, 4, 6, 8, 11, 13, 15, 17</td>
</tr>
<tr>
<td>Output voltage for signal 1</td>
<td>On Request</td>
</tr>
<tr>
<td>Max. leakage current with signal 0</td>
<td>1.1 mA root mean square at 132 VAC and 1.8 mA root mean square at 264 VAC</td>
</tr>
<tr>
<td>Output voltage</td>
<td>120 VAC or 240 VAC</td>
</tr>
<tr>
<td>Indication of the output signals</td>
<td>1 yellow LED per channel; the LED is on when the output signal is high (signal 1) and the module is powered via the I/O bus</td>
</tr>
<tr>
<td>Parameter</td>
<td>Value</td>
</tr>
<tr>
<td>------------------------------------------------</td>
<td>-----------------------------------------------------------------------</td>
</tr>
<tr>
<td>Way of operation</td>
<td>Non-latching type</td>
</tr>
<tr>
<td>Output delay</td>
<td>On Request</td>
</tr>
<tr>
<td>Output data length</td>
<td>1 byte</td>
</tr>
<tr>
<td>Output current</td>
<td></td>
</tr>
<tr>
<td>Rated current per channel (max.)</td>
<td>0.3 A</td>
</tr>
<tr>
<td>Rated current per group (max.)</td>
<td>0.3 A</td>
</tr>
<tr>
<td>Surge current (max.)</td>
<td>On request</td>
</tr>
<tr>
<td>Lamp load (max.)</td>
<td>On request</td>
</tr>
<tr>
<td>Spark suppression with inductive AC loads</td>
<td>Must be performed externally according to driven load specification</td>
</tr>
<tr>
<td>Switching Frequencies</td>
<td></td>
</tr>
<tr>
<td>With resistive loads</td>
<td>Max. 10 Hz</td>
</tr>
<tr>
<td>With inductive loads</td>
<td>Not applicable</td>
</tr>
<tr>
<td>With lamp loads</td>
<td>Max. 10 Hz</td>
</tr>
<tr>
<td>Output type</td>
<td>Non-protected</td>
</tr>
<tr>
<td>Protection type</td>
<td>External fuse on each channel</td>
</tr>
<tr>
<td>Rated protection fuse</td>
<td>2 A fast</td>
</tr>
<tr>
<td>Short-circuit-proof / Overload-proof</td>
<td>No, should be provided by an external fuse or circuit breaker</td>
</tr>
<tr>
<td>Overload message</td>
<td>No</td>
</tr>
<tr>
<td>Output current limitation</td>
<td>No</td>
</tr>
<tr>
<td>Resistance to feedback against 230 VAC</td>
<td>No</td>
</tr>
<tr>
<td>Connection of 2 outputs in parallel</td>
<td>Not applicable</td>
</tr>
<tr>
<td>Max. cable length</td>
<td></td>
</tr>
<tr>
<td>Shielded</td>
<td>500 m</td>
</tr>
<tr>
<td>Unshielded</td>
<td>150 m</td>
</tr>
</tbody>
</table>

### Ordering data

<table>
<thead>
<tr>
<th>Part no.</th>
<th>Description</th>
<th>Product life cycle phase *)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1TNE 968 902 R2203</td>
<td>DO572, digital output module, 8 DO, triac output</td>
<td>Active</td>
</tr>
<tr>
<td>1TNE 968 901 R3101</td>
<td>Terminal block TA563-9, 9 pins, screw front, cable side, 6 pieces per unit</td>
<td>Active</td>
</tr>
<tr>
<td>1TNE 968 901 R3102</td>
<td>Terminal block TA563-11, 11 pins, screw front, cable side, 6 pieces per unit</td>
<td>Active</td>
</tr>
<tr>
<td>1TNE 968 901 R3103</td>
<td>Terminal block TA564-9, 9 pins, screw front, cable front, 6 pieces per unit</td>
<td>Active</td>
</tr>
<tr>
<td>1TNE 968 901 R3104</td>
<td>Terminal block TA564-11, 11 pins, screw front, cable front, 6 pieces per unit</td>
<td>Active</td>
</tr>
<tr>
<td>Part no.</td>
<td>Description</td>
<td>Product life cycle phase *)</td>
</tr>
<tr>
<td>---------------------</td>
<td>------------------------------------------------------------------------------</td>
<td>-----------------------------</td>
</tr>
<tr>
<td>1TNE 968 901 R3105</td>
<td>Terminal block TA565-9, 9 pins, spring front, cable front, 6 pieces per unit</td>
<td>Active</td>
</tr>
<tr>
<td>1TNE 968 901 R3106</td>
<td>Terminal block TA565-11, 11 pins, spring front, cable front, 6 pieces per unit</td>
<td>Active</td>
</tr>
</tbody>
</table>

*) For planning and commissioning of new installations use modules in Active status only.

**DO573 - Digital output module**
- 16 digital normally open relay outputs 24 VDC or 100-240 VAC (NO0 to NO15) in 2 groups, 2 A max.
- Group-wise electrically isolated

![Diagram of DO573 module]

1 I/O bus
2 16 yellow LEDs to display the signal states of the outputs O0 to O15
3 Terminal number
4 Allocation of signal name
5 Terminal block for output signals (9-pin)
6 Terminal block for output signals (11-pin)
7 2 holes for wall-mounting with screws
8 DIN rail

Intended purpose

The device can be used as a decentralized I/O extension module for S500 Communication Interface Modules (e.g. CI592-CS31, CI501-PNIO, CI541-DP, CI581-CN) or as a centralized extension module for AC500 CPUs (PM5xx).

The outputs are group-wise electrically isolated from each other.
All other circuitry of the module is electrically isolated from the outputs.

The I/O module must not be used as a decentralized I/O module with CI590-CS31-HA bus modules.

Functionality

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>LED displays</td>
<td>For signal states</td>
</tr>
<tr>
<td>Internal power supply</td>
<td>Via I/O bus</td>
</tr>
<tr>
<td>External power supply</td>
<td>Via the terminals L+ (process voltage 24 VDC) and M (0 VDC); the M terminal is connected to the M terminal of the CPU via the I/O bus</td>
</tr>
</tbody>
</table>

Electrical connection

For a detailed description of the mounting, disassembly and electrical connection of the module, please refer to the System Assembly chapter.

The electrical connection is carried out by using removable 9-pin and 11-pin terminal blocks. These terminal blocks differ in their connection system (spring terminals or screw terminals, cable mounting from the front or from the side). For more information, please refer to the chapter Terminal Blocks for S500-eCo I/O Modules. The terminal blocks are not included in the module's scope of delivery and must be ordered separately.

The following block diagram shows the internal construction of the digital outputs:
Table 547: Assignment of the terminals:

<table>
<thead>
<tr>
<th>Terminal</th>
<th>Signal</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>NO0</td>
<td>Normally-open contact of the output NO0</td>
</tr>
<tr>
<td>2</td>
<td>NO1</td>
<td>Normally-open contact of the output NO1</td>
</tr>
<tr>
<td>3</td>
<td>NO2</td>
<td>Normally-open contact of the output NO2</td>
</tr>
<tr>
<td>4</td>
<td>NO3</td>
<td>Normally-open contact of the output NO3</td>
</tr>
<tr>
<td>5</td>
<td>NO4</td>
<td>Normally-open contact of the output NO4</td>
</tr>
<tr>
<td>6</td>
<td>NO5</td>
<td>Normally-open contact of the output NO5</td>
</tr>
<tr>
<td>7</td>
<td>NO6</td>
<td>Normally-open contact of the output NO6</td>
</tr>
<tr>
<td>8</td>
<td>NO7</td>
<td>Normally-open contact of the output NO7</td>
</tr>
<tr>
<td>9</td>
<td>R0..7</td>
<td>Output common for signals NO0 to NO7</td>
</tr>
<tr>
<td>10</td>
<td>NO8</td>
<td>Normally-open contact of the output NO8</td>
</tr>
<tr>
<td>11</td>
<td>NO9</td>
<td>Normally-open contact of the output NO9</td>
</tr>
<tr>
<td>12</td>
<td>NO10</td>
<td>Normally-open contact of the output NO10</td>
</tr>
<tr>
<td>13</td>
<td>NO11</td>
<td>Normally-open contact of the output NO11</td>
</tr>
<tr>
<td>14</td>
<td>NO12</td>
<td>Normally-open contact of the output NO12</td>
</tr>
</tbody>
</table>

Table 547: Assignment of the terminals:
<table>
<thead>
<tr>
<th>Terminal</th>
<th>Signal</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>NO13</td>
<td>Normally-open contact of the output NO13</td>
</tr>
<tr>
<td>16</td>
<td>NO14</td>
<td>Normally-open contact of the output NO14</td>
</tr>
<tr>
<td>17</td>
<td>NO15</td>
<td>Normally-open contact of the output NO15</td>
</tr>
<tr>
<td>18</td>
<td>R8..15</td>
<td>Output common for signals NO8 to NO15</td>
</tr>
<tr>
<td>19</td>
<td>L+</td>
<td>Process voltage L+ (24 VDC)</td>
</tr>
<tr>
<td>20</td>
<td>M</td>
<td>Process voltage M (0 VDC)</td>
</tr>
</tbody>
</table>

The internal power supply voltage for the module's circuitry is carried out via the I/O bus (provided by a bus module or a CPU). Thus, the current consumption from 24 VDC power supply at the terminals L+/UP and M/ZP of the CPU/bus module increases by 5 mA per DO573.

The external power supply connection is carried out via the L+ (+24 VDC) and the M (0 VDC) terminals. The M terminal is electrically interconnected to the M/ZP terminal of the CPU/bus module.

**WARNING!**

**Risk of death by electric shock!**

Hazardous voltages can be present at the terminals of the module.

Make sure that all voltage sources (supply voltage and process supply voltage) are switched off before you begin with operations at the system.

For screw-type terminals only:

**WARNING!**

**For screw terminals only: Danger of death by electric shock!**

The IP 20 protection degree is only provided if all terminal screws are tightened.

Tighten all screws of unused load terminals of relay outputs if voltages > 24 V are connected to the relay group.

**WARNING!**

**Removal/Insertion under power**

The devices are not designed for removal or insertion under power. Because of unforeseeable consequences, it is not allowed to plug or unplug devices with the power being ON.

Make sure that all voltage sources (supply and process voltage) are switched off before you

- connect or disconnect any signal or terminal block
- remove, mount or replace a module.

Disconnecting any powered devices while energized in a hazardous location could result in an electric arc, which could create a flammable ignition resulting in fire or explosion.

Make sure that power is removed and that the area has been thoroughly checked to ensure that flammable materials are not present prior to proceeding.

The devices must not be opened when in operation. The same applies to the network interfaces.
**NOTICE!**

**Risk of damaging the I/O module!**

The outputs are not protected against short circuit and overload.

- Never short-circuit or overload the outputs.
- Never connect inductive loads without an external suppression against voltage peaks due to inductive kickback.
- Never connect voltages > 240 V. All outputs must be supplied from the same phase.
- Use an external 5 A fast protection fuse for the outputs.

---

**NOTICE!**

**Risk of damaging the PLC modules!**

The PLC modules can be damaged by overload.

Make sure that the total current of each output common terminal (R0..7 and R8..15) does not exceed 10 A.

Never connect total currents > 10 A per group.

The following figure shows the electrical connection of the module:
Fig. 102: Connection of 24 VDC actuators
The module provides several diagnosis functions (see section Diagnosis "Chapter 1.5.3.6.1.11.6 "Diagnosis" on page 2424). The meaning of the LEDs is described in the section State LEDs "Chapter 1.5.3.6.1.10.7 "State LEDs" on page 2414.

Fig. 103: Connection of 100-240 VAC actuators

The module provides several diagnosis functions (see section Diagnosis "Chapter 1.5.3.6.1.11.6 "Diagnosis" on page 2424). The meaning of the LEDs is described in the section State LEDs "Chapter 1.5.3.6.1.10.7 "State LEDs" on page 2414.
The L+ connection of the DO573 and the 24 V supply of the CPU/bus module must be connected to the same 24 V power supply.

I/O configuration

The module itself does not store configuration data. It receives its parameterization data from the master device of the I/O bus (CPU or bus module) during power-up of the system. Hence, replacing I/O modules is possible without any re-parameterization via software.

If the external power supply voltage via UP/ZP terminals fails, the I/O module loses its configuration data. The whole station has to be switched off and on again to re-configure the module.

Parameterization

The arrangement of the parameter data is performed with Automation Builder software. The parameter data directly influences the functionality of modules. For non-standard applications, it is necessary to adapt the parameters to your system configuration.
<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>Internal Value</th>
<th>Internal Value, Type</th>
<th>Default Value</th>
<th>Min.</th>
<th>Max.</th>
<th>EDS Slot Index</th>
</tr>
</thead>
<tbody>
<tr>
<td>Module ID</td>
<td>Internal</td>
<td>6150 1)</td>
<td>WORD</td>
<td>6150</td>
<td>0</td>
<td>65535</td>
<td>xx01</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0x1806</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ignore module</td>
<td>No</td>
<td>0</td>
<td>BYTE</td>
<td>No</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Yes</td>
<td>1</td>
<td></td>
<td>(0x00)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Parameter length</td>
<td>Internal</td>
<td>1</td>
<td>BYTE</td>
<td>0</td>
<td>0</td>
<td>255</td>
<td>xx02 2)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0x01</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Check supply</td>
<td>Off</td>
<td>0</td>
<td>BYTE</td>
<td>On</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>On</td>
<td>1</td>
<td></td>
<td>0x01</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

1) with CS31 and addresses less than 70, the value is increased by 1
2) Value is hexadecimal: HighByte is slot (xx: 0...7), LowByte is index (1...n)

GSD file:

```
Ext_User_Prm_Data.Len =
Ext_User_Prm_Data.Const(0) =
0x07 0x18, 0x07, 0x00, 0x03, 0x01, 0x00, 0x00;
```

**Diagnosis**

<table>
<thead>
<tr>
<th>E1...E4</th>
<th>d1</th>
<th>d2</th>
<th>d3</th>
<th>d4</th>
<th>Identifier 000...063</th>
<th>AC500-Display</th>
<th>&lt;- Display in PLC Browser</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class</td>
<td>Comp</td>
<td>Dev</td>
<td>Mod</td>
<td>Ch</td>
<td>Err</td>
<td>PS501 PLC Browser</td>
<td></td>
</tr>
<tr>
<td>Byte 6 Bit 6...7</td>
<td>Byte 3</td>
<td>Byte 4</td>
<td>Byte 5</td>
<td>Byte 6 Bit 0...5</td>
<td>PNIO diagnosis block</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Class</td>
<td>Interface</td>
<td>Device</td>
<td>Module</td>
<td>Channel</td>
<td>Error-Identifier</td>
<td>Error message</td>
<td>Remedy</td>
</tr>
<tr>
<td>---------</td>
<td>------------</td>
<td>---------</td>
<td>---------</td>
<td>---------</td>
<td>----------------</td>
<td>---------------</td>
<td>--------</td>
</tr>
<tr>
<td>1)</td>
<td>2)</td>
<td>3)</td>
<td>4)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Module error**

| 3 | 14   | 1...10 | 31   | 31   | 19 | Checksum error in the I/O module | Replace I/O module |
|   | 11 / 12   | ADR | 1...10   |       |       |               |                      |
| 3 | 14   | 1...10 | 31   | 31   | 43 | Internal error in the module | Replace I/O module |
|   | 11 / 12   | ADR | 1...10   |       |       |               |                      |
| 3 | 14   | 1...10 | 31   | 31   | 9  | Overflow diagnosis buffer | Restart |
|   | 11 / 12   | ADR | 1...10   |       |       |               |                      |
| 4 | 14   | 1...10 | 31   | 31   | 26 | Parameter error | Check master |
|   | 11 / 12   | ADR | 1...10   |       |       |               |                      |
### Remarks:

1) In AC500 the following interface identifier applies:
   14 = I/O bus, 11 = COM1 (e.g. CS31-Bus), 12 = COM2.
   The PNIO diagnosis block does not contain this identifier.

2) With "Device" the following allocation applies:
   31 = Module itself, 1...10 = decentralized communication interface module 1...10,
   ADR = Hardware address (e.g. of the DC551-CS31)

3) With "Module" the following allocation applies depending on the master:
   Module error: I/O bus or PNIO: 31 = module itself; COM1/COM2: 1...10 = expansion 1...10
   Channel error: I/O bus or PNIO = module type (2 = DO); COM1/COM2: 1...10 = expansion 1...10

4) In case of module errors, with channel "31 = Module itself" is output.

### Technical data

The System Data of AC500-eCo apply
Only additional details are therefore documented below.
### Technical data of the digital outputs

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of channels per module</td>
<td>16 normally-open relay outputs</td>
</tr>
<tr>
<td>Distribution of the channels into groups</td>
<td>2 (8 channels per group)</td>
</tr>
<tr>
<td>Connection of the channels NO0 to NO7</td>
<td>Terminals 1 to 8</td>
</tr>
<tr>
<td>Connection of the channels NO8 to NO15</td>
<td>Terminals 10 to 17</td>
</tr>
<tr>
<td>Reference potential for the channels NO0 to NO7</td>
<td>Terminal 9 (signal name R0..7)</td>
</tr>
<tr>
<td>Reference potential for the channels NO8 to NO15</td>
<td>Terminal 18 (signal name R8..15)</td>
</tr>
<tr>
<td>Relay coil power supply</td>
<td>Terminals 19 and 20 (signal names L+ and M)</td>
</tr>
<tr>
<td>Indication of the output signals</td>
<td>1 yellow LED per channel; the LED is on when the output signal is high (signal 1) and the module is powered via the I/O bus</td>
</tr>
<tr>
<td>Way of operation</td>
<td>Non-latching type</td>
</tr>
<tr>
<td>Relay output voltage</td>
<td>24 VDC or 120/240 VAC</td>
</tr>
</tbody>
</table>

---

**No effects of multiple overloads**

No effects of multiple overloads on isolated multi-channel modules occur, as every channel is protected individually by an external fuse.
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output delay</td>
<td></td>
</tr>
<tr>
<td>Switching 0 to 1 (max.)</td>
<td>Typ. 10 ms</td>
</tr>
<tr>
<td>Switching 1 to 0 (max.)</td>
<td>Typ. 10 ms</td>
</tr>
<tr>
<td>Output data length</td>
<td>2 bytes</td>
</tr>
<tr>
<td>Output current</td>
<td></td>
</tr>
<tr>
<td>Rated current per channel (max.)</td>
<td>2.0 A (24 VDC / 24 VAC / 48 VAC / 120 VAC / 240 VAC, only resistive loads)</td>
</tr>
<tr>
<td></td>
<td>2.0 A (24 VAC / 48 VAC / 120 VAC, only pilot duty)</td>
</tr>
<tr>
<td></td>
<td>1.5 A (240 VAC, only pilot duty)</td>
</tr>
<tr>
<td>Rated current per group (max.)</td>
<td>10 A</td>
</tr>
<tr>
<td>Lamp load (max.)</td>
<td>200 W (230 VAC), 30 W (24 VDC)</td>
</tr>
<tr>
<td>Spark suppression with inductive AC loads</td>
<td>Must be performed externally according to driven load specification</td>
</tr>
<tr>
<td>Switching Frequencies</td>
<td></td>
</tr>
<tr>
<td>With resistive loads</td>
<td>Max. 1 Hz</td>
</tr>
<tr>
<td>With inductive loads</td>
<td>On Request</td>
</tr>
<tr>
<td>With lamp loads</td>
<td>Max. 1 Hz</td>
</tr>
<tr>
<td>Output type</td>
<td>Non-protected</td>
</tr>
<tr>
<td>Protection type</td>
<td>External fuse on each channel</td>
</tr>
<tr>
<td>Rated protection fuse (for each channel)</td>
<td>5 A fast</td>
</tr>
<tr>
<td>Short-circuit-proof / Overload-proof</td>
<td>No, should be provided by an external fuse or circuit breaker</td>
</tr>
<tr>
<td>Overload message</td>
<td>No</td>
</tr>
<tr>
<td>Output current limitation</td>
<td>No</td>
</tr>
<tr>
<td>Connection of 2 outputs in parallel</td>
<td>Not possible</td>
</tr>
<tr>
<td>Life time of relay contacts (cycles)</td>
<td>100.000 at rated load</td>
</tr>
<tr>
<td>Max. cable length</td>
<td></td>
</tr>
<tr>
<td>Shielded</td>
<td>500 m</td>
</tr>
<tr>
<td>Unshielded</td>
<td>150 m</td>
</tr>
</tbody>
</table>

### Ordering data

<table>
<thead>
<tr>
<th>Part no.</th>
<th>Description</th>
<th>Product life cycle phase *)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1SAP 231 300 R0000</td>
<td>DO573, digital output module, 16 DO, relay output</td>
<td>Active</td>
</tr>
<tr>
<td>1TNE 968 901 R3101</td>
<td>Terminal block TA563-9, 9 pins, screw front, cable side, 6 pieces per unit</td>
<td>Active</td>
</tr>
<tr>
<td>1TNE 968 901 R3102</td>
<td>Terminal block TA563-11, 11 pins, screw front, cable side, 6 pieces per unit</td>
<td>Active</td>
</tr>
<tr>
<td>1TNE 968 901 R3103</td>
<td>Terminal block TA564-9, 9 pins, screw front, cable front, 6 pieces per unit</td>
<td>Active</td>
</tr>
<tr>
<td>Part no.</td>
<td>Description</td>
<td>Product life cycle phase *)</td>
</tr>
<tr>
<td>--------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>------------------------------</td>
</tr>
<tr>
<td>1TNE 968 901 R3104</td>
<td>Terminal block TA564-11, 11 pins, screw front, cable front, 6 pieces per unit</td>
<td>Active</td>
</tr>
<tr>
<td>1TNE 968 901 R3105</td>
<td>Terminal block TA565-9, 9 pins, spring front, cable front, 6 pieces per unit</td>
<td>Active</td>
</tr>
<tr>
<td>1TNE 968 901 R3106</td>
<td>Terminal block TA565-11, 11 pins, spring front, cable front, 6 pieces per unit</td>
<td>Active</td>
</tr>
</tbody>
</table>

*) For planning and commissioning of new installations use modules in Active status only.

DX561 - Digital input/output module

- 8 digital inputs 24 VDC (I0 to I7) in 1 group
- 8 digital transistor outputs 24 VDC (O0 to O7) in 1 group
- Group-wise electrically isolated
1 I/O bus
2 8 yellow LEDs to display the signal states of the inputs I0 to I7
3 8 yellow LEDs to display the signal states of the outputs O0 to O7
4 Terminal number
5 Allocation of signal name
6 Terminal block for input signals (9-pin)
7 Terminal block for output signals (11-pin)
8 2 holes for wall-mounting with screws
9 DIN rail

Intended purpose

The device can be used as a decentralized I/O extension module for S500 Communication Interface Modules (e.g. CI592-CS31, CI501-PNIO, CI541-DP, CI581-CN) or as a centralized extension module for AC500 CPUs (PM5xx).

The inputs and outputs are group-wise electrically isolated from each other.
All other circuitry of the module is electrically isolated from the inputs.
The I/O module must not be used as a decentralized I/O module with CI590-CS31-HA bus modules.

### Functionality

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>LED displays</td>
<td>For signal states</td>
</tr>
<tr>
<td>Internal power supply</td>
<td>Via I/O bus</td>
</tr>
<tr>
<td>External power supply</td>
<td>Via the terminals ZP and UP (process voltage 24 VDC)</td>
</tr>
</tbody>
</table>

### Electrical connection

For a detailed description of the mounting, disassembly and electrical connection of the module, please refer to the System Assembly chapter.

The electrical connection is carried out by using removable 9-pin and 11-pin terminal blocks. These terminal blocks differ in their connection system (spring terminals or screw terminals, cable mounting from the front or from the side). For more information, please refer to the chapter Terminal Blocks for S500-eCo I/O Modules. The terminal blocks are not included in the module’s scope of delivery and must be ordered separately.

The following block diagram shows the internal construction of the digital inputs and outputs:
Table 548: Assignment of the terminals:

<table>
<thead>
<tr>
<th>Terminal</th>
<th>Signal</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>C0..7</td>
<td>Input common for signals I0 to I7</td>
</tr>
<tr>
<td>2</td>
<td>I0</td>
<td>Input signal I0</td>
</tr>
<tr>
<td>3</td>
<td>I1</td>
<td>Input signal I1</td>
</tr>
<tr>
<td>4</td>
<td>I2</td>
<td>Input signal I2</td>
</tr>
<tr>
<td>5</td>
<td>I3</td>
<td>Input signal I3</td>
</tr>
<tr>
<td>6</td>
<td>I4</td>
<td>Input signal I4</td>
</tr>
<tr>
<td>7</td>
<td>I5</td>
<td>Input signal I5</td>
</tr>
<tr>
<td>8</td>
<td>I6</td>
<td>Input signal I6</td>
</tr>
<tr>
<td>9</td>
<td>I7</td>
<td>Input signal I7</td>
</tr>
<tr>
<td>10</td>
<td>---</td>
<td>Reserved</td>
</tr>
<tr>
<td>11</td>
<td>O0</td>
<td>Output signal O0</td>
</tr>
<tr>
<td>12</td>
<td>O1</td>
<td>Output signal O1</td>
</tr>
<tr>
<td>13</td>
<td>O2</td>
<td>Output signal O2</td>
</tr>
</tbody>
</table>
### Table

<table>
<thead>
<tr>
<th>Terminal</th>
<th>Signal</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>14</td>
<td>O3</td>
<td>Output signal O3</td>
</tr>
<tr>
<td>15</td>
<td>O4</td>
<td>Output signal O4</td>
</tr>
<tr>
<td>16</td>
<td>O5</td>
<td>Output signal O5</td>
</tr>
<tr>
<td>17</td>
<td>O6</td>
<td>Output signal O6</td>
</tr>
<tr>
<td>18</td>
<td>O7</td>
<td>Output signal O7</td>
</tr>
<tr>
<td>19</td>
<td>UP</td>
<td>Process voltage UP +24 VDC</td>
</tr>
<tr>
<td>20</td>
<td>ZP</td>
<td>Process voltage ZP 0 VDC</td>
</tr>
</tbody>
</table>

The internal power supply voltage for the module's circuitry is carried out via the I/O bus (provided by a bus module or a CPU). Thus, the current consumption from 24 VDC power supply at the terminals L+/UP and M/ZP of the CPU/bus module increases by 10 mA per DX561.

The external power supply connection is carried out via the UP (+24 VDC) and ZP (0 VDC) terminals.

---

**WARNING!**

**Removal/Insertion under power**

The devices are not designed for removal or insertion under power. Because of unforeseeable consequences, it is not allowed to plug or unplug devices with the power being ON.

Make sure that all voltage sources (supply and process voltage) are switched off before you

– connect or disconnect any signal or terminal block
– remove, mount or replace a module.

Disconnecting any powered devices while energized in a hazardous location could result in an electric arc, which could create a flammable ignition resulting in fire or explosion.

Make sure that power is removed and that the area has been thoroughly checked to ensure that flammable materials are not present prior to proceeding.

The devices must not be opened when in operation. The same applies to the network interfaces.

---

**NOTICE!**

**Risk of damaging the PLC modules!**

Overvoltages and short circuits might damage the PLC modules.

– Make sure that all voltage sources (supply voltage and process supply voltage) are switched off before you begin with operations at the system.
– Never connect any voltages or signals to reserved terminals (marked with ---). Reserved terminals may carry internal voltages.

The digital inputs can be used as source inputs or as sink inputs.

---

**NOTICE!**

**Risk of malfunctions in the plant!**

A ground closure, e. g. caused by a damaged cable insulation, can bridge switches accidentally.

Use sink inputs when possible or make sure that, in case of error, there will be no risks to persons or plant.
The following figure shows the electrical connection of the inputs to the digital input/output module DX561:

**Fig. 105: Electrical connection of inputs - sink inputs**

The following figure shows the electrical connection of the outputs to the module:
NOTICE!
Risk of malfunctions in the plant!
The outputs may switch on for a period of 10 to 50 \(\mu\)s if the process supply voltage UP/ZP is switched on.
This must be considered in the planning of the application.

NOTICE!
Risk of damaging the I/O module!
The outputs are not protected against short circuits and overload.
- Never short-circuit or overload the outputs.
- Never connect the outputs to other voltages.
- Use an external 3 A fast-protection fuse for the outputs.

The module provides several diagnosis functions (see chapter Diagnosis \(\odot\) Chapter 1.5.3.6.1.1.12.6 “Diagnosis” on page 2436).
The meaning of the LEDs is described in the Displays section \(\odot\) Chapter 1.5.3.6.1.1.12.7 “State LEDs” on page 2437 chapter.

I/O configuration
The module itself does not store configuration data. It receives its parameterization data from
the master device of the I/O bus (CPU or bus module) during power-up of the system.
Hence, replacing I/O modules is possible without any re-parameterization via software.
If the external power supply voltage via UP/ZP terminals fails, the I/O module loses its configuration data. The whole station has to be switched off and on again to re-configure the module.

Parameterization

The arrangement of the parameter data is performed with Automation Builder software. The parameter data directly influences the functionality of modules. For non-standard applications, it is necessary to adapt the parameters to your system configuration.

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>Internal Value</th>
<th>Internal Value, Type</th>
<th>Default</th>
<th>Min.</th>
<th>Max.</th>
<th>EDS Slot Index</th>
</tr>
</thead>
<tbody>
<tr>
<td>Module ID</td>
<td>Internal</td>
<td>6135 ¹)</td>
<td>WORD</td>
<td>6135 0x17F7</td>
<td>0</td>
<td>65535</td>
<td>xx01</td>
</tr>
<tr>
<td>Ignore module</td>
<td>No</td>
<td>0</td>
<td>BYTE</td>
<td>No</td>
<td>(0x00)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Yes</td>
<td>1</td>
<td>BYTE</td>
<td>0</td>
<td></td>
<td></td>
<td>xx02 ²)</td>
</tr>
<tr>
<td>Parameter length</td>
<td>Internal</td>
<td>1</td>
<td>BYTE</td>
<td>0</td>
<td>0</td>
<td>255</td>
<td>xx02 ²)</td>
</tr>
</tbody>
</table>

¹) with CS31 and addresses smaller than 70, the value is increased by 1
²) Value is hexadecimal: HighByte is slot (xx: 0...7), LowByte is index (1...n)

GSD file:

```
Ext_User_Prm_Data_Len = 0x03
Ext_User_Prm_Data_Const(0) = 0xF8, 0x17, 0x00, 
(0) = 0x01;
```
## Diagnosis

<table>
<thead>
<tr>
<th>Class</th>
<th>Identifier</th>
<th>AC500-Display</th>
<th>&lt; Display in</th>
</tr>
</thead>
<tbody>
<tr>
<td>Identifier 000...063</td>
<td>PS501 PLC Browser</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Byte 6 Bit 6...7

<table>
<thead>
<tr>
<th>Class</th>
<th>Interface</th>
<th>Device</th>
<th>Module</th>
<th>Channel</th>
<th>Error Identifier</th>
<th>Error message</th>
<th>Remedy</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Module error

<table>
<thead>
<tr>
<th>Class</th>
<th>Interface</th>
<th>Device</th>
<th>Module</th>
<th>Channel</th>
<th>Error Identifier</th>
<th>Error message</th>
<th>Remedy</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td></td>
<td>1...10</td>
<td>31</td>
<td>31</td>
<td>19</td>
<td>Checksum error in the I/O module</td>
<td>Replace I/O module</td>
</tr>
<tr>
<td>11 / 12</td>
<td>ADR</td>
<td>1...10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td></td>
<td>1...10</td>
<td>31</td>
<td>31</td>
<td>43</td>
<td>Internal error in the module</td>
<td>Replace I/O module</td>
</tr>
<tr>
<td>11 / 12</td>
<td>ADR</td>
<td>1...10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td></td>
<td>1...10</td>
<td>31</td>
<td>31</td>
<td>9</td>
<td>Overflow diagnosis buffer</td>
<td>Restart</td>
</tr>
<tr>
<td>11 / 12</td>
<td>ADR</td>
<td>1...10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td></td>
<td>1...10</td>
<td>31</td>
<td>31</td>
<td>26</td>
<td>Parameter error</td>
<td>Check master</td>
</tr>
<tr>
<td>11 / 12</td>
<td>ADR</td>
<td>1...10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Remarks:

1) In AC500 the following interface identifier applies:
   14 = I/O bus, 11 = COM1 (e.g. CS31 bus), 12 = COM2.
   The PNIO diagnosis block does not contain this identifier.

2) With "Device" the following allocation applies:
   31 = module itself,
   1...10 = decentralized communication interface module 1...10,
   ADR = hardware address (e.g. of the DC551-CS31)

3) With "Module" the following allocation applies depending on the master:
   Module error: I/O bus or PNIO: 31 = module itself; COM1/COM2: 1...10 = expansion 1...10
   Channel error: I/O bus or PNIO = module type (2 = DO); COM1/COM2: 1...10 = expansion 1...10

4) In case of module errors, with channel "31 = module itself" is output.
State LEDs

<table>
<thead>
<tr>
<th>LED</th>
<th>State</th>
<th>Color</th>
<th>LED = OFF</th>
<th>LED = ON</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Inputs I0...I7</td>
<td>Digital input</td>
<td>Yellow</td>
<td>Input is OFF</td>
</tr>
<tr>
<td></td>
<td>Outputs O0...O7</td>
<td>Digital output</td>
<td>Yellow</td>
<td>Output is OFF</td>
</tr>
</tbody>
</table>

Technical data

The System Data of AC500-eCo apply
Only additional details are therefore documented below.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Process supply voltage UP</td>
<td></td>
</tr>
<tr>
<td>Connections</td>
<td>Terminal 19 for UP (+24 VDC) and terminal 20 for ZP (0 VDC)</td>
</tr>
<tr>
<td>Rated value</td>
<td>24 VDC</td>
</tr>
<tr>
<td>Current consumption via UP terminal</td>
<td>5 mA + max. 0.5 A per output</td>
</tr>
<tr>
<td>Max. ripple</td>
<td>5 %</td>
</tr>
<tr>
<td>Inrush current</td>
<td>0.000002 A²s</td>
</tr>
<tr>
<td>Protection against reversed voltage</td>
<td>Yes</td>
</tr>
<tr>
<td>Rated protection fuse for UP</td>
<td>Recommended; the outputs must be protected by an 3 A fast fuse</td>
</tr>
<tr>
<td>Current consumption from 24 VDC power supply at the L+/UP and M/ZP terminals of the CPU/bus module</td>
<td>Ca. 10 mA</td>
</tr>
<tr>
<td>Galvanic isolation</td>
<td>Yes, between the input group and the output group and the rest of the module</td>
</tr>
<tr>
<td>Isolated groups</td>
<td>2 groups (1 group for 8 input channels, 1 group for 8 output channels)</td>
</tr>
<tr>
<td>Surge-voltage (max.)</td>
<td>35 VDC for 0.5 s</td>
</tr>
<tr>
<td>Max. power dissipation within the module</td>
<td>2.3 W</td>
</tr>
<tr>
<td>Weight</td>
<td>ca. 120 g</td>
</tr>
<tr>
<td>Mounting position</td>
<td>Horizontal or vertical</td>
</tr>
<tr>
<td>Cooling</td>
<td>The natural convection cooling must not be hindered by cable ducts or other parts in the switch-gear cabinet.</td>
</tr>
</tbody>
</table>

No effects of multiple overloads

No effects of multiple overloads on isolated multi-channel modules occur, as every channel is protected individually by an external fuse.
Technical data of the digital inputs

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of channels per module</td>
<td>8</td>
</tr>
<tr>
<td>Distribution of the channels into groups</td>
<td>1 group for 8 channels</td>
</tr>
<tr>
<td>Connections of the channels I0 to I7</td>
<td>Terminals 2 to 9</td>
</tr>
<tr>
<td>Reference potential for the channels I0 to I7</td>
<td>Terminal 1</td>
</tr>
<tr>
<td>Indication of the input signals</td>
<td>1 yellow LED per channel; the LED is ON when the input signal is high (signal 1)</td>
</tr>
<tr>
<td>Monitoring point of input indicator</td>
<td>LED is part of the input circuitry</td>
</tr>
<tr>
<td>Input type according to EN 61131-2</td>
<td>Type 1 source Type 1 sink</td>
</tr>
<tr>
<td>Input signal range</td>
<td>-24 VDC +24 VDC</td>
</tr>
<tr>
<td>Signal 0</td>
<td>-5 V...+3 V -3 V...+5 V</td>
</tr>
<tr>
<td>Undefined signal</td>
<td>-15 V...+5 V +5 V...+15 V</td>
</tr>
<tr>
<td>Signal 1</td>
<td>-30 V...-15 V +15 V...+30 V</td>
</tr>
<tr>
<td>Ripple with signal 0</td>
<td>-5 V...+3 V -3 V...+5 V</td>
</tr>
<tr>
<td>Ripple with signal 1</td>
<td>-30 V...-15 V +15 V...+30 V</td>
</tr>
<tr>
<td>Input current per channel</td>
<td></td>
</tr>
<tr>
<td>Input voltage +24 V</td>
<td>Typ. 5 mA</td>
</tr>
<tr>
<td>Input voltage +5 V</td>
<td>Typ. 1 mA</td>
</tr>
<tr>
<td>Input voltage +15 V</td>
<td>&gt; 2.5 mA</td>
</tr>
<tr>
<td>Input voltage +30 V</td>
<td>&lt; 8 mA</td>
</tr>
<tr>
<td>Max. permissible leakage current (at 2-wire proximity switches)</td>
<td>1 mA</td>
</tr>
<tr>
<td>Input delay (0-&gt;1 or 1-&gt;0)</td>
<td>Typ. 8 ms</td>
</tr>
<tr>
<td>Input data length</td>
<td>1 byte</td>
</tr>
<tr>
<td>Max. cable length</td>
<td></td>
</tr>
<tr>
<td>Shielded</td>
<td>500 m</td>
</tr>
<tr>
<td>Unshielded</td>
<td>300 m</td>
</tr>
</tbody>
</table>

Technical data of the digital outputs

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of channels per module</td>
<td>8 transistor outputs (24 VDC, 0.5 A max.)</td>
</tr>
<tr>
<td>Distribution of the channels into groups</td>
<td>1 group of 8 channels</td>
</tr>
<tr>
<td>Connection of the channels O0 to O7</td>
<td>Terminals 11 to 18</td>
</tr>
<tr>
<td>Reference potential for the channels O0 to O7</td>
<td>Terminal 20 (negative pole of the process voltage, name ZP)</td>
</tr>
<tr>
<td>Common power supply voltage</td>
<td>Terminal 19 (positive pole of the process voltage, name UP)</td>
</tr>
<tr>
<td>Indication of the output signals</td>
<td>1 yellow LED per channel; the LED is on when the output signal is high (signal 1) and the module is powered via the I/O bus</td>
</tr>
<tr>
<td>Monitoring point of output indicator</td>
<td>Controlled together with transistor</td>
</tr>
<tr>
<td>Parameter</td>
<td>Value</td>
</tr>
<tr>
<td>------------------------------------------------</td>
<td>-----------------------------------------------------------------------</td>
</tr>
<tr>
<td>Way of operation</td>
<td>Non-latching type</td>
</tr>
<tr>
<td>Max. output voltage at signal 1</td>
<td>20 VDC at max. current consumption</td>
</tr>
<tr>
<td>Output delay</td>
<td></td>
</tr>
<tr>
<td>0 to 1</td>
<td>50 µs</td>
</tr>
<tr>
<td>1 to 0</td>
<td>200 µs</td>
</tr>
<tr>
<td>Output data length</td>
<td>1 byte</td>
</tr>
<tr>
<td>Output current</td>
<td></td>
</tr>
<tr>
<td>Rated current per channel (max.)</td>
<td>0.5 A at UP 24 VDC</td>
</tr>
<tr>
<td>Rated current per group (max.)</td>
<td>4 A</td>
</tr>
<tr>
<td>Rated current (all channels together, max.)</td>
<td>4 A</td>
</tr>
<tr>
<td>Lamp load (max.)</td>
<td>5 W</td>
</tr>
<tr>
<td>Max. leakage current with signal 0</td>
<td>0.5 mA</td>
</tr>
<tr>
<td>Output type</td>
<td>Non-protected</td>
</tr>
<tr>
<td>Protection type</td>
<td>External fuse on each channel</td>
</tr>
<tr>
<td>Rated protection fuse (for each channel)</td>
<td>3 A fast</td>
</tr>
<tr>
<td>Demagnetization when inductive loads are switched off</td>
<td>Must be performed externally according to driven load specification</td>
</tr>
<tr>
<td>Switching Frequencies</td>
<td></td>
</tr>
<tr>
<td>With inductive loads</td>
<td>Max. 0.5 Hz</td>
</tr>
<tr>
<td>With lamp loads</td>
<td>Max. 11 Hz at max. 5 W</td>
</tr>
<tr>
<td>Short-circuit-proof / Overload-proof</td>
<td>No</td>
</tr>
<tr>
<td>Overload message</td>
<td>No</td>
</tr>
<tr>
<td>Output current limitation</td>
<td>No</td>
</tr>
<tr>
<td>Resistance to feedback against 24 VDC</td>
<td>No</td>
</tr>
<tr>
<td>Connection of 2 outputs in parallel</td>
<td>Not possible</td>
</tr>
<tr>
<td>Max. cable length</td>
<td></td>
</tr>
<tr>
<td>Shielded</td>
<td>500 m</td>
</tr>
<tr>
<td>Unshielded</td>
<td>150 m</td>
</tr>
</tbody>
</table>

**Ordering data**

<table>
<thead>
<tr>
<th>Part no.</th>
<th>Description</th>
<th>Product life cycle phase *)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1TNE 968 902 R2301</td>
<td>DX561, digital input/output module, 8 DI 24 VDC, 8 DO 24 VDC, transistor output</td>
<td>Active</td>
</tr>
<tr>
<td>1TNE 968 901 R3101</td>
<td>Terminal block TA563-9, 9 pins, screw front, cable side, 6 pieces per unit</td>
<td>Active</td>
</tr>
<tr>
<td>1TNE 968 901 R3102</td>
<td>Terminal block TA563-11, 11 pins, screw front, cable side, 6 pieces per unit</td>
<td>Active</td>
</tr>
<tr>
<td>1TNE 968 901 R3103</td>
<td>Terminal block TA564-9, 9 pins, screw front, cable front, 6 pieces per unit</td>
<td>Active</td>
</tr>
</tbody>
</table>
### Terminal Blocks

<table>
<thead>
<tr>
<th>Part no.</th>
<th>Description</th>
<th>Product life cycle phase *</th>
</tr>
</thead>
<tbody>
<tr>
<td>1TNE 968 901 R3104</td>
<td>Terminal block TA564-11, 11 pins, screw front, cable front, 6 pieces per unit</td>
<td>Active</td>
</tr>
<tr>
<td>1TNE 968 901 R3105</td>
<td>Terminal block TA565-9, 9 pins, spring front, cable front, 6 pieces per unit</td>
<td>Active</td>
</tr>
<tr>
<td>1TNE 968 901 R3106</td>
<td>Terminal block TA565-11, 11 pins, spring front, cable front, 6 pieces per unit</td>
<td>Active</td>
</tr>
</tbody>
</table>

* For planning and commissioning of new installations use modules in Active status only.

---

### DX571 - Digital input/output module

- 8 digital inputs 24 VDC / 24 VAC (I0 to I7) in 1 group
- 8 digital normally open relay outputs 24 VDC / 24 VAC or 100-240 VAC, 2 A max. (NO0 to NO7) in 2 groups
- Group-wise electrically isolated
1. I/O bus
2. 8 yellow LEDs to display the signal states of the inputs I0 to I7
3. 8 yellow LEDs to display the signal states of the outputs NO0 to NO7
4. Terminal number
5. Allocation of signal name
6. Terminal block for input signals (9-pin)
7. Terminal block for output signals (11-pin)
8. 2 holes for wall-mounting with screws
9. DIN rail

**Intended purpose**

The device can be used as a decentralized I/O extension module for S500 Communication Interface Modules (e.g., CI592-CS31, CI501-PNIO, CI541-DP, CI581-CN) or as a centralized extension module for AC500 CPUs (PM5xx).

The inputs and outputs are group-wise electrically isolated from each other.

All other circuitry of the module is electrically isolated from the inputs.
The I/O module must not be used as a decentralized I/O module with CI590-CS31-HA bus modules.

### Functionality

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>LED displays</td>
<td>For signal states</td>
</tr>
<tr>
<td>Internal power supply</td>
<td>Via I/O bus</td>
</tr>
<tr>
<td>External power supply</td>
<td>Via the terminal L+ (process voltage 24 VDC). The negative pole is provided by the I/O bus.</td>
</tr>
</tbody>
</table>

### Electrical connection

For a detailed description of the mounting, disassembly and electrical connection of the module, please refer to the System Assembly chapter.

The electrical connection is carried out by using removable 9-pin and 11-pin terminal blocks. These terminal blocks differ in their connection system (spring terminals or screw terminals, cable mounting from the front or from the side). For more information, please refer to the chapter Terminal Blocks for S500-eCo I/O Modules. The terminal blocks are not included in the module's scope of delivery and must be ordered separately.

The following block diagram shows the internal construction of the digital inputs and outputs:
Table 549: Assignment of the terminals:

<table>
<thead>
<tr>
<th>Terminal</th>
<th>Signal</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>C0..7</td>
<td>Input common for signals I0 to I7</td>
</tr>
<tr>
<td>2</td>
<td>I0</td>
<td>Input signal I0</td>
</tr>
<tr>
<td>3</td>
<td>I1</td>
<td>Input signal I1</td>
</tr>
<tr>
<td>4</td>
<td>I2</td>
<td>Input signal I2</td>
</tr>
<tr>
<td>5</td>
<td>I3</td>
<td>Input signal I3</td>
</tr>
<tr>
<td>6</td>
<td>I4</td>
<td>Input signal I4</td>
</tr>
<tr>
<td>7</td>
<td>I5</td>
<td>Input signal I5</td>
</tr>
<tr>
<td>8</td>
<td>I6</td>
<td>Input signal I6</td>
</tr>
<tr>
<td>9</td>
<td>I7</td>
<td>Input signal I7</td>
</tr>
<tr>
<td>10</td>
<td>NO0</td>
<td>Normally-open contact of the output 0</td>
</tr>
<tr>
<td>11</td>
<td>NO1</td>
<td>Normally-open contact of the output 1</td>
</tr>
<tr>
<td>12</td>
<td>NO2</td>
<td>Normally-open contact of the output 2</td>
</tr>
</tbody>
</table>
### Terminal Signal Description

<table>
<thead>
<tr>
<th>Terminal</th>
<th>Signal</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>13</td>
<td>NO3</td>
<td>Normally-open contact of the output 3</td>
</tr>
<tr>
<td>14</td>
<td>R0...3</td>
<td>Output common for signals O0 to O3</td>
</tr>
<tr>
<td>15</td>
<td>NO4</td>
<td>Normally-open contact of the output 4</td>
</tr>
<tr>
<td>16</td>
<td>NO5</td>
<td>Normally-open contact of the output 5</td>
</tr>
<tr>
<td>17</td>
<td>NO6</td>
<td>Normally-open contact of the output 6</td>
</tr>
<tr>
<td>18</td>
<td>NO7</td>
<td>Normally-open contact of the output 7</td>
</tr>
<tr>
<td>19</td>
<td>R4...7</td>
<td>Output common for signals O4 to O7</td>
</tr>
<tr>
<td>20</td>
<td>L+</td>
<td>Process voltage +24 VDC</td>
</tr>
</tbody>
</table>

The internal power supply voltage for the module's circuitry is carried out via the I/O bus (provided by a bus module or a CPU). Thus, the current consumption from 24 VDC power supply at the terminals L+/UP and M/ZP of the CPU/bus module increases by 5 mA per DX571.

The external power supply connection is carried out via the L+ (+24 VDC) terminal. The negative pole of the external power supply is realized via the I/O bus. Therefore, the CPU/bus module and the DX571 must have a common power supply.

---

**WARNING!**
**Removal/Insertion under power**

The devices are not designed for removal or insertion under power. Because of unforeseeable consequences, it is not allowed to plug or unplug devices with the power being ON.

Make sure that all voltage sources (supply and process voltage) are switched off before you
- connect or disconnect any signal or terminal block
- remove, mount or replace a module.

Disconnecting any powered devices while energized in a hazardous location could result in an electric arc, which could create a flammable ignition resulting in fire or explosion.

Make sure that power is removed and that the area has been thoroughly checked to ensure that flammable materials are not present prior to proceeding.

The devices must not be opened when in operation. The same applies to the network interfaces.

---

**NOTICE!**
**Risk of damaging the PLC modules!**

Overvoltages and short circuits might damage the PLC modules.
- Make sure that all voltage sources (supply voltage and process supply voltage) are switched off before you begin with operations at the system.
- Never connect any voltages or signals to reserved terminals (marked with ---). Reserved terminals may carry internal voltages.

The module provides several diagnosis functions (see Diagnosis \(\$\) Chapter 1.5.3.6.1.13.6 "Diagnosis" on page 2449).
The digital inputs can be used as source inputs or as sink inputs.

**NOTICE!**  
Risk of malfunctions in the plant!  
A ground closure, e.g. caused by a damaged cable insulation, can bridge switches accidentally.  
Use sink inputs when possible or make sure that, in case of error, there will be no risks to persons or plant.

The following figures show the electrical connection of the inputs to the digital input/output module DX571:

**Fig. 108: Electrical connection of inputs - sink inputs**

![Diagram of sink input connection]

**Fig. 109: Electrical connection of inputs - source inputs**

![Diagram of source input connection]

The following figures show the electrical connection of the outputs to the module:
Fig. 110: Connection of 24 VDC actuators

Fig. 111: Connection of 24 VAC or 100-240 VAC actuators

The L+ connection of the DX571 and the 24 V supply of the CPU/bus module must be connected to the same 24 V power supply.
WARNING!
Risk of death by electric shock!
Hazardous voltages can be present at the terminals of the module.
Make sure that all voltage sources (supply voltage and process supply voltage) are switched off before you begin with operations at the system.

For screw-type terminals only:

WARNING!
For screw terminals only: Danger of death by electric shock!
The IP 20 protection degree is only provided if all terminal screws are tightened.
Tighten all screws of unused load terminals of relay outputs if voltages > 24 V are connected to the relay group.

NOTICE!
Risk of damaging the I/O module!
The outputs are not protected against short circuit and overload.
- Never short-circuit or overload the outputs.
- Never connect inductive loads without an external suppression against voltage peaks due to inductive kickback.
- Never connect voltages > 240 V. All outputs must be supplied from the same phase.
- Use an external 5 A fast protection fuse for the outputs.

The meaning of the LEDs is described in the Displays section Chapter 1.5.3.6.1.13.7 “State LEDs” on page 2450.

I/O configuration
The module itself does not store configuration data. It receives its parameterization data from the master device of the I/O bus (CPU or bus module) during power-up of the system.
Hence, replacing I/O modules is possible without any re-parameterization via software.

If the external power supply voltage via UP/ZP terminals fails, the I/O module loses its configuration data. The whole station has to be switched off and on again to re-configure the module.

Parameterization

The arrangement of the parameter data is performed with Automation Builder software.

The parameter data directly influences the functionality of modules.

For non-standard applications, it is necessary to adapt the parameters to your system configuration.

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>Internal Value</th>
<th>Internal Value, Type</th>
<th>Default</th>
<th>Min.</th>
<th>Max.</th>
<th>EDS Slot Index</th>
</tr>
</thead>
<tbody>
<tr>
<td>Module ID</td>
<td>Internal</td>
<td>6140 1)</td>
<td>WORD</td>
<td>6140</td>
<td>0</td>
<td>65535</td>
<td>xx01</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0x17FC</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ignore module</td>
<td>No</td>
<td>0</td>
<td>BYTE</td>
<td>No</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Yes</td>
<td>1</td>
<td></td>
<td>(0x00)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Parameter length</td>
<td>Internal</td>
<td>1</td>
<td>BYTE</td>
<td>0</td>
<td>0</td>
<td>255</td>
<td>xx02 2)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Check supply</td>
<td>Off</td>
<td>0</td>
<td>BYTE</td>
<td>On</td>
<td>0</td>
<td>0x01</td>
<td></td>
</tr>
<tr>
<td></td>
<td>On</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

1) with CS31 and addresses smaller than 70, the value is increased by 1

2) Value is hexadecimal: HighByte is slot (xx: 0...7), LowByte is index (1…n)

GSD file:

```
Ext_User_Prm_Data_Len = 0x04
Ext_User_Prm_Data_Const(0) = 0xFD, 0x17, 0x00, \n(0) = 0x01;
```
### Diagnosis

<table>
<thead>
<tr>
<th>E1...E4</th>
<th>d1</th>
<th>d2</th>
<th>d3</th>
<th>d4</th>
<th>Identifier 000...063</th>
<th>AC500-Display</th>
<th>Display in PS501 PLC Browser</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class</td>
<td>Comp</td>
<td>Dev</td>
<td>Mod</td>
<td>Ch</td>
<td>Err</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Byte 6</td>
<td>-</td>
<td>Byte 3</td>
<td>Byte 4</td>
<td>Byte 5</td>
<td>Byte 6 Bit 0...5</td>
<td>PNIO diagnosis block</td>
<td></td>
</tr>
</tbody>
</table>

#### Module error

<table>
<thead>
<tr>
<th>Class</th>
<th>Interface</th>
<th>Device</th>
<th>Module</th>
<th>Channel</th>
<th>Error Identifier</th>
<th>Error message</th>
<th>Remedy</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>14</td>
<td>1...10</td>
<td>31</td>
<td>31</td>
<td>19</td>
<td>Checksum error in the I/O module</td>
<td>Replace I/O module</td>
</tr>
<tr>
<td></td>
<td>11 / 12</td>
<td>ADR</td>
<td>1...10</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>14</td>
<td>1...10</td>
<td>31</td>
<td>31</td>
<td>43</td>
<td>Internal error in the module</td>
<td>Replace I/O module</td>
</tr>
<tr>
<td></td>
<td>11 / 12</td>
<td>ADR</td>
<td>1...10</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>14</td>
<td>1...10</td>
<td>31</td>
<td>31</td>
<td>9</td>
<td>Overflow diagnosis buffer</td>
<td>Restart</td>
</tr>
<tr>
<td></td>
<td>11 / 12</td>
<td>ADR</td>
<td>1...10</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>14</td>
<td>1...10</td>
<td>31</td>
<td>31</td>
<td>26</td>
<td>Parameter error</td>
<td>Check master</td>
</tr>
<tr>
<td></td>
<td>11 / 12</td>
<td>ADR</td>
<td>1...10</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>14</td>
<td>1...10</td>
<td>31</td>
<td>31</td>
<td>11</td>
<td>Process voltage too low</td>
<td>Check process voltage</td>
</tr>
<tr>
<td></td>
<td>11 / 12</td>
<td>ADR</td>
<td>1...10</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### Remarks:

1) In AC500 the following interface identifier applies:
   14 = I/O bus, 11 = COM1 (e.g. CS31 bus), 12 = COM2.
   The PNIO diagnosis block does not contain this identifier.

2) With "Device" the following allocation applies:
   31 = Module itself,
   1...10 = communication interface module 1...10,
   ADR = hardware address (e.g. of the DC551-CS31)

3) With "Module" the following allocation applies depending on the master:
   Module error: I/O bus or PNIO: 31 = Module itself; COM1/COM2: 1...10 = expansion 1...10
   Channel error: I/O bus or PNIO = Module type (2 = DO); COM1/COM2: 1...10 = expansion 1...10

4) In case of module errors, with channel "31 = module itself" is output.
State LEDs

<table>
<thead>
<tr>
<th>LED State</th>
<th>Color</th>
<th>LED = OFF</th>
<th>LED = ON</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inputs I0...I7</td>
<td>Digital input</td>
<td>Yellow</td>
<td>Input is OFF</td>
</tr>
<tr>
<td>Outputs NO0...NO7</td>
<td>Digital output</td>
<td>Yellow</td>
<td>Output is OFF</td>
</tr>
</tbody>
</table>

*In the undefined signal range, the state LED for the inputs can be ON although the input state detected by the module is OFF.*

Technical data

The System Data of AC500-eCo apply

Only additional details are therefore documented below.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Process supply voltage L+</td>
<td>Terminal 20 for L+ (+24 VDC). The minus pole is provided by the I/O-Bus.</td>
</tr>
<tr>
<td>Connections</td>
<td></td>
</tr>
<tr>
<td>Rated value</td>
<td>24 VDC</td>
</tr>
<tr>
<td>Current consumption via L+</td>
<td>50 mA</td>
</tr>
<tr>
<td>Inrush current (at power-up)</td>
<td>0.0035 A's</td>
</tr>
<tr>
<td>Max. ripple</td>
<td>5 %</td>
</tr>
<tr>
<td>Protection against reversed voltage</td>
<td>Yes</td>
</tr>
<tr>
<td>Rated protection fuse for L+</td>
<td>Recommended; the outputs must be protected by a 3 A fast fuse</td>
</tr>
<tr>
<td>Current consumption from 24 VDC power supply at the L+/UP and M/ZP terminals of the CPU/bus module</td>
<td>Ca. 5 mA</td>
</tr>
<tr>
<td>Galvanic isolation</td>
<td>Yes, between the input group and the output group and the rest of the module</td>
</tr>
<tr>
<td>Isolated groups</td>
<td>3 groups (1 group for 8 input channels, 2 groups for 8 output channels)</td>
</tr>
<tr>
<td>Surge-voltage (max.)</td>
<td>35 VDC for 0.5 s</td>
</tr>
<tr>
<td>Max. power dissipation within the module</td>
<td>2.3 W</td>
</tr>
<tr>
<td>Weight</td>
<td>Ca. 150 g</td>
</tr>
<tr>
<td>Mounting position</td>
<td>Horizontal or vertical</td>
</tr>
<tr>
<td>Cooling</td>
<td>The natural convection cooling must not be hindered by cable ducts or other parts in the switch-gear cabinet.</td>
</tr>
</tbody>
</table>
No effects of multiple overloads

No effects of multiple overloads on isolated multi-channel modules occur, as every channel is protected individually by an external fuse.

Technical data of the digital inputs

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of channels per module</td>
<td>8</td>
</tr>
<tr>
<td>Distribution of the channels into groups</td>
<td>1 group for 8 channels</td>
</tr>
<tr>
<td>Connections of the channels I0 to I7</td>
<td>Terminals 2 to 9</td>
</tr>
<tr>
<td>Reference potential for the channels I0 to I7</td>
<td>Terminal 1</td>
</tr>
<tr>
<td>Indication of the input signals</td>
<td>1 yellow LED per channel; the LED is ON when the input signal is high (signal 1)</td>
</tr>
<tr>
<td>Monitoring point of input indicator</td>
<td>LED is part of the input circuitry</td>
</tr>
<tr>
<td>Input type according to EN 61131-2</td>
<td>Type 1 source</td>
</tr>
<tr>
<td>Input signal range</td>
<td>-24 VDC</td>
</tr>
<tr>
<td>Signal 0</td>
<td>+24 VDC</td>
</tr>
<tr>
<td>Undefined signal</td>
<td>-5 V...+3 V</td>
</tr>
<tr>
<td>Signal 1</td>
<td>-3 V...+5 V</td>
</tr>
<tr>
<td>Input signal range</td>
<td>+5 V...+15 V</td>
</tr>
<tr>
<td>Signal 1</td>
<td>-15 V...+5 V</td>
</tr>
<tr>
<td>Input signal range</td>
<td>+15 V...+30 V</td>
</tr>
<tr>
<td>Input current per channel</td>
<td>-30 V...-15 V</td>
</tr>
<tr>
<td>Input voltage 24 V</td>
<td>-30 V...+30 V</td>
</tr>
<tr>
<td>Input voltage 5 V</td>
<td>15 V...27 V</td>
</tr>
<tr>
<td>Input voltage 14 V</td>
<td>24 VAC 50/60 Hz</td>
</tr>
<tr>
<td>Input voltage 15 V</td>
<td>-30 V...+30 V</td>
</tr>
<tr>
<td>Input voltage 17 V</td>
<td>14 VAC 50/60 Hz</td>
</tr>
<tr>
<td>Input voltage 27 V</td>
<td>+15 V...+30 V</td>
</tr>
<tr>
<td>Input voltage 15 V</td>
<td>-15 V...+5 V</td>
</tr>
<tr>
<td>Input voltage 24 V</td>
<td>14 VAC 50/60 Hz</td>
</tr>
<tr>
<td>Input voltage 5 V</td>
<td>24 VAC 50/60 Hz</td>
</tr>
<tr>
<td>Input voltage 14 V</td>
<td>-30 V...+30 V</td>
</tr>
<tr>
<td>Input voltage 15 V</td>
<td>-15 V...+5 V</td>
</tr>
<tr>
<td>Input voltage 27 V</td>
<td>14 VAC 50/60 Hz</td>
</tr>
<tr>
<td>Input voltage 30 V</td>
<td>24 VAC 50/60 Hz</td>
</tr>
<tr>
<td>Input delay (0-&gt;1 or 1-&gt;0)</td>
<td>Typ. 8 ms</td>
</tr>
<tr>
<td>Input data length</td>
<td>1 byte</td>
</tr>
<tr>
<td>Max. cable length</td>
<td>Shielded 500 m</td>
</tr>
<tr>
<td></td>
<td>Unshielded 300 m</td>
</tr>
</tbody>
</table>

1) When inputs are used with 24 VAC, external surge limiting filters are required.

Technical data of the digital outputs

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of channels per module</td>
<td>8 normally-open relay outputs</td>
</tr>
<tr>
<td>Distribution of the channels into groups</td>
<td>2 (4 channels per group)</td>
</tr>
<tr>
<td>Connection of the channels O0 to O3</td>
<td>Terminals 10 to 13</td>
</tr>
<tr>
<td>Connection of the channels O4 to O7</td>
<td>Terminals 15 to 18</td>
</tr>
<tr>
<td>Parameter</td>
<td>Value</td>
</tr>
<tr>
<td>----------------------------------------------------</td>
<td>----------------------------------------------------------------------</td>
</tr>
<tr>
<td>Reference potential for the channels O0 to O3</td>
<td>Terminal 14 (signal name R0..3)</td>
</tr>
<tr>
<td>Reference potential for the channels O4 to O7</td>
<td>Terminal 19 (signal name R4..7)</td>
</tr>
<tr>
<td>Relay coil power supply</td>
<td>Terminal 20 (positive pole of the process supply voltage, signal name L+) The negative pole is provided by the I/O bus</td>
</tr>
<tr>
<td>Indication of the output signals</td>
<td>1 yellow LED per channel; the LED is on when the output signal is high (signal 1) and the module is powered through the I/O bus</td>
</tr>
<tr>
<td>Monitoring point of output indicator</td>
<td>Controlled together with relay</td>
</tr>
<tr>
<td>Way of operation</td>
<td>Non-latching type</td>
</tr>
<tr>
<td>Relay output voltage</td>
<td></td>
</tr>
<tr>
<td>Rated value</td>
<td>24 VDC / 24 VAC or 120/240 VAC</td>
</tr>
<tr>
<td>Output delay</td>
<td></td>
</tr>
<tr>
<td>Switching 0 to 1 (max.)</td>
<td>Typ. 10 ms</td>
</tr>
<tr>
<td>Switching 1 to 0 (max.)</td>
<td>Typ. 10 ms</td>
</tr>
<tr>
<td>Output data length</td>
<td>1 byte</td>
</tr>
<tr>
<td>Output current</td>
<td></td>
</tr>
<tr>
<td>Rated current per channel (max.)</td>
<td>2.0 A (24 VDC / 24 VAC / 48 VAC / 120 VAC / 240 VAC, only resistive loads)</td>
</tr>
<tr>
<td></td>
<td>2.0 A (24 VAC / 48 VAC / 120 VAC, only pilot duty)</td>
</tr>
<tr>
<td></td>
<td>1.5 A (240 VAC, only pilot duty)</td>
</tr>
<tr>
<td>Rated current per group (max.)</td>
<td>8 A</td>
</tr>
<tr>
<td>Lamp load (max.)</td>
<td>200 W (230 VAC), 30 W (24 VDC)</td>
</tr>
<tr>
<td>Spark suppression with inductive AC loads</td>
<td>Must be performed externally according to driven load specification</td>
</tr>
<tr>
<td>Switching Frequencies</td>
<td></td>
</tr>
<tr>
<td>With resistive loads</td>
<td>Max. 1 Hz</td>
</tr>
<tr>
<td>With inductive loads</td>
<td>On Request</td>
</tr>
<tr>
<td>With lamp loads</td>
<td>Max. 1 Hz</td>
</tr>
<tr>
<td>Output type</td>
<td>Non-protected</td>
</tr>
<tr>
<td>Protection type</td>
<td>External fuse on each channel</td>
</tr>
<tr>
<td>Rated protection fuse (for each channel)</td>
<td>5 A fast</td>
</tr>
<tr>
<td>Short-circuit-proof / Overload-proof</td>
<td>No, should be provided by an external fuse or circuit breaker</td>
</tr>
<tr>
<td>Overload message</td>
<td>No</td>
</tr>
<tr>
<td>Output current limitation</td>
<td>No</td>
</tr>
<tr>
<td>Connection of 2 outputs in parallel</td>
<td>Not possible</td>
</tr>
<tr>
<td>Life time of relay contacts (cycles)</td>
<td>100,000 at rated load</td>
</tr>
<tr>
<td>Max. cable length</td>
<td></td>
</tr>
<tr>
<td>Shielded</td>
<td>500 m</td>
</tr>
<tr>
<td>Unshielded</td>
<td>150 m</td>
</tr>
</tbody>
</table>
### Ordering data

<table>
<thead>
<tr>
<th>Part no.</th>
<th>Description</th>
<th>Product life cycle phase *)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1TNE 968 902</td>
<td>DX571, digital input/output module, 8 DI 24 VDC / 24 VAC, 8 DO, relay output</td>
<td>Active</td>
</tr>
<tr>
<td>1TNE 968 901</td>
<td>Terminal block TA563-9, 9 pins, screw front, cable side, 6 pieces per unit</td>
<td>Active</td>
</tr>
<tr>
<td>R3101</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1TNE 968 901</td>
<td>Terminal block TA563-11, 11 pins, screw front, cable side, 6 pieces per unit</td>
<td>Active</td>
</tr>
<tr>
<td>R3102</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1TNE 968 901</td>
<td>Terminal block TA564-9, 9 pins, screw front, cable front, 6 pieces per unit</td>
<td>Active</td>
</tr>
<tr>
<td>R3103</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1TNE 968 901</td>
<td>Terminal block TA564-11, 11 pins, screw front, cable front, 6 pieces per unit</td>
<td>Active</td>
</tr>
<tr>
<td>R3104</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1TNE 968 901</td>
<td>Terminal block TA565-9, 9 pins, spring front, cable front, 6 pieces per unit</td>
<td>Active</td>
</tr>
<tr>
<td>R3105</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1TNE 968 901</td>
<td>Terminal block TA565-11, 11 pins, spring front, cable front, 6 pieces per unit</td>
<td>Active</td>
</tr>
<tr>
<td>R3106</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

*) For planning and commissioning of new installations use modules in Active status only.

### S500

#### DC522 - Digital input/output module

- 16 configurable digital inputs/outputs
- Module-wise electrically isolated
- Fast counter
- XC version for use in extreme ambient conditions available
I/O bus
Allocation between terminal number and signal name
Sensor power supply 24 VDC / 0.5 A
16 yellow LEDs to display the signal states at the digital inputs/outputs (C0 - C15)
1 green LED to display the state of the process supply voltage UP
4 red LEDs to display errors
Label
Terminal unit
DIN rail
Sign for XC version

Intended purpose
The device can be used as a decentralized I/O extension module for S500 Communication Interface Modules (e.g. CI592-CS31, CI501-PNIO, CI541-DP, CI581-CN) or as a centralized extension module for AC500 CPUs (PM5xx).
Digital configurable input/output unit.

- 2 sensor supply voltages 24 VDC, 0.5 A, with short-circuit and overload protection
- 16 digital configurable inputs/outputs 24 VDC (C0 to C15) in 1 group (2.0...2.7 and 4.0...4.7), each of which can be used
  - as an input,
  - as a transistor output with short-circuit and overload protection, 0.5 A rated current or
  - as a re-readable output (combined input/output) with the technical data of the digital inputs and outputs.
- Optional with fast counter

The configuration is performed by software. The modules are supplied with a process supply voltage of 24 VDC.

All available inputs/outputs are electrically isolated from all other circuitry of the module. There is no potential separation between the channels within the same group.

For use in extreme ambient conditions (e.g. wider temperature and humidity range), a special XC version of the device is available.

### Functionality

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fast counter</td>
<td>Integrated, many configurable operating modes (only with AC500)</td>
</tr>
<tr>
<td>LED displays</td>
<td>For signal states, errors and supply voltage</td>
</tr>
<tr>
<td>Internal power supply</td>
<td>Through the expansion bus interface (I/O bus)</td>
</tr>
<tr>
<td>External power supply</td>
<td>Via the terminals ZP and UP (process voltage 24 VDC)</td>
</tr>
<tr>
<td>Required terminal unit</td>
<td>TU515 or TU516 % Chapter 1.5.3.5.2 “TU515, TU516, TU541 and TU542 for I/O modules” on page 2311</td>
</tr>
<tr>
<td>Effect of incorrect input terminal connection</td>
<td>Wrong or no signal detected, no damage up to 35 V</td>
</tr>
</tbody>
</table>

The device is plugged on a terminal unit % Chapter 1.5.3.5.2 “TU515, TU516, TU541 and TU542 for I/O modules” on page 2311. Position the module properly and press until it locks in place. The terminal unit is either mounted on a DIN rail or to the wall using 2 screws plus the additional accessory for wall mounting (TA526 % Chapter 1.5.3.8.1.4 “TA526 - Wall mounting accessory” on page 3035).

### Electrical connection

The electrical connection of the I/O channels is carried out using the 40 terminals of the I/O terminal unit. I/O modules can be replaced without re-wiring the terminal units.

The terminals 1.8 to 4.8 and 1.9 to 4.9 are electrically interconnected within the I/O terminal unit and always have the same assignment, irrespective of the inserted module:

Terminals 1.8 to 4.8: process voltage UP = +24 VDC
Terminals 1.9 to 4.9: process voltage ZP = 0 VDC
1. I/O bus
2. 4.0 - 4.7: Connected with UP (switch) -> Input; Connected with ZP (load) -> Output
3. Switch-gear cabinet earth

The assignment of the other terminals:

<table>
<thead>
<tr>
<th>Terminals</th>
<th>Signal</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0 to 1.3</td>
<td>+24 V</td>
<td>4 x sensor power supply sources (loadable with 0.5 A in total)</td>
</tr>
<tr>
<td>1.4 to 1.7</td>
<td>0 V</td>
<td>0 V (reference potential)</td>
</tr>
<tr>
<td>2.0 to 2.7</td>
<td>C0 to C7</td>
<td>8 digital inputs/outputs</td>
</tr>
<tr>
<td>3.0 to 3.3</td>
<td>+24 V</td>
<td>4 x sensor power supply sources (loadable with 0.5 A in total)</td>
</tr>
<tr>
<td>3.4 to 3.7</td>
<td>0 V</td>
<td>0 V (reference potential)</td>
</tr>
<tr>
<td>4.0 to 4.7</td>
<td>C8 to C15</td>
<td>8 digital inputs/outputs</td>
</tr>
</tbody>
</table>
CAUTION!
The process supply voltage must be included in the earthing concept (e.g. earthing of the negative pole).

The internal power supply voltage for the module's circuitry is carried out via the I/O bus (provided by a bus module or a CPU). Thus, the current consumption from 24 VDC power supply at the terminals L+/UP and M/ZP of the CPU/bus module increases by 2 mA per DC522.

The external power supply connection is carried out via the UP (+24 VDC) and the ZP (0 VDC) terminals.

WARNING!
Removal/Insertion under power
The devices are not designed for removal or insertion under power. Because of unforeseeable consequences, it is not allowed to plug or unplug devices with the power being ON.

Make sure that all voltage sources (supply and process voltage) are switched off before you
- connect or disconnect any signal or terminal block
- remove, mount or replace a module.

Disconnecting any powered devices while energized in a hazardous location could result in an electric arc, which could create a flammable ignition resulting in fire or explosion.

Make sure that power is removed and that the area has been thoroughly checked to ensure that flammable materials are not present prior to proceeding.

The devices must not be opened when in operation. The same applies to the network interfaces.

NOTICE!
Risk of damaging the PLC modules!
- Overvoltages and short circuits might damage the PLC modules.
  - Make sure that all voltage sources (supply voltage and process supply voltage) are switched off before you begin with operations at the system.
  - Never connect any voltages or signals to reserved terminals (marked with ---). Reserved terminals may carry internal voltages.

NOTICE!
Risk of influences to the connected sensors!
- Some sensors may be influenced by the deactivated module outputs of DC522.
  - Connect a 470 Ω / 1 W resistor in series to inputs C8/C9 if they are used as fast counter inputs to avoid any influences.

The modules provide several diagnosis functions § Chapter 1.5.3.6.1.2.1.7 “Diagnosis” on page 2459.
Internal data exchange

<table>
<thead>
<tr>
<th></th>
<th>Without the Fast Counter</th>
<th>With the Fast Counter (only with AC500)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Digital inputs (bytes)</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>Digital outputs (bytes)</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>Counter input data (words)</td>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td>Counter output data (words)</td>
<td>0</td>
<td>8</td>
</tr>
</tbody>
</table>

I/O Configuration

The module itself does not store configuration data. It receives its parameterization data from the master device of the I/O bus (CPU or bus module) during power-up of the system.

Hence, replacing I/O modules is possible without any re-parameterization via software.

*If the external power supply voltage via UP/ZP terminals fails, the I/O module loses its configuration data. The whole station has to be switched off and on again to re-configure the module.*

Parameterization

<table>
<thead>
<tr>
<th>Firmware version</th>
<th>Configuration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Firmware version &gt; V2.0.0</td>
<td>The arrangement of the parameter data is performed by Control Builder Plus/ Automation Builder software.</td>
</tr>
</tbody>
</table>

The parameter data directly influences the functionality of modules.

For non-standard applications, it is necessary to adapt the parameters to your system configuration.

Module: Module slot address: Y = 1...10

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>Internal value</th>
<th>Internal value, type</th>
<th>Default</th>
<th>Min.</th>
<th>Max.</th>
<th>EDS Slot/ Index</th>
</tr>
</thead>
<tbody>
<tr>
<td>Module ID</td>
<td>Internal</td>
<td>1220 1)</td>
<td>Word</td>
<td>1220</td>
<td>0</td>
<td>65535</td>
<td>0x0Y01</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0x04C4</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ignore module</td>
<td>No</td>
<td>0</td>
<td>Byte</td>
<td>No</td>
<td>0</td>
<td>1</td>
<td>Not for FBP</td>
</tr>
<tr>
<td></td>
<td>Yes</td>
<td>1</td>
<td></td>
<td>0x00</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Parameter length</td>
<td>Internal</td>
<td>7</td>
<td>Byte</td>
<td>7-CPU</td>
<td>0</td>
<td>255</td>
<td>0x0Y02</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>6-FBP</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Check supply</td>
<td>Off</td>
<td>0</td>
<td>Byte</td>
<td>On</td>
<td>0</td>
<td>1</td>
<td>0x0Y03</td>
</tr>
<tr>
<td></td>
<td>On</td>
<td>1</td>
<td></td>
<td>0x01</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Name</td>
<td>Value</td>
<td>Internal value</td>
<td>Internal value, type</td>
<td>Default</td>
<td>Min.</td>
<td>Max.</td>
<td>EDS Slot/Index</td>
</tr>
<tr>
<td>-------------------------------------------</td>
<td>------------</td>
<td>----------------</td>
<td>----------------------</td>
<td>-------------</td>
<td>------</td>
<td>-------</td>
<td>----------------</td>
</tr>
<tr>
<td>Input delay</td>
<td>0.1 ms</td>
<td>0</td>
<td>Byte</td>
<td>8 ms</td>
<td>0</td>
<td>3</td>
<td>0x0Y04</td>
</tr>
<tr>
<td></td>
<td>1 ms</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>8 ms</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>32 ms</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Fast counter 4)</td>
<td>0</td>
<td>0</td>
<td>Byte</td>
<td>Mode 0</td>
<td>0</td>
<td></td>
<td>Not for FBP</td>
</tr>
<tr>
<td></td>
<td>: 10</td>
<td>:</td>
<td></td>
<td>0x00</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Short-circuit detection of output or sensor supply</td>
<td>Off</td>
<td>0</td>
<td>Byte</td>
<td>On</td>
<td>0</td>
<td>1</td>
<td>0x0Y05</td>
</tr>
<tr>
<td></td>
<td>On</td>
<td>1</td>
<td></td>
<td>0x01</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Behaviour of outputs at communication errors</td>
<td>Off</td>
<td>0</td>
<td>Byte</td>
<td>Off</td>
<td>0</td>
<td>2</td>
<td>0x0Y06</td>
</tr>
<tr>
<td></td>
<td>Last value</td>
<td>1+(n*5)</td>
<td></td>
<td>0x00</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Substitute value</td>
<td>2+(n*5), n ≤ 2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Substitute value at outputs</td>
<td>0...</td>
<td>0</td>
<td>Word</td>
<td>0</td>
<td>0</td>
<td>65535</td>
<td>0x0Y07</td>
</tr>
<tr>
<td>Bit 15 = Output 15</td>
<td>65535</td>
<td>0</td>
<td></td>
<td>0x0000</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bit 0 = Output 0</td>
<td>0...</td>
<td>0x0000</td>
<td></td>
<td>65555</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0xffff</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Remarks:

1) With CS31 and addresses smaller than 70 and FBP, the value is increased by 1
2) Not with FBP
3) For a description of the counter operating modes, please refer to the Fast Counter section "Chapter 1.5.3.6.1.2.9 “Fast counter” on page 2543"
4) With FBP or CS31 without the parameter Fast counter

GSD file:

```
Ext_User_Prm_Data_Len = 9
Ext_User_Prm_Data_Const(0) = 0x04, 0xc5, 0x06, \n0x01, 0x02, 0x01, 0x00, 0x00, 0x00;
```

**Diagnosis**

In case of overload or short-circuit, the outputs switch off automatically and try to switch on again cyclically. Therefore an acknowledgement of the outputs is not necessary. The LED error message, however, is stored.
### Remarks:

1) In AC500 the following interface identifier applies:
   14 = I/O-Bus, 11 = COM1 (e.g. CS31 bus), 12 = COM2.
   The FBP diagnosis block does not contain this identifier.

2) With "Device" the following allocation applies:
   31 = module itself,
   1...10 = communication interface module 1...10,
   ADR = hardware address (e.g. of the DC551)
With "Module" the following allocation applies depending on the master:

Module error: I/O bus or FBP: 31 = module itself; COM1/COM2: 1...10 = expansion 1...10
Channel error: I/O bus or FBP = module type (4 = DC); COM1/COM2: 1...10 = expansion 1...10

In case of module errors, with channel "31 = module itself" is output.

State LEDs

During the power ON procedure, the module initializes automatically. All LEDs (except the channel LEDs) are ON during this time.

<table>
<thead>
<tr>
<th>LED</th>
<th>State</th>
<th>Color</th>
<th>LED = OFF</th>
<th>LED = ON</th>
<th>LED flashes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inputs/outputs C0...C15</td>
<td>Digital input or digital output</td>
<td>Yellow</td>
<td>Input/output = OFF</td>
<td>Input/output = ON ¹)</td>
<td>--</td>
</tr>
<tr>
<td>UP</td>
<td>Process supply voltage 24 VDC via terminal</td>
<td>Green</td>
<td>Process supply voltage is missing</td>
<td>Process supply voltage OK</td>
<td>--</td>
</tr>
<tr>
<td>CH-ERR1</td>
<td>Channel Error, error messages in groups (digital inputs/outputs combined into the groups 1, 2, 3, 4)</td>
<td>Red</td>
<td>No error or process supply voltage is missing</td>
<td>Severe error within the corresponding group</td>
<td>Error on one channel of the corresponding group (e.g. short circuit at an output)</td>
</tr>
<tr>
<td>CH-ERR2</td>
<td></td>
<td>Red</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CH-ERR3</td>
<td></td>
<td>Red</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CH-ERR4</td>
<td></td>
<td>Red</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CH-ERR ²)</td>
<td>Module error</td>
<td>Red</td>
<td>--</td>
<td>Internal error</td>
<td>--</td>
</tr>
</tbody>
</table>

¹) Indication LED is ON even if an input signal is applied to the channel and the supply voltage is off. In this case the module is not operating and does not generate an input signal.

²) All of the LEDs CH-ERR1 to CH-ERR4 light up together

Technical data

The system data of AC500 and S500  "System data AC500" on page 3058 are valid for standard version.

The system data of AC500-XC  "System data AC500-XC" on page 3099 are valid for the XC version.

Only additional details are therefore documented below.

The technical data are also valid for the XC version.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Process supply voltage UP</td>
<td></td>
</tr>
<tr>
<td>Connections</td>
<td>Terminals 1.8, 2.8, 3.8 and 4.8 for +24 V (UP) as well as 1.9, 2.9, 3.9 and 4.9 for 0 V (ZP)</td>
</tr>
<tr>
<td>Parameter</td>
<td>Value</td>
</tr>
<tr>
<td>------------------------------------------------</td>
<td>------------------------------------------------------------</td>
</tr>
<tr>
<td>Rated value</td>
<td>24 VDC</td>
</tr>
<tr>
<td>Max. ripple</td>
<td>5 %</td>
</tr>
<tr>
<td>Protection against reversed voltage</td>
<td>Yes</td>
</tr>
<tr>
<td>Rated protection fuse on UP</td>
<td>10 A fast</td>
</tr>
<tr>
<td>Galvanic isolation</td>
<td>Yes, per module</td>
</tr>
<tr>
<td>Current consumption</td>
<td></td>
</tr>
<tr>
<td>From 24 VDC power supply at the L+/UP and M/ZP terminals of the CPU/bus module</td>
<td>Ca. 2 mA</td>
</tr>
<tr>
<td>From UP at normal operation / with outputs</td>
<td>0.15 A + max. 0.5 A per output</td>
</tr>
<tr>
<td>Inrush current from UP (at power up)</td>
<td>0.005 A·s</td>
</tr>
<tr>
<td>Max. power dissipation within the module</td>
<td>6 W (outputs unloaded)</td>
</tr>
<tr>
<td>Sensor power supply</td>
<td></td>
</tr>
<tr>
<td>Connections</td>
<td>Terminals 1.0...1.3 = +24 V, 1.4...1.7 = 0 V</td>
</tr>
<tr>
<td></td>
<td>Terminals 3.0...3.3 = +24 V, 3.4...3.7 = 0 V</td>
</tr>
<tr>
<td>Voltage</td>
<td>24 VDC with short-circuit and overload protection</td>
</tr>
<tr>
<td>Loadability</td>
<td>Terminals 1.0...1.3, in total max. 0.5 A</td>
</tr>
<tr>
<td></td>
<td>Terminals 3.0...3.3, in total max. 0.5 A</td>
</tr>
<tr>
<td>Weight (without terminal unit)</td>
<td>Ca. 125 g</td>
</tr>
<tr>
<td>Mounting position</td>
<td>Horizontal</td>
</tr>
<tr>
<td></td>
<td>Or vertical with deration (output load reduced to 50 % at 40 °C per group)</td>
</tr>
<tr>
<td>Cooling</td>
<td>The natural convection cooling must not be hindered by cable ducts or other parts in the switch-gear cabinet.</td>
</tr>
</tbody>
</table>

**NOTICE!**

**Attention:**

All I/O channels (digital and analog) are protected against reverse polarity, reverse supply, short circuit and continuous overvoltage up to 30 VDC.

**Multiple overloads**

No effects of multiple overloads on isolated multi-channel modules occur, as every channel is protected individually by an internal smart high-side switch.

**Technical data of the configurable digital inputs/outputs**

Each of the configurable I/O channels is defined as input or output by the user program. This is done by interrogating or allocating the corresponding channel.
## Parameter Values

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of channels per module</td>
<td>16 inputs/outputs (with transistors)</td>
</tr>
<tr>
<td>Distribution of the channels into groups</td>
<td>1 group of 16 channels</td>
</tr>
<tr>
<td>If the channels are used as inputs</td>
<td></td>
</tr>
<tr>
<td>Channels C0...C7</td>
<td>Terminals 2.0...2.7</td>
</tr>
<tr>
<td>Channels C8...C15</td>
<td>Terminals 4.0...4.7</td>
</tr>
<tr>
<td>If the channels are used as outputs</td>
<td></td>
</tr>
<tr>
<td>Channels C0...C7</td>
<td>Terminals 2.0...2.7</td>
</tr>
<tr>
<td>Channels C8...C15</td>
<td>Terminals 4.0...4.7</td>
</tr>
<tr>
<td>Indication of the input/output signals</td>
<td>1 yellow LED per channel, the LED is ON when the input/output signal is high (signal 1)</td>
</tr>
<tr>
<td>Monitoring point of input/output indicator</td>
<td>LED is part of the input circuitry</td>
</tr>
<tr>
<td>Galvanic isolation</td>
<td>From the rest of the module</td>
</tr>
</tbody>
</table>

## Technical data of the digital inputs/outputs if used as inputs

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of channels per module</td>
<td>Max. 16 digital inputs</td>
</tr>
<tr>
<td>Reference potential for all inputs</td>
<td>Terminals 1.9, 2.9, 3.9 and 4.9 (negative pole of the process supply voltage, signal name ZP)</td>
</tr>
<tr>
<td>Galvanic isolation</td>
<td>From the rest of the module</td>
</tr>
<tr>
<td>Indication of the input signals</td>
<td>1 yellow LED per channel, the LED is ON when the input signal is high (signal 1)</td>
</tr>
<tr>
<td>Monitoring point of input/output indicator</td>
<td>LED is part of the input circuitry</td>
</tr>
<tr>
<td>Input type acc. to EN 61131-2</td>
<td>Type 1</td>
</tr>
<tr>
<td>Input delay (0-&gt;1 or 1-&gt;0)</td>
<td>Typ. 8 ms, configurable from 0.1 to 32 ms</td>
</tr>
<tr>
<td>Input signal voltage</td>
<td>24 VDC</td>
</tr>
<tr>
<td>Signal 0</td>
<td>-3 V...+5 V *)</td>
</tr>
<tr>
<td>Undefined signal</td>
<td>&gt; +5 V...&lt; +15 V</td>
</tr>
<tr>
<td>Signal 1</td>
<td>+15 V...+30 V</td>
</tr>
<tr>
<td>Ripple with signal 0</td>
<td>Within -3 V...+5 V *)</td>
</tr>
<tr>
<td>Ripple with signal 1</td>
<td>Within +15 V...+30 V</td>
</tr>
<tr>
<td>Input current per channel</td>
<td></td>
</tr>
<tr>
<td>Input voltage +24 V</td>
<td>Typ. 5 mA</td>
</tr>
<tr>
<td>Input voltage +5 V</td>
<td>&gt; 1 mA</td>
</tr>
<tr>
<td>Input voltage +15 V</td>
<td>&gt; 5 mA</td>
</tr>
<tr>
<td>Input voltage +30 V</td>
<td>&lt; 8 mA</td>
</tr>
<tr>
<td>Max. cable length</td>
<td></td>
</tr>
<tr>
<td>Shielded</td>
<td>1000 m</td>
</tr>
<tr>
<td>Unshielded</td>
<td>600 m</td>
</tr>
</tbody>
</table>
*) Due to the direct connection to the output, the demagnetizing varistor is also effective at the input (see figure) above. This is why the difference between UPx and the input signal may not exceed the clamp voltage of the varistor. The varistor limits the voltage to approx. 36 V. Consequently, the input voltage must range from -12 V to +30 V when UPx = 24 V and from -6 V to +30 V when UPx = 30 V.

Technical data of the digital inputs/outputs if used as outputs

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of channels per module</td>
<td>Max. 16 transistor outputs</td>
</tr>
<tr>
<td>Reference potential for all outputs</td>
<td>Terminals 1.9, 2.9, 3.9 and 4.9 (negative pole of the process supply voltage, signal name ZP)</td>
</tr>
<tr>
<td>Common power supply voltage</td>
<td>For all outputs: terminals 1.8, 2.8, 3.8 and 4.8 (positive pole of the process supply voltage, signal name UP)</td>
</tr>
<tr>
<td>Output voltage for signal 1</td>
<td>UP (-0.8 V)</td>
</tr>
<tr>
<td>Output delay (0-&gt;1 or 1-&gt;0)</td>
<td>On request</td>
</tr>
<tr>
<td>Output current</td>
<td></td>
</tr>
<tr>
<td>Rated value, per channel</td>
<td>500 mA at UP = 24 V</td>
</tr>
<tr>
<td>Maximum value (all channels together)</td>
<td>8 A</td>
</tr>
<tr>
<td>Leakage current with signal 0</td>
<td>&lt; 0.5 mA</td>
</tr>
<tr>
<td>Rated protection fuse on UP</td>
<td>10 A fast</td>
</tr>
<tr>
<td>Demagnetization when inductive loads are switched off</td>
<td>With varistors integrated in the module (see figure below)</td>
</tr>
<tr>
<td>Switching frequency</td>
<td></td>
</tr>
<tr>
<td>With resistive load</td>
<td>On request</td>
</tr>
<tr>
<td>With inductive loads</td>
<td>Max. 0.5 Hz</td>
</tr>
<tr>
<td>With lamp loads</td>
<td>Max. 11 Hz with max. 5 W</td>
</tr>
<tr>
<td>Short-circuit-proof / overload-proof</td>
<td>Yes</td>
</tr>
<tr>
<td>Overload message (I &gt; 0.7 A)</td>
<td>Yes, after ca. 100 ms</td>
</tr>
<tr>
<td>Output current limitation</td>
<td>Yes, automatic reactivation after short circuit/overload</td>
</tr>
<tr>
<td>Resistance to feedback against 24 V signals</td>
<td>Yes</td>
</tr>
<tr>
<td>Max. cable length</td>
<td></td>
</tr>
<tr>
<td>Shielded</td>
<td>1000 m</td>
</tr>
<tr>
<td>Unshielded</td>
<td>600 m</td>
</tr>
</tbody>
</table>

The following drawing shows the circuitry of a digital input/output with the varistors for demagnetization when inductive loads are switched off.
Technical data of the fast counter

- The fast counter of the module does not work if the module is connected to a
  - FBP interface module
  - CS31 bus module
  - CANopen bus module

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Used inputs</td>
<td>C8 / C9</td>
</tr>
<tr>
<td>Used outputs</td>
<td>C10</td>
</tr>
<tr>
<td>Counting frequency</td>
<td>Max. 50 kHz</td>
</tr>
</tbody>
</table>
| Detailed description    | See Fast Counter % Chapter 1.5.5.1.10 “Fast
counters” on page 3194                      |
| Operating modes         | See Operating modes                        |

Ordering data

<table>
<thead>
<tr>
<th>Part no.</th>
<th>Description</th>
<th>Product life cycle phase *)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1SAP 240 600 R0001</td>
<td>DC522, digital input/output module, 16 DC, 24 VDC / 0.5 A, 2-wires</td>
<td>Active</td>
</tr>
<tr>
<td>1SAP 440 600 R0001</td>
<td>DC522-XC, digital input/output module, 16 DC, 24 VDC / 0.5 A, 2-wires, XC version</td>
<td>Active</td>
</tr>
</tbody>
</table>

*) For planning and commissioning of new installations use modules in Active status only.
DC523 - Digital input/output module

- 24 configurable digital inputs/outputs
- Module-wise electrically isolated
- Fast counter
- XC version for use in extreme ambient conditions available

Intended purpose

The device can be used as a decentralized I/O extension module for S500 Communication Interface Modules (e.g. CI592-CS31, CI501-PNIO, CI541-DP, CI581-CN) or as a centralized extension module for AC500 CPUs (PM5xx).
Digital configurable input/output unit.

- 1 sensor supply voltage 24 VDC, 0.5 A, with short circuit and overload protection
- 24 digital configurable inputs/outputs 24 VDC (C0 to C23) in 1 group (2.0...2.7, 3.0...3.7 and 4.0...4.7), of which each can be used
  - as an input,
  - as a transistor output with short circuit and overload protection, 0.5 A rated current or
  - as a re-readable output (combined input/output) with the technical data of the digital inputs and outputs.
- Optional with fast counter

The configuration is performed by software. The modules are supplied with a process supply voltage of 24 VDC.

All available inputs/outputs are electrically isolated from all other circuitry of the module. There is no potential separation between the channels within the same group.

For use in extreme ambient conditions (e.g. wider temperature and humidity range), a special XC version of the device is available.

### Functionality

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fast counter</td>
<td>Integrated, many configurable operating modes (only with AC500)</td>
</tr>
<tr>
<td>LED displays</td>
<td>For signal states, errors and supply voltage</td>
</tr>
<tr>
<td>Internal power supply</td>
<td>Through the expansion bus interface (I/O bus)</td>
</tr>
<tr>
<td>External power supply</td>
<td>Via the terminals ZP and UP (process voltage 24 VDC)</td>
</tr>
<tr>
<td>Required terminal unit</td>
<td>TU515 or TU516 ≡ Chapter 1.5.3.5.2 “TU515, TU516, TU541 and TU542 for I/O modules” on page 2311</td>
</tr>
<tr>
<td>Effect of incorrect input terminal connection</td>
<td>Wrong or no signal detected, no damage up to 35 V</td>
</tr>
</tbody>
</table>

**NOTICE!**

Attention:

All I/O channels (digital and analog) are protected against reverse polarity, reverse supply, short circuit and continuous overvoltage up to 30 VDC.

The device is plugged on a terminal unit ≡ Chapter 1.5.3.5.2 “TU515, TU516, TU541 and TU542 for I/O modules” on page 2311. Position the module properly and press until it locks in place. The terminal unit is either mounted on a DIN rail or to the wall using 2 screws plus the additional accessory for wall mounting (TA526 ≡ Chapter 1.5.3.8.1.4 “TA526 - Wall mounting accessory” on page 3035).

### Electrical connection

The electrical connection of the I/O channels is carried out using the 40 terminals of the I/O terminal unit. I/O modules can be replaced without re-wiring the terminal units.

The terminals 1.8 to 4.8 and 1.9 to 4.9 are electrically interconnected within the I/O terminal unit and always have the same assignment, irrespective of the inserted module:
Terminals 1.8 to 4.8: process voltage UP = +24 VDC
Terminals 1.9 to 4.9: process voltage ZP = 0 VDC

1 I/O bus
2 4.0 - 4.7: Connected with UP (switch) -> Input;
   Connected with ZP (load) -> Output
3 Switch-gear cabinet earth

The assignment of the other terminals:

<table>
<thead>
<tr>
<th>Terminals</th>
<th>Signal</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0 to 1.3</td>
<td>+24 V</td>
<td>4 x sensor power supply sources (loadable with 0.5 A in total)</td>
</tr>
<tr>
<td>1.4 to 1.7</td>
<td>0 V</td>
<td>0 V (reference potential)</td>
</tr>
<tr>
<td>2.0 to 2.7</td>
<td>C0 to C7</td>
<td>8 digital inputs/outputs</td>
</tr>
<tr>
<td>Terminals</td>
<td>Signal</td>
<td>Description</td>
</tr>
<tr>
<td>-----------</td>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td>3.0 to 3.7</td>
<td>C8 to C15</td>
<td>8 digital inputs/outputs</td>
</tr>
<tr>
<td>4.0 to 4.7</td>
<td>C16 to C23</td>
<td>8 digital inputs/outputs</td>
</tr>
</tbody>
</table>

**CAUTION!**
The process supply voltage must be included in the earthing concept (e.g. earthing of the negative pole).

The internal power supply voltage for the module's circuitry is carried out via the I/O bus (provided by a bus module or a CPU). Thus, the current consumption from 24 VDC power supply at the terminals L+/UP and M/ZP of the CPU/bus module increases by 2 mA per DC523.

The external power supply connection is carried out via the UP (+24 VDC) and the ZP (0 VDC) terminals.

**WARNING!**
Removal/Insertion under power
The devices are not designed for removal or insertion under power. Because of unforeseeable consequences, it is not allowed to plug or unplug devices with the power being ON.

Make sure that all voltage sources (supply and process voltage) are switched off before you
- connect or disconnect any signal or terminal block
- remove, mount or replace a module.

Disconnecting any powered devices while energized in a hazardous location could result in an electric arc, which could create a flammable ignition resulting in fire or explosion.

Make sure that power is removed and that the area has been thoroughly checked to ensure that flammable materials are not present prior to proceeding.

The devices must not be opened when in operation. The same applies to the network interfaces.

**NOTICE!**
Risk of damaging the PLC modules!
Overvoltages and short circuits might damage the PLC modules.
- Make sure that all voltage sources (supply voltage and process supply voltage) are switched off before you begin with operations at the system.
- Never connect any voltages or signals to reserved terminals (marked with ---). Reserved terminals may carry internal voltages.

**NOTICE!**
Risk of influences to the connected sensors!
Some sensors may be influenced by the deactivated module outputs of DC523.
Connect a 470 Ω / 1 W resistor in series to inputs C16/C17 if they are used as fast counter inputs to avoid any influences.

The modules provide several diagnosis functions "Chapter 1.5.3.6.1.2.2.7 "Diagnosis" on page 2472."
**Internal data exchange**

<table>
<thead>
<tr>
<th></th>
<th>Without the Fast Counter</th>
<th>With the Fast Counter (only with AC500)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Digital inputs (bytes)</td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>Digital outputs (bytes)</td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>Counter input data (words)</td>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td>Counter output data (words)</td>
<td>0</td>
<td>8</td>
</tr>
</tbody>
</table>

**I/O configuration**

The module itself does not store configuration data. It receives its parameterization data from the master device of the I/O bus (CPU or bus module) during power-up of the system.

Hence, replacing I/O modules is possible without any re-parameterization via software.

*If the external power supply voltage via UP/ZP terminals fails, the I/O module loses its configuration data. The whole station has to be switched off and on again to re-configure the module.*

**Parameterization**

**Firmware version**

<table>
<thead>
<tr>
<th>Firmware version</th>
<th>Configuration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Firmware version &gt; V2.0.0</td>
<td>The arrangement of the parameter data is performed by Control Builder Plus/ Automation Builder software.</td>
</tr>
</tbody>
</table>

The parameter data directly influences the functionality of modules.

For non-standard applications, it is necessary to adapt the parameters to your system configuration.

Module: Module slot address: Y = 1...10

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>Internal value</th>
<th>Internal value, type</th>
<th>Default</th>
<th>Min.</th>
<th>Max.</th>
<th>EDS Slot/Index</th>
</tr>
</thead>
<tbody>
<tr>
<td>Module ID</td>
<td>Internal</td>
<td>1215 (1)</td>
<td>Word</td>
<td>1215 0x04BF</td>
<td>0</td>
<td>65535</td>
<td>0x0Y01</td>
</tr>
<tr>
<td>Ignore module</td>
<td>No Yes</td>
<td>0 1</td>
<td>Byte</td>
<td>No 0x00</td>
<td></td>
<td></td>
<td>Not for FBP</td>
</tr>
<tr>
<td>Parameter length</td>
<td>Internal</td>
<td>9</td>
<td>Byte</td>
<td>9-CPU 8-FBP</td>
<td>0</td>
<td>255</td>
<td>0x0Y02</td>
</tr>
<tr>
<td>Check supply</td>
<td>Off on</td>
<td>0 1</td>
<td>Byte</td>
<td>On 0x01</td>
<td>0</td>
<td>1</td>
<td>0x=Y03</td>
</tr>
<tr>
<td>Name</td>
<td>Value</td>
<td>Internal value</td>
<td>Internal value, type</td>
<td>Default</td>
<td>Min.</td>
<td>Max.</td>
<td>EDS Slot/Index</td>
</tr>
<tr>
<td>-------------------------------------</td>
<td>-------------</td>
<td>----------------</td>
<td>---------------------</td>
<td>-----------</td>
<td>------</td>
<td>------</td>
<td>-----------------</td>
</tr>
<tr>
<td>Input delay</td>
<td>0.1 ms</td>
<td>0</td>
<td>Byte</td>
<td>8 ms</td>
<td>0</td>
<td>3</td>
<td>0x0Y04</td>
</tr>
<tr>
<td></td>
<td>1 ms</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>8 ms</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>32 ms</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Fast counter 4)</td>
<td>0</td>
<td>0</td>
<td>Byte</td>
<td>Mode 0</td>
<td>0</td>
<td></td>
<td>Not for FBP</td>
</tr>
<tr>
<td></td>
<td>:</td>
<td>:</td>
<td></td>
<td>0x00</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Short circuit detection of output or sensor supply</td>
<td>Off</td>
<td>0</td>
<td>Byte</td>
<td>On</td>
<td>0</td>
<td>1</td>
<td>0x0Y05</td>
</tr>
<tr>
<td></td>
<td>On</td>
<td>1</td>
<td></td>
<td>0x01</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Behaviour of outputs at communication errors</td>
<td>Off</td>
<td>0</td>
<td>Byte</td>
<td>Off</td>
<td>0</td>
<td>2</td>
<td>0x0Y06</td>
</tr>
<tr>
<td></td>
<td>Last value</td>
<td>1+(n*5)</td>
<td></td>
<td>0x00</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Substitute value</td>
<td>2+(n*5), n ≤ 2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Substitute value at outputs B23 = Output 23 Bit 0 = Output 0</td>
<td>0...</td>
<td>0...</td>
<td>DWord</td>
<td>0</td>
<td>0</td>
<td>224-1</td>
<td>0x0Y07</td>
</tr>
<tr>
<td></td>
<td>16777215</td>
<td>0x00ff-ffff</td>
<td></td>
<td>0x0000</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>-0000</td>
<td></td>
<td>-0000</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Remarks:

1) With CS31 and addresses smaller than 70 and FBP, the value is increased by 1
2) Not with FBP
3) For a description of the counter operating modes, please refer to the Fast Counter section "Chapter 1.5.3.6.1.2.9 "Fast counter" on page 2543"
4) With FBP or CS31 without the parameter fast counter

GSD file:

```
Ext_User_Prm_Data.Len = 11
Ext_User_Prm_Data(Const(0) =
0x04, 0xc0, 0x08, \n0x01, 0x02, 0x01, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00;
```
**Diagnosis**

In case of overload or short circuit, the outputs switch off automatically and try to switch on again cyclically. Therefore an acknowledgement of the outputs is not necessary. The LED error message, however, is stored.

<table>
<thead>
<tr>
<th>E1...E4</th>
<th>d1</th>
<th>d2</th>
<th>d3</th>
<th>d4</th>
<th>Identifier 000...063</th>
<th>AC500 display</th>
<th>&lt;= Display in</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class</td>
<td>Comp</td>
<td>Dev</td>
<td>Mod</td>
<td>Ch</td>
<td>Err</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Byte 6</td>
<td>Bit 6...7</td>
<td>-</td>
<td>Byte 3</td>
<td>Byte 4</td>
<td>Byte 5</td>
<td>Byte 6 Bit 0...5</td>
<td>FBP diagnosis block</td>
</tr>
<tr>
<td>Class</td>
<td>Interface</td>
<td>Device</td>
<td>Module</td>
<td>Channel</td>
<td>Error identifier</td>
<td>Error message</td>
<td>Remedy</td>
</tr>
<tr>
<td>Remarks:</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Module error

<table>
<thead>
<tr>
<th>1)</th>
<th>2)</th>
<th>3)</th>
<th>4)</th>
</tr>
</thead>
<tbody>
<tr>
<td>11 / 12</td>
<td>ADR</td>
<td>1...10</td>
<td>1...10</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Class</th>
<th>Interface</th>
<th>Device</th>
<th>Module</th>
<th>Channel</th>
<th>Error identifier</th>
<th>Error message</th>
<th>Remedy</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>14</td>
<td>1...10</td>
<td>31</td>
<td>31</td>
<td>19</td>
<td>Checksum error in the I/O module</td>
<td>Replace I/O module</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>14</td>
<td>1...10</td>
<td>31</td>
<td>31</td>
<td>3</td>
<td>Timeout in the I/O module</td>
<td>Replace I/O module</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>14</td>
<td>1...10</td>
<td>31</td>
<td>31</td>
<td>40</td>
<td>Different hard-/firmware versions in the module</td>
<td>Replace I/O module</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>14</td>
<td>1...10</td>
<td>31</td>
<td>31</td>
<td>43</td>
<td>Internal error in the module</td>
<td>Replace I/O module</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>14</td>
<td>1...10</td>
<td>31</td>
<td>31</td>
<td>36</td>
<td>Internal data exchange failure</td>
<td>Replace I/O module</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>14</td>
<td>1...10</td>
<td>31</td>
<td>31</td>
<td>9</td>
<td>Overflow diagnosis buffer</td>
<td>New start</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>14</td>
<td>1...10</td>
<td>31</td>
<td>31</td>
<td>26</td>
<td>Parameter error</td>
<td>Check master</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>14</td>
<td>1...10</td>
<td>31</td>
<td>31</td>
<td>11</td>
<td>Process voltage too low</td>
<td>Check process voltage</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>14</td>
<td>1...10</td>
<td>31</td>
<td>31</td>
<td>45</td>
<td>Process voltage is switched off (ON -&gt; OFF)</td>
<td>Process voltage ON</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Channel error

| 4 | 14 | 1...10 | 2 | 0...23 | 47 | Short circuit at an output | Check connection |
|   |    |        |   |        |    |                            |                |
|   | 11 / 12 | ADR | 1...10 | | | | |

Remarks:
1) In AC500 the following interface identifier applies:
14 = I/O bus, 11 = COM1 (e.g. CS31 bus), 12 = COM2.
The FBP diagnosis block does not contain this identifier.

2) With "Device" the following allocation applies:
31 = module itself,
1...10 = decentralized communication interface module 1...10,
ADR = Hardware address (e.g. of the DC551)

3) With "Module" the following allocation applies depending on the master:
Module error: I/O bus or FBP: 31 = module itself; COM1/COM2: 1...10 = expansion 1...10
Channel error: I/O bus or FBP = Module type (4 = DC); COM1/COM2: 1...10 = expansion 1...10

4) In case of module errors, with channel "31 = module itself" is output.

State LEDs

During the power ON procedure, the module initializes automatically. All LEDs (except the channel LEDs) are ON during this time.

<table>
<thead>
<tr>
<th>LED</th>
<th>State</th>
<th>Color</th>
<th>LED = OFF</th>
<th>LED = ON</th>
<th>LED flashes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inputs/ outputs C0...C23</td>
<td>Digital input or digital output</td>
<td>Yellow</td>
<td>Input/output = OFF</td>
<td>Input/output = ON ¹)</td>
<td>--</td>
</tr>
<tr>
<td>UP</td>
<td>Process supply voltage 24 VDC via terminal</td>
<td>Green</td>
<td>Process supply voltage is missing</td>
<td>Process supply voltage OK</td>
<td>--</td>
</tr>
<tr>
<td>CH-ERR1</td>
<td>Channel error, error messages in groups (digital inputs/outputs combined into the groups 1, 2, 3, 4)</td>
<td>Red</td>
<td>No error or process supply voltage is missing</td>
<td>Severe error within the corresponding group</td>
<td>Error on one channel of the corresponding group (e.g. short circuit at an output)</td>
</tr>
<tr>
<td>CH-ERR2</td>
<td>Red</td>
<td>--</td>
<td>Internal error</td>
<td>--</td>
<td></td>
</tr>
<tr>
<td>CH-ERR3</td>
<td>Red</td>
<td>--</td>
<td>Internal error</td>
<td>--</td>
<td></td>
</tr>
<tr>
<td>CH-ERR4</td>
<td>Red</td>
<td>--</td>
<td>Internal error</td>
<td>--</td>
<td></td>
</tr>
</tbody>
</table>

¹) Indication LED is ON even if an input signal is applied to the channel and the supply voltage is off. In this case the module is not operating and does not generate an input signal.

²) All of the LEDs CH-ERR1 to CH-ERR4 light up together

Technical data

The system data of AC500 and S500 ☞ Chapter 1.5.4.5.1 “System data AC500” on page 3058 are valid for standard version.

The system data of AC500-XC ☞ Chapter 1.5.4.6.1 “System data AC500-XC” on page 3099 are valid for the XC version.
Only additional details are therefore documented below.
The technical data are also valid for the XC version.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Process supply voltage UP</td>
<td></td>
</tr>
<tr>
<td>Connections</td>
<td>Terminals 1.8, 2.8, 3.8 and 4.8 for +24 V (UP) as well as 1.9, 2.9, 3.9 and 4.9 for 0 V (ZP)</td>
</tr>
<tr>
<td>Rated value</td>
<td>24 VDC</td>
</tr>
<tr>
<td>Max. ripple</td>
<td>5 %</td>
</tr>
<tr>
<td>Protection against reversed voltage</td>
<td>Yes</td>
</tr>
<tr>
<td>Rated protection fuse on UP</td>
<td>10 A fast</td>
</tr>
<tr>
<td>Galvanic isolation</td>
<td>Yes, per module</td>
</tr>
<tr>
<td>Current consumption</td>
<td></td>
</tr>
<tr>
<td>From 24 VDC power supply at the terminals UP/L+ and ZP/M of the CPU/Bus Module</td>
<td>Ca. 2 mA</td>
</tr>
<tr>
<td>From UP at normal operation / with outputs</td>
<td>0.1 A + max. 0.5 A per output</td>
</tr>
<tr>
<td>Inrush current from UP (at power up)</td>
<td>0.008 A s</td>
</tr>
<tr>
<td>Max. power dissipation within the module</td>
<td>6 W (outputs unloaded)</td>
</tr>
<tr>
<td>Sensor power supply</td>
<td></td>
</tr>
<tr>
<td>Connections</td>
<td>Terminals 1.0...1.3 = +24 V, 1.4...1.7 = 0 V</td>
</tr>
<tr>
<td>Voltage</td>
<td>24 VDC with short circuit and overload protection</td>
</tr>
<tr>
<td>Loadability</td>
<td>Terminals 1.0...1.3, in total max. 0.5 A</td>
</tr>
<tr>
<td>Weight (without terminal unit)</td>
<td>Ca. 125 g</td>
</tr>
<tr>
<td>Mounting position</td>
<td>Horizontal</td>
</tr>
<tr>
<td></td>
<td>Or vertical with derating (output load reduced to 50 % at 40 °C per group)</td>
</tr>
<tr>
<td>Cooling</td>
<td>The natural convection cooling must not be hindered by cable ducts or other parts in the switchgear cabinet.</td>
</tr>
</tbody>
</table>

**NOTICE!**

**Attention:**

All I/O channels (digital and analog) are protected against reverse polarity, reverse supply, short circuit and continuous overvoltage up to 30 VDC.

---

**Multiple overloads**

No effects of multiple overloads on isolated multi-channel modules occur, as every channel is protected individually by an internal smart high-side switch.
Technical data of the configurable digital inputs/outputs

Each of the configurable I/O channels is defined as input or output by the user program. This is done by interrogating or allocating the corresponding channel.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of channels per module</td>
<td>24 inputs/outputs (with transistors)</td>
</tr>
<tr>
<td>Distribution of the channels into groups</td>
<td>1 group of 24 channels</td>
</tr>
<tr>
<td>If the channels are used as inputs</td>
<td>Channels C0...C7 Terminus 2.0...2.7</td>
</tr>
<tr>
<td></td>
<td>Channels C8...C15 Terminus 3.0...3.7</td>
</tr>
<tr>
<td></td>
<td>Channels C16...C23 Terminus 4.0...4.7</td>
</tr>
<tr>
<td>If the channels are used as outputs</td>
<td>Channels C0...C7 Terminus 2.0...2.7</td>
</tr>
<tr>
<td></td>
<td>Channels C8...C15 Terminus 3.0...3.7</td>
</tr>
<tr>
<td></td>
<td>Channels C16...C23 Terminus 4.0...4.7</td>
</tr>
<tr>
<td>Indication of the input/output signals</td>
<td>1 yellow LED per channel, the LED is ON when</td>
</tr>
<tr>
<td></td>
<td>the input/output signal is high (signal 1)</td>
</tr>
<tr>
<td>Monitoring point of input/output indicator</td>
<td>LED is part of the input circuitry</td>
</tr>
<tr>
<td>Galvanic isolation</td>
<td>From the rest of the module</td>
</tr>
</tbody>
</table>

Technical data of the digital inputs/outputs if used as inputs

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of channels per module</td>
<td>Max. 24 digital inputs</td>
</tr>
<tr>
<td>Reference potential for all inputs</td>
<td>Terminals 1.9, 2.9, 3.9 and 4.9 (minus pole of the process supply voltage, signal name ZP)</td>
</tr>
<tr>
<td>Galvanic isolation</td>
<td>From the rest of the module</td>
</tr>
<tr>
<td>Indication of the input signals</td>
<td>1 yellow LED per channel, the LED is ON when</td>
</tr>
<tr>
<td></td>
<td>the input signal is high (signal 1)</td>
</tr>
<tr>
<td>Monitoring point of input/output indicator</td>
<td>LED is part of the input circuitry</td>
</tr>
<tr>
<td>Input type acc. to EN 61131-2</td>
<td>Type 1</td>
</tr>
<tr>
<td>Input delay (0-&gt;1 or 1-&gt;0)</td>
<td>Typ. 8 ms, configurable from 0.1 to 32 ms</td>
</tr>
<tr>
<td>Input signal voltage</td>
<td>24 VDC</td>
</tr>
<tr>
<td>Signal 0</td>
<td>-3 V...+5 V *)</td>
</tr>
<tr>
<td>Undefined signal</td>
<td>&gt;+5 V...&lt;+15 V</td>
</tr>
<tr>
<td>Signal 1</td>
<td>+15 V...+30 V</td>
</tr>
<tr>
<td>Ripple with signal 0</td>
<td>Within -3 V...+5 V *)</td>
</tr>
<tr>
<td>Ripple with signal 1</td>
<td>Within +15 V...+30 V</td>
</tr>
<tr>
<td>Input current per channel</td>
<td></td>
</tr>
<tr>
<td>Input voltage +24 V</td>
<td>Typ. 5 mA</td>
</tr>
<tr>
<td>Input voltage +5 V</td>
<td>&gt; 1 mA</td>
</tr>
<tr>
<td>Input voltage +15 V</td>
<td>&gt; 5 mA</td>
</tr>
<tr>
<td>Parameter</td>
<td>Value</td>
</tr>
<tr>
<td>----------------------------</td>
<td>----------------</td>
</tr>
<tr>
<td>Input voltage +30 V</td>
<td>&lt; 8 mA</td>
</tr>
<tr>
<td>Max. cable length</td>
<td></td>
</tr>
<tr>
<td>Shielded</td>
<td>1000 m</td>
</tr>
<tr>
<td>Unshielded</td>
<td>600 m</td>
</tr>
</tbody>
</table>

*) Due to the direct connection to the output, the demagnetizing varistor is also effective at the input (see figure) above. This is why the difference between UPx and the input signal must not exceed the clamp voltage of the varistor. The varistor limits the clamp voltage to approx. 36 V. Consequently, the input voltage must range from -12 V to +30 V when UPx = 24 V and from -6 V to +30 V when UPx = 30 V.

**Technical data of the digital inputs/outputs if used as outputs**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of channels per module</td>
<td>Max. 24 transistor outputs</td>
</tr>
<tr>
<td>Reference potential for all outputs</td>
<td>Terminals 1.9, 2.9, 3.9 and 4.9 (negative pole of the process supply voltage, signal name ZP)</td>
</tr>
<tr>
<td>Common power supply voltage</td>
<td>For all outputs: terminals 1.8, 2.8, 3.8 and 4.8 (positive pole of the process supply voltage, signal name UP)</td>
</tr>
<tr>
<td>Output voltage for signal 1</td>
<td>UP (-0.8 V)</td>
</tr>
<tr>
<td>Output delay (0-&gt;1 or 1-&gt;0)</td>
<td>On request</td>
</tr>
<tr>
<td>Output current</td>
<td></td>
</tr>
<tr>
<td>Rated value, per channel</td>
<td>500 mA at UP = 24 V</td>
</tr>
<tr>
<td>Maximum value (all channels together)</td>
<td>8 A</td>
</tr>
<tr>
<td>Leakage current with signal 0</td>
<td>&lt; 0.5 mA</td>
</tr>
<tr>
<td>Rated protection fuse on UP</td>
<td>10 A fast</td>
</tr>
<tr>
<td>Demagnetization when inductive loads are switched off</td>
<td>With varistors integrated in the module (see figure below)</td>
</tr>
<tr>
<td>Switching frequency</td>
<td></td>
</tr>
<tr>
<td>With resistive load</td>
<td>On request</td>
</tr>
<tr>
<td>With inductive loads</td>
<td>Max. 0.5 Hz</td>
</tr>
<tr>
<td>With lamp loads</td>
<td>Max. 11 Hz with max. 5 W</td>
</tr>
<tr>
<td>Short-circuit-proof / overload-proof</td>
<td>Yes</td>
</tr>
<tr>
<td>Overload message (I &gt; 0.7 A)</td>
<td>Yes, after ca. 100 ms</td>
</tr>
<tr>
<td>Output current limitation</td>
<td>Yes, automatic reactivation after short circuit/overload</td>
</tr>
<tr>
<td>Resistance to feedback against 24 V signals</td>
<td>Yes</td>
</tr>
<tr>
<td>Max. cable length</td>
<td></td>
</tr>
<tr>
<td>Shielded</td>
<td>1000 m</td>
</tr>
<tr>
<td>Unshielded</td>
<td>600 m</td>
</tr>
</tbody>
</table>

The following drawing shows the circuitry of a digital input/output with the varistors for demagnetization when inductive loads are switched off.
Technical data of the fast counter

The fast counter of the module does not work if the module is connected to a
- FBP interface module
- CS31 bus module
- CANopen bus module

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Used inputs</td>
<td>C16 / C17</td>
</tr>
<tr>
<td>Used outputs</td>
<td>C18</td>
</tr>
<tr>
<td>Counting frequency</td>
<td>Max. 50 kHz</td>
</tr>
<tr>
<td>Detailed description</td>
<td>See Fast Counter Chapter 1.5.5.1.10 “Fast counters” on page 3194</td>
</tr>
<tr>
<td>Operating modes</td>
<td>See Operating modes</td>
</tr>
</tbody>
</table>

Ordering data

<table>
<thead>
<tr>
<th>Part no.</th>
<th>Description</th>
<th>Product life cycle phase *)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1SAP 240 500 R0001</td>
<td>DC523, digital input/output module, 24 DC, 24 VDC / 0.5 A, 1-wire</td>
<td>Active</td>
</tr>
<tr>
<td>1SAP 440 500 R0001</td>
<td>DC523-XC, digital input/output module, 24 DC, 24 VDC / 0.5 A, 1-wire, XC Version</td>
<td>Active</td>
</tr>
</tbody>
</table>

*) For planning and commissioning of new installations use modules in Active status only.

DC532 - Digital input/output module

- 16 digital inputs 24 VDC, 16 configurable digital inputs/outputs
- Module-wise electrically isolated
- Fast counter
- XC version for use in extreme ambient conditions available

**Intended purpose**

The device can be used as a decentralized I/O extension module for S500 Communication Interface Modules (e.g. CI592-CS31, CI501-PNIO, CI541-DP, CI581-CN) or as a centralized extension module for AC500 CPUs (PM5xx).
Digital configurable input / output unit.

- 16 digital inputs 24 VDC in 2 groups (1.0...1.7 and 2.0...2.7)
- 16 digital configurable inputs/outputs 24 VDC (C16 to C31) in 1 group (3.0...3.7 and 4.0...4.7), of which each can be used
  - as an input,
  - as a transistor output with short circuit and overload protection, 0.5 A rated current or
  - as a re-readable output (combined input/output) with the technical data of the digital
    inputs and outputs.
- Optional with fast counter

The configuration is performed by software. The modules are supplied with a process supply voltage of 24 VDC.

All available inputs/outputs are electrically isolated from all other circuitry of the module. There is no potential separation between the channels within the same group.

For use in extreme ambient conditions (e.g. wider temperature and humidity range), a special XC version of the device is available.

### Functionality

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Digital inputs</td>
<td>16 (24 VDC)</td>
</tr>
<tr>
<td>Digital inputs/outputs</td>
<td>16 (24 VDC)</td>
</tr>
<tr>
<td>Fast counter</td>
<td>Integrated, many configurable operating modes (only with AC500)</td>
</tr>
<tr>
<td>LED displays</td>
<td>For signal states, errors and supply voltage</td>
</tr>
<tr>
<td>Internal power supply</td>
<td>Through the expansion bus interface (I/O bus)</td>
</tr>
<tr>
<td>External power supply</td>
<td>Via the terminals ZP and UP (process voltage 24 VDC)</td>
</tr>
<tr>
<td>Required terminal unit</td>
<td>TU515 or TU516 (Chapter 1.5.3.5.2 “TU515, TU516, TU541 and TU542 for I/O modules” on page 2311)</td>
</tr>
<tr>
<td>Effect of incorrect input terminal connection</td>
<td>Wrong or no signal detected, no damage up to 35 V</td>
</tr>
</tbody>
</table>

The device is plugged on a terminal unit (Chapter 1.5.3.5.2 “TU515, TU516, TU541 and TU542 for I/O modules” on page 2311). Position the module properly and press until it locks in place. The terminal unit is either mounted on a DIN rail or to the wall using 2 screws plus the additional accessory for wall mounting (TA526 (Chapter 1.5.3.8.1.4 “TA526 - Wall mounting accessory” on page 3035).

For a detailed description of the mounting, disassembly and electrical connection of the module, please refer to the System Assembly, Construction and Connection chapter (Chapter 1.5.4.5 “AC500 (Standard)” on page 3058).
### Electrical connection

The electrical connection of the I/O channels is carried out using the 40 terminals of the I/O terminal unit. I/O modules can be replaced without re-wiring the terminal units.

The terminals 1.8 to 4.8 and 1.9 to 4.9 are electrically interconnected within the I/O terminal unit and always have the same assignment, irrespective of the inserted module:

Terminals 1.8 to 4.8: process voltage UP = +24 VDC

Terminals 1.9 to 4.9: process voltage ZP = 0 VDC

<table>
<thead>
<tr>
<th>I/O Bus</th>
<th>Terminal</th>
<th>Assignment</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1.0</td>
<td>2.0</td>
</tr>
<tr>
<td></td>
<td>1.1</td>
<td>2.1</td>
</tr>
<tr>
<td></td>
<td>1.2</td>
<td>2.2</td>
</tr>
<tr>
<td></td>
<td>1.3</td>
<td>2.3</td>
</tr>
<tr>
<td></td>
<td>1.4</td>
<td>2.4</td>
</tr>
<tr>
<td></td>
<td>1.5</td>
<td>2.5</td>
</tr>
<tr>
<td></td>
<td>1.6</td>
<td>2.6</td>
</tr>
<tr>
<td></td>
<td>1.7</td>
<td>2.7</td>
</tr>
<tr>
<td></td>
<td>1.8</td>
<td>2.8</td>
</tr>
<tr>
<td></td>
<td>1.9</td>
<td>2.9</td>
</tr>
<tr>
<td>2</td>
<td>3.0</td>
<td>4.0</td>
</tr>
<tr>
<td></td>
<td>3.1</td>
<td>4.1</td>
</tr>
<tr>
<td></td>
<td>3.2</td>
<td>4.2</td>
</tr>
<tr>
<td></td>
<td>3.3</td>
<td>4.3</td>
</tr>
<tr>
<td></td>
<td>3.4</td>
<td>4.4</td>
</tr>
<tr>
<td></td>
<td>3.5</td>
<td>4.5</td>
</tr>
<tr>
<td></td>
<td>3.6</td>
<td>4.6</td>
</tr>
<tr>
<td></td>
<td>3.7</td>
<td>4.7</td>
</tr>
<tr>
<td></td>
<td>3.8</td>
<td>4.8</td>
</tr>
<tr>
<td></td>
<td>3.9</td>
<td>4.9</td>
</tr>
</tbody>
</table>

1. I/O bus
2. 4.0 - 4.7: Connected with UP (switch) -> Input;
   Connected with ZP (load) -> Output
3. Switch-gear cabinet earth

The assignment of the other terminals:
<table>
<thead>
<tr>
<th>Terminals</th>
<th>Signal</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0 to 1.7</td>
<td>I0 to I7</td>
<td>8 digital inputs</td>
</tr>
<tr>
<td>2.0 to 2.7</td>
<td>I8 to I15</td>
<td>8 digital inputs</td>
</tr>
<tr>
<td>3.0 to 3.7</td>
<td>C16 to C23</td>
<td>8 digital inputs/outputs</td>
</tr>
<tr>
<td>4.0 to 4.7</td>
<td>C24 to C31</td>
<td>8 digital inputs/outputs</td>
</tr>
</tbody>
</table>

**CAUTION!**
The process supply voltage must be included in the earthing concept (e.g. earthing of the negative pole).

The internal power supply voltage for the module's circuitry is carried out via the I/O bus (provided by a bus module or a CPU). Thus, the current consumption from 24 VDC power supply at the terminals L+/UP and M/ZP of the CPU/bus module increases by 2 mA per DC532.

The external power supply connection is carried out via the UP (+24 VDC) and the ZP (0 VDC) terminals.

**WARNING!**
Removal/Insertion under power
The devices are not designed for removal or insertion under power. Because of unforeseeable consequences, it is not allowed to plug or unplug devices with the power being ON.

Make sure that all voltage sources (supply and process voltage) are switched off before you
   – connect or disconnect any signal or terminal block
   – remove, mount or replace a module.

Disconnecting any powered devices while energized in a hazardous location could result in an electric arc, which could create a flammable ignition resulting in fire or explosion.

Make sure that power is removed and that the area has been thoroughly checked to ensure that flammable materials are not present prior to proceeding.

The devices must not be opened when in operation. The same applies to the network interfaces.

**NOTICE!**
Risk of damaging the PLC modules!
Overvoltages and short circuits might damage the PLC modules.

- Make sure that all voltage sources (supply voltage and process supply voltage) are switched off before you begin with operations at the system.
- Never connect any voltages or signals to reserved terminals (marked with ---). Reserved terminals may carry internal voltages.

**NOTICE!**
Risk of influences to the connected sensors!
Some sensors may be influenced by the deactivated module outputs of DC532.
Connect a 470 Ω / 1 W resistor in series to inputs C24/C25 if using them as fast counter inputs to avoid any influences.
The module provides several diagnosis functions \( \text{\S} \) Chapter 1.5.3.6.1.2.3.7 “Diagnosis” on page 2484.

Internal data exchange

<table>
<thead>
<tr>
<th></th>
<th>Without the Fast Counter</th>
<th>With the Fast Counter (only with AC500)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Digital inputs (bytes)</td>
<td>4</td>
<td>6</td>
</tr>
<tr>
<td>Digital outputs (bytes)</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>Counter input data (words)</td>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td>Counter output data (words)</td>
<td>0</td>
<td>8</td>
</tr>
</tbody>
</table>

I/O configuration

The module itself does not store configuration data. It receives its parameterization data from the master device of the I/O bus (CPU or bus module) during power-up of the system. Hence, replacing I/O modules is possible without any re-parameterization via software.

\[\text{If the external power supply voltage via UP/ZP terminals fails, the I/O module loses its configuration data. The whole station has to be switched off and on again to re-configure the module.}\]

Parameterization

<table>
<thead>
<tr>
<th>Firmware version</th>
<th>Configuration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Firmware version &gt; V2.0.0</td>
<td>The arrangement of the parameter data is performed by Control Builder Plus/ Automation Builder software.</td>
</tr>
</tbody>
</table>

The parameter data directly influences the functionality of modules.

For non-standard applications, it is necessary to adapt the parameters to your system configuration.

Module: Module slot address: Y = 1...10

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>Internal value</th>
<th>Internal value, type</th>
<th>Default</th>
<th>Min.</th>
<th>Max.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Module ID</td>
<td>Internal</td>
<td>1200</td>
<td>Word</td>
<td>1200 0x04B0</td>
<td>0</td>
<td>65535</td>
</tr>
<tr>
<td>Ignore module</td>
<td>No</td>
<td>0</td>
<td>Byte</td>
<td>No 0x00</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Parameter length</td>
<td>Internal</td>
<td>7</td>
<td>Byte</td>
<td>7-CPU 6-FBP</td>
<td>0</td>
<td>255</td>
</tr>
</tbody>
</table>

\(1)\) Not for FBP

\(2)\) Value '1' does not make sense; the parameter is always set to '0'.
<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>Internal value</th>
<th>Internal value, type</th>
<th>Default</th>
<th>Min.</th>
<th>Max.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Check supply</td>
<td>Off</td>
<td>0</td>
<td>1 Byte</td>
<td>On</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>on</td>
<td></td>
<td></td>
<td>0x01</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Input delay</td>
<td>0.1 ms</td>
<td>0</td>
<td>1 Byte</td>
<td>8 ms</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>1 ms</td>
<td>1</td>
<td></td>
<td>0x02</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>8 ms</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>32 ms</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Fast counter 4)</td>
<td>0</td>
<td>0</td>
<td>1 Byte</td>
<td>Mode 0</td>
<td>0</td>
<td>Not for FBP</td>
</tr>
<tr>
<td></td>
<td>:</td>
<td></td>
<td></td>
<td>0x00</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Output short circuit detection</td>
<td>Off</td>
<td>0</td>
<td>1 Byte</td>
<td>On</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>On</td>
<td></td>
<td></td>
<td>0x01</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Behaviour of outputs at communication errors</td>
<td>Off</td>
<td>0</td>
<td>1 Byte</td>
<td>Off</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>Last value Substitute value</td>
<td>1+(n*5)</td>
<td></td>
<td>0x00</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>2+(n*5), n ≤ 2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Substitute value at outputs Bit 15 = Output 15 Bit 0 = Output 0</td>
<td>0...65535</td>
<td>0...0xffff</td>
<td>Word</td>
<td>0x0000</td>
<td>0</td>
<td>65535</td>
</tr>
</tbody>
</table>

Remarks:

1) With CS31 and addresses smaller than 70 and FBP, the value is increased by 1
2) Not with FBP
3) For a description of the counter operating modes, please refer to the Fast Counter section % Chapter 1.5.3.6.1.2.9 “Fast counter” on page 2543
4) With FBP or CS31 without the parameter Fast Counter

GSD file:

```
Ext_User_Prm_Data_Len = 9
Ext_User_Prm_Data_Const(0) = 0x04, 0xb1, 0x06, \0x01, 0x02, 0x01, 0x00, 0x00, 0x00;
```
Diagnosis

In case of overload or short circuit, the outputs switch off automatically and try to switch on again cyclically. Therefore, an acknowledgement of the outputs is not necessary. The LED error message, however, is stored.

<table>
<thead>
<tr>
<th>E1...E4</th>
<th>d1</th>
<th>d2</th>
<th>d3</th>
<th>d4</th>
<th>Identifier 000...063</th>
<th>AC500 display</th>
<th>&lt;- Display in</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class</td>
<td>Comp</td>
<td>Dev</td>
<td>Mod</td>
<td>Ch</td>
<td>Err</td>
<td>PS501 PLC browser</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Byte 6</th>
<th>Bit 6...7</th>
<th>Byte 3</th>
<th>Byte 4</th>
<th>Byte 5</th>
<th>Byte 6</th>
<th>Bit 0...5</th>
<th>FBP diagnosis block</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class</td>
<td>Interface</td>
<td>Device</td>
<td>Module</td>
<td>Channel</td>
<td>Error Identifier</td>
<td>Error message</td>
<td>Remedy</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Module error</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
</tr>
<tr>
<td>11 / 12</td>
</tr>
<tr>
<td>3</td>
</tr>
<tr>
<td>11 / 12</td>
</tr>
<tr>
<td>3</td>
</tr>
<tr>
<td>11 / 12</td>
</tr>
<tr>
<td>3</td>
</tr>
<tr>
<td>11 / 12</td>
</tr>
<tr>
<td>3</td>
</tr>
<tr>
<td>11 / 12</td>
</tr>
<tr>
<td>3</td>
</tr>
<tr>
<td>11 / 12</td>
</tr>
<tr>
<td>3</td>
</tr>
<tr>
<td>11 / 12</td>
</tr>
<tr>
<td>3</td>
</tr>
<tr>
<td>11 / 12</td>
</tr>
<tr>
<td>4</td>
</tr>
<tr>
<td>11 / 12</td>
</tr>
</tbody>
</table>

Channel error DC532

<table>
<thead>
<tr>
<th>Channel error DC532</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
</tr>
<tr>
<td>11 / 12</td>
</tr>
</tbody>
</table>

Remarks:
1) In AC500 the following interface identifier applies:
   14 = I/O bus, 11 = COM1 (e.g. CS31 bus), 12 = COM2.
   The FBP diagnosis block does not contain this identifier.

2) With "Device" the following allocation applies:
   31 = module itself,
   1...10 = decentralized communication interface module 1...10,
   ADR = hardware address (e.g. of the DC551)

3) With "Module" the following allocation applies depending on the master:
   Module error: I/O bus or FBP: 31 = module itself; COM1/COM2: 1...10 = expansion 1...10
   Channel error: I/O bus or FBP = module type (4 = DC); COM1/COM2: 1...10 = expansion 1...10

4) In case of module errors, with channel "31 = module itself" is output.

State LEDs

During the power ON procedure, the module initializes automatically. All LEDs (except the channel LEDs) are ON during this time.

<table>
<thead>
<tr>
<th>LED</th>
<th>State</th>
<th>Color</th>
<th>LED = OFF</th>
<th>LED = ON</th>
<th>LED flashes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inputs I0...I15</td>
<td>Digital input</td>
<td>Yellow</td>
<td>Input = OFF</td>
<td>Input = ON ¹</td>
<td>--</td>
</tr>
<tr>
<td>Inputs/ outputs C16...C31</td>
<td>Digital input/output</td>
<td>Yellow</td>
<td>Input/output = OFF</td>
<td>Input/output = ON ¹</td>
<td>--</td>
</tr>
<tr>
<td>UP</td>
<td>Process supply voltage 24 VDC via terminal</td>
<td>Green</td>
<td>Process supply voltage is missing</td>
<td>Process supply voltage OK</td>
<td>--</td>
</tr>
</tbody>
</table>
| CH-ERR1 | Channel Error, error messages in groups (digital inputs/outputs combined into the groups 1, 2, 3, 4) | Red | No error or process supply voltage is missing | Severe error within the corresponding group | Error on one channel of the corresponding group (e.g. short circuit at an output)
| CH-ERR2 |  | Red |  |  |  |
| CH-ERR3 |  | Red |  |  |  |
| CH-ERR4 |  | Red |  |  |  |
| CH-ERR ² | Module Error | Red | -- | Internal error | -- |

¹) Indication LED is ON even if an input signal is applied to the channel and the supply voltage is off. In this case the module is not operating and does not generate an input signal.

²) All of the LEDs CH-ERR1 to CH-ERR4 light up together
Technical data

The system data of AC500 and S500 "Chapter 1.5.4.5.1 “System data AC500” on page 3058 are valid for the standard version.

The system data of AC500-XC "Chapter 1.5.4.6.1 “System data AC500-XC” on page 3099 are valid for the XC version.

Only additional details are therefore documented below.

The technical data are also valid for the XC version.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Process supply voltage UP</td>
<td></td>
</tr>
<tr>
<td>Connections</td>
<td>Terminals 1.8, 2.8, 3.8 and 4.8 for +24 V (UP) as well as 1.9, 2.9, 3.9 and 4.9 for 0 V (ZP)</td>
</tr>
<tr>
<td>Rated value</td>
<td>24 VDC</td>
</tr>
<tr>
<td>Max. ripple</td>
<td>5 %</td>
</tr>
<tr>
<td>Protection against reversed voltage</td>
<td>Yes</td>
</tr>
<tr>
<td>Rated protection fuse on UP</td>
<td>10 A fast</td>
</tr>
<tr>
<td>Galvanic isolation</td>
<td>Yes, per module</td>
</tr>
<tr>
<td>Current consumption</td>
<td></td>
</tr>
<tr>
<td>From 24 VDC power supply at the terminals UP/L+ and ZP/M of the CPU/bus module</td>
<td>Ca. 2 mA</td>
</tr>
<tr>
<td>From UP at normal operation / with outputs</td>
<td>0.15 A + max. 0.5 A per output</td>
</tr>
<tr>
<td>Inrush current from UP (at power up)</td>
<td>0.007 A·s</td>
</tr>
<tr>
<td>Max. power dissipation within the module</td>
<td>6 W (outputs unloaded)</td>
</tr>
<tr>
<td>Weight (without terminal unit)</td>
<td>ca. 125 g</td>
</tr>
<tr>
<td>Mounting position</td>
<td>Horizontal</td>
</tr>
<tr>
<td>Or vertical with derating (output load reduced to 50 % at 40 °C per group)</td>
<td></td>
</tr>
<tr>
<td>Cooling</td>
<td>The natural convection cooling must not be hindered by cable ducts or other parts in the switch-gear cabinet.</td>
</tr>
</tbody>
</table>

**NOTICE!**

Attention:

All I/O channels (digital and analog) are protected against reverse polarity, reverse supply, short circuit and continuous overvoltage up to 30 VDC.

**Multiple overloads**

No effects of multiple overloads on isolated multi-channel modules occur, as every channel is protected individually by an internal smart high-side switch.
## Technical data of the digital inputs

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of channels per module</td>
<td>16</td>
</tr>
<tr>
<td>Distribution of the channels into groups</td>
<td>1 group of 16 channels</td>
</tr>
<tr>
<td>Terminals of the channels I0 to I7</td>
<td>1.0 to 1.7</td>
</tr>
<tr>
<td>Terminals of the channels I8 to I15</td>
<td>2.0 to 2.7</td>
</tr>
<tr>
<td>Reference potential for all inputs</td>
<td>Terminals 1.9, 2.8, 3.8 and 4.9 (negative pole of the process supply voltage, signal name ZP)</td>
</tr>
<tr>
<td>Galvanic isolation</td>
<td>From the rest of the module (I/O bus)</td>
</tr>
<tr>
<td>Indication of the input signals</td>
<td>1 yellow LED per channel, the LED is ON when the input signal is high (signal 1)</td>
</tr>
<tr>
<td>Monitoring point of input indicator</td>
<td>LED is part of the input circuitry</td>
</tr>
<tr>
<td>Input type acc. to EN 61131-2</td>
<td>Type 1</td>
</tr>
<tr>
<td>Input delay (0-&gt;1 or 1-&gt;0)</td>
<td>Typ. 8 ms, configurable from 0.1 to 32 ms</td>
</tr>
<tr>
<td>Input signal voltage</td>
<td>24 VDC</td>
</tr>
<tr>
<td>Signal 0</td>
<td>-3 V...+5 V</td>
</tr>
<tr>
<td>Undefined signal</td>
<td>&gt; +5 V...&lt; +15 V</td>
</tr>
<tr>
<td>Parameter</td>
<td></td>
</tr>
<tr>
<td>Signal 1</td>
<td>+15 V...+30 V</td>
</tr>
<tr>
<td>Ripple with signal 0</td>
<td>Within -3 V...+5 V</td>
</tr>
<tr>
<td>Ripple with signal 1</td>
<td>Within +15 V...+30 V</td>
</tr>
<tr>
<td>Input current per channel</td>
<td></td>
</tr>
<tr>
<td>Input voltage +24 V</td>
<td>Typ. 5 mA</td>
</tr>
<tr>
<td>Input voltage +5 V</td>
<td>&gt; 1 mA</td>
</tr>
<tr>
<td>Input voltage +15 V</td>
<td>&gt; 5 mA</td>
</tr>
<tr>
<td>Input voltage +30 V</td>
<td>&lt; 8 mA</td>
</tr>
<tr>
<td>Max. cable length</td>
<td></td>
</tr>
<tr>
<td>Shielded</td>
<td>1000 m</td>
</tr>
<tr>
<td>Unshielded</td>
<td>600 m</td>
</tr>
</tbody>
</table>

## Technical data of the configurable digital inputs/outputs

Each of the configurable I/O channels is defined as input or output by the user program. This is done by interrogating or allocating the corresponding channel.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of channels per module</td>
<td>16 inputs/outputs (with transistors)</td>
</tr>
<tr>
<td>Distribution of the channels into groups</td>
<td>1 group of 16 channels</td>
</tr>
<tr>
<td>If the channels are used as inputs</td>
<td></td>
</tr>
<tr>
<td>Channels I16...I23</td>
<td>Terminals 3.0...3.7</td>
</tr>
<tr>
<td>Channels I24...I31</td>
<td>Terminals 4.0...4.7</td>
</tr>
<tr>
<td>If the channels are used as outputs</td>
<td></td>
</tr>
</tbody>
</table>
Parameter | Value
--- | ---
Channels Q16...Q23 | Terminals 3.0...3.7
Channels Q24...Q31 | Terminals 4.0...4.7
Indication of the input/output signals | 1 yellow LED per channel, the LED is ON when the input/output signal is high (signal 1)
Monitoring point of input/output indicator | LED is part of the input circuitry
Galvanic isolation | From the rest of the module

Technical data of the digital inputs/outputs if used as inputs

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of channels per module</td>
<td>Max. 16 digital inputs</td>
</tr>
<tr>
<td>Reference potential for all inputs</td>
<td>Terminals 1.9, 2.9, 3.9 and 4.9 (negative pole of the process supply voltage, signal name ZP)</td>
</tr>
<tr>
<td>Input current, per channel</td>
<td>See Technical Data of the Digital Inputs % Chapter 1.5.3.6.1.2.3.9.1 &quot;Technical data of the digital inputs&quot; on page 2487</td>
</tr>
<tr>
<td>Input type acc. to EN 61131-2</td>
<td>Type 1</td>
</tr>
<tr>
<td>Input delay (0-&gt;1 or 1-&gt;0)</td>
<td>Typ. 8 ms, configurable from 0.1 to 32 ms</td>
</tr>
<tr>
<td>Input signal voltage</td>
<td>24 VDC</td>
</tr>
<tr>
<td>Signal 0</td>
<td>-3 V...+5 V *)</td>
</tr>
<tr>
<td>undefined signal</td>
<td>&gt; +5 V...&lt; +15 V</td>
</tr>
<tr>
<td>Signal 1</td>
<td>+15 V...+30 V</td>
</tr>
<tr>
<td>Ripple with signal 0</td>
<td>Within -3 V...+5 V *)</td>
</tr>
<tr>
<td>Ripple with signal 1</td>
<td>Within +15 V...+30 V</td>
</tr>
<tr>
<td>Max. cable length</td>
<td>1000 m</td>
</tr>
<tr>
<td>Shielded</td>
<td>600 m</td>
</tr>
</tbody>
</table>

*) Due to the direct connection to the output, the demagnetizing varistor is also effective at the input (see figure) above. This is why the difference between UPx and the input signal must not exceed the clamp voltage of the varistor. The varistor limits the clamp voltage to approx. 36 V. Consequently, the input voltage must range from -12 V to +30 V when UPx = 24 V and from -6 V to +30 V when UPx = 30 V.

Technical data of the digital inputs/outputs if used as outputs

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of channels per module</td>
<td>Max. 16 transistor outputs</td>
</tr>
<tr>
<td>Reference potential for all outputs</td>
<td>Terminals 1.9, 2.9, 3.9 and 4.9 (negative pole of the process supply voltage, signal name ZP)</td>
</tr>
<tr>
<td>Common power supply voltage</td>
<td>For all outputs: terminals 1.8, 2.8, 3.8 and 4.8 (positive pole of the process supply voltage, signal name UP)</td>
</tr>
<tr>
<td>Output voltage for signal 1</td>
<td>UP (-0.8 V)</td>
</tr>
<tr>
<td>Output delay (0-&gt;1 or 1-&gt;0)</td>
<td>On request</td>
</tr>
</tbody>
</table>
### Parameter

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output current</td>
<td></td>
</tr>
<tr>
<td>Rated value, per channel</td>
<td>500 mA at UP = 24 V</td>
</tr>
<tr>
<td>Maximum value (all channels together)</td>
<td>8 A</td>
</tr>
<tr>
<td>Leakage current with signal 0</td>
<td>&lt; 0.5 mA</td>
</tr>
<tr>
<td>Rated protection fuse on UP</td>
<td>10 A fast</td>
</tr>
<tr>
<td>Demagnetization when inductive loads are switched off</td>
<td>With varistors integrated in the module (see figure below)</td>
</tr>
<tr>
<td>Switching frequency</td>
<td></td>
</tr>
<tr>
<td>With resistive load</td>
<td>On request</td>
</tr>
<tr>
<td>With inductive loads</td>
<td>Max. 0.5 Hz</td>
</tr>
<tr>
<td>With lamp loads</td>
<td>Max. 11 Hz with max. 5 W</td>
</tr>
<tr>
<td>Short-circuit-proof / overload-proof</td>
<td>Yes</td>
</tr>
<tr>
<td>Overload message (I &gt; 0.7 A)</td>
<td>Yes, after ca. 100 ms</td>
</tr>
<tr>
<td>Output current limitation</td>
<td>Yes, automatic reactivation after short circuit/overload</td>
</tr>
<tr>
<td>Resistance to feedback against 24 V signals</td>
<td>Yes</td>
</tr>
<tr>
<td>Max. cable length</td>
<td></td>
</tr>
<tr>
<td>Shielded</td>
<td>1000 m</td>
</tr>
<tr>
<td>Unshielded</td>
<td>600 m</td>
</tr>
</tbody>
</table>

The following drawing shows the circuitry of a digital input/output with the varistors for demagnetization when inductive loads are switched off.

![Circuit Diagram](image)

**Technical data of the fast counter**

*The fast counter of the module does not work if the module is connected to a*

- FBP interface module
- CS31 bus module
- CANopen bus module
### Parameter Table

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Used inputs</td>
<td>C24/C25</td>
</tr>
<tr>
<td>Used outputs</td>
<td>C26</td>
</tr>
<tr>
<td>Counting frequency</td>
<td>Max. 50 kHz</td>
</tr>
<tr>
<td>Detailed description</td>
<td>See <em>Fast Counter</em> § Chapter 1.5.5.1.10 “Fast counters” on page 3194</td>
</tr>
<tr>
<td>Operating modes</td>
<td>See <em>Operating modes</em></td>
</tr>
</tbody>
</table>

### Ordering Data

<table>
<thead>
<tr>
<th>Part no.</th>
<th>Description</th>
<th>Product life cycle phase *)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1SAP 240 100 R0001</td>
<td>DC532, digital input/output module, 16 DI, 16 DC, 24 VDC / 0.5 A, 1-wire</td>
<td>Active</td>
</tr>
<tr>
<td>1SAP 440 100 R0001</td>
<td>DC532-XC, digital input/output module, 16 DI, 16 DC, 24 VDC / 0.5 A, 1-wire, XC version</td>
<td>Active</td>
</tr>
</tbody>
</table>

* *) For planning and commissioning of new installations use modules in Active status only.

### DI524 - Digital Input Module

- 32 digital inputs 24 VDC in 4 groups (1.0...1.7, 2.0...2.7, 3.0...3.7 and 4.0...4.7)
- Fast counter
- Module-wise electrically isolated
- XC version for use in extreme ambient conditions available
Fig. 114: Digital input module DI524, plugged on a terminal unit TU516

1 I/O bus
2 Allocation between terminal number and signal name
3 32 yellow LEDs to display the signal states at the digital inputs (I0 - I31)
4 1 green LED to display the state of the process supply voltage UP
5 4 red LEDs to display errors
6 Label
7 Terminal unit
8 DIN rail
* Sign for XC version

**Intended purpose**

The device can be used as a decentralized I/O extension module for S500 Communication Interface Modules (e.g. CI592-CS31, CI501-PNIO, CI541-DP, CI581-CN) or as a centralized extension module for AC500 CPUs (PM5xx).

The configuration is performed by software. The modules are supplied with a process supply voltage of 24 VDC.

All available inputs/outputs are electrically isolated from all other circuitry of the module. There is no potential separation between the channels within the same group.

For use in extreme ambient conditions (e.g. wider temperature and humidity range), a special XC version of the device is available.
### Functionality

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fast counter</td>
<td>Integrated, many configurable operating modes (only with AC500)</td>
</tr>
<tr>
<td>LED displays</td>
<td>For signal states, errors and supply voltage</td>
</tr>
<tr>
<td>Internal power supply</td>
<td>Via the expansion bus interface (I/O bus)</td>
</tr>
<tr>
<td>External power supply</td>
<td>Via the terminals ZP and UP (process voltage 24 VDC)</td>
</tr>
<tr>
<td>Required terminal units</td>
<td>TU515 or TU516 [Chapter 1.5.3.5.2 “TU515, TU516, TU541 and TU542 for I/O modules” on page 2311]</td>
</tr>
<tr>
<td>Effect of incorrect input terminal connection</td>
<td>Wrong or no signal detected, no damage up to 35 V</td>
</tr>
</tbody>
</table>

The device is plugged on a terminal unit [Chapter 1.5.3.5.2 “TU515, TU516, TU541 and TU542 for I/O modules” on page 2311]. Position the module properly and press until it locks in place. The terminal unit is either mounted on a DIN rail or to the wall using 2 screws plus the additional accessory for wall mounting (TA526 [Chapter 1.5.3.8.1.4 “TA526 - Wall mounting accessory” on page 3035]).

### Electrical connection

> For a detailed description of the mounting, disassembly and electrical connection of the module, please refer to the System Assembly, Construction and Connection chapter [Chapter 1.5.4.5 “AC500 (Standard)” on page 3058].

The electrical connection of the I/O channels is carried out using the 40 terminals of the I/O terminal unit. I/O modules can be replaced without re-wiring the terminal units.

The terminals 1.8 to 4.8 and 1.9 to 4.9 are electrically interconnected within the I/O terminal unit and have always the same assignment, irrespective of the inserted module:

- Terminals 1.8 to 4.8: process voltage UP = +24 VDC
- Terminals 1.9 to 4.9: process voltage ZP = 0 VDC

**Table 550: Assignment of the other terminals:**

<table>
<thead>
<tr>
<th>Terminals</th>
<th>Signal</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0 to 1.7</td>
<td>I0 to I7</td>
<td>8 digital inputs</td>
</tr>
<tr>
<td>2.0 to 2.7</td>
<td>I8 to I15</td>
<td>8 digital inputs</td>
</tr>
<tr>
<td>3.0 to 3.7</td>
<td>I16 to I23</td>
<td>8 digital inputs</td>
</tr>
<tr>
<td>4.0 to 4.7</td>
<td>I24 to I31</td>
<td>8 digital inputs</td>
</tr>
</tbody>
</table>

The internal power supply voltage for the module's circuitry is carried out via the I/O bus (provided by a bus module or a CPU). Thus, the current consumption from 24 VDC power supply at the terminals L+/UP and M/ZP of the CPU/bus module increases by 2 mA per DI524.

The external power supply connection is carried out via the UP (+24 VDC) and the ZP (0 VDC) terminals.
**WARNING!**

**Removal/Insertion under power**

The devices are not designed for removal or insertion under power. Because of unforeseeable consequences, it is not allowed to plug or unplug devices with the power being ON.

Make sure that all voltage sources (supply and process voltage) are switched off before you

- connect or disconnect any signal or terminal block
- remove, mount or replace a module.

Disconnecting any powered devices while energized in a hazardous location could result in an electric arc, which could create a flammable ignition resulting in fire or explosion.

Make sure that power is removed and that the area has been thoroughly checked to ensure that flammable materials are not present prior to proceeding.

The devices must not be opened when in operation. The same applies to the network interfaces.

---

**NOTICE!**

**Risk of damaging the PLC modules!**

Overvoltages and short circuits might damage the PLC modules.

- Make sure that all voltage sources (supply voltage and process supply voltage) are switched off before you begin with operations at the system.
- Never connect any voltages or signals to reserved terminals (marked with ---). Reserved terminals may carry internal voltages.
The process supply voltage must be included in the earthing concept (e.g. earthing of the negative pole).

The module provides several diagnosis functions. Chapter 1.5.3.6.1.2.4.7 “Diagnosis” on page 2496.

Internal data exchange

<table>
<thead>
<tr>
<th></th>
<th>Without the Fast Counter</th>
<th>With the Fast Counter (only with AC500)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Digital inputs (bytes)</td>
<td>4</td>
<td>6</td>
</tr>
<tr>
<td>Digital outputs (bytes)</td>
<td>0</td>
<td>2</td>
</tr>
</tbody>
</table>
Without the Fast Counter | With the Fast Counter (only with AC500)
---|---
Counter input data (words) | 0 | 4
Counter output data (words) | 0 | 8

**I/O configuration**

The module itself does not store configuration data. It receives its parameterization data from the master device of the I/O bus (CPU or bus module) during power-up of the system. Hence, replacing I/O modules is possible without any re-parameterization via software.

*If the external power supply voltage via UP/ZP terminals fails, the I/O module loses its configuration data. The whole station has to be switched off and on again to re-configure the module.*

**Parameterization**

<table>
<thead>
<tr>
<th>Firmware version</th>
<th>Configuration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Firmware version &gt; V2.0.0</td>
<td>The arrangement of the parameter data is performed by Control Builder Plus/ Automation Builder software.</td>
</tr>
</tbody>
</table>

The parameter data directly influences the functionality of modules. For non-standard applications, it is necessary to adapt the parameters to your system configuration.

Module: Module slot address: \( Y = 1 \ldots 10 \)

<table>
<thead>
<tr>
<th>No.</th>
<th>Name</th>
<th>Value</th>
<th>Internal value</th>
<th>Internal value, type</th>
<th>Default</th>
<th>Min.</th>
<th>Max.</th>
<th>EDS Slot/Index</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Module ID</td>
<td>Internal</td>
<td>1000 (^1)</td>
<td>Word</td>
<td>1000 0x03E8</td>
<td>0</td>
<td>65535</td>
<td>0x0Y01</td>
</tr>
<tr>
<td>2</td>
<td>Ignore module (^2)</td>
<td>No Yes</td>
<td>0 1</td>
<td>Byte</td>
<td>No 0x00</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Parameter length</td>
<td>Internal</td>
<td>3-CPU 2-FBP</td>
<td>Byte</td>
<td>3 2</td>
<td>0</td>
<td>255</td>
<td>0x0Y02</td>
</tr>
<tr>
<td>4</td>
<td>Check supply</td>
<td>Off On</td>
<td>0 1</td>
<td>Byte</td>
<td>On 0x01</td>
<td>0</td>
<td>1</td>
<td>0x0Y03</td>
</tr>
<tr>
<td>No.</td>
<td>Name</td>
<td>Value</td>
<td>Internal value</td>
<td>Internal value, type</td>
<td>Default Value</td>
<td>Min.</td>
<td>Max.</td>
<td>EDS Slot/Index</td>
</tr>
<tr>
<td>-----</td>
<td>-----------------------</td>
<td>------------------</td>
<td>----------------</td>
<td>----------------------</td>
<td>---------------</td>
<td>------</td>
<td>------</td>
<td>----------------</td>
</tr>
<tr>
<td>5</td>
<td>Input delay</td>
<td>0.1 ms</td>
<td>0</td>
<td>1</td>
<td>Byte</td>
<td>8 ms</td>
<td>0x02</td>
<td>0x0Y04</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 ms</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>8 ms</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>32 ms</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>Fast counter 4)</td>
<td>0</td>
<td>0</td>
<td></td>
<td>Byte</td>
<td>Mode 0</td>
<td></td>
<td>Not for FBP</td>
</tr>
<tr>
<td></td>
<td></td>
<td>10</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>10</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Remarks:

1) With CS31 and addresses smaller than 70 and FBP, the value is increased by 1
2) Not with FBP
3) For a description of the counter operating modes, please refer to the Fast Counter section in Chapter 1.5.3.6.1.2.9 "Fast counter on page 2543"
4) With FBP or CS31 without the parameter Fast counter

GSD file:
```
Ext_User_Prm_Data_Len = 5
Ext_User_Prm_Data_Const(0) = 0x03, 0xe9, 0x02, 0x01, 0x02;
```

Diagnosis

<table>
<thead>
<tr>
<th>E1...E4</th>
<th>d1</th>
<th>d2</th>
<th>d3</th>
<th>d4</th>
<th>Identifier 000...063</th>
<th>AC500 display</th>
<th>&lt;- Display in</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class</td>
<td>Comp</td>
<td>Dev</td>
<td>Mod</td>
<td>Ch</td>
<td>Err</td>
<td>PS501 PLC browser</td>
<td></td>
</tr>
<tr>
<td>Byte 6</td>
<td>Byte 3</td>
<td>Byte 4</td>
<td>Byte 5</td>
<td>Byte 6</td>
<td>Bit 0...5</td>
<td>FBP diagnosis block</td>
<td></td>
</tr>
<tr>
<td>Bit 6...7</td>
<td>-</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Class</td>
<td>Interface</td>
<td>Device</td>
<td>Module</td>
<td>Channel</td>
<td>Error code</td>
<td>Error message</td>
<td>Remedy</td>
</tr>
<tr>
<td></td>
<td>1)</td>
<td>2)</td>
<td>3)</td>
<td>4)</td>
<td>19</td>
<td>Checksum error in the I/O module</td>
<td>Replace I/O module</td>
</tr>
<tr>
<td>Module error</td>
<td>3</td>
<td>14</td>
<td>1...10</td>
<td>31</td>
<td>31</td>
<td>3</td>
<td>Timeout in the I/O module</td>
</tr>
<tr>
<td></td>
<td>11 / 12</td>
<td>ADR</td>
<td>1...10</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

2496 3ADR010583, 1, en_US 2020/12/10
### Remarks:

1) In AC500 the following interface identifier applies:
   
   14 = I/O bus, 11 = COM1 (e.g. CS31 bus), 12 = COM2.
   
   The FB-P diagnosis block does not contain this identifier.

2) With "Device" the following allocation applies:
   
   31 = module itself,
   
   1...10 = decentralized communication interface module 1...10,
   
   ADR = hardware address (e.g. of the DC551)

3) With "Module" the following allocation applies depending on the master:
   
   Module error: I/O bus or FBP: 31 = module itself; COM1/COM2: 1...10 = expansion 1...10

4) In case of module errors, with channel "31 = module itself" is output.

### State LEDs

During the power ON procedure, the module initializes automatically. All LEDs (except the channel LEDs) are ON during this time.
<table>
<thead>
<tr>
<th>LED</th>
<th>State</th>
<th>Color</th>
<th>LED = OFF</th>
<th>LED = ON</th>
<th>LED flashes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Digital input</td>
<td>Digital input</td>
<td>Yellow</td>
<td>Input = OFF</td>
<td>Input = ON ¹)</td>
<td>--</td>
</tr>
<tr>
<td>Process supply voltage</td>
<td>24 VDC via terminal</td>
<td>Green</td>
<td>Process supply voltage is missing</td>
<td>Process supply voltage OK</td>
<td>--</td>
</tr>
<tr>
<td>CH-ERR1</td>
<td>Channel error, error messages in groups (digital inputs combined into the groups 1, 2, 3, 4)</td>
<td>Red</td>
<td>No error or process supply voltage is missing</td>
<td>Severe error within the corresponding group</td>
<td>Error on one channel of the corresponding group</td>
</tr>
<tr>
<td>CH-ERR2</td>
<td>Module error</td>
<td>Red</td>
<td>--</td>
<td>Internal error</td>
<td>--</td>
</tr>
<tr>
<td>CH-ERR3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CH-ERR4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

¹) Indication LED is ON even if an input signal is applied to the channel and the supply voltage is off. In this case the module is not operating and does not generate an input signal.

²) All of the LEDs CH-ERR1 to CH-ERR4 light up together

### Technical data

The system data of AC500 and S500 **Chapter 1.5.4.5.1 “System data AC500” on page 3058** are valid for standard version.

The system data of AC500-XC **Chapter 1.5.4.6.1 “System data AC500-XC” on page 3099** are valid for the XC version.

Only additional details are therefore documented below.

The technical data are also valid for the XC version.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Process supply voltage UP</td>
<td></td>
</tr>
<tr>
<td>Connections</td>
<td>Terminals 1.8, 2.8, 3.8 and 4.8 for +24 V (UP) as well as 1.9, 2.9, 3.9 and 4.9 for 0 V (ZP)</td>
</tr>
<tr>
<td>Rated value</td>
<td>24 VDC</td>
</tr>
<tr>
<td>Max. ripple</td>
<td>5 %</td>
</tr>
<tr>
<td>Protection against reversed voltage</td>
<td>Yes</td>
</tr>
<tr>
<td>Rated protection fuse for UP</td>
<td>10 A fast</td>
</tr>
<tr>
<td>Galvanic isolation</td>
<td>Yes, per module</td>
</tr>
<tr>
<td>Current consumption</td>
<td></td>
</tr>
<tr>
<td>From 24 VDC power supply at the terminals UP/L+ and ZP/M of the CPU/bus module</td>
<td>ca. 2 mA</td>
</tr>
<tr>
<td>From UP at normal operation</td>
<td>0.15 A</td>
</tr>
<tr>
<td>Inrush current from UP (at power up)</td>
<td>0.008 A²s</td>
</tr>
<tr>
<td>Weight (without terminal unit)</td>
<td>ca. 105 g</td>
</tr>
</tbody>
</table>

---

2498 3ADR010583, 1, en_US 2020/12/10
### Parameter | Value
--- | ---
Mounting position | Horizontal or vertical with derating (output load reduced to 50 % at 40 °C per group)
Cooling | The natural convection cooling must not be hindered by cable ducts or other parts in the switch-gear cabinet.

**NOTICE!**

Attention:

All I/O channels (digital and analog) are protected against reverse polarity, reverse supply, short circuit and continuous overvoltage up to 30 VDC.

---

**Technical data of the digital inputs**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of channels per module</td>
<td>32</td>
</tr>
<tr>
<td>Distribution of the channels into groups</td>
<td>1 group of 32 channels</td>
</tr>
<tr>
<td>Terminals of the channels I0 to I7</td>
<td>1.0 to 1.7</td>
</tr>
<tr>
<td>Terminals of the channels I8 to I15</td>
<td>2.0 to 2.7</td>
</tr>
<tr>
<td>Terminals of the channels I16 to I23</td>
<td>3.0 to 3.7</td>
</tr>
<tr>
<td>Terminals of the channels I24 to I31</td>
<td>4.0 to 4.7</td>
</tr>
<tr>
<td>Reference potential for all inputs</td>
<td>Terminals 1.9, 2.9, 3.9 and 4.9 (negative pole of the process supply voltage, signal name ZP)</td>
</tr>
<tr>
<td>Galvanic isolation</td>
<td>From the rest of the module (I/O bus)</td>
</tr>
<tr>
<td>Indication of the input signals</td>
<td>One yellow LED per channel, the LED is ON when the input signal is high (signal 1)</td>
</tr>
<tr>
<td>Monitoring point of input indicator</td>
<td>LED is part of the input circuitry</td>
</tr>
<tr>
<td>Input type acc. to EN 61131-2</td>
<td>Type 1</td>
</tr>
<tr>
<td>Input delay (0 -&gt; 1 or 1 -&gt; 0)</td>
<td>Typ. 8 ms, configurable from 0.1 to 32 ms</td>
</tr>
<tr>
<td>Input signal voltage</td>
<td>24 VDC</td>
</tr>
<tr>
<td>Signal 0</td>
<td>-3 V...+5 V</td>
</tr>
<tr>
<td>Undefined signal</td>
<td>&gt; +5 V...&lt; +15 V</td>
</tr>
<tr>
<td>Signal 1</td>
<td>+15 V...+30 V</td>
</tr>
<tr>
<td>Ripple with signal 0</td>
<td>Within -3 V...+5 V</td>
</tr>
<tr>
<td>Ripple with signal 1</td>
<td>Within +15 V...+30 V</td>
</tr>
<tr>
<td>Input current per channel</td>
<td></td>
</tr>
<tr>
<td>Input voltage +24 V</td>
<td>Typ. 5 mA</td>
</tr>
<tr>
<td>Input voltage +5 V</td>
<td>&gt; 1 mA</td>
</tr>
<tr>
<td>Input voltage +15 V</td>
<td>&gt; 5 mA</td>
</tr>
<tr>
<td>Input voltage +30 V</td>
<td>&lt; 8 mA</td>
</tr>
<tr>
<td>Max. cable length</td>
<td></td>
</tr>
<tr>
<td>Parameter</td>
<td>Value</td>
</tr>
<tr>
<td>---------------</td>
<td>--------</td>
</tr>
<tr>
<td>Shielded</td>
<td>1000 m</td>
</tr>
<tr>
<td>Unshielded</td>
<td>600 m</td>
</tr>
</tbody>
</table>

### Technical data of the fast counter

The fast counter of the module does not work if the module is connected to a
- FBP interface module
- CS31 bus module
- CANopen bus module

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Used inputs</td>
<td>I24 / I25</td>
</tr>
<tr>
<td>Used outputs</td>
<td>None</td>
</tr>
<tr>
<td>Counting frequency</td>
<td>Max. 50 kHz</td>
</tr>
<tr>
<td>Detailed description</td>
<td>See Fast Counter Chapter 1.5.5.1.10 “Fast counters” on page 3194</td>
</tr>
<tr>
<td>Operating modes</td>
<td>See Operating modes</td>
</tr>
</tbody>
</table>

### Ordering data

<table>
<thead>
<tr>
<th>Part no.</th>
<th>Description</th>
<th>Product life cycle phase *)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1SAP 240 000 R0001</td>
<td>DI524, digital input module, 32 DI, 24 VDC, 1-wire</td>
<td>Active</td>
</tr>
<tr>
<td>1SAP 440 000 R0001</td>
<td>DI524-XC, digital input module, 32 DI, 24 VDC, 1-wire, XC version</td>
<td>Active</td>
</tr>
</tbody>
</table>

*) For planning and commissioning of new installations use modules in Active status only.

**DO524 - Digital output module**

- 32 digital outputs 24 VDC / 0.5 A in 4 groups (1.0...4.7) with short circuit and overload protection
- Module-wise electrically isolated
- XC version for use in extreme ambient conditions available
1 I/O bus
2 Allocation between terminal number and signal name
3 32 yellow LEDs to display the signal states at the digital outputs (O0 - O31)
4 1 green LED to display the state of the process supply voltage UP
5 4 red LEDs to display errors
6 Label
7 Terminal unit
8 DIN rail
9 SIGN for XC version

**Intended purpose**

The device can be used as a decentralized I/O extension module for S500 Communication Interface Modules (e.g., CI592-CS31, CI501-PNIO, CI541-DP, CI581-CN) or as a centralized extension module for AC500 CPUs (PM5xx).

The outputs are electrically isolated from all other circuitry of the module. There is no potential separation between the channels.
### Functionality

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>LED displays</td>
<td>For signal states, errors and supply voltage</td>
</tr>
<tr>
<td>Internal power supply</td>
<td>Via the expansion bus interface (I/O bus)</td>
</tr>
<tr>
<td>External power supply</td>
<td>Via the terminals ZP and UP (process voltage 24 VDC)</td>
</tr>
<tr>
<td>Required terminal unit</td>
<td>TU515 or TU516 [Chapter 1.5.3.5.2 “TU515, TU516, TU541 and TU542 for I/O modules” on page 2311]</td>
</tr>
</tbody>
</table>

The device is plugged on a terminal unit [Chapter 1.5.3.5.2 “TU515, TU516, TU541 and TU542 for I/O modules” on page 2311]. Position the module properly and press until it locks in place. The terminal unit is either mounted on a DIN rail or to the wall using 2 screws plus the additional accessory for wall mounting (TA526 [Chapter 1.5.3.8.1.4 “TA526 - Wall mounting accessory” on page 3035]).

**Multiple overloads**

No effects of multiple overloads on isolated multi-channel modules occur, as every channel is protected individually by an internal smart high-side switch.

### Electrical connection

For a detailed description of the mounting, disassembly and electrical connection of the module, please refer to the System Assembly, Construction and Connection chapter [Chapter 1.5.4.5 “AC500 (Standard)” on page 3058].

The electrical connection of the I/O channels is carried out using the 40 terminals of the I/O terminal unit. I/O modules can be replaced without re-wiring the terminal units.

The terminals 1.8 to 4.8 and 1.9 to 4.9 are electrically interconnected within the I/O terminal unit and have always the same assignment, independent of the inserted module:

- Terminals 1.8 to 4.8: process voltage UP = +24 VDC
- Terminals 1.9 to 4.9: process voltage ZP = 0 VDC

The assignment of the other terminals:

```
<table>
<thead>
<tr>
<th>Terminals</th>
<th>Signal</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0 to 1.7</td>
<td>O0 to O7</td>
<td>8 digital outputs</td>
</tr>
<tr>
<td>2.0 to 2.7</td>
<td>O8 to O15</td>
<td>8 digital outputs</td>
</tr>
<tr>
<td>3.0 to 3.7</td>
<td>O16 to O23</td>
<td>8 digital outputs</td>
</tr>
<tr>
<td>4.0 to 4.7</td>
<td>O24 to O31</td>
<td>8 digital outputs</td>
</tr>
</tbody>
</table>
```

The internal power supply voltage for the module’s circuitry is carried out via the I/O bus (provided by a bus module or a CPU). Thus, the current consumption from 24 VDC power supply at the terminals L+/UP and M/ZP of the CPU/bus module increases by 2 mA per DO524.

The external power supply connection is carried out via the UP (+24 VDC) and the ZP (0 VDC) terminals.
NOTICE!
Risk of damaging the PLC modules!
Overvoltages and short circuits might damage the PLC modules.
- Make sure that all voltage sources (supply voltage and process supply voltage) are switched off before you begin with operations at the system.
- Never connect any voltages or signals to reserved terminals (marked with ---). Reserved terminals may carry internal voltages.

The following block diagram shows the internal construction of the digital outputs:

The module provides several diagnosis functions see Chapter 1.5.3.6.1.2.5.7 “Diagnosis” on page 2505.

Internal data exchange

<table>
<thead>
<tr>
<th>Digital inputs (bytes)</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Digital outputs (bytes)</td>
<td>4</td>
</tr>
</tbody>
</table>

I/O configuration

The module itself does not store configuration data. It receives its parameterization data from the master device of the I/O bus (CPU or bus module) during power-up of the system.
Hence, replacing I/O modules is possible without any re-parameterization via software.

If the external power supply voltage via UP/ZP terminals fails, the I/O module loses its configuration data. The whole station has to be switched off and on again to re-configure the module.
Parameterization

<table>
<thead>
<tr>
<th>Firmware version</th>
<th>Configuration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Firmware version &gt; V2.0.0</td>
<td>The arrangement of the parameter data is performed by Control Builder Plus/ Automation Builder software.</td>
</tr>
</tbody>
</table>

The parameter data directly influences the functionality of modules.

For non-standard applications, it is necessary to adapt the parameters to your system configuration.

Module: Module slot address: Y = 1...10

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>Internal value</th>
<th>Internal value, type</th>
<th>Default</th>
<th>Min.</th>
<th>Max.</th>
<th>Max.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Module ID</td>
<td>Internal</td>
<td>1101</td>
<td>WORD</td>
<td>1101</td>
<td>0x044D</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ignore module 2)</td>
<td>No</td>
<td>0</td>
<td>BYTE</td>
<td>No</td>
<td>0x00</td>
<td></td>
<td>not for FBP</td>
</tr>
<tr>
<td>Parameter length</td>
<td>Internal</td>
<td>7</td>
<td>BYTE</td>
<td>7-CPU</td>
<td>0</td>
<td>255</td>
<td>0x0Y02</td>
</tr>
<tr>
<td>Check supply</td>
<td>Off</td>
<td>0</td>
<td>BYTE</td>
<td>On</td>
<td>0x01</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Output short circuit detection</td>
<td>Off</td>
<td>0</td>
<td>BYTE</td>
<td>On</td>
<td>0x01</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Behaviour of outputs at communication errors</td>
<td>Off</td>
<td>0</td>
<td>BYTE</td>
<td>Off</td>
<td>0x00</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Substitute value at outputs</td>
<td>0...4294967295</td>
<td>0...0xffffffff</td>
<td>DWORD</td>
<td>0</td>
<td>0x00000000 00</td>
<td>42949672 95</td>
<td>0x0Y06</td>
</tr>
</tbody>
</table>

1) With CS31 and addresses smaller than 70 and FBP, the value is increased by 1
2) Not with FBP

GSD file:

```plaintext
Ext_User_Prm_Data_Len =
Ext_User_Prm_Data_Const(0) =
```

10
0x04, 0x4d, 0x07, 0x01, 0x01, 0x00, 0x00, 0x00, 0x00;
Diagnosis

In case of overload or short circuit, the outputs switch off automatically and try to switch on again cyclically. Therefore, an acknowledgement of the outputs is not necessary. The LED error message, however, is stored.

<table>
<thead>
<tr>
<th>E1...E4</th>
<th>d1</th>
<th>d2</th>
<th>d3</th>
<th>d4</th>
<th>Identifier 000...063</th>
<th>AC500 display</th>
<th>&lt;- Display in</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class</td>
<td>Comp</td>
<td>Dev</td>
<td>Mod</td>
<td>Ch</td>
<td>Err</td>
<td>PS501 PLC browser</td>
<td></td>
</tr>
<tr>
<td>Byte 6</td>
<td>Bit 6...7</td>
<td>Byte 3</td>
<td>Byte 4</td>
<td>Byte 5</td>
<td>Byte 6 Bit 0...5</td>
<td>FBP diagnosis block</td>
<td></td>
</tr>
</tbody>
</table>

Class | Interface | Device | Module | Channel | Error Identifier | Error message | Remedy |
|------|-----------|--------|--------|---------|------------------|---------------|--------|
1)    |           |        |        |         |                  |               |        |
2)    |           |        |        |         |                  |               |        |
3)    |           |        |        |         |                  |               |        |
4)    |           |        |        |         |                  |               |        |

Module error

<table>
<thead>
<tr>
<th>Class</th>
<th>Interface</th>
<th>Device</th>
<th>Module</th>
<th>Channel</th>
<th>Error Identifier</th>
<th>Error message</th>
<th>Remedy</th>
</tr>
</thead>
</table>
3      | 14        | 1...10 | 31     | 31      | 19               | Checksum error in the I/O module | Replace I/O module |
|       | 11 / 12   | ADR    | 1...10 |         |                  |               |        |
3      | 14        | 1...10 | 31     | 31      | 3                | Timeout in the I/O module | Replace I/O module |
|       | 11 / 12   | ADR    | 1...10 |         |                  |               |        |
3      | 14        | 1...10 | 31     | 31      | 40               | Different hard-/firmware versions in the module | Replace I/O module |
|       | 11 / 12   | ADR    | 1...10 |         |                  |               |        |
3      | 14        | 1...10 | 31     | 31      | 43               | Internal error in the module | Replace I/O module |
|       | 11 / 12   | ADR    | 1...10 |         |                  |               |        |
3      | 14        | 1...10 | 31     | 31      | 36               | Internal data exchange failure | Replace I/O module |
|       | 11 / 12   | ADR    | 1...10 |         |                  |               |        |
3      | 14        | 1...10 | 31     | 31      | 9                | Overflow diagnosis buffer | New start |
|       | 11 / 12   | ADR    | 1...10 |         |                  |               |        |
3      | 14        | 1...10 | 31     | 31      | 26               | Parameter error | Check master |
|       | 11 / 12   | ADR    | 1...10 |         |                  |               |        |
3      | 14        | 1...10 | 31     | 31      | 11               | Process voltage too low | Check process voltage |
|       | 11 / 12   | ADR    | 1...10 |         |                  |               |        |
4      | 14        | 1...10 | 2      | 0...31   | 47               | Short circuit at a digital output | Check connection |
|       | 11 / 12   | ADR    | 1...10 |         |                  |               |        |

Channel error

<table>
<thead>
<tr>
<th>Class</th>
<th>Interface</th>
<th>Device</th>
<th>Channel</th>
<th>Error Identifier</th>
<th>Error message</th>
<th>Remedy</th>
</tr>
</thead>
</table>
4      | 14        | 1...10 | 2       | 0...31           | 47            | Short circuit at a digital output | Check connection |
|       | 11 / 12   | ADR    | 1...10 |                  |               |        |
1) In AC500 the following interface identifier applies:
   14 = I/O bus, 11 = COM1 (e.g. CS31 bus), 12 = COM2.
The FBP diagnosis block does not contain this identifier.

2) With "Device" the following allocation applies:
   31 = module itself, 1...10 = decentralized communication interface module
   1...10, ADR = Hardware address (e.g. of the DC551)

3) With "Module" the following allocation applies dependent of the master:
   Module error: I/O bus or FBP: 31 = module itself; COM1/COM2: 1...10 =
   expansion 1...10
   Channel error: I/O bus or FBP = module type (4 = DC); COM1/COM2:
   1...10 = expansion 1...10

4) In case of module errors, with channel "31 = Module itself" is output.

State LEDs

During the power ON procedure, the module initializes automatically. All LEDs (except the channel LEDs) are ON during this time.

<table>
<thead>
<tr>
<th>LED</th>
<th>State</th>
<th>Color</th>
<th>LED = OFF</th>
<th>LED = ON</th>
<th>LED flashes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Outputs</td>
<td>Digital output</td>
<td>Yellow</td>
<td>Output = OFF</td>
<td>Output = ON</td>
<td>--</td>
</tr>
<tr>
<td>UP</td>
<td>Process supply voltage 24 VDC via terminal</td>
<td>Green</td>
<td>Process supply voltage is missing</td>
<td>Process supply voltage OK</td>
<td>--</td>
</tr>
<tr>
<td>CH-ERR1</td>
<td>Channel error, error messages in groups (digital outputs combined into the groups 1, 2, 3, 4)</td>
<td>Red</td>
<td>No error or process supply voltage is missing</td>
<td>Severe error within the corresponding group</td>
<td>Error on one channel of the corresponding group (e.g. short circuit at an output)</td>
</tr>
<tr>
<td>CH-ERR2</td>
<td></td>
<td>Red</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CH-ERR3</td>
<td></td>
<td>Red</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CH-ERR4</td>
<td></td>
<td>Red</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CH-ERR *)</td>
<td>Module error</td>
<td>Red</td>
<td>--</td>
<td>Internal error</td>
<td>--</td>
</tr>
</tbody>
</table>

*) All of the LEDs CH-ERR1 to CH-ERR4 light up together

Technical data

The system data of AC500 and S500 © Chapter 1.5.4.5.1 “System data AC500” on page 3058 are valid for standard version.

The system data of AC500-XC © Chapter 1.5.4.6.1 “System data AC500-XC” on page 3099 are valid for the XC version.

Only additional details are therefore documented below.
The technical data are also valid for the XC version.
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Process supply voltage UP</td>
<td></td>
</tr>
<tr>
<td>Connections</td>
<td>Terminals 1.8, 2.8, 3.8 and 4.8 for +24 V (UP) as well as 1.9, 2.9, 3.9 and 4.9 for 0 V (ZP)</td>
</tr>
<tr>
<td>Rated value</td>
<td>24 VDC</td>
</tr>
<tr>
<td>Max. ripple</td>
<td>5 %</td>
</tr>
<tr>
<td>Protection against reversed voltage</td>
<td>Yes</td>
</tr>
<tr>
<td>Rated protection fuse on UP</td>
<td>10 A fast</td>
</tr>
<tr>
<td>Galvanic isolation</td>
<td>Yes, per module</td>
</tr>
<tr>
<td>Current consumption</td>
<td></td>
</tr>
<tr>
<td>From 24 VDC power supply at the terminals UP/L+ and ZP/M of the CPU/bus module</td>
<td>Ca. 2 mA</td>
</tr>
<tr>
<td>From UP at normal operation / with outputs</td>
<td>0.10 A + max. 0.5 A per output</td>
</tr>
<tr>
<td>Inrush current from UP (at power up)</td>
<td>0.005 A^2s</td>
</tr>
<tr>
<td>Max. power dissipation within the module</td>
<td>6 W (outputs unloaded)</td>
</tr>
<tr>
<td>Weight (without terminal unit)</td>
<td>Ca. 100 g</td>
</tr>
<tr>
<td>Mounting position</td>
<td>Horizontal</td>
</tr>
<tr>
<td></td>
<td>Or vertical with derating (output load reduced to 50 % at 40 °C per group)</td>
</tr>
<tr>
<td>Cooling</td>
<td>The natural convection cooling must not be hindered by cable ducts or other parts in the switch-gear cabinet.</td>
</tr>
</tbody>
</table>

**NOTICE!**

**Attention:**

All I/O channels (digital and analog) are protected against reverse polarity, reverse supply, short circuit and continuous overvoltage up to 30 VDC.

**Multiple overloads**

No effects of multiple overloads on isolated multi-channel modules occur, as every channel is protected individually by an internal smart high-side switch.

---

### Technical data of the digital outputs

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of channels per module</td>
<td>32 outputs (with transistors)</td>
</tr>
<tr>
<td>Distribution of the channels into groups</td>
<td>1 group of 32 channels</td>
</tr>
<tr>
<td>Connection of the channels</td>
<td></td>
</tr>
<tr>
<td>O0 to O7</td>
<td>Terminals 1.0 to 1.7</td>
</tr>
<tr>
<td>O8 to O15</td>
<td>Terminals 2.0 to 2.7</td>
</tr>
<tr>
<td>O16 to O23</td>
<td>Terminals 3.0 to 3.7</td>
</tr>
<tr>
<td>Parameter</td>
<td>Value</td>
</tr>
<tr>
<td>-----------------------------------------------</td>
<td>----------------------------------------------------------------------</td>
</tr>
<tr>
<td>O24 to O31</td>
<td>Terminals 4.0 to 4.7</td>
</tr>
<tr>
<td>Indication of the output signals</td>
<td>1 yellow LED per channel, the LED is ON if the output signal is high (signal 1)</td>
</tr>
<tr>
<td>Reference potential for all outputs</td>
<td>Terminals 1.9, 2.9, 3.9 and 4.9 (negative pole of the process supply voltage, signal name ZP)</td>
</tr>
<tr>
<td>Common power supply voltage</td>
<td>For all outputs: terminals 1.8, 2.8, 3.8 and 4.8 (positive pole of the process supply voltage, signal name UP)</td>
</tr>
<tr>
<td>Output voltage for signal 1</td>
<td>UP (-0.8 V)</td>
</tr>
<tr>
<td>Output delay (0 -&gt; 1 or 1 -&gt; 0)</td>
<td>On request</td>
</tr>
<tr>
<td>Output current</td>
<td></td>
</tr>
<tr>
<td>Rated value, per channel</td>
<td>500 mA at UP = 24 V</td>
</tr>
<tr>
<td>Maximum value (channels O0 to O15)</td>
<td>4 A</td>
</tr>
<tr>
<td>Maximum value (channels O16 to O31)</td>
<td>4 A</td>
</tr>
<tr>
<td>Maximum value (all channels together)</td>
<td>8 A</td>
</tr>
<tr>
<td>Max. leakage current with signal 0</td>
<td>&lt; 0.5 mA</td>
</tr>
<tr>
<td>Rated protection fuse on UP</td>
<td>10 A fast</td>
</tr>
<tr>
<td>Demagnetization when inductive loads are switched off</td>
<td>With varistors integrated in the module (see figure below)</td>
</tr>
<tr>
<td>Switching frequency</td>
<td></td>
</tr>
<tr>
<td>With resistive load</td>
<td>On request</td>
</tr>
<tr>
<td>With inductive loads</td>
<td>Max. 0.5 Hz</td>
</tr>
<tr>
<td>With lamp loads</td>
<td>Max. 11 Hz with max. 5 W</td>
</tr>
<tr>
<td>Short-circuit proof / overload proof</td>
<td>Yes</td>
</tr>
<tr>
<td>Overload message (I &gt; 0.7 A)</td>
<td>Yes, after ca. 100 ms</td>
</tr>
<tr>
<td>Output current limitation</td>
<td>Yes, automatic reactivation after short-circuit/overload</td>
</tr>
<tr>
<td>Resistance to feedback against 24 V signals</td>
<td>Yes</td>
</tr>
<tr>
<td>Max. cable length</td>
<td></td>
</tr>
<tr>
<td>Shielded</td>
<td>1000 m</td>
</tr>
<tr>
<td>Unshielded</td>
<td>600 m</td>
</tr>
</tbody>
</table>

The following drawing shows the circuitry of a digital output with the varistors for demagnetization when inductive loads are switched off.
### Ordering data

<table>
<thead>
<tr>
<th>Part no.</th>
<th>Description</th>
<th>Product life cycle phase *)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1SAP 240 700 R0001</td>
<td>DO524, digital output module, 32 DO, 24 VDC / 0.5 A, 1-wire</td>
<td>Active</td>
</tr>
<tr>
<td>1SAP 440 700 R0001</td>
<td>DO524-XC, digital output module, 32 DO, 24 VDC / 0.5 A, 1-wire, XC version</td>
<td>Active</td>
</tr>
</tbody>
</table>

*) For planning and commissioning of new installations use modules in Active status only.

### DO526 - Digital output module

- 8 digital outputs 24 VDC (O0 to O7) in 2 groups without short circuit and without overload protection.
- Module and group-wise electrically isolated
- XC version for use in extreme ambient conditions available
Fig. 115: DO526-XC, plugged on a terminal unit TU542-XC

1 I/O bus
2 Allocation between terminal number and signal name
3 8 yellow LEDs to display the signal states of the outputs O0 to O7
4 3 green LEDs to display the states of the process supply voltage UP, UP3 and UP4
5 2 red LEDs to display errors
6 Label
7 Terminal unit
8 DIN-rail
9 Sign for XC version

Intended purpose

The device can be used as a decentralized I/O extension module for S500 Communication Interface Modules (e.g. CI592-CS31, CI501-PNIO, CI541-DP, CI581-CN) or as a centralized extension module for AC500 CPUs (PM5xx).

The outputs are group-wise electrically isolated from each other.
All other circuitry of the module is electrically isolated from the outputs.
Potential separation between the channel groups.
Functionality

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>LED displays</td>
<td>For signal states, errors and supply voltages</td>
</tr>
<tr>
<td>Internal power supply</td>
<td>Via I/O bus</td>
</tr>
<tr>
<td>External power supply</td>
<td>Via the terminals ZP, ZP3, UP, UP3 and UP4 (process voltage 24 VDC)</td>
</tr>
<tr>
<td>Required terminal unit</td>
<td>TU542 § Chapter 1.5.3.5.2 “TU515, TU516, TU541 and TU542 for I/O modules” on page 2311</td>
</tr>
</tbody>
</table>

The output module is plugged on the terminal unit TU542. Properly position the module and press until it locks in place. The terminal unit is mounted on a DIN rail or with 2 screws plus the additional accessory for wall mounting (TA526 § Chapter 1.5.3.8.1.4 “TA526 - Wall mounting accessory” on page 3035).

Electrical connection

For a detailed description of the mounting, disassembly and electrical connection of the module, please refer to the System Assembly, Construction and Connection chapter § Chapter 1.5.4.5 “AC500 (Standard)” on page 3058.

The electrical connection of the I/O channels is carried out using the 40 terminals of the I/O terminal unit. I/O modules can be replaced without re-wiring the terminal units.

The terminals 1.8 to 2.8 and 1.9 to 2.9 are electrically interconnected within the I/O terminal unit and always have the same assignment, irrespective of the inserted module:

<table>
<thead>
<tr>
<th>Terminals 1.8 to 2.8:</th>
<th>Process voltage UP = +24 VDC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Terminals 1.9 to 2.9:</td>
<td>Process voltage ZP = 0 V</td>
</tr>
<tr>
<td>Terminal 3.8:</td>
<td>Process voltage UP3 = +24 VDC</td>
</tr>
<tr>
<td>Terminal 3.9:</td>
<td>Process voltage ZP3 = 0 V</td>
</tr>
<tr>
<td>Terminal 4.8:</td>
<td>Process voltage UP4 = +24 VDC</td>
</tr>
<tr>
<td>Terminal 4.9:</td>
<td>Process voltage ZP4 = 0 V</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Terminals 3.0, 3.1, 3.4, 3.5</th>
<th>Signal</th>
<th>4 digital outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.0, 4.1, 4.4, 4.5</td>
<td>O4 to O7</td>
<td>4 digital outputs</td>
</tr>
</tbody>
</table>

The internal power supply voltage for the module's circuitry is carried out via the I/O bus (provided by a bus Module or a CPU). Thus, the current consumption from 24 VDC power supply at the terminals L+/UP and M/ZP of the CPU/bus module increases by 2 mA per DO526.

The external power supply connection is carried out via the UP, UP3, UP4 (+24 VDC) and the ZP, ZP3, ZP4 (0 VDC) terminals.
**NOTICE!**

**Risk of damaging the PLC modules!**

Overvoltages and short circuits might damage the PLC modules.

- Make sure that all voltage sources (supply voltage and process supply voltage) are switched off before you begin with operations at the system.
- Never connect any voltages or signals to reserved terminals (marked with ---). Reserved terminals may carry internal voltages.

The following block diagram shows the internal construction of the digital outputs:

```
1.0 1.1 1.2 1.3 1.4 1.5 1.6 1.7 1.8 UP +24V 1.9 ZP 0V 2.0 2.1 2.2 2.3 2.4 2.5 2.6 2.7 2.8 UP +24V 3.0 O0 3.1 O1 3.2 3.3 3.4 O2 3.5 O3 3.6 3.7 3.8 UP3 +24V 3.9 ZP3 0V 4.0 O4 4.1 O5 4.2 4.3 4.4 O6 4.5 O7 4.6 4.7 4.8 UP4 +24V 4.9 ZP4 0V
```
1 I/O bus
2 4.0 - 4.7: Connected with UP (switch) -> Input;
   Connected with ZP (load) -> Output
3 Switch-gear cabinet earth

CAUTION!
The process supply voltage must be included in the earthing concept (e. g. earthing of the negative pole).

The module provides several diagnosis functions ⇨ Chapter 1.5.3.6.1.2.6.7 “Diagnosis” on page 2515.
Internal data exchange

<table>
<thead>
<tr>
<th>Digital inputs (bytes)</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Digital outputs (bytes)</td>
<td>1</td>
</tr>
</tbody>
</table>

I/O configuration

The module itself does not store configuration data. It receives its parameterization data from the master device of the I/O bus (CPU or bus module) during power-up of the system. Hence, replacing I/O modules is possible without any re-parameterization via software.

If the external power supply voltage via UP/ZP terminals fails, the I/O module loses its configuration data. The whole station has to be switched off and on again to re-configure the module.

Parameterization

The arrangement of the parameter data is performed with Automation Builder software, versions ≥ 1.2.3.

The parameter data directly influences the functionality of modules.

For non-standard applications, it is necessary to adapt the parameters to your system configuration.

Module: Module slot address: Y = 1...7

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>Internal value</th>
<th>Internal value, type</th>
<th>Default</th>
<th>Min.</th>
<th>Max.</th>
<th>Max.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Module ID</td>
<td>Internal</td>
<td>1105</td>
<td>WORD</td>
<td>1105</td>
<td>0</td>
<td>65535</td>
<td>0x0Y01</td>
</tr>
<tr>
<td>Ignore module</td>
<td>No</td>
<td>0</td>
<td>BYTE</td>
<td>No</td>
<td>0</td>
<td></td>
<td>not for FBP</td>
</tr>
<tr>
<td></td>
<td>Yes</td>
<td>1</td>
<td></td>
<td>0x00</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Parameter length</td>
<td>Internal</td>
<td>6</td>
<td>BYTE</td>
<td>6-CPU</td>
<td>0</td>
<td>6</td>
<td>0x0Y02</td>
</tr>
<tr>
<td>Check supply</td>
<td>Off</td>
<td>0</td>
<td>BYTE</td>
<td>On</td>
<td>0</td>
<td>1</td>
<td>0x0Y03</td>
</tr>
<tr>
<td></td>
<td>on</td>
<td>1</td>
<td></td>
<td>0x01</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Reserve</td>
<td>0...255</td>
<td>0...0xff</td>
<td>BYTE</td>
<td>On</td>
<td>0</td>
<td>1</td>
<td>0x0Y04</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0x01</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Behaviour of outputs at communica-tion errors</td>
<td>Off</td>
<td>0</td>
<td>BYTE</td>
<td>Off</td>
<td>0</td>
<td>2</td>
<td>0x0Y05</td>
</tr>
<tr>
<td></td>
<td>Last value</td>
<td>1+(n*5)</td>
<td></td>
<td>0x00</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Substitute value</td>
<td>2+(n*5), n ≤ 2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Name</td>
<td>Value</td>
<td>Internal value</td>
<td>Internal value, type</td>
<td>Default</td>
<td>Min.</td>
<td>Max.</td>
<td>Max.</td>
</tr>
<tr>
<td>-------------------------------------------</td>
<td>-------</td>
<td>----------------</td>
<td>----------------------</td>
<td>---------</td>
<td>------</td>
<td>------</td>
<td>------</td>
</tr>
<tr>
<td>Substitute value at outputs</td>
<td>0...255</td>
<td>0...0xff</td>
<td>BYTE</td>
<td>0x00</td>
<td>0</td>
<td>255</td>
<td>0x0Y06</td>
</tr>
<tr>
<td>Bit 7 = Output 7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bit 0 = Output 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Reserve</td>
<td>0...255</td>
<td>0...0xff</td>
<td>BYTE</td>
<td>0x00</td>
<td>0</td>
<td>255</td>
<td>0x0Y07</td>
</tr>
<tr>
<td>Reserve</td>
<td>0...255</td>
<td>0...0xff</td>
<td>BYTE</td>
<td>0x00</td>
<td>0</td>
<td>255</td>
<td>0x0Y08</td>
</tr>
</tbody>
</table>

1) With CS31 and addresses smaller than 70 and FBP, the value is increased by 1
2) Not with FBP

GSD file:

Ext_User_Prm_Data_Len = 10
Ext_User_Prm_Data_Const(0) = 0x04, 0x51, 0x00, 0x06, 0x01, 0x01, 0x00, 0x00, 0x00

Diagnosis

<table>
<thead>
<tr>
<th>E1...E4</th>
<th>d1</th>
<th>d2</th>
<th>d3</th>
<th>d4</th>
<th>Identifier 000...063</th>
<th>AC500 display</th>
<th>← Display in</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class</td>
<td>Comp</td>
<td>Dev</td>
<td>Mod</td>
<td>Ch</td>
<td>Err</td>
<td>PS501 PLC browser</td>
<td></td>
</tr>
<tr>
<td>Byte 6</td>
<td>-</td>
<td>Byte 3</td>
<td>Byte 4</td>
<td>Byte 5</td>
<td>Byte 6 Bit 0...5</td>
<td>FBP diagnosis block</td>
<td></td>
</tr>
<tr>
<td>Bit 6...7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Class Interface Device Module Channel Error Identifier Error message Remedy
1) 2) 3) 4) Module error

| 3 | 14 | 1...10 | 31 | 31 | 19 | Checksum error in the I/O module | Replace I/O module |
| 11 / 12 | ADR | 1...10 | | | | | |
| 3 | 14 | 1...10 | 31 | 31 | 3 | Timeout in the I/O module | Replace I/O module |
| 11 / 12 | ADR | 1...10 | | | | | |
| 3 | 14 | 1...10 | 31 | 31 | 40 | Different hard-/firmware versions in the module | Replace I/O module |
| 11 / 12 | ADR | 1...10 | | | | | |
| 3 | 14 | 1...10 | 31 | 31 | 43 | Internal error in the module | Replace I/O module |
| 11 / 12 | ADR | 1...10 | | | | | |
| 3 | 14 | 1...10 | 31 | 31 | 36 | Internal data exchange failure | Replace I/O module |
### PLC Automation with V3 CPUs

#### PLC integration > Device specifications

<table>
<thead>
<tr>
<th>Class</th>
<th>Interface</th>
<th>Device</th>
<th>Module</th>
<th>Channel</th>
<th>Error Identifier</th>
<th>Error message</th>
<th>Remedy</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>14</td>
<td>1...10</td>
<td>31</td>
<td>31</td>
<td>9</td>
<td>Overflow diagnosis buffer</td>
<td>New start</td>
</tr>
<tr>
<td></td>
<td>11 / 12</td>
<td>ADR</td>
<td>1...10</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>14</td>
<td>1...10</td>
<td>31</td>
<td>31</td>
<td>26</td>
<td>Parameter error</td>
<td>Check master</td>
</tr>
<tr>
<td></td>
<td>11 / 12</td>
<td>ADR</td>
<td>1...10</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>14</td>
<td>1...10</td>
<td>31</td>
<td>31</td>
<td>11</td>
<td>Process voltage UP3 and/or UP4 too low</td>
<td>Check process voltage</td>
</tr>
<tr>
<td></td>
<td>11 / 12</td>
<td>ADR</td>
<td>1...10</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>14</td>
<td>1...10</td>
<td>31</td>
<td>31</td>
<td>45</td>
<td>Process voltage UP is switched off (ON -&gt; OFF)</td>
<td>Process voltage ON</td>
</tr>
<tr>
<td></td>
<td>11 / 12</td>
<td>ADR</td>
<td>1...10</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### Channel error

<table>
<thead>
<tr>
<th>Class</th>
<th>Interface</th>
<th>Device</th>
<th>Module</th>
<th>Channel</th>
<th>Error Identifier</th>
<th>Error message</th>
<th>Remedy</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>14</td>
<td>1...10</td>
<td>31</td>
<td>0(UP3)</td>
<td>11</td>
<td>Process voltage too low</td>
<td>Check process voltage</td>
</tr>
<tr>
<td></td>
<td>11 / 12</td>
<td>ADR</td>
<td>1...10</td>
<td>4(UP4)</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Remarks:

1) In AC500 the following interface identifier applies:
   - 14 = I/O bus
   - 11 = COM1 (e.g. CS31-Bus), 12 = COM2.
   - The FBP diagnosis block does not contain this identifier.

2) With "Device" the following allocation applies:
   - 31 = module itself
   - 1...10 = decentralized communication interface module 1...10, ADR = hardware address (e.g. of the DC551-CS31)

3) With "Module" the following allocation applies depending on the master:
   - Module error: I/O bus or FBP: 31 = module itself; COM1/COM2: 1...10 = expansion 1...10
   - Channel error: I/O bus or FBP = module type (2 = DO); COM1/COM2: 1...10 = expansion 1...10

4) In case of module errors, with channel "31 = Module itself" is output.

### State LEDs

During the power ON procedure, the module initializes automatically. All LEDs (except the channel LEDs) are ON during this time.
### Technical data

The system data of AC500 and S500 % Chapter 1.5.4.5.1 “System data AC500” on page 3058 are valid for standard version.

The system data of AC500-XC % Chapter 1.5.4.6.1 “System data AC500-XC” on page 3099 are valid for the XC version.

Only additional details are therefore documented below.

The technical data are also valid for the XC version.

#### Parameter | Value
---|---
Process supply voltage UP, UP3 and UP4 | Terminals 1.8 and 2.8 for +24 V (UP) as well as 1.9 and 2.9 0 V (ZP), Terminals 3.8 for +24 V (UP3) as well as 3.9 for 0 V (ZP3), Terminals 4.8 for +24 V (UP4) as well as 4.9 for 0 V (ZP4)
Connections |
### Parameter Value

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rated value</td>
<td>24 VDC</td>
</tr>
<tr>
<td>Max. ripple</td>
<td>5 %</td>
</tr>
<tr>
<td>Protection against reversed voltage</td>
<td>Yes</td>
</tr>
<tr>
<td>Rated protection fuse on UP, UP3 and UP4</td>
<td>10 A fast (for each process supply voltage)</td>
</tr>
<tr>
<td>Galvanic isolation</td>
<td>Yes, per module and per output channel groups</td>
</tr>
</tbody>
</table>

#### Current consumption

<table>
<thead>
<tr>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>From 24 VDC power supply at the terminals UP/L+ and ZP/M of the CPU/bus module</td>
<td>Ca. 2 mA</td>
</tr>
<tr>
<td>From UP at normal operation / with outputs</td>
<td>Ca. 20 mA + 1.5 mA per output</td>
</tr>
<tr>
<td>From UP3 or UP4 at normal operation / with outputs</td>
<td>Ca. 0.01 A + max. 2 A per output</td>
</tr>
<tr>
<td>Inrush current from UP (at power up)</td>
<td>0.015 A(^2)s</td>
</tr>
<tr>
<td>Inrush current from UP3 or UP4 (at power up)</td>
<td>0.005 A(^2)s (without output load)</td>
</tr>
<tr>
<td>Max. power dissipation within the module</td>
<td>6 W</td>
</tr>
<tr>
<td>Weight (without terminal unit)</td>
<td>Ca. 135 g</td>
</tr>
<tr>
<td>Mounting position</td>
<td>Horizontal Or vertical with derating (output load reduced to 50 % at 40 °C per group)</td>
</tr>
<tr>
<td>Cooling</td>
<td>The natural convection cooling must not be hindered by cable ducts or other parts in the switch-gear cabinet.</td>
</tr>
</tbody>
</table>

### NOTICE!

**Attention:**

All I/O channels (digital and analog) are protected against reverse polarity, reverse supply and continuous overvoltage up to 30 VDC.

### No effects of multiple overloads

No effects of multiple overloads on isolated multi-channel modules occur, as every channel is protected individually by an external fuse.

### Technical data of the digital outputs

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of channels per module</td>
<td>8 outputs (with transistors, non-latching type)</td>
</tr>
<tr>
<td>Distribution of the channels into groups</td>
<td>2 groups of 4 channels</td>
</tr>
<tr>
<td>Connection of the channels</td>
<td></td>
</tr>
<tr>
<td>O0 to O3</td>
<td>Terminals 3.0, 3.1, 3.4, 3.5</td>
</tr>
<tr>
<td>O4 to O7</td>
<td>Terminals 4.0, 4.1, 4.4, 4.5</td>
</tr>
<tr>
<td>Indication of the output signals</td>
<td>1 yellow LED per channel, the LED is ON if the output signal is high (signal 1)</td>
</tr>
<tr>
<td>Parameter</td>
<td>Value</td>
</tr>
<tr>
<td>---------------------------------------------------------------------------</td>
<td>----------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Power supply voltage for the module</td>
<td>Terminals 1.8 and 2.8 (positive pole of the process supply voltage, signal name UP)</td>
</tr>
<tr>
<td>Reference potential for module power supply</td>
<td>Terminals 1.9 and 2.9 (negative pole of the process supply voltage, signal name ZP)</td>
</tr>
<tr>
<td>Power supply voltage for the outputs O0 to O3</td>
<td>Terminal 3.8 (positive pole of the process supply voltage, signal name UP3)</td>
</tr>
<tr>
<td>Reference potential for the outputs O0 to O3</td>
<td>Terminal 3.9 (negative pole of the process supply voltage, signal name ZP3)</td>
</tr>
<tr>
<td>Power supply voltage for the outputs O4 to O7</td>
<td>Terminal 4.8 (positive pole of the process supply voltage, signal name UP4)</td>
</tr>
<tr>
<td>Reference potential for the outputs O4 to O7</td>
<td>Terminal 4.9 (negative pole of the process supply voltage, signal name ZP4)</td>
</tr>
<tr>
<td>Output voltage for signal 1</td>
<td>UP (-0.4 V)</td>
</tr>
<tr>
<td>Output delay (0-&gt;1 or 1-&gt;0)</td>
<td>On request</td>
</tr>
<tr>
<td>Output current</td>
<td></td>
</tr>
<tr>
<td>Rated value, per channel</td>
<td>2 A at UP3 or UP4 = 24 V</td>
</tr>
<tr>
<td>Maximum value (channels O0 to O3)</td>
<td>8 A</td>
</tr>
<tr>
<td>Maximum value (channels O4 to O7)</td>
<td>8 A</td>
</tr>
<tr>
<td>Leakage current with signal 0</td>
<td>&lt; 0.1 mA</td>
</tr>
<tr>
<td>Rated protection fuse on UP</td>
<td>10 A fast</td>
</tr>
<tr>
<td>Demagnetization when inductive loads are switched off</td>
<td>With clamp diode in output high side driver</td>
</tr>
<tr>
<td>Switching frequency</td>
<td></td>
</tr>
<tr>
<td>With resistive load</td>
<td>On request</td>
</tr>
<tr>
<td>With inductive loads</td>
<td>Max. 2 Hz</td>
</tr>
<tr>
<td>With lamp loads</td>
<td>Max. 11 Hz with max. 48 W</td>
</tr>
<tr>
<td>Short-circuit proof / overload proof</td>
<td>No (should be done externally)</td>
</tr>
<tr>
<td>Overload message</td>
<td>No</td>
</tr>
<tr>
<td>Output current limitation</td>
<td>No (should be done externally)</td>
</tr>
<tr>
<td>Resistance to feedback against 24 V signals</td>
<td>Yes to UP3 or UP4. No to outputs in same group.</td>
</tr>
<tr>
<td>Max. cable length</td>
<td></td>
</tr>
<tr>
<td>Shielded</td>
<td>1000 m</td>
</tr>
<tr>
<td>Unshielded</td>
<td>600 m</td>
</tr>
</tbody>
</table>

### Ordering data

<table>
<thead>
<tr>
<th>Part no.</th>
<th>Description</th>
<th>Product life cycle phase *)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1SAP 240 800 R0001</td>
<td>DO526, digital output module, 8 DO, 24 VDC / 2 A, 1-wire</td>
<td>Active</td>
</tr>
<tr>
<td>1SAP 440 800 R0001</td>
<td>DO526-XC, digital output module, 8 DO, 24 VDC / 2 A, 1-wire, XC version</td>
<td>Active</td>
</tr>
<tr>
<td>Part no.</td>
<td>Description</td>
<td>Product life cycle phase *)</td>
</tr>
<tr>
<td>------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>------------------------------</td>
</tr>
<tr>
<td>1SAP 213 200 R0001</td>
<td>TU542, I/O terminal unit, 24 VDC, spring terminals</td>
<td>Active</td>
</tr>
<tr>
<td>1SAP 413 200 R0001</td>
<td>TU542-XC, I/O terminal unit, 24 VDC, spring terminals, XC version</td>
<td>Active</td>
</tr>
</tbody>
</table>

**DX522 - Digital input/output module**

- 8 digital inputs 24 VDC, module-wise electrically isolated
- 8 relay outputs
- Fast counter
- XC version for use in extreme ambient conditions available

---

1. I/O bus
2. Allocation between terminal number and signal name
3. 8 yellow LEDs to display the signal states at the digital inputs (I0 - I7)
4. 8 yellow LEDs to display the signal states at the digital relay outputs (R0 - R7)
5. 1 green LED to display the state of the process supply voltage UP
6. 2 red LEDs to display errors
7. Label
8. Terminal unit
9. DIN rail
10. Sign for XC version
Intended purpose

The device can be used as a decentralized I/O extension module for S500 Communication Interface Modules (e.g. CI592-CS31, CI501-PNIO, CI541-DP, CI581-CN) or as a centralized extension module for AC500 CPUs (PM5xx).

Digital configurable input/output unit.

- 8 digital inputs 24 VDC in 1 group (1.0...1.7)
- 8 digital relay outputs with one switch-over contact each (R0...R7). All output channels are electrically isolated from each other.
- Fast counter

The configuration is performed by software. The modules are supplied with a process supply voltage of 24 VDC.

All available inputs/outputs are electrically isolated from all other circuitry of the module. There is no potential separation between the channels within the same group.

For use in extreme ambient conditions (e.g. wider temperature and humidity range), a special XC version of the device is available.

Functionality

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fast counter</td>
<td>Integrated, many configurable operating modes (only with AC500)</td>
</tr>
<tr>
<td>LED displays</td>
<td>For signal states, errors and supply voltage</td>
</tr>
<tr>
<td>Internal power supply</td>
<td>Through the expansion bus interface (I/O bus)</td>
</tr>
<tr>
<td>External power supply</td>
<td>Via the terminals ZP and UP (process supply voltage 24 VDC)</td>
</tr>
<tr>
<td>Required terminal units</td>
<td>TU531 or TU532 (\text{\textcopyright Chapter 1.5.3.5.4 &quot;TU531 and TU532 for I/O modules&quot; on page 2320})</td>
</tr>
</tbody>
</table>

The device is plugged on a terminal unit \(\text{\textcopyright Chapter 1.5.3.5.4 "TU531 and TU532 for I/O modules" on page 2320}\). Position the module properly and press until it locks in place. The terminal unit is either mounted on a DIN rail or to the wall using 2 screws plus the additional accessory for wall mounting (TA526 \(\text{\textcopyright Chapter 1.5.3.8.1.4 "TA526 - Wall mounting accessory" on page 3035}\)).

Electrical connection

WARNING!

Risk of death by electric shock!

Hazardous voltages can be present at the terminals of the module.

Make sure that all voltage sources (supply voltage and process supply voltage) are switched off before you begin with operations at the system.
For a detailed description of the mounting, disassembly and electrical connection of the module, please refer to the System Assembly, Construction and Connection chapter § Chapter 1.5.4.5 “AC500 (Standard)” on page 3058.

The electrical connection of the I/O channels is carried out using the 40 terminals of the I/O terminal unit. I/O modules can be replaced without re-wiring the terminal units.

The terminals 1.8 to 4.8 and 1.9 to 4.9 are electrically interconnected within the I/O terminal unit and have always the same assignment, irrespective of the inserted module:

- Terminals 1.8 to 4.8: process supply voltage UP = +24 VDC
- Terminals 1.9 to 4.9: process supply voltage ZP = 0 VDC

Table 551: Assignment of the other terminals:

<table>
<thead>
<tr>
<th>Terminals</th>
<th>Signal</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0 to 1.7</td>
<td>I0 to I7</td>
<td>Input signals of the 8 digital inputs</td>
</tr>
<tr>
<td>1.8 to 4.8</td>
<td>UP</td>
<td>Process supply voltage +24 VDC</td>
</tr>
<tr>
<td>1.9 to 4.9</td>
<td>ZP</td>
<td>Reference potential for the 8 digital inputs and the process supply voltage</td>
</tr>
<tr>
<td>2.0</td>
<td>R0</td>
<td>Common contact of the first relay output</td>
</tr>
<tr>
<td>3.0</td>
<td>NO 0</td>
<td>Normally-open contact of the first relay output</td>
</tr>
<tr>
<td>4.0</td>
<td>NC 0</td>
<td>Normally-closed contact of the first relay output</td>
</tr>
<tr>
<td>2.1</td>
<td>R1</td>
<td>Common contact of the second relay output</td>
</tr>
<tr>
<td>3.1</td>
<td>NO 1</td>
<td>Normally-open contact of the second relay output</td>
</tr>
<tr>
<td>4.1</td>
<td>NC 1</td>
<td>Normally-closed contact of the second relay output</td>
</tr>
<tr>
<td>2.7</td>
<td>R7</td>
<td>Common contact of the eighth relay output</td>
</tr>
<tr>
<td>3.7</td>
<td>NO 7</td>
<td>Normally-open contact of the eighth relay output</td>
</tr>
<tr>
<td>4.7</td>
<td>NC 7</td>
<td>Normally-closed contact of the eighth relay output</td>
</tr>
</tbody>
</table>

The internal power supply voltage for the module’s circuitry is carried out via the I/O bus (provided by a bus module or a CPU). Thus, the current consumption from 24 VDC power supply at the terminals L+/UP and M/ZP of the CPU/bus module increases by 2 mA per DX522.

The external power supply connection is carried out via the UP (+24 VDC) and the ZP (0 VDC) terminals.
**NOTICE!**

**Risk of damaging the PLC modules!**

Overvoltages and short circuits might damage the PLC modules.
- Make sure that all voltage sources (supply voltage and process supply voltage) are switched off before you begin with operations at the system.
- Never connect any voltages or signals to reserved terminals (marked with ---). Reserved terminals may carry internal voltages.

The module provides several diagnosis functions (see Diagnosis and State LEDs "Chapter 1.5.3.6.1.2.7.7 "Diagnosis" on page 2526).

The following figure shows the electrical connection of the digital input/output module DX522.

---

*Fig. 116: Electrical connection of the module*

1 I/O bus
2 Switch-gear cabinet earth
NOTICE!
- If the relay outputs have to switch inductive DC loads, free-wheeling diodes must be circuited in parallel to these loads.
- If the relay outputs have to switch inductive AC loads, spark suppressors are required.

CAUTION!
The process supply voltage must be included in the earthing concept (e.g., earthing of the negative pole).

NOTICE!
Risk of damaging the PLC module!
The following things have to be considered when connecting input and output voltages to the module:
- All 230 VAC feeds must be single-phase from the same supply system.
- Connection of 2 or more relay contacts in series is possible; however, voltages above 230 VAC and 3-phase loads are not allowed.
- The 8 switch-over contacts of the relays are electrically isolated from channel to channel. This allows to connect loads of 24 VDC and 230 VAC to relay outputs of the same module. In such cases it is necessary that both supply voltages are grounded to prevent unsafe floating grounds.

NOTICE!
Risk of damaging the PLC module!
There is no internal short-circuit or overload protection for the relay outputs. Protect the relay contacts by back-up fuses of 6 A max. (characteristic gG/gL). Depending on the application, fuses can be used for single channels or module-wise.

Internal data exchange

<table>
<thead>
<tr>
<th></th>
<th>Without the Fast Counter</th>
<th>With the Fast Counter (only with AC500)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Digital inputs (bytes)</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>Digital outputs (bytes)</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>Counter input data (words)</td>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td>Counter output data (words)</td>
<td>0</td>
<td>8</td>
</tr>
</tbody>
</table>

I/O configuration
The module itself does not store configuration data. It receives its parameterization data from the master device of the I/O bus (CPU or bus module) during power-up of the system. Hence, replacing I/O modules is possible without any re-parameterization via software.
If the external power supply voltage via UP/ZP terminals fails, the I/O module loses its configuration data. The whole station has to be switched off and on again to re-configure the module.

Parameterization

<table>
<thead>
<tr>
<th>Firmware version</th>
<th>Configuration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Firmware version &gt; V2.0.0</td>
<td>The arrangement of the parameter data is performed by Control Builder Plus/ Automation Builder software.</td>
</tr>
</tbody>
</table>

The parameter data directly influences the functionality of modules.

For non-standard applications, it is necessary to adapt the parameters to your system configuration.

Module: Module slot address: Y = 1…10

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>Internal value</th>
<th>Internal value, type</th>
<th>Default</th>
<th>Min.</th>
<th>Max.</th>
<th>EDS Slot/ Index</th>
</tr>
</thead>
<tbody>
<tr>
<td>Module ID</td>
<td>Internal</td>
<td>1210 1)</td>
<td>Word</td>
<td>1210</td>
<td>0</td>
<td>65535</td>
<td>0x0Y01</td>
</tr>
<tr>
<td>Ignore module 2)</td>
<td>No</td>
<td>0</td>
<td>Byte</td>
<td>No</td>
<td>0</td>
<td>0x00</td>
<td>Not for FBP</td>
</tr>
<tr>
<td>Parameter length</td>
<td>Internal</td>
<td>5</td>
<td>Byte</td>
<td>5-CPU 4-FBP</td>
<td>0</td>
<td>255</td>
<td>0x0Y02</td>
</tr>
<tr>
<td>Check supply</td>
<td>Off</td>
<td>0</td>
<td>Byte</td>
<td>On</td>
<td>0</td>
<td>1</td>
<td>0x0Y03</td>
</tr>
<tr>
<td>Input delay</td>
<td>0.1 ms 1 ms 8 ms 32 ms</td>
<td>0 1 2 3</td>
<td>Byte</td>
<td>8 ms 0x02</td>
<td>0</td>
<td>3</td>
<td>0x0Y04</td>
</tr>
<tr>
<td>Fast Counter 4)</td>
<td>0 : 10 3)</td>
<td>0 : 10</td>
<td>Byte</td>
<td>Mode 0 0x00</td>
<td>Not for FBP</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Name</td>
<td>Value</td>
<td>Internal value</td>
<td>Internal value, type</td>
<td>Default</td>
<td>Min.</td>
<td>Max.</td>
<td>EDS Slot/Index</td>
</tr>
<tr>
<td>------</td>
<td>-------</td>
<td>----------------</td>
<td>---------------------</td>
<td>---------</td>
<td>------</td>
<td>------</td>
<td>----------------</td>
</tr>
<tr>
<td>Behaviour of outputs at communication errors</td>
<td>Off</td>
<td>Last value Substitute value</td>
<td>0</td>
<td>1+(n*5)</td>
<td>2+(n*5), n ≤ 2</td>
<td>Byte</td>
<td>Off</td>
</tr>
<tr>
<td>Substitute value at outputs</td>
<td>0...255</td>
<td>0...255</td>
<td>0</td>
<td>0xff</td>
<td></td>
<td>Byte</td>
<td>0</td>
</tr>
</tbody>
</table>

Remarks:

1) With CS31 and addresses smaller than 70 and FBP, the value is increased by 1
2) Not with FBP
3) For a description of the counter operating modes, please refer to the Fast Counter section “Chapter 1.5.3.6.1.2.9 “Fast counter” on page 2543
4) With FBP and without the parameter Fast Counter

GSD file:

```
Ext_User_Prm_Data_Len =
Ext_User_Prm_Data_Const (0) =
```

Diagnosis

<table>
<thead>
<tr>
<th>E1...E4</th>
<th>d1</th>
<th>d2</th>
<th>d3</th>
<th>d4</th>
<th>Identifier 000...063</th>
<th>AC500 display</th>
<th>← Display in</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class</td>
<td>Comp</td>
<td>Dev</td>
<td>Mod</td>
<td>Ch</td>
<td>Err</td>
<td>PS501 PLC browser</td>
<td></td>
</tr>
<tr>
<td>Byte 6</td>
<td>-</td>
<td>Byte 3</td>
<td>Byte 4</td>
<td>Byte 5</td>
<td>Byte 6 Bit 0...5</td>
<td>FBP diagnosis block</td>
<td></td>
</tr>
<tr>
<td>Bit 6...7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Class</th>
<th>Interface</th>
<th>Device</th>
<th>Module</th>
<th>Channel</th>
<th>Error Identifier</th>
<th>Error message</th>
<th>Remedy</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1)</td>
<td>2)</td>
<td>3)</td>
<td>4)</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Module error

<p>| | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>14</td>
<td>1...10</td>
<td>31</td>
<td>31</td>
<td>19</td>
<td>Checksum error in the I/O module</td>
<td>Replace I/O module</td>
</tr>
<tr>
<td>11 / 12</td>
<td>ADR</td>
<td>1...10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### PLC Automation with V3 CPUs
#### PLC integration > Device specifications

<table>
<thead>
<tr>
<th>Class</th>
<th>Interface</th>
<th>Device</th>
<th>Module</th>
<th>Channel</th>
<th>Error Identifier</th>
<th>Error message</th>
<th>Remedy</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>14</td>
<td>1...10</td>
<td>31</td>
<td>31</td>
<td>3</td>
<td>Timeout in the I/O module</td>
<td>Replace I/O module</td>
</tr>
<tr>
<td></td>
<td>11 / 12</td>
<td>ADR</td>
<td>1...10</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>14</td>
<td>1...10</td>
<td>31</td>
<td>31</td>
<td>40</td>
<td>Different hard-/firmware versions in the module</td>
<td>Replace I/O module</td>
</tr>
<tr>
<td></td>
<td>11 / 12</td>
<td>ADR</td>
<td>1...10</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>14</td>
<td>1...10</td>
<td>31</td>
<td>31</td>
<td>43</td>
<td>Internal error in the module</td>
<td>Replace I/O module</td>
</tr>
<tr>
<td></td>
<td>11 / 12</td>
<td>ADR</td>
<td>1...10</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>14</td>
<td>1...10</td>
<td>31</td>
<td>31</td>
<td>36</td>
<td>Internal data exchange failure</td>
<td>Replace I/O module</td>
</tr>
<tr>
<td></td>
<td>11 / 12</td>
<td>ADR</td>
<td>1...10</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>14</td>
<td>1...10</td>
<td>31</td>
<td>31</td>
<td>9</td>
<td>Overflow diagnosis buffer</td>
<td>New start</td>
</tr>
<tr>
<td></td>
<td>11 / 12</td>
<td>ADR</td>
<td>1...10</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>14</td>
<td>1...10</td>
<td>31</td>
<td>31</td>
<td>26</td>
<td>Parameter error</td>
<td>Check master</td>
</tr>
<tr>
<td></td>
<td>11 / 12</td>
<td>ADR</td>
<td>1...10</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>14</td>
<td>1...10</td>
<td>31</td>
<td>31</td>
<td>11</td>
<td>Process supply voltage too low</td>
<td>Check process supply voltage</td>
</tr>
<tr>
<td></td>
<td>11 / 12</td>
<td>ADR</td>
<td>1...10</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>14</td>
<td>1...10</td>
<td>31</td>
<td>31</td>
<td>45</td>
<td>Process supply voltage is switched off (ON -&gt; OFF)</td>
<td>Process supply voltage ON</td>
</tr>
<tr>
<td></td>
<td>11 / 12</td>
<td>ADR</td>
<td>1...10</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Remarks:

1) In AC500 the following interface identifier applies:
   - 14 = I/O bus, 11 = COM1 (e.g. CS31 bus), 12 = COM2.
   - The FBP diagnosis block does not contain this identifier.

2) With "Device" the following allocation applies:
   - 31 = module itself,
   - 1...10 = decentralized communication interface module 1...10,
   - ADR = hardware address (e.g. of the DC551)

3) With "Module" the following allocation applies dependent of the master:
   - Module error: I/O bus or FBP: 31 = module itself; COM1/COM2: 1...10 = expansion 1...10
   - Channel error: I/O bus or FBP = module type (2 = DO); COM1/COM2: 1...10 = expansion 1...10

4) In case of module errors, with channel "31 = module itself" is output.
State LEDs

During the power ON procedure, the module initializes automatically. All LEDs (except the channel LEDs) are ON during this time.

<table>
<thead>
<tr>
<th>LED</th>
<th>State</th>
<th>Color</th>
<th>LED = OFF</th>
<th>LED = ON</th>
<th>LED flashes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inputs I0...I7</td>
<td>Digital input</td>
<td>Yellow</td>
<td>Input = OFF</td>
<td>Input = ON 1)</td>
<td>--</td>
</tr>
<tr>
<td>Outputs R0...R7 (relays)</td>
<td>Digital output</td>
<td>Yellow</td>
<td>Relay output = OFF</td>
<td>Relay output = ON</td>
<td>--</td>
</tr>
<tr>
<td>UP</td>
<td>Process supply voltage 24 VDC via terminal</td>
<td>Green</td>
<td>Process supply voltage is missing</td>
<td>Process supply voltage OK</td>
<td>--</td>
</tr>
<tr>
<td>CH-ERR1</td>
<td>Channel Error, error messages in groups (digital inputs/outputs combined into the groups 1 and 2)</td>
<td>Red</td>
<td>No error or process supply voltage is missing</td>
<td>Severe error within the corresponding group</td>
<td>Error on one channel of the corresponding group</td>
</tr>
<tr>
<td>CH-ERR2</td>
<td>Module Error</td>
<td>Red</td>
<td>--</td>
<td>Internal error</td>
<td>--</td>
</tr>
</tbody>
</table>

1) Indication LED is ON even if an input signal is applied to the channel and the supply voltage is off. In this case the module is not operating and does not generate an input signal.

2) All of the LEDs CH-ERR1 to CH-ERR2 light up together

Technical data

The system data of AC500 and S500 © Chapter 1.5.4.5.1 “System data AC500” on page 3058 are valid for standard version.

The system data of AC500-XC © Chapter 1.5.4.6.1 “System data AC500-XC” on page 3099 are valid for the XC version.

Only additional details are therefore documented below.

The technical data are also valid for the XC version.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Process supply voltage UP</td>
<td></td>
</tr>
<tr>
<td>Connections</td>
<td>Terminals 1.8, 2.8, 3.8 and 4.8 for +24 V (UP) as well as 1.9, 2.9, 3.9 and 4.9 for 0 V (ZP)</td>
</tr>
<tr>
<td>Rated value</td>
<td>24 VDC</td>
</tr>
<tr>
<td>Max. ripple</td>
<td>5 %</td>
</tr>
<tr>
<td>Protection against reversed voltage</td>
<td>Yes</td>
</tr>
<tr>
<td>Rated protection fuse on UP</td>
<td>10 A fast</td>
</tr>
<tr>
<td>Galvanic isolation</td>
<td>Yes, per module</td>
</tr>
<tr>
<td>Parameter</td>
<td>Value</td>
</tr>
<tr>
<td>-----------------------------------------------------</td>
<td>-----------------------------------------------------------------------</td>
</tr>
<tr>
<td>Current consumption</td>
<td></td>
</tr>
<tr>
<td>From 24 VDC power supply at the terminals UP/L+ and ZP/M of the CPU/Bus Module</td>
<td>ca. 2 mA</td>
</tr>
<tr>
<td>From UP at normal operation / with outputs</td>
<td>0.05 A + output loads</td>
</tr>
<tr>
<td>Inrush current from UP (at power up)</td>
<td>0.010 A²s</td>
</tr>
<tr>
<td>Max. power dissipation within the module</td>
<td>6 W (outputs OFF)</td>
</tr>
<tr>
<td>Weight (without terminal unit)</td>
<td>ca. 300 g</td>
</tr>
<tr>
<td>Mounting position</td>
<td>Horizontal or vertical with derating (output load reduced to 50 % at 40 °C per group)</td>
</tr>
<tr>
<td>Cooling</td>
<td>The natural convection cooling must not be hindered by cable ducts or other parts in the switch-gear cabinet.</td>
</tr>
</tbody>
</table>

**NOTICE!**

Attention:

All I/O channels (digital and analog) are protected against reverse polarity, reverse supply, short circuit and continuous overvoltage up to 30 VDC.

---

**No effects of multiple overloads**

No effects of multiple overloads on isolated multi-channel modules occur, as every channel is protected individually by an external fuse.

---

**Technical data of the digital inputs**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of channels per module</td>
<td>8</td>
</tr>
<tr>
<td>Distribution of the channels into groups</td>
<td>1 group of 8 channels</td>
</tr>
<tr>
<td>Terminals of the channels I0 to I7</td>
<td>1.0 to 1.7</td>
</tr>
<tr>
<td>Reference potential for all inputs</td>
<td>Terminals 1.9, 2.9, 3.9 and 4.9 (minus pole of the process supply voltage, signal name ZP)</td>
</tr>
<tr>
<td>Galvanic isolation</td>
<td>From the rest of the module (I/O bus)</td>
</tr>
<tr>
<td>Indication of the input signals</td>
<td>One yellow LED per channel, the LED is ON when the input signal is high (signal 1)</td>
</tr>
<tr>
<td>Monitoring point of input indicator</td>
<td>LED is part of the input circuitry</td>
</tr>
<tr>
<td>Input type acc. to EN 61131-2</td>
<td>Type 1</td>
</tr>
<tr>
<td>Input delay (0-&gt;1 or 1-&gt;0)</td>
<td>Typ. 8 ms, configurable from 0.1 to 32 ms</td>
</tr>
<tr>
<td>Input signal voltage</td>
<td>24 VDC</td>
</tr>
<tr>
<td>Signal 0</td>
<td>-3 V...+5 V</td>
</tr>
<tr>
<td>Undefined signal</td>
<td>&gt; +5 V...&lt; +15 V</td>
</tr>
<tr>
<td>Signal 1</td>
<td>+15 V...+30 V</td>
</tr>
<tr>
<td>Ripple with signal 0</td>
<td>Within -3 V...+5 V</td>
</tr>
<tr>
<td>Ripple with signal 1</td>
<td>Within +15 V...+30 V</td>
</tr>
<tr>
<td>Parameter</td>
<td>Value</td>
</tr>
<tr>
<td>-----------------------------------</td>
<td>----------------------------</td>
</tr>
<tr>
<td>Input current per channel</td>
<td></td>
</tr>
<tr>
<td>Input voltage +24 V</td>
<td>Typ. 5 mA</td>
</tr>
<tr>
<td>Input voltage +5 V</td>
<td>&gt; 1 mA</td>
</tr>
<tr>
<td>Input voltage +15 V</td>
<td>&gt; 5 mA</td>
</tr>
<tr>
<td>Input voltage +30 V</td>
<td>&lt; 8 mA</td>
</tr>
<tr>
<td>Max. cable length</td>
<td></td>
</tr>
<tr>
<td>Shielded</td>
<td>1000 m</td>
</tr>
<tr>
<td>Unshielded</td>
<td>600 m</td>
</tr>
</tbody>
</table>

### Technical data of the relay outputs

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of channels per module</td>
<td>8 relay outputs</td>
</tr>
<tr>
<td>Distribution of channels into groups</td>
<td>8 groups of 1 channel each</td>
</tr>
<tr>
<td>Connection of the channel R0</td>
<td>Terminal 2.0 (common), 3.0 (NO) and 4.0 (NC)</td>
</tr>
<tr>
<td>Connection of the channel R1</td>
<td>Terminal 2.1 (common), 3.1 (NO) and 4.1 (NC)</td>
</tr>
<tr>
<td>Connection of the channel R6</td>
<td>Terminal 2.6 (common), 3.6 (NO) and 4.6 (NC)</td>
</tr>
<tr>
<td>Connection of the channel R7</td>
<td>Terminal 2.7 (common), 3.7 (NO) and 4.7 (NC)</td>
</tr>
<tr>
<td>Galvanic isolation</td>
<td>Between the channels and from the rest of the module</td>
</tr>
<tr>
<td>Indication of the output signals</td>
<td>One yellow LED per channel, the LED is ON when the relay coil is energized</td>
</tr>
<tr>
<td>Monitoring point of output indicator</td>
<td>LED is controlled by process CPU</td>
</tr>
<tr>
<td>Way of operation</td>
<td>Non-latching type</td>
</tr>
<tr>
<td>Output delay (0-&gt;1 or 1-&gt;0)</td>
<td>On request</td>
</tr>
<tr>
<td>Relay power supply</td>
<td>By UP process supply voltage</td>
</tr>
<tr>
<td>Relay outputs</td>
<td></td>
</tr>
<tr>
<td>Output short circuit protection</td>
<td>Should be provided externally with a fuse or circuit breaker</td>
</tr>
<tr>
<td>Rated protection fuse</td>
<td>6 A gL/gG per channel</td>
</tr>
<tr>
<td>Min. switching current</td>
<td>10 mA</td>
</tr>
<tr>
<td>Output switching capacity</td>
<td></td>
</tr>
<tr>
<td>Resistive load, max.</td>
<td>3 A; 3 A (230 VAC), 2 A (24 VDC)</td>
</tr>
<tr>
<td>Inductive load, max.</td>
<td>1.5 A; 1.5 A (230 VAC), 1.5 A (24 VDC)</td>
</tr>
<tr>
<td>Lamp load</td>
<td>60 W (230 VAC), 10 W (24 VDC)</td>
</tr>
<tr>
<td>Output switching capacity (XC version above 60 °C)</td>
<td>On request</td>
</tr>
<tr>
<td>Life time (cycles)</td>
<td>Mechanical: 300 000; Under load: 300 000 (24 VDC at 2 A), 200 000 (120 VAC at 2 A), 100 000 (230 VAC at 3 A)</td>
</tr>
<tr>
<td>Spark suppression with inductive AC load</td>
<td>Must be performed externally according to driven load specifications</td>
</tr>
</tbody>
</table>
### Technical data of the fast counter

The fast counter of the module does not work if the module is connected to a
- FBP interface module
- CS31 bus module
- CANopen bus module

### Parameter | Value
--- | ---
Demagnetization with inductive DC load | A free-wheeling diode must be circuited in parallel to the inductive load
Switching frequency
  - With resistive load | Max. 10 Hz
  - With inductive load | Max. 2 Hz
  - With lamp load | On request
Max. cable length
  - Shielded | 1000 m
  - Unshielded | 600 m

### Ordering data

| Part no. | Description | Product life cycle phase *
--- | --- | ---
1SAP 245 200 R0001 | DX522, digital input/output module, 8 DI, 24 VDC, 8 DO relays | Active
1SAP 445 200 R0001 | DX522-XC, digital input/output module, 8 DI, 24 VDC, 8 DO relays, XC version | Active

*) For planning and commissioning of new installations use modules in Active status only.
DX531 - Digital input/output module

- 8 digital inputs 120/230 VAC
- 4 relay outputs with one switch-over contact each
- Module-wise electrically isolated

1 I/O bus
2 Allocation between terminal number and signal name
3 8 yellow LEDs to display the signal states at the digital inputs (I0 - I7)
4 4 yellow LEDs to display the signal states at the digital relay outputs (R0 - R3)
5 1 green LED to display the state of the process supply voltage UP
6 2 red LEDs to display errors
7 Label
8 Terminal unit
9 DIN rail

Intended purpose

The device can be used as a decentralized I/O extension module for S500 Communication Interface Modules (e.g. CI592-CS31, CI501-PNIO, CI541-DP, CI581-CN) or as a centralized extension module for AC500 CPUs (PM5xx).
Digital configurable input / output unit.

- 8 digital inputs 120/230 VAC in 1 group (2.0...2.3 and 3.0...3.3)
- 4 digital relay outputs with one switch-over contact each (R0...R3). All output channels are electrically isolated from each other.

The configuration is performed by software. The modules are supplied with a process supply voltage of 24 VDC.

All available inputs/outputs are electrically isolated from all other circuitry of the module. There is no potential separation between the channels within the same group.

For use in extreme ambient conditions (e.g. wider temperature and humidity range), a special XC version of the device is available.

### Functionality

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>LED displays</td>
<td>For signal states, errors and supply voltage</td>
</tr>
<tr>
<td>Internal power supply</td>
<td>Through the expansion bus interface (I/O bus)</td>
</tr>
<tr>
<td>External power supply</td>
<td>Via the terminals ZP and UP (process supply voltage 24 VDC)</td>
</tr>
<tr>
<td>Required terminal units</td>
<td>TU531 or TU532 &amp; Chapter 1.5.3.5.4 “TU531 and TU532 for I/O modules” on page 2320</td>
</tr>
</tbody>
</table>

The device is plugged on a terminal unit & Chapter 1.5.3.5.4 “TU531 and TU532 for I/O modules” on page 2320. Position the module properly and press until it locks in place. The terminal unit is either mounted on a DIN rail or to the wall using 2 screws plus the additional accessory for wall mounting (TA526 & Chapter 1.5.3.8.1.4 “TA526 - Wall mounting accessory” on page 3035).

### Electrical connection

**WARNING!**

Risk of death by electric shock!

Hazardous voltages can be present at the terminals of the module.

Make sure that all voltage sources (supply voltage and process supply voltage) are switched off before you begin with operations at the system.

For a detailed description of the mounting, disassembly and electrical connection of the module, please refer to the System Assembly, Construction and Connection chapter & Chapter 1.5.4.5 “AC500 (Standard)” on page 3058.

The electrical connection of the I/O channels is carried out using the 40 terminals of the I/O terminal unit. I/O modules can be replaced without re-wiring the terminal units.

The terminals 1.8 to 4.8 and 1.9 to 4.9 are electrically interconnected within the I/O terminal unit and always have the same assignment, irrespective of the inserted module.
- Terminals 1.8 to 4.8: process supply voltage UP = +24 VDC
- Terminals 1.9 to 4.9: process supply voltage ZP = 0 VDC

The assignment of the other terminals:

<table>
<thead>
<tr>
<th>Terminals</th>
<th>Signal</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0 to 1.7</td>
<td>unused</td>
<td></td>
</tr>
<tr>
<td>2.0 and 3.0</td>
<td>I0 and I1</td>
<td>Input signals for the digital inputs I0 and I1</td>
</tr>
<tr>
<td>4.0</td>
<td>N01</td>
<td>Neutral conductor for the digital inputs I0 and I1</td>
</tr>
<tr>
<td>2.1 and 3.1</td>
<td>I2 and I3</td>
<td>Input signals for the digital inputs I2 and I3</td>
</tr>
<tr>
<td>4.1</td>
<td>N23</td>
<td>Neutral conductor for the digital inputs I2 and I3</td>
</tr>
<tr>
<td>2.2 and 3.2</td>
<td>I4 and I5</td>
<td>Input signals for the digital inputs I4 and I5</td>
</tr>
<tr>
<td>4.2</td>
<td>N45</td>
<td>Neutral conductor for the digital inputs I4 and I5</td>
</tr>
<tr>
<td>2.3 and 3.3</td>
<td>I6 and I7</td>
<td>Input signals for the digital inputs I6 and I7</td>
</tr>
<tr>
<td>4.3</td>
<td>N67</td>
<td>Neutral conductor for the digital inputs I6 and I7</td>
</tr>
<tr>
<td>2.4</td>
<td>R0</td>
<td>Common contact of the first relay output</td>
</tr>
<tr>
<td>3.4 and 4.4</td>
<td>NO0 and NC0</td>
<td>NO and NC contacts of the first relay output</td>
</tr>
<tr>
<td>2.5</td>
<td>R1</td>
<td>Common contact of the second relay output</td>
</tr>
<tr>
<td>3.5 and 4.5</td>
<td>NO1 and NC1</td>
<td>NO and NC contacts of the second relay output</td>
</tr>
<tr>
<td>2.6</td>
<td>R2</td>
<td>Common contact of the third relay output</td>
</tr>
<tr>
<td>3.6 and 4.6</td>
<td>NO2 and NC2</td>
<td>NO and NC contacts of the third relay output</td>
</tr>
<tr>
<td>2.7</td>
<td>R3</td>
<td>Common contact of the fourth relay output</td>
</tr>
<tr>
<td>3.7 and 4.7</td>
<td>NO3 and NC3</td>
<td>NO and NC contacts of the fourth relay output</td>
</tr>
</tbody>
</table>
The internal power supply voltage for the module's circuitry is carried out via the I/O bus (provided by a bus module or a CPU). Thus, the current consumption from 24 VDC power supply at the terminals L+/UP and M/ZP of the CPU/bus module increases by 2 mA per DX531. The external power supply connection is carried out via the UP (+24 VDC) and the ZP (0 VDC) terminals.

**NOTICE!**

Risk of damaging the PLC modules!
- Overvoltages and short circuits might damage the PLC modules.
  - Make sure that all voltage sources (supply voltage and process supply voltage) are switched off before you begin with operations at the system.
  - Never connect any voltages or signals to reserved terminals (marked with ---). Reserved terminals may carry internal voltages.

The following figure shows the electrical connection of the module:
NOTICE!
- If the relay outputs have to switch inductive DC loads, free-wheeling diodes must be circulated in parallel to these loads.
- If the relay outputs have to switch inductive AC loads, spark suppressors are required.

CAUTION!
The process supply voltage must be included in the earthing concept (e.g., earthing of the negative pole).
NOTICE!
Risk of damaging the PLC module!
The following things have to be considered when connecting input and output voltages to the module:

– All 230 VAC feeds must be single phase from the same supply system.
– Connection of 2 or more relay contacts in series is possible; however, voltages above 230 VAC and 3-phase loads are not allowed.
– The 4 switch-over contacts of the relays are electrically isolated from channel to channel. This allows to connect loads of 24 VDC and 230 VAC to relay outputs of the same module. In such cases it is necessary that both supply voltages are grounded to prevent unsafe floating grounds.
– All input signals must come from the same phase of the same supply system (together with the used neutral conductor). The module is designed for 120/230 VAC max., not for 400 VAC, not even between two input terminals.
– All neutral conductor connections must be common to the same supply system, since the terminals 4.0 to 4.3 are interconnected within the module. Otherwise, accidental energization could occur.

NOTICE!
Risk of damaging the PLC module!
There is no internal short-circuit or overload protection for the relay outputs.
Protect the relay contacts by back-up fuses of 6 A max. (characteristic gG/gL). Depending on the application, fuses can be used for single channels or module-wise.

The module provides several diagnosis functions (see chapter Diagnosis and State LEDs Chapter 1.5.3.6.1.2.8.7 “Diagnosis” on page 2539).

Internal data exchange

<table>
<thead>
<tr>
<th>Digital inputs (bytes)</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Digital outputs (bytes)</td>
<td>1</td>
</tr>
<tr>
<td>Counter input data (words)</td>
<td>0</td>
</tr>
<tr>
<td>Counter output data (words)</td>
<td>0</td>
</tr>
</tbody>
</table>

I/O configuration

The module itself does not store configuration data. It receives its parameterization data from the master device of the I/O bus (CPU or bus module) during power-up of the system.
Hence, replacing I/O modules is possible without any re-parameterization via software.

If the external power supply voltage via UP/ZP terminals fails, the I/O module loses its configuration data. The whole station has to be switched off and on again to re-configure the module.
Parameterization

The arrangement of the parameter data is performed by Control Builder Plus/ Automation Builder software.

The parameter data directly influences the functionality of modules.

For non-standard applications, it is necessary to adapt the parameters to your system configuration.

Module: Module slot address: Y = 1...10

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>Internal value</th>
<th>Internal value, type</th>
<th>Default</th>
<th>Min.</th>
<th>Max.</th>
<th>EDS Slot/Index</th>
</tr>
</thead>
<tbody>
<tr>
<td>Module ID</td>
<td>Internal</td>
<td>1205 1)</td>
<td>Word</td>
<td>1205 0x04B5</td>
<td>0</td>
<td>65535</td>
<td>0x0Y01</td>
</tr>
<tr>
<td>Ignore module</td>
<td>No Yes</td>
<td>0 1</td>
<td>Byte</td>
<td>No 0x00</td>
<td></td>
<td></td>
<td>not for FBP</td>
</tr>
<tr>
<td>Parameter length</td>
<td>Internal</td>
<td>4</td>
<td>Byte</td>
<td>4-CPU 4-FBP</td>
<td>0</td>
<td>255 0x0Y02</td>
<td></td>
</tr>
<tr>
<td>Check supply</td>
<td>Off on</td>
<td>0 1</td>
<td>Byte</td>
<td>On 0x01</td>
<td>0</td>
<td>1</td>
<td>0x0Y03</td>
</tr>
<tr>
<td>Input delay</td>
<td>20 ms 100 ms</td>
<td>0 1</td>
<td>Byte</td>
<td>20 ms 0x00</td>
<td>0</td>
<td>1</td>
<td>0x0Y04</td>
</tr>
<tr>
<td>Behaviour of outputs at communication errors</td>
<td>Off Last value Substitute value</td>
<td>0 1+(n<em>5) 2+(n</em>5), n ≤ 2</td>
<td>Byte</td>
<td>Off 0x00</td>
<td>0</td>
<td>2</td>
<td>0x0Y05</td>
</tr>
<tr>
<td>Substitute value at outputs</td>
<td>Bit 3 = Output 3 Bit 0 = Output 0</td>
<td>0...15 0...0x0f</td>
<td>Byte</td>
<td>0 0x00</td>
<td>0</td>
<td>15 0x0Y06</td>
<td></td>
</tr>
</tbody>
</table>

1) With CS31 and addresses smaller than 70 and FBP, the value is increased by 1
2) Not with FBP

GSD file:

```
Ext_User_Prm_Data_Len = 7
Ext_User_Prm_Data_Const (0) = 0x04, 0xb6, 0x04, \0x01, 0x00, 0x00, 0x00;
```
## Diagnosis

<table>
<thead>
<tr>
<th>Class</th>
<th>Comp</th>
<th>Dev</th>
<th>Mod</th>
<th>Ch</th>
<th>Err</th>
<th>AC500 display</th>
<th>⇐ Display in</th>
</tr>
</thead>
<tbody>
<tr>
<td>E1...E4</td>
<td>d1</td>
<td>d2</td>
<td>d3</td>
<td>d4</td>
<td>Identifier 000...063</td>
<td>PS501 PLC browser</td>
<td></td>
</tr>
</tbody>
</table>

## Byte 6 Bit 6...7

<table>
<thead>
<tr>
<th>Class</th>
<th>Interface</th>
<th>Device</th>
<th>Module</th>
<th>Channel</th>
<th>Error Identifier</th>
<th>Error message</th>
<th>Remedy</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>14</td>
<td>1...10</td>
<td>31</td>
<td>19</td>
<td>Checksum error in the I/O module</td>
<td>Replace I/O module</td>
<td></td>
</tr>
<tr>
<td></td>
<td>11 / 12</td>
<td>ADR</td>
<td>1...10</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>14</td>
<td>1...10</td>
<td>31</td>
<td>3</td>
<td>Timeout in the I/O module</td>
<td>Replace I/O module</td>
<td></td>
</tr>
<tr>
<td></td>
<td>11 / 12</td>
<td>ADR</td>
<td>1...10</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>14</td>
<td>1...10</td>
<td>31</td>
<td>40</td>
<td>Different hard-/firmware versions in the module</td>
<td>Replace I/O module</td>
<td></td>
</tr>
<tr>
<td></td>
<td>11 / 12</td>
<td>ADR</td>
<td>1...10</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>14</td>
<td>1...10</td>
<td>31</td>
<td>43</td>
<td>Internal error in the module</td>
<td>Replace I/O module</td>
<td></td>
</tr>
<tr>
<td></td>
<td>11 / 12</td>
<td>ADR</td>
<td>1...10</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>14</td>
<td>1...10</td>
<td>31</td>
<td>36</td>
<td>Internal data exchange failure</td>
<td>Replace I/O module</td>
<td></td>
</tr>
<tr>
<td></td>
<td>11 / 12</td>
<td>ADR</td>
<td>1...10</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>14</td>
<td>1...10</td>
<td>31</td>
<td>9</td>
<td>Overflow diagnosis buffer</td>
<td>New start</td>
<td></td>
</tr>
<tr>
<td></td>
<td>11 / 12</td>
<td>ADR</td>
<td>1...10</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>14</td>
<td>1...10</td>
<td>31</td>
<td>26</td>
<td>Parameter error</td>
<td>Check master</td>
<td></td>
</tr>
<tr>
<td></td>
<td>11 / 12</td>
<td>ADR</td>
<td>1...10</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>14</td>
<td>1...10</td>
<td>31</td>
<td>11</td>
<td>Process supply voltage too low</td>
<td>Check process supply voltage</td>
<td></td>
</tr>
<tr>
<td></td>
<td>11 / 12</td>
<td>ADR</td>
<td>1...10</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>14</td>
<td>1...10</td>
<td>31</td>
<td>45</td>
<td>Process supply voltage is switched off (ON −&gt; OFF)</td>
<td>Process supply voltage ON</td>
<td></td>
</tr>
<tr>
<td></td>
<td>11 / 12</td>
<td>ADR</td>
<td>1...10</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Remarks:

1) In AC500 the following interface identifier applies:
   14 = I/O bus, 11 = COM1 (e.g. CS31 bus), 12 = COM2.
   The FBP diagnosis block does not contain this identifier.

2) With "Device" the following allocation applies:
   31 = module itself,
   1...10 = decentralized communication interface module 1...10,
   ADR = hardware address (e.g. of the DC551)
With "Module" the following allocation applies depending on the master:
Module error: I/O bus or FBP: 31 = module itself; COM1/COM2: 1...10 = expansion 1...10
Channel error: I/O bus or FBP = module type (2 = DO); COM1/COM2: 1...10 = expansion 1...10

In case of module errors, with channel "31 = module itself" is output.

State LEDs

During the power ON procedure, the module initializes automatically. All LEDs (except the channel LEDs) are ON during this time.

<table>
<thead>
<tr>
<th>LED</th>
<th>State</th>
<th>Color</th>
<th>LED = OFF</th>
<th>LED = ON</th>
<th>LED flashes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inputs</td>
<td>Digital input</td>
<td>Yellow</td>
<td>Input = OFF</td>
<td>Input = ON</td>
<td>--</td>
</tr>
<tr>
<td>Outputs</td>
<td>Digital output (relays)</td>
<td>Yellow</td>
<td>Relay output = OFF</td>
<td>Relay output = ON</td>
<td>--</td>
</tr>
<tr>
<td>UP</td>
<td>Process supply voltage 24 VDC via terminal</td>
<td>Green</td>
<td>Process supply voltage is missing</td>
<td>Process supply voltage OK</td>
<td>--</td>
</tr>
<tr>
<td>CH-ERR2</td>
<td>Channel error, error messages in groups (digital inputs/outputs combined into the groups 2 and 3)</td>
<td>Red</td>
<td>No error or process supply voltage is missing</td>
<td>Severe error within the corresponding group</td>
<td>Error on one channel of the corresponding group</td>
</tr>
<tr>
<td>CH-ERR3</td>
<td>Module Error</td>
<td>Red</td>
<td>--</td>
<td>Internal error</td>
<td>--</td>
</tr>
</tbody>
</table>

*) All of the LEDs CH-ERR2 to CH-ERR3 light up together

Technical data

The system data of AC500 and S500 % Chapter 1.5.4.5.1 “System data AC500” on page 3058 are valid for standard version.

The system data of AC500-XC % Chapter 1.5.4.6.1 “System data AC500-XC” on page 3099 are valid for the XC version.

Only additional details are therefore documented below.

The technical data are also valid for the XC version.
### Parameter | Value
--- | ---
Process supply voltage UP |  
Connections | Terminals 1.8, 2.8, 3.8 and 4.8 for +24 VDC (UP) as well as 1.9, 2.9, 3.9 and 4.9 for 0 VDC (ZP)
Rated value | 24 VDC
Max. ripple | 5 %
Protection against reversed voltage | Yes
Rated protection fuse on UP | 10 A fast
Galvanic isolation | Yes, per module

#### Current consumption
- From 24 VDC power supply at the terminals UP/L + and ZP/M of the CPU/bus module: ca. 2 mA
- From UP at normal operation / with outputs: 0.15 A + output loads

Inrush current from UP (at power up): 0.004 A²s
Max. power dissipation within the module: 6 W (outputs OFF)
Weight (without terminal unit): Ca. 300 g

Mounting position
- Horizontal or vertical with derating (output load reduced to 50 % at 40 °C per group)

Cooling
- The natural convection cooling must not be hindered by cable ducts or other parts in the switchgear cabinet.

#### NOTICE!
**Attention:**
All I/O channels (digital and analog) are protected against reverse polarity, reverse supply, short circuit and continuous overvoltage up to 30 VDC.

---

**No effects of multiple overloads:**
No effects of multiple overloads on isolated multi-channel modules occur, as every channel is protected individually by an external fuse.

---

### Technical data of the digital inputs

| Parameter | Value |
--- | --- |
Number of channels per module | 8 |
Distribution of the channels into groups | 4 groups of 2 channels each |
Terminals of the channels I0 to I7
| Chapter 1.5.3.6.1.2.8.3 “Electrical connection” on page 2533 |
Galvanic isolation | 2500 VAC from the rest of the module (I/O bus) |
Indication of the input signals | 1 yellow LED per channel |
| The LEDs are only operating if the module is initialized |
Monitoring point of input indicator | LED is controlled by process CPU |
**Technical data of the relay outputs**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of channels per module</td>
<td>4 relay outputs</td>
</tr>
<tr>
<td>Distribution of channels into groups</td>
<td>4 groups of 1 channel each</td>
</tr>
<tr>
<td>Connection of the four relays</td>
<td>% Chapter 1.5.3.6.1.2.8.3 “Electrical connection” on page 2533</td>
</tr>
<tr>
<td>Galvanic isolation</td>
<td>Between the channels and from the rest of the module</td>
</tr>
<tr>
<td>Indication of the output signals</td>
<td>1 yellow LED per channel, the LED is ON when the relay coil is energized</td>
</tr>
<tr>
<td>Monitoring point of output indicator</td>
<td>LED is controlled by process CPU</td>
</tr>
<tr>
<td>Way of operation</td>
<td>Non-latching type</td>
</tr>
<tr>
<td>Output delay (0-&gt;1 or 1-&gt;0)</td>
<td>On request</td>
</tr>
<tr>
<td>Relay power supply</td>
<td>By UP process supply voltage</td>
</tr>
<tr>
<td>Relay outputs</td>
<td></td>
</tr>
<tr>
<td>Output short circuit protection</td>
<td>Must be provided externally with a fuse or circuit breaker</td>
</tr>
<tr>
<td>Rated protection fuse</td>
<td>6 A gL/gG per channel</td>
</tr>
<tr>
<td>Output switching capacity</td>
<td></td>
</tr>
<tr>
<td>Resistive load, max.</td>
<td>3 A; 3 A (230 VAC), 2 A (24 VDC)</td>
</tr>
<tr>
<td>Inductive load, max.</td>
<td>1.5 A; 1.5 A (230 VAC), 1.5 A (24 VDC)</td>
</tr>
<tr>
<td>Lamp load</td>
<td>60 W (230 VAC), 10 W (24 VDC)</td>
</tr>
</tbody>
</table>
### Parameter | Value
--- | ---
Life time (cycles) | Mechanical: 300 000; Under load: 300 000 (24 VDC at 2 A), 200 000 (120 VAC at 2 A), 100 000 (230 VAC at 3 A)
Spark suppression with inductive AC load | Must be performed externally according to driven load specifications
Demagnetization with inductive DC load | A free-wheeling diode must be circuited in parallel to the inductive load
Switching frequency | 
  - With resistive load: Max. 10 Hz
  - With inductive load: Max. 2 Hz
  - With lamp load: On request
Max. cable length | 
  - Shielded: 1000 m
  - Unshielded: 600 m

### Ordering data

<table>
<thead>
<tr>
<th>Part no.</th>
<th>Description</th>
<th>Product life cycle phase *)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1SAP 245 000 R0001</td>
<td>DX531, digital input/output module, 8 DI, 230 VAC, 4 DO relays, 2-wires</td>
<td>Active</td>
</tr>
</tbody>
</table>

*) For planning and commissioning of new installations use modules in Active status only.

### Fast counter
Details on fast counters: System Technology  Chapter 1.5.5.1.10 “Fast counters” on page 3194

### 1.5.3.6.2 Analog I/O modules
S500-eCo
AI561 - Analog input module
- 4 configurable analog inputs (I0 to I3) in 1 group
- Resolution: 11 bits plus sign or 12 bits
<table>
<thead>
<tr>
<th>Number</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>I/O bus</td>
</tr>
<tr>
<td>2</td>
<td>1 green LED to display power supply, 1 red LED to display error</td>
</tr>
<tr>
<td>3</td>
<td>Terminal number</td>
</tr>
<tr>
<td>4</td>
<td>Allocation of signal name</td>
</tr>
<tr>
<td>5</td>
<td>Terminal block for input signals (9-pin)</td>
</tr>
<tr>
<td>6</td>
<td>Terminal block for input signals (11-pin)</td>
</tr>
<tr>
<td>7</td>
<td>2 holes for wall-mounting with screws</td>
</tr>
<tr>
<td>8</td>
<td>DIN rail</td>
</tr>
</tbody>
</table>

### Intended purpose

The device can be used as a decentralized I/O extension module for S500 Communication Interface Modules (e.g. CI592-CS31, CI501-PNIO, CI541-DP, CI581-CN) or as a centralized extension module for AC500 CPUs (PM5xx).

The inputs are not electrically isolated from each other. All other circuitry of the module is not electrically isolated from the inputs or from the I/O bus.

*The I/O module must not be used as a decentralized I/O module with CI590-CS31-HA bus modules.*
Functionality

4 analog inputs, individually configurable for
- Not used (default setting)
- -2.5 V...+2.5 V
- -5 V...+5 V
- 0 V...+5 V
- 0 V...+10 V
- 0 mA...20 mA
- 4 mA...20 mA

Parameter | Value
---|---
Resolution of the analog channels | 11 bits plus sign
Voltage bipolar (-2.5 V...+2.5 V; -5 V...+5 V) | 11 bits plus sign
Voltage unipolar (0 V...5 V; 0 V...10 V) | 12 bits
Current (0 mA...20 mA; 4 mA...20 mA) | 12 bits
LED displays | 2 LEDs for process voltage and error messages
Internal supply | Via I/O bus
External supply | Via the terminals L+ (process voltage 24 VDC) and M (0 VDC); the M terminal is connected to the M terminal of the CPU via the I/O bus

Electrical connection

For a detailed description of the mounting, disassembly and electrical connection of the module, please refer to the System Assembly chapter.

The electrical connection is carried out by using a removable 9-pin and 11-pin terminal block. These terminal blocks differ in their connection system (spring terminals or screw terminals, cable mounting from the front or from the side). The terminal blocks are not included in the module's scope of delivery and must be ordered separately.

The following block diagram shows the internal construction of the analog inputs:
The assignment of the terminals:

<table>
<thead>
<tr>
<th>Terminal</th>
<th>Signal</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>R0</td>
<td>Burden resistor for input signal 0 for current sensing</td>
</tr>
<tr>
<td>2</td>
<td>I0+</td>
<td>Positive pole of input signal 0</td>
</tr>
<tr>
<td>3</td>
<td>I0-</td>
<td>Negative pole of input signal 0</td>
</tr>
<tr>
<td>4</td>
<td>R1</td>
<td>Burden resistor for input signal 1 for current sensing</td>
</tr>
<tr>
<td>5</td>
<td>I1+</td>
<td>Positive pole of input signal 1</td>
</tr>
<tr>
<td>6</td>
<td>I1-</td>
<td>Negative pole of input signal 1</td>
</tr>
<tr>
<td>7</td>
<td>R2</td>
<td>Burden resistor for input signal 2 for current sensing</td>
</tr>
<tr>
<td>8</td>
<td>I2+</td>
<td>Positive pole of input signal 2</td>
</tr>
<tr>
<td>9</td>
<td>I2-</td>
<td>Negative pole of input signal 2</td>
</tr>
<tr>
<td>10</td>
<td>R3</td>
<td>Burden resistor for input signal 3 for current sensing</td>
</tr>
<tr>
<td>11</td>
<td>I3+</td>
<td>Positive pole of input signal 3</td>
</tr>
</tbody>
</table>
The internal power supply voltage for the module's circuitry is carried out via the I/O bus (provided by a bus module or a CPU). Thus, the current consumption from 24 VDC power supply at the terminals L+/UP and M/ZP of the CPU/bus module increases by 10 mA per AI561.

The external power supply connection is carried out via the L+ (+24 VDC) and the M (0 VDC) terminals. The M terminal is electrically interconnected to the M/ZP terminal of the CPU/bus module.

### Terminal Signal Description

<table>
<thead>
<tr>
<th>Terminal</th>
<th>Signal</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>12</td>
<td>I3-</td>
<td>Negative pole of input signal 3</td>
</tr>
<tr>
<td>13</td>
<td>---</td>
<td>Reserved</td>
</tr>
<tr>
<td>14</td>
<td>---</td>
<td>Reserved</td>
</tr>
<tr>
<td>15</td>
<td>---</td>
<td>Reserved</td>
</tr>
<tr>
<td>16</td>
<td>---</td>
<td>Reserved</td>
</tr>
<tr>
<td>17</td>
<td>---</td>
<td>Reserved</td>
</tr>
<tr>
<td>18</td>
<td>SG</td>
<td>Shield grounding</td>
</tr>
<tr>
<td>19</td>
<td>L+</td>
<td>Process voltage L+ (24 VDC)</td>
</tr>
<tr>
<td>20</td>
<td>M</td>
<td>Process voltage M (0 VDC)</td>
</tr>
</tbody>
</table>

NOTICE!

**Risk of imprecise and faulty measurements!**

Analog signals may be distorted seriously by external electromagnetic influences.

Use shielded wires when wiring analog signal sources. The cable shield must be grounded at both ends of the cable. Provide a potential equalisation of a low resistance to avoid high potential differences between different parts of the plant.

NOTICE!

**Risk of damaging the PLC modules!**

The PLC modules must not be removed while the plant is connected to a power supply.

Make sure that all voltage sources (supply and process voltage) are switched off before you
- connect or disconnect any signal or terminal block
- remove or replace a module.

NOTICE!

**Risk of damaging the PLC modules!**

Overvoltages and short circuits might damage the PLC modules.
- Make sure that all voltage sources (supply voltage and process supply voltage) are switched off before you begin with operations at the system.
- Never connect any voltages or signals to reserved terminals (marked with ---). Reserved terminals may carry internal voltages.

The module provides several diagnosis functions "Chapter 1.5.3.6.2.1.1.6 “Diagnosis” on page 2550.

The following figure is an example of the internal construction of the analog input AI0. The analog inputs AI1...AI3 are designed in the same way.
The following figures are an example of the electrical connection of analog sensors (voltage) to the input I0 of the analog input module AI561. Proceed with the inputs I1 to I3 in the same way.

The meaning of the LEDs is described in the Displays section Chapter 1.5.3.6.2.1.1.7 “State LEDs” on page 2551.

I/O configuration

The analog input module AI561 does not store configuration data itself.
Parameterization

The arrangement of the parameter data is performed with Automation Builder software. The parameter data directly influences the functionality of modules. For non-standard applications, it is necessary to adapt the parameters to your system configuration.

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>Internal Value</th>
<th>Internal value, Type</th>
<th>Default</th>
<th>Min.</th>
<th>Max.</th>
<th>EDS Slot Index</th>
</tr>
</thead>
<tbody>
<tr>
<td>Module ID</td>
<td>Intern</td>
<td>6500 (^1)</td>
<td>WORD</td>
<td>0x1964</td>
<td>0</td>
<td>65535</td>
<td>xx01</td>
</tr>
<tr>
<td>Ignore module</td>
<td>No</td>
<td>0</td>
<td>BYTE</td>
<td>No</td>
<td>0x00</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Yes</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Parameter length</td>
<td>Internal</td>
<td>6</td>
<td>BYTE</td>
<td>0</td>
<td>0</td>
<td>255</td>
<td>xx02 (^2)</td>
</tr>
<tr>
<td>Check Supply</td>
<td>Off</td>
<td>0</td>
<td>BYTE</td>
<td>On</td>
<td>0x01</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>On</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Analog Data Format</td>
<td>Default</td>
<td>0</td>
<td>BYTE</td>
<td>Default</td>
<td>0x00</td>
<td></td>
<td>255</td>
</tr>
</tbody>
</table>

\(^1\) with CS31 and addresses smaller than 70, the value is increased by 1
\(^2\) Value is hexadecimal: HighByte is slot (xx: 0 ... 7), LowByte is index (1...n)

GSD file:

```
Ext_User_Prm_Data_Len = Ext_User_Prm_Data_Const(0) = 0x09
0x65, 0x19, 0x06, \0x01, 0x00, \0x00, 0x00, 0x00;
```

Input channel (4x)

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>Internal value</th>
<th>Internal value, Type</th>
<th>Default</th>
<th>Min.</th>
<th>Max.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Channel configuration</td>
<td>see table (^2)</td>
<td>see table (^2)</td>
<td>BYTE</td>
<td>0</td>
<td>0x00</td>
<td>65535</td>
</tr>
</tbody>
</table>

\(^2\) Value is hexadecimal: HighByte is slot (xx: 0 ... 7), LowByte is index (1...n)

Table 552: Channel configuration \(^2\)

<table>
<thead>
<tr>
<th>Internal value</th>
<th>Operating modes for the analog inputs, individually configurable</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Not used (default)</td>
</tr>
<tr>
<td>1</td>
<td>0 V...10 V</td>
</tr>
<tr>
<td>3</td>
<td>0 mA...20 mA</td>
</tr>
<tr>
<td>4</td>
<td>4 mA...20 mA</td>
</tr>
<tr>
<td>6</td>
<td>0 V...5 V</td>
</tr>
</tbody>
</table>
### Internal value

<table>
<thead>
<tr>
<th>Internal value</th>
<th>Operating modes for the analog inputs, individually configurable</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>-5 V...+5 V</td>
</tr>
<tr>
<td>20</td>
<td>-2.5 V...+2.5 V</td>
</tr>
</tbody>
</table>

### Diagnosis

<table>
<thead>
<tr>
<th>E1...E4</th>
<th>d1</th>
<th>d2</th>
<th>d3</th>
<th>d4</th>
<th>Identifier</th>
<th>AC500 display</th>
<th>&lt;- Display in</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Byte 6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bit 6...7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Class</th>
<th>Interface</th>
<th>Device</th>
<th>Module</th>
<th>Channel</th>
<th>Error Identifier</th>
<th>Error message</th>
<th>Remedy</th>
</tr>
</thead>
<tbody>
<tr>
<td>1)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### Module error

<table>
<thead>
<tr>
<th>Module error</th>
<th>14</th>
<th>1...10</th>
<th>31</th>
<th>31</th>
<th>19</th>
<th>Checksum error in the I/O module</th>
<th>Replace I/O module</th>
</tr>
</thead>
<tbody>
<tr>
<td>11 / 12 ADR</td>
<td>1...10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Module error</th>
<th>14</th>
<th>1...10</th>
<th>31</th>
<th>31</th>
<th>9</th>
<th>Overflow diagnosis buffer</th>
<th>Restart</th>
</tr>
</thead>
<tbody>
<tr>
<td>11 / 12 ADR</td>
<td>1...10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Module error</th>
<th>14</th>
<th>1...10</th>
<th>31</th>
<th>31</th>
<th>26</th>
<th>Parameter error</th>
<th>Check master</th>
</tr>
</thead>
<tbody>
<tr>
<td>11 / 12 ADR</td>
<td>1...10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Module error</th>
<th>14</th>
<th>1...10</th>
<th>31</th>
<th>31</th>
<th>11</th>
<th>Process voltage too low</th>
<th>Check process voltage</th>
</tr>
</thead>
<tbody>
<tr>
<td>11 / 12 ADR</td>
<td>1...10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### Channel error

<table>
<thead>
<tr>
<th>Channel error</th>
<th>14</th>
<th>1...10</th>
<th>1</th>
<th>0...3</th>
<th>48</th>
<th>Analog value overflow at an analog input</th>
<th>Check input value or terminal</th>
</tr>
</thead>
<tbody>
<tr>
<td>11 / 12 ADR</td>
<td>1...0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Channel error</th>
<th>14</th>
<th>1...10</th>
<th>1</th>
<th>0...3</th>
<th>7</th>
<th>Analog value underflow at an analog input</th>
<th>Check input value</th>
</tr>
</thead>
<tbody>
<tr>
<td>11 / 12 ADR</td>
<td>1...0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Remarks:

1) In AC500, the following interface identifier applies:
   14 = I/O bus, 11 = COM1 (e.g. CS31 bus), 12 = COM2.
   The PNIO diagnosis block does not contain this identifier.

2) With "Device" the following allocation applies:
   31 = module itself, 1...10 = decentralized communication interface module 1...10, ADR = hardware address (e.g. of the DC551-CS31)
With "Module" the following allocation applies depending on the master:

- Module error: I/O bus or PNIO: 31 = module itself; COM1/COM2: 1...10 = expansion 1...10
- Channel error: I/O bus or PNIO = module type (1 = AI); COM1/COM2: 1...10 = expansion 1..10

In case of module errors, with channel "31 = Module itself" is output.

### State LEDs

<table>
<thead>
<tr>
<th>LED</th>
<th>State</th>
<th>Color</th>
<th>LED = OFF</th>
<th>LED = ON</th>
<th>LED flashes</th>
</tr>
</thead>
<tbody>
<tr>
<td>PWR</td>
<td>Process voltage 24 VDC via terminal</td>
<td>Green</td>
<td>CPU module voltage or external 24 VDC supply voltage is missing</td>
<td>3.3 V system voltage (I/O bus) and external 24 VDC supply voltage are present</td>
<td>---</td>
</tr>
<tr>
<td>ERR</td>
<td>Channel or module error</td>
<td>Red</td>
<td>No error or process voltage is missing</td>
<td>Severe error in the module</td>
<td>Error on 1 or more channels of the module</td>
</tr>
</tbody>
</table>

### Measuring ranges

**Risk of invalid analog input values!**

The analog input values may be invalid if the measuring range of the inputs is exceeded.

Make sure that the analog signal at the connection terminals is always within the signal range.

<table>
<thead>
<tr>
<th>Range</th>
<th>-2.5 ... +2.5 V</th>
<th>-5 ... +5 V</th>
<th>0 ... 5 V</th>
<th>0 ... 10 V mA</th>
<th>0 ... 20 mA</th>
<th>4 ... 20 mA</th>
<th>Digital value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Overflow</td>
<td>&gt;2.9397</td>
<td>&gt;5.8795</td>
<td>&gt;5.8795</td>
<td>&gt;11.7589</td>
<td>&gt;23.5178</td>
<td>&gt;22.8142</td>
<td>32767</td>
</tr>
<tr>
<td>Measured value too high</td>
<td>2.9397</td>
<td>5.8795</td>
<td>5.8795</td>
<td>11.7589</td>
<td>23.5178</td>
<td>22.8142</td>
<td>32511</td>
</tr>
<tr>
<td></td>
<td>2.5014</td>
<td>5.0029</td>
<td>5.0015</td>
<td>10.0029</td>
<td>20.0058</td>
<td>20.0058</td>
<td>27656</td>
</tr>
</tbody>
</table>
### Technical data

The System Data of AC500-eCo apply

Only additional details are therefore documented below.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Process supply voltage L+</td>
<td></td>
</tr>
<tr>
<td>Connections</td>
<td>Terminal 19 for L+ (+24 VDC) and terminal 20 for M (0 V)</td>
</tr>
<tr>
<td>Rated value</td>
<td>24 VDC</td>
</tr>
<tr>
<td>Current consumption via L+ terminal</td>
<td>0.1 A</td>
</tr>
<tr>
<td>Inrush current (at power up)</td>
<td>0.05 A²s</td>
</tr>
<tr>
<td>Max. ripple</td>
<td>5 %</td>
</tr>
<tr>
<td>Protection against reversed voltage</td>
<td>Yes</td>
</tr>
<tr>
<td>Protection fuse for L+</td>
<td>Recommended</td>
</tr>
<tr>
<td>Current consumption from 24 VDC power supply at the terminals UP/L+ and ZP/M of the CPU/bus module</td>
<td>Ca. 10 mA</td>
</tr>
<tr>
<td>Galvanic isolation</td>
<td>No</td>
</tr>
<tr>
<td>Surge-voltage (max.)</td>
<td>35 VDC for 0.5 s</td>
</tr>
<tr>
<td>Max. power dissipation within the module</td>
<td>2.7 W</td>
</tr>
</tbody>
</table>
### Parameter

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Weight</td>
<td>Ca. 120 g</td>
</tr>
<tr>
<td>Mounting position</td>
<td>Horizontal or vertical</td>
</tr>
<tr>
<td>Cooling</td>
<td>The natural convection cooling must not be hindered by cable ducts or other parts in the switch-gear cabinet.</td>
</tr>
</tbody>
</table>

#### NOTICE!

**Attention:**

All I/O channels (digital and analog) are protected against reverse polarity, reverse supply, short circuit and continuous overvoltage up to 30 VDC.

---

### Technical data of the analog inputs

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of channels per module</td>
<td>4 individually configurable voltage or current inputs</td>
</tr>
<tr>
<td>Distribution of channels into groups</td>
<td>1 (4 channels per group)</td>
</tr>
<tr>
<td>Resolution</td>
<td></td>
</tr>
<tr>
<td>Unipolar</td>
<td>Voltage: 0 V...+5 V; 0 V...+10 V: 12 bits</td>
</tr>
<tr>
<td></td>
<td>Current 0 mA...20 mA; 4 mA...20 mA: 12 bits</td>
</tr>
<tr>
<td>Bipolar</td>
<td>Voltage -2.5 V...+2.5 V; -5 V...+5 V: 11 bits plus sign</td>
</tr>
<tr>
<td>Connection of the signals I0- to I3-</td>
<td>Terminals 3, 6, 9, 12</td>
</tr>
<tr>
<td>Connection of the signals I0+ to I3+</td>
<td>Terminals 2, 5, 8, 11</td>
</tr>
<tr>
<td>Input type</td>
<td>Differential</td>
</tr>
<tr>
<td>Galvanic isolation</td>
<td>No galvanic isolation between the inputs and the I/O bus</td>
</tr>
<tr>
<td>Common mode input range</td>
<td>Signal voltage plus common mode voltage must be within ±12 V</td>
</tr>
<tr>
<td>Indication of the input signals</td>
<td>No</td>
</tr>
<tr>
<td>Channel input resistance</td>
<td>Voltage: &gt; 1 MΩ</td>
</tr>
<tr>
<td></td>
<td>Current: ca. 250 Ω</td>
</tr>
<tr>
<td>Conversion error of the analog values caused by non-linearity, adjustment error at factory and resolution within the normal range</td>
<td>Typ. ±0.5 % of full scale (voltage)</td>
</tr>
<tr>
<td></td>
<td>±0.5 % of full scale (current 0 mA...20 mA)</td>
</tr>
<tr>
<td></td>
<td>±0.7 % of full scale (current 4 mA...20 mA) at 25 °C</td>
</tr>
<tr>
<td></td>
<td>Max. ±2 % of full scale (all ranges) at 0 °C...60 °C or EMC disturbance</td>
</tr>
<tr>
<td>Time constant of the input filter</td>
<td>Voltage: 300 µs</td>
</tr>
<tr>
<td></td>
<td>Current: 300 µs</td>
</tr>
<tr>
<td>Relationship between input signal and hex code</td>
<td>Chapter 1.5.3.6.2.1.1.8 “Measuring ranges” on page 2551</td>
</tr>
</tbody>
</table>

---

2020/12/10

3ADR010583, 1, en_US

2553
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Analog to digital conversion time</td>
<td>Typ. 500 µs per channel</td>
</tr>
<tr>
<td>Unused inputs</td>
<td>Can be left open and should be configured as &quot;unused&quot;</td>
</tr>
<tr>
<td>Input data length</td>
<td>8 bytes</td>
</tr>
<tr>
<td>Overvoltage protection</td>
<td>Yes, up to 30 VDC only for voltage input</td>
</tr>
<tr>
<td>Max. cable length (conductor cross section &gt; 0.14 mm²)</td>
<td></td>
</tr>
<tr>
<td>Unshielded wire</td>
<td>10 m</td>
</tr>
<tr>
<td>Shielded wire</td>
<td>100 m</td>
</tr>
</tbody>
</table>

### Ordering data

<table>
<thead>
<tr>
<th>Part no.</th>
<th>Description</th>
<th>Product life cycle phase *)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1TNE 968 902 R1101</td>
<td>AI561, analog input module, 4 AI, U/I</td>
<td>Active</td>
</tr>
<tr>
<td>1TNE 968 901 R3101</td>
<td>Terminal block TA563-9, 9 pins, screw front, cable side, 6 pieces per unit</td>
<td>Active</td>
</tr>
<tr>
<td>1TNE 968 901 R3102</td>
<td>Terminal block TA563-11, 11 pins, screw front, cable side, 6 pieces per unit</td>
<td>Active</td>
</tr>
<tr>
<td>1TNE 968 901 R3103</td>
<td>Terminal block TA564-9, 9 pins, screw front, cable front, 6 pieces per unit</td>
<td>Active</td>
</tr>
<tr>
<td>1TNE 968 901 R3104</td>
<td>Terminal block TA564-11, 11 pins, screw front, cable front, 6 pieces per unit</td>
<td>Active</td>
</tr>
<tr>
<td>1TNE 968 901 R3105</td>
<td>Terminal block TA565-9, 9 pins, spring front, cable front, 6 pieces per unit</td>
<td>Active</td>
</tr>
<tr>
<td>1TNE 968 901 R3106</td>
<td>Terminal block TA565-11, 11 pins, spring front, cable front, 6 pieces per unit</td>
<td>Active</td>
</tr>
</tbody>
</table>

*) For planning and commissioning of new installations use modules in Active status only.

### AI562 - Analog input module

- 2 configurable analog resistance temperature detector (RTD) inputs (I0 and I1) in 1 group
- Resolution: 15 bits plus sign
1. I/O bus
2. 1 green LED to display power supply, 1 red LED to display error
3. Terminal number
4. Allocation of signal name
5. Terminal block for input signals (11-pin)
6. 2 holes for wall-mounting with screws
7. DIN rail

**Intended purpose**

The device can be used as a decentralized I/O extension module for S500 Communication Interface Modules (e.g. CI592-CS31, CI501-PNIO, CI541-DP, CI581-CN) or as a centralized extension module for AC500 CPUs (PM5xx).

The inputs are not electrically isolated from each other.

All other circuitry of the module is electrically isolated from the inputs.

*The I/O module must not be used as a decentralized I/O module with CI590-CS31-HA bus modules.*
Functionality

2 analog RTD-inputs, individually configurable for

- Not used (default)
- Pt100, -50 °C...+400 °C, 2-wire
- Pt100, -50 °C...+400 °C, 3-wire
- Pt1000, -50 °C...+400 °C, 2-wire
- Pt1000, -50 °C...+400 °C, 3-wire
- Ni1000, -50 °C...+150 °C, 2-wire
- Ni1000, -50 °C...+150 °C, 3-wire
- Ni100, -50 °C...+150 °C, 2-wire
- Ni100, -50 °C...+150 °C, 3-wire
- Analog input resistance 0 Ω...150 Ω
- Analog input resistance 0 Ω...300 Ω

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Resolution of the analog channels</td>
<td></td>
</tr>
<tr>
<td>Temperature</td>
<td>0.1 °C</td>
</tr>
<tr>
<td>LED displays</td>
<td>2 LEDs for process voltage and error messages</td>
</tr>
<tr>
<td>Internal supply</td>
<td>Via I/O bus</td>
</tr>
<tr>
<td>External supply</td>
<td>Via the terminals UP (process voltage 24 VDC) and ZP (0 VDC)</td>
</tr>
</tbody>
</table>

Electrical connection

> For a detailed description of the mounting, disassembly and electrical connection of the module, please refer to the System Assembly chapter.

The electrical connection is carried out by using a removable 11-pin terminal block. These terminal blocks differ in their connection system (spring terminals or screw terminals, cable mounting from the front or from the side). The terminal blocks are not included in the module's scope of delivery and must be ordered separately.

The following block diagram shows the internal construction of the analog inputs:
The assignment of the terminals:

<table>
<thead>
<tr>
<th>Terminal</th>
<th>Signal</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>O0+</td>
<td>Current source of channel 0</td>
</tr>
<tr>
<td>11</td>
<td>I0+</td>
<td>Sense input of channel 0</td>
</tr>
<tr>
<td>12</td>
<td>I0-</td>
<td>Return input of channel 0</td>
</tr>
<tr>
<td>13</td>
<td>O1+</td>
<td>Current source of channel 1</td>
</tr>
<tr>
<td>14</td>
<td>I1+</td>
<td>Sense input of channel 1</td>
</tr>
<tr>
<td>15</td>
<td>I1-</td>
<td>Return input of channel 1</td>
</tr>
<tr>
<td>16</td>
<td>---</td>
<td>Reserved</td>
</tr>
<tr>
<td>17</td>
<td>SG</td>
<td>Shield grounding</td>
</tr>
<tr>
<td>18</td>
<td>SG</td>
<td>Shield grounding</td>
</tr>
<tr>
<td>19</td>
<td>UP</td>
<td>Process voltage UP (24 VDC)</td>
</tr>
<tr>
<td>20</td>
<td>ZP</td>
<td>Process voltage ZP (0 VDC)</td>
</tr>
</tbody>
</table>

The internal power supply voltage for the module's circuitry is carried out via the I/O bus (provided by a bus module or a CPU). Thus, the current consumption from 24 VDC power supply at the terminals L+/UP and M/ZP of the CPU/bus module increases by 5 mA per AI562.

The external power supply connection is carried out via the UP (+24 VDC) and the ZP (0 VDC) terminals.

**NOTICE!**

Risk of imprecise and faulty measurements!

Analog signals may be distorted seriously by external electromagnetic influences.

Use shielded wires when wiring analog signal sources. The cable shield must be grounded at both ends of the cable. Provide a potential equalisation of a low resistance to avoid high potential differences between different parts of the plant.
NOTICE!
Risk of damaging the PLC modules!
The PLC modules must not be removed while the plant is connected to a power supply. Make sure that all voltage sources (supply and process voltage) are switched off before you
– connect or disconnect any signal or terminal block
– remove or replace a module.

NOTICE!
Risk of damaging the PLC modules!
Overvoltages and short circuits might damage the PLC modules.
– Make sure that all voltage sources (supply voltage and process supply voltage) are switched off before you begin with operations at the system.
– Never connect any voltages or signals to reserved terminals (marked with --). Reserved terminals may carry internal voltages.

The module provides several diagnosis functions  \(\text{Chapter 1.5.3.6.2.1.2.6 “Diagnosis” on page 2560.}\)

The following figures show the electrical connection of RTDs to the inputs of the analog input module AI562.

\[\begin{array}{|c|c|}
\hline
10 & O0+ \\
11 & I0+ \\
12 & I0- \\
13 & O1+ \\
14 & I1+ \\
15 & I1- \\
16 & --- \\
17 & SG \\
18 & SG \\
19 & UP \\
20 & ZP \\
\hline
\end{array}\]

\[\begin{array}{|c|c|}
\hline
10 & O0+ \\
11 & I0+ \\
12 & I0- \\
13 & O1+ \\
14 & I1+ \\
15 & I1- \\
16 & --- \\
17 & SG \\
18 & SG \\
19 & UP \\
20 & ZP \\
\hline
\end{array}\]

2-wires input 3-wires input

\[\text{With 2-wires connection, the resistance of the connection wires influences the accuracy of the measured value. Use 3-wires connection to achieve the guaranteed measuring accuracy.}\]

The meaning of the LEDs is described in the Displays section  \(\text{Chapter 1.5.3.6.2.1.2.7 “State LEDs” on page 2561.}\)

I/O configuration
The analog input module AI562 does not store configuration data itself.
Parameterization

The arrangement of the parameter data is performed with Automation Builder software.

The parameter data directly influences the functionality of modules.

For non-standard applications, it is necessary to adapt the parameters to your system configuration.

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>Internal value</th>
<th>Internal value, Type</th>
<th>Default</th>
<th>Min.</th>
<th>Max.</th>
<th>EDS Slot Index</th>
</tr>
</thead>
<tbody>
<tr>
<td>Module ID</td>
<td>Intern</td>
<td>6505 1)</td>
<td>WORD</td>
<td>0x1969</td>
<td>0</td>
<td>65535</td>
<td>xx01</td>
</tr>
<tr>
<td>Ignore module</td>
<td>No</td>
<td>0</td>
<td>BYTE</td>
<td>No</td>
<td>0</td>
<td>0</td>
<td>xx02 2)</td>
</tr>
<tr>
<td></td>
<td>Yes</td>
<td>1</td>
<td></td>
<td>0x00</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Parameter length</td>
<td>Intern</td>
<td>4</td>
<td>BYTE</td>
<td>0</td>
<td>0</td>
<td>255</td>
<td>xx02 2)</td>
</tr>
<tr>
<td>Check Supply</td>
<td>Off</td>
<td>0</td>
<td>BYTE</td>
<td>On</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>On</td>
<td>1</td>
<td></td>
<td>0x01</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Analog Data Format</td>
<td>Default</td>
<td>0</td>
<td>BYTE</td>
<td>Default</td>
<td>0</td>
<td>255</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0x00</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

1) with CS31 and addresses less than 70, the value is increased by 1
2) Value is hexadecimal: HighByte is slot (xx: 0...7), LowByte is index (1...n)

GSD file:

```
Ext_User_Prm_Data_Len = 0x07
Ext_User_Prm_Data_Const(0) = 0x6A, 0x19, 0x04, \
0x01, 0x00, \
0x00, 0x00;
```

Input channel (2x)

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>Internal value</th>
<th>Internal value, Type</th>
<th>Default</th>
<th>Min.</th>
<th>Max.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Channel configuration</td>
<td>see table 2)</td>
<td>see table 2)</td>
<td>BYTE</td>
<td>0</td>
<td>0</td>
<td>65535</td>
</tr>
<tr>
<td></td>
<td>see table 3)</td>
<td>see table 3)</td>
<td></td>
<td>0x00 see table 3)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 553: Channel configuration 2)

<table>
<thead>
<tr>
<th>Internal value</th>
<th>Operating modes for the analog inputs, individually configurable</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Not used (default)</td>
</tr>
<tr>
<td>8</td>
<td>2-wire Pt100, -50 °C...+400 °C</td>
</tr>
<tr>
<td>9</td>
<td>3-wire Pt100, -50 °C...+400 °C</td>
</tr>
<tr>
<td>16</td>
<td>2-wire Pt1000, -50 °C...+400 °C</td>
</tr>
</tbody>
</table>
### Internal value

<table>
<thead>
<tr>
<th>Internal value</th>
<th>Operating modes for the analog inputs, individually configurable</th>
</tr>
</thead>
<tbody>
<tr>
<td>17</td>
<td>3-wire Pt1000, -50 °C...+400 °C</td>
</tr>
<tr>
<td>18</td>
<td>2-wire Ni1000 -50 °C...+150 °C</td>
</tr>
<tr>
<td>19</td>
<td>3-wire Ni1000 -50 °C...+150 °C</td>
</tr>
<tr>
<td>22</td>
<td>2-wire Ni100, -50 °C...+150 °C</td>
</tr>
<tr>
<td>23</td>
<td>3-wire Ni100, -50 °C...+150 °C</td>
</tr>
<tr>
<td>32</td>
<td>Analog input resistor 0 Ω...150 Ω</td>
</tr>
<tr>
<td>33</td>
<td>Analog input resistor 0 Ω...300 Ω</td>
</tr>
</tbody>
</table>

### Diagnosis

<table>
<thead>
<tr>
<th>E1...E4</th>
<th>d1</th>
<th>d2</th>
<th>d3</th>
<th>d4</th>
<th>Identifier 000...063</th>
<th>AC500 display</th>
<th>&lt;- Display in</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class</td>
<td>Comp</td>
<td>Dev</td>
<td>Mod</td>
<td>Ch</td>
<td>Err</td>
<td>PS501 PLC Browser</td>
<td></td>
</tr>
<tr>
<td>Byte 6</td>
<td>-</td>
<td>Byte 3</td>
<td>Byte 4</td>
<td>Byte 5</td>
<td>Byte 6 Bit 0...5</td>
<td>PNIO diagnosis block</td>
<td></td>
</tr>
<tr>
<td>Bit 6...7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### Class Interface Device Module Channel Error Identifier Error message Remedy
1) 2) 3) 4)

### Module error

<table>
<thead>
<tr>
<th>3</th>
<th>14</th>
<th>1...10</th>
<th>31</th>
<th>31</th>
<th>19</th>
<th>Checksum error in the I/O module</th>
<th>Replace I/O module</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>11 / 12</td>
<td>ADR</td>
<td>1...10</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>14</td>
<td>1...10</td>
<td>31</td>
<td>31</td>
<td>9</td>
<td>Overflow diagnosis buffer</td>
<td>Restart</td>
</tr>
<tr>
<td></td>
<td>11 / 12</td>
<td>ADR</td>
<td>1...10</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>14</td>
<td>1...10</td>
<td>31</td>
<td>31</td>
<td>26</td>
<td>Parameter error</td>
<td>Check master</td>
</tr>
<tr>
<td></td>
<td>11 / 12</td>
<td>ADR</td>
<td>1...10</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>14</td>
<td>1...10</td>
<td>31</td>
<td>31</td>
<td>11</td>
<td>Process voltage too low</td>
<td>Check process voltage</td>
</tr>
<tr>
<td></td>
<td>11 / 12</td>
<td>ADR</td>
<td>1...10</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Channel error

<table>
<thead>
<tr>
<th>4</th>
<th>14</th>
<th>1...10</th>
<th>1</th>
<th>0...1</th>
<th>48</th>
<th>Analog value overflow at an analog input</th>
<th>Check input value or terminal</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>11 / 12</td>
<td>ADR</td>
<td>1...10</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>14</td>
<td>1...10</td>
<td>1</td>
<td>0...1</td>
<td>7</td>
<td>Analog value underflow at an analog input</td>
<td>Check input value</td>
</tr>
<tr>
<td></td>
<td>11 / 12</td>
<td>ADR</td>
<td>1...10</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Remarks:
In AC500 the following interface identifier applies:
14 = I/O bus, 11 = COM1 (e.g. CS31 bus), 12 = COM2.
The PNIO diagnosis block does not contain this identifier.

With "Device" the following allocation applies:
31 = module itself, 1...10 = decentralized communication interface module 1...10, ADR = hardware address (e.g. of the DC551-CS31)

With "Module" the following allocation applies dependent of the master:
Module error: I/O bus or PNIO: 31 = module itself; COM1/COM2: 1...10 = expansion 1...10
Channel error: I/O bus or PNIO = module type (1 = AI); COM1/COM2: 1...10 = expansion 1...10

In case of module errors, with channel "31 = Module itself" is output.

State LEDs

<table>
<thead>
<tr>
<th>LED</th>
<th>State</th>
<th>Color</th>
<th>LED = OFF</th>
<th>LED = ON</th>
<th>LED flashes</th>
</tr>
</thead>
<tbody>
<tr>
<td>PWR</td>
<td>Process voltage 24 VDC via terminal</td>
<td>Green</td>
<td>CPU module voltage or external 24 VDC supply voltage is missing</td>
<td>3.3 V system voltage (I/O bus) and external 24 VDC supply voltage are present</td>
<td>---</td>
</tr>
<tr>
<td>ERR</td>
<td>Channel or module error</td>
<td>Red</td>
<td>No error or process voltage is missing</td>
<td>Severe error in the module</td>
<td>Error on 1 or more channels of the module</td>
</tr>
</tbody>
</table>

Measuring ranges

Risk of invalid analog input values!
The analog input values may be invalid if the measuring range of the inputs is exceeded.
Make sure that the analog signal at the connection terminals is always within the signal range.

Resistance temperature detectors

<table>
<thead>
<tr>
<th>Range</th>
<th>Pt100 / Pt1000 -50 ... +400 °C</th>
<th>Ni1000 / Ni1000 -50 ... +150 °C</th>
<th>Digital value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Overflow</td>
<td>&gt; 450.0 °C</td>
<td>&gt; 160.0 °C</td>
<td>32767</td>
</tr>
<tr>
<td>Measured value</td>
<td>&gt; 450.0 °C</td>
<td></td>
<td></td>
</tr>
<tr>
<td>too high</td>
<td>450.0 °C</td>
<td>4001</td>
<td>1194</td>
</tr>
<tr>
<td></td>
<td>400.1 °C</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Risk of invalid analog input values!
The analog input values may be invalid if the measuring range of the inputs is exceeded.
Make sure that the analog signal at the connection terminals is always within the signal range.
### Resistances

<table>
<thead>
<tr>
<th>Range</th>
<th>Resistance 0 ... 150 Ω</th>
<th>Resistance 0 ... 300 Ω</th>
<th>Digital value</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>Decimal</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Overflow</td>
<td>&gt;176.383</td>
<td>&gt;352.767</td>
<td>32767</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Measured value</td>
<td>176.383</td>
<td>352.767</td>
<td>32511</td>
</tr>
<tr>
<td>too high</td>
<td>150.005</td>
<td>300.011</td>
<td>27649</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Normal range</td>
<td>150.000</td>
<td>300.000</td>
<td>27648</td>
</tr>
<tr>
<td></td>
<td>0.005</td>
<td>0.011</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

### Technical data

The System Data of AC500-eCo apply
Only additional details are therefore documented below.
### Parameter | Value
--- | ---
Process supply voltage UP | Terminal 19 for UP (+24 VDC) and terminal 20 for ZP (0 V)
Connections | 24 VDC
Rated value | 0.04 A
Current consumption | 0.05 A²s
Inrush current (at power-up) | 5 %
Max. ripple | Protection against reversed voltage
Protection fuse for UP | Yes
| Recommended
Current consumption from 24 VDC power supply at the terminals UP/L+ and ZP/M of the CPU/Bus Module | Ca. 5 mA
Galvanic isolation | Yes, between the input group and the rest of the module
Isolated groups | 1 (2 channels per group)
Surge-voltage (max.) | 35 VDC for 0.5 s
Max. power dissipation within the module | 1.1 W
Weight | Ca. 120 g
Mounting position | Horizontal or vertical
Cooling | The natural convection cooling must not be hindered by cable ducts or other parts in the switch-gear cabinet.

**NOTICE!**

**Attention:**

All I/O channels (digital and analog) are protected against reverse polarity, reverse supply, short circuit and continuous overvoltage up to 30 VDC.

---

### Technical data of the analog inputs

| Parameter | Value |
--- | ---
Number of channels per module | 2 configurable RTD (resistance temperature detector) inputs
Distribution of channels into groups | 1 (2 channels per group)
Resolution | RTD 0.1 °C / 0.1 °F
 | Resistance 15 bits + sign
Connection of the signals O0+ and O1+ | Terminals 10 and 13
Connection of the signals I0- and I1- | Terminals 11 and 14
Connection of the signals I0+ and I1+ | Terminals 12 and 15
Input type | Module ground referenced RTD for 2-wire and 3-wire resistance temperature detectors
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Galvanic isolation</td>
<td>Against internal power supply and other modules</td>
</tr>
<tr>
<td>Input ranges</td>
<td>Pt100, Pt1000, Ni100, Ni1000</td>
</tr>
<tr>
<td>Indication of the input signals</td>
<td>No</td>
</tr>
<tr>
<td>Module update time</td>
<td>All channels: &lt; 1 s</td>
</tr>
<tr>
<td>Channel input resistance</td>
<td>&gt; 100 kΩ</td>
</tr>
<tr>
<td>Input filter attenuation</td>
<td>-3 dB at 3.6 kHz</td>
</tr>
<tr>
<td>Conversion error of the analog values</td>
<td>Typ. Depending on RTD max. ±0.6 % of full scale (guaranteed for 3-wires connection only) at 25 °C</td>
</tr>
<tr>
<td>caused by non-linearity, adjustment error at factory and resolution within the normal range</td>
<td>Max. ±2 % of full scale (guaranteed for 3-wires connection only) at 0 °C...60 °C or EMC disturbances</td>
</tr>
<tr>
<td>Measuring range</td>
<td>% Chapter 1.5.3.6.2.1.2.8 “Measuring ranges” on page 2561</td>
</tr>
<tr>
<td>Analog to digital conversion time</td>
<td>Typ. 140 ms per channel</td>
</tr>
<tr>
<td>Unused inputs</td>
<td>Can be left open and should be configured as &quot;unused&quot;</td>
</tr>
<tr>
<td>Input data length</td>
<td>4 bytes</td>
</tr>
<tr>
<td>Power dissipation inside the sensor</td>
<td>1 mW</td>
</tr>
<tr>
<td>Suppression of interference</td>
<td>On request</td>
</tr>
<tr>
<td>Maximum input voltage</td>
<td>30 VDC (sense), 5 VDC (source)</td>
</tr>
<tr>
<td>Basic error (resistance)</td>
<td>0.1 % of full-scale</td>
</tr>
<tr>
<td>Repeatability</td>
<td>0.05 % of full-scale</td>
</tr>
<tr>
<td>Overvoltage protection</td>
<td>Yes, up to 30 VDC</td>
</tr>
<tr>
<td>Wire loop resistance</td>
<td>&lt; 20 Ω</td>
</tr>
<tr>
<td>Max. cable length (conductor cross section &gt; 0.14 mm²)</td>
<td></td>
</tr>
<tr>
<td>Unshielded wire</td>
<td>10 m</td>
</tr>
<tr>
<td>Shielded wire</td>
<td>100 m</td>
</tr>
</tbody>
</table>

### Ordering data

<table>
<thead>
<tr>
<th>Part no.</th>
<th>Description</th>
<th>Product life cycle phase * )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1TNE 968 902 R1102</td>
<td>AI562, analog input module, 2 AI, RTD</td>
<td>Active</td>
</tr>
<tr>
<td>1TNE 968 901 R3102</td>
<td>Terminal block TA563-11, 11 pins, screw front, cable side, 6 pieces per unit</td>
<td>Active</td>
</tr>
</tbody>
</table>
### AI563 - Analog input module

- 4 configurable thermocouple (TC) / -80 mV...+80 mV inputs (I0 to I3) in 1 group
- Resolution: 15 bits plus sign

---

**Table:**

<table>
<thead>
<tr>
<th>Part no.</th>
<th>Description</th>
<th>Product life cycle phase *)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1TNE 968 901 R3104</td>
<td>Terminal block TA564-11, 11 pins, screw front, cable front, 6 pieces per unit</td>
<td>Active</td>
</tr>
<tr>
<td>1TNE 968 901 R3106</td>
<td>Terminal block TA565-11, 11 pins, spring front, cable front, 6 pieces per unit</td>
<td>Active</td>
</tr>
</tbody>
</table>

*) For planning and commissioning of new installations use modules in Active status only.

---

**Diagram Notes:**

1. I/O bus
2. 1 green LED to display power supply, 1 red LED to display error
3. Terminal number
Intended purpose

The device can be used as a decentralized I/O extension module for S500 Communication Interface Modules (e.g. CI592-CS31, CI501-PNIO, CI541-DP, CI581-CN) or as a centralized extension module for AC500 CPUs (PM5xx).

The inputs are group-wise electrically isolated from each other.

The other electronic circuitry of the module is electrically isolated from the inputs.

The I/O module must not be used as a decentralized I/O module with CI590-CS31-HA bus modules.

Functionality

4 analog TC inputs, individually configurable for
- Not used (default)
- Voltage -80 mV ... + 80 mV
- Thermocouple J-type -210 °C...+1200 °C
- Thermocouple K-type -270 °C...+1372 °C
- Thermocouple R-type -50 °C...+1768 °C
- Thermocouple S-type -50 °C...+1768 °C
- Thermocouple T-type -270 °C...+400 °C
- Thermocouple E-type -270 °C...+1000 °C
- Thermocouple N-type -270 °C...+1300 °C

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Resolution of the analog channels</td>
<td></td>
</tr>
<tr>
<td>Temperature</td>
<td>0.1 °C</td>
</tr>
<tr>
<td>LED displays</td>
<td>2 LEDs for process voltage and error messages</td>
</tr>
<tr>
<td>Internal supply</td>
<td>Via I/O bus</td>
</tr>
<tr>
<td>External supply</td>
<td>Via the terminals UP (process voltage 24 VDC) and ZP (0 VDC)</td>
</tr>
</tbody>
</table>

Electrical connection

For a detailed description of the mounting, disassembly and electrical connection of the module, please refer to the System Assembly chapter.
After powering up the system, input channels, which are configured will have undefined values/diagnosis message for typically 45 seconds, if the wires of all configured channels are broken.

If the AI563 is connected to a PROFINET Bus Module, the firmware version of PROFINET Bus Modules must be 1.2 or above.

The electrical connection is carried out by using a removable 9-pin and 11-pin terminal block. These terminal blocks differ in their connection system (spring terminals or screw-type terminals, cable mounting from the front or from the side). The terminal blocks are not included in the module's scope of delivery and must be ordered separately.

The following block diagram shows the internal construction of the analog inputs:

```
    I0+ 1
  /       \
 /         \  
I0- 2
    \       /  
      \     /   
      I1+ 3
      |     |   |
      |     |   |
      I1- 4

    I2+ 5
  /       \
 /         \  
I2- 6
    \       /  
      \     /   
      I3+ 7
      |     |   |
      |     |   |
      I3- 8

    --- 9
  /       \
 /         \  
 --- 10
  /       \
 --- 11
  /       \
 --- 12
  /       \
 --- 13
  /       \
 --- 14

SG 15
SG 16
SG 17
SG 18

UP 19

ZP 20
```

The assignment of the terminals:

<table>
<thead>
<tr>
<th>Terminal</th>
<th>Signal</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>I0+</td>
<td>Plus pole of channel 0</td>
</tr>
<tr>
<td>2</td>
<td>I0-</td>
<td>Minus pole of channel 0</td>
</tr>
<tr>
<td>Terminal</td>
<td>Signal</td>
<td>Description</td>
</tr>
<tr>
<td>---------</td>
<td>--------</td>
<td>----------------------------------</td>
</tr>
<tr>
<td>3</td>
<td>I1+</td>
<td>Plus pole of channel 1</td>
</tr>
<tr>
<td>4</td>
<td>I1-</td>
<td>Minus pole of channel 1</td>
</tr>
<tr>
<td>5</td>
<td>I2+</td>
<td>Plus pole of channel 2</td>
</tr>
<tr>
<td>6</td>
<td>I2-</td>
<td>Minus pole of channel 2</td>
</tr>
<tr>
<td>7</td>
<td>I3+</td>
<td>Plus pole of channel 3</td>
</tr>
<tr>
<td>8</td>
<td>I3-</td>
<td>Minus pole of channel 3</td>
</tr>
<tr>
<td>9</td>
<td>---</td>
<td>Reserved</td>
</tr>
<tr>
<td>10</td>
<td>---</td>
<td>Reserved</td>
</tr>
<tr>
<td>11</td>
<td>---</td>
<td>Reserved</td>
</tr>
<tr>
<td>12</td>
<td>---</td>
<td>Reserved</td>
</tr>
<tr>
<td>13</td>
<td>---</td>
<td>Reserved</td>
</tr>
<tr>
<td>14</td>
<td>---</td>
<td>Reserved</td>
</tr>
<tr>
<td>15</td>
<td>SG</td>
<td>Shield grounding</td>
</tr>
<tr>
<td>16</td>
<td>SG</td>
<td>Shield grounding</td>
</tr>
<tr>
<td>17</td>
<td>SG</td>
<td>Shield grounding</td>
</tr>
<tr>
<td>18</td>
<td>SG</td>
<td>Shield grounding</td>
</tr>
<tr>
<td>19</td>
<td>UP</td>
<td>Process voltage UP (24 VDC)</td>
</tr>
<tr>
<td>20</td>
<td>ZP</td>
<td>Process voltage ZP (0 VDC)</td>
</tr>
</tbody>
</table>

The internal power supply voltage for the module’s circuitry is carried out via the I/O bus (provided by a bus module or a CPU). Thus, the current consumption from 24 VDC power supply at the terminals UP/L+ and ZP/M of the CPU/bus module increases by 5 mA per AI563.

The external power supply connection is carried out via the UP (+24 VDC) and the ZP (0 VDC) terminals.

**NOTICE!**
**Risk of imprecise and faulty measurements!**
Analog signals may be distorted seriously by external electromagnetic influences.

Use shielded wires when wiring analog signal sources. The cable shield must be grounded at both ends of the cable. Provide a potential equalisation of a low resistance to avoid high potential differences between different parts of the plant.

**NOTICE!**
**Risk of damaging the PLC modules!**
The PLC modules must not be removed while the plant is connected to a power supply.
Make sure that all voltage sources (supply and process voltage) are switched off before you
- connect or disconnect any signal or terminal block
- remove or replace a module.
NOTICE!

Risk of damaging the PLC modules!

Overvoltages and short circuits might damage the PLC modules.
- Make sure that all voltage sources (supply voltage and process supply voltage) are switched off before you begin with operations at the system.
- Never connect any voltages or signals to reserved terminals (marked with ---). Reserved terminals may carry internal voltages.

The module provides several diagnosis functions "Diagnosis" on page 2571.

The following figure shows the electrical connection of thermocouples to the inputs of the module:

The meaning of the LEDs is described in Displays "State LEDs" on page 2572 chapter.
I/O configuration

The analog input module AI563 does not store configuration data itself.

Parameterization

The arrangement of the parameter data is performed with Automation Builder software. The parameter data directly influences the functionality of modules. For non-standard applications, it is necessary to adapt the parameters to your system configuration.

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>Internal value</th>
<th>Internal value, Type</th>
<th>Default</th>
<th>Min.</th>
<th>Max.</th>
<th>EDS Slot Index</th>
</tr>
</thead>
<tbody>
<tr>
<td>Module ID</td>
<td>Intern</td>
<td>6510 1)</td>
<td>WORD</td>
<td>0x196E</td>
<td>0</td>
<td>65535</td>
<td>xx01</td>
</tr>
<tr>
<td>Ignore module</td>
<td>No</td>
<td>0</td>
<td>BYTE</td>
<td>No</td>
<td>0x00</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Yes</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Parameter length</td>
<td>Intern</td>
<td>6</td>
<td>BYTE</td>
<td>0</td>
<td>0</td>
<td>255</td>
<td>xx02 2)</td>
</tr>
<tr>
<td>Check Supply</td>
<td>Off</td>
<td>0</td>
<td>BYTE</td>
<td>On</td>
<td>0x01</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>On</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Analog Data Format</td>
<td>Default</td>
<td>0</td>
<td>BYTE</td>
<td>Default</td>
<td>0x00</td>
<td>255</td>
<td></td>
</tr>
</tbody>
</table>

1) with CS31 and addresses less than 70, the value is increased by 1
2) Value is hexadecimal: HighByte is slot (xx: 0...7), LowByte is index (1...n)

GSD file:

```plaintext
Ext_User_Prm_Data_Len = 0x09
Ext_User_Prm_Data_Const(0) = 0x6F, 0x19, 0x06, \n0x01, 0x00, \n0x00, 0x00, 0x00, 0x00;
```

Input channel (4x)

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>Internal value</th>
<th>Internal value, Type</th>
<th>Default</th>
<th>Min.</th>
<th>Max.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Channel configuration</td>
<td>see table 2)</td>
<td>see table 2)</td>
<td>BYTE</td>
<td>0</td>
<td>0x00</td>
<td>65535</td>
</tr>
</tbody>
</table>
### Table 554: Channel configuration

<table>
<thead>
<tr>
<th>Internal value</th>
<th>Operating modes for the analog inputs, individually configurable</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Not used (default)</td>
</tr>
<tr>
<td>21</td>
<td>Voltage -80 mV...+80 mV</td>
</tr>
<tr>
<td>24</td>
<td>Thermocouple J-type -210 °C...+1200 °C</td>
</tr>
<tr>
<td>25</td>
<td>Thermocouple K-type -270 °C...+1372 °C</td>
</tr>
<tr>
<td>26</td>
<td>Thermocouple R-type -50 °C...+1768 °C</td>
</tr>
<tr>
<td>27</td>
<td>Thermocouple S-type -50 °C...+1768 °C</td>
</tr>
<tr>
<td>28</td>
<td>Thermocouple T-type -270 °C...+400 °C</td>
</tr>
<tr>
<td>29</td>
<td>Thermocouple E-type -270 °C...+1000 °C</td>
</tr>
<tr>
<td>30</td>
<td>Thermocouple N-type -270 °C...+1300 °C</td>
</tr>
</tbody>
</table>

### Diagnosis

<table>
<thead>
<tr>
<th>E1...E4</th>
<th>d1</th>
<th>d2</th>
<th>d3</th>
<th>d4</th>
<th>Identifier 000...063</th>
<th>AC500 display</th>
<th>&lt;- Display in PS501 PLC browser</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class</td>
<td>Comp</td>
<td>Dev</td>
<td>Mod</td>
<td>Ch</td>
<td>Err</td>
<td>PNIO diagnosis block</td>
<td></td>
</tr>
<tr>
<td>Byte 6</td>
<td>Byte 3</td>
<td>Byte 4</td>
<td>Byte 5</td>
<td>Byte 6 Bit 0...5</td>
<td>PNIO diagnosis block</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bit 6...7</td>
<td>-</td>
<td>Byte 6 Bit 0...5</td>
<td>PNIO diagnosis block</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Class</th>
<th>Interface</th>
<th>Device</th>
<th>Module</th>
<th>Module</th>
<th>Channel</th>
<th>Error Identifier</th>
<th>Error message</th>
<th>Remedy</th>
</tr>
</thead>
<tbody>
<tr>
<td>1)</td>
<td>2)</td>
<td>3)</td>
<td>4)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### Module error

<table>
<thead>
<tr>
<th>3</th>
<th>14</th>
<th>1...10</th>
<th>31</th>
<th>31</th>
<th>19</th>
<th>Checksum error in the I/O module</th>
<th>Replace I/O module</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>14</td>
<td>1...10</td>
<td>31</td>
<td>31</td>
<td>9</td>
<td>Overflow diagnosis buffer</td>
<td>Restart</td>
</tr>
<tr>
<td>3</td>
<td>14</td>
<td>1...10</td>
<td>31</td>
<td>31</td>
<td>26</td>
<td>Parameter error</td>
<td>Check master</td>
</tr>
<tr>
<td>3</td>
<td>14</td>
<td>1...10</td>
<td>31</td>
<td>31</td>
<td>11</td>
<td>Process voltage too low</td>
<td>Check process voltage</td>
</tr>
</tbody>
</table>

#### Channel error

<table>
<thead>
<tr>
<th>4</th>
<th>14</th>
<th>1...10</th>
<th>1</th>
<th>0...3</th>
<th>48</th>
<th>Analog value overflow or broken wire at an analog input</th>
<th>Check input value or terminal</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>14</td>
<td>1...10</td>
<td>1</td>
<td>0...3</td>
<td>7</td>
<td>Analog value underflow at an analog input</td>
<td>Check input value</td>
</tr>
</tbody>
</table>

### Remarks:

- PLC Automation with V3 CPUs
- PLC integration > Device specifications

2020/12/10

3ADR010583, 1, en_US

2571
1) In AC500 the following interface identifier applies:
14 = I/O bus, 11 = COM1 (e.g. CS31-Bus), 12 = COM2.
The PNIO diagnosis block does not contain this identifier.

2) With "Device" the following allocation applies:
31 = module itself, 1...10 = expansion module 1...10, ADR = hardware address (e.g. of the DC551-CS31)

3) With "Module" the following allocation applies dependent of the master:
Module error: I/O bus or PNIO: 31 = module itself; COM1/COM2: 1...10 = expansion 1...10
Channel error: I/O bus or PNIO = module type (1 = AI); COM1/COM2: 1...10 = expansion 1...10

4) In case of module errors, with channel "31 = Module itself" is output.

State LEDs

<table>
<thead>
<tr>
<th>LED</th>
<th>State</th>
<th>Color</th>
<th>LED = OFF</th>
<th>LED = ON</th>
<th>LED flashes</th>
</tr>
</thead>
<tbody>
<tr>
<td>PWR</td>
<td>Process voltage 24 VDC via terminal</td>
<td>Green</td>
<td>CPU module voltage or external 24 VDC supply voltage is missing</td>
<td>3.3 V system voltage (I/O bus) and external 24 VDC supply voltage are present</td>
<td>---</td>
</tr>
<tr>
<td>ERR</td>
<td>Channel or module error</td>
<td>Red</td>
<td>No error or process voltage is missing</td>
<td>Severe error in the module</td>
<td>Error on 1 or more channels of the module</td>
</tr>
</tbody>
</table>

Measuring ranges

AI563 needs 4 seconds for initialization after applying the process supply voltage to clamp UP/ZP. During these 4 seconds, the measurement values are set to '0'. After that, valid measurement values are provided by the module.

After an interruption of the process supply voltage > 10 ms, a re-initialization is performed by AI563.

Risk of invalid analog input values!
The analog input values may be invalid if the measuring range of the inputs is exceeded.

Make sure that the analog signal at the connection terminals is always within the signal range.

When a wire break occurs on a sensor wire, the temperature measurement value of the corresponding channel changes to Overflow (Hexadecimal 7FFF).
### Range

<table>
<thead>
<tr>
<th>Type</th>
<th>-210 ... +1200 °C</th>
<th>-270 ... +1372 °C</th>
<th>-270 ... +1300 °C</th>
<th>-270 ... +400 °C</th>
<th>Digital value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type J</td>
<td>&gt; 1200.0 °C</td>
<td>&gt; 1372.0 °C</td>
<td>&gt; 1300.0 °C</td>
<td>&gt; 400.0 °C</td>
<td>32767 7FFF</td>
</tr>
<tr>
<td>Type K</td>
<td>-270.0 °C</td>
<td>:</td>
<td>1372.0 °C</td>
<td>13720 5958</td>
<td></td>
</tr>
<tr>
<td>Type N</td>
<td>1200.0 °C</td>
<td>:</td>
<td>1300.0 °C</td>
<td>13000 32C8</td>
<td></td>
</tr>
<tr>
<td>Type T</td>
<td>:</td>
<td>:</td>
<td>400.0 °C</td>
<td>4000 0FA0</td>
<td></td>
</tr>
<tr>
<td>Normal range</td>
<td>0.1 °C</td>
<td>:</td>
<td>0.1 °C</td>
<td>:</td>
<td>1 1</td>
</tr>
<tr>
<td></td>
<td>:</td>
<td>:</td>
<td>0.1 °C</td>
<td>:</td>
<td>0 0000</td>
</tr>
<tr>
<td></td>
<td>:</td>
<td>:</td>
<td>-0.1 °C</td>
<td>:</td>
<td>1 -1 FFFF</td>
</tr>
<tr>
<td></td>
<td>:</td>
<td>:</td>
<td>-210.0 °C</td>
<td>:</td>
<td>:</td>
</tr>
<tr>
<td></td>
<td>:</td>
<td>:</td>
<td>-270.0 °C</td>
<td>:</td>
<td>:</td>
</tr>
<tr>
<td>Underflow</td>
<td>&lt; -210.0 °C</td>
<td>&lt; -270.0 °C</td>
<td>&lt; -270.0 °C</td>
<td>&lt; -270.0 °C</td>
<td>-32768 8000</td>
</tr>
</tbody>
</table>

### Range

<table>
<thead>
<tr>
<th>-80 mV ... +80 mV</th>
<th>Type E</th>
<th>-270 ... +1000 °C</th>
<th>Types R, S</th>
<th>-50 ... +1768 °C</th>
<th>Digital value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Overflow</td>
<td>&gt; +90 mV</td>
<td>&gt; 1000.0 °C</td>
<td>&gt; 17680.0 °C</td>
<td>32767 7FFF</td>
<td></td>
</tr>
<tr>
<td>Normal range</td>
<td>+80 mV</td>
<td>1768.0 °C</td>
<td>17680 4510</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1000.0 °C</td>
<td>10000 2710</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>9000</td>
<td>2328</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>3 µV</td>
<td>0.1 °C</td>
<td>0.1 °C</td>
<td>1 1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0 µV</td>
<td>0.0 °C</td>
<td>0.0 °C</td>
<td>0 0000</td>
<td></td>
</tr>
<tr>
<td></td>
<td>-3 µV</td>
<td>-0.1 °C</td>
<td>-0.1 °C</td>
<td>-1 FFFF</td>
<td></td>
</tr>
<tr>
<td></td>
<td>:</td>
<td>:</td>
<td>:</td>
<td>:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>:</td>
<td>:</td>
<td>-50.0 °C</td>
<td>-500 FE0C</td>
<td></td>
</tr>
<tr>
<td></td>
<td>:</td>
<td>-270.0 °C</td>
<td>-2700 F574</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Underflow</td>
<td>&lt; -90 mV</td>
<td>&lt; -270.0 °C</td>
<td>&lt; -50.0 °C</td>
<td>-32768 8000</td>
<td></td>
</tr>
</tbody>
</table>

### Technical data

The System Data of AC500-eCo apply
Only additional details are therefore documented below.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Process supply voltage UP</td>
<td></td>
</tr>
<tr>
<td>Connections</td>
<td>Terminal 19 for UP (+24 VDC) and terminal 20 for ZP (0 V)</td>
</tr>
<tr>
<td>Rated value</td>
<td>24 VDC</td>
</tr>
<tr>
<td>Current consumption</td>
<td>0.10 A</td>
</tr>
<tr>
<td>Inrush current (at power-up)</td>
<td>0.07 A*s</td>
</tr>
<tr>
<td>Max. ripple</td>
<td>5 %</td>
</tr>
<tr>
<td>Protection against reversed voltage</td>
<td>Yes</td>
</tr>
<tr>
<td>Rated protection fuse for UP</td>
<td>Not necessary</td>
</tr>
<tr>
<td>Current consumption from 24 VDC power supply at the terminals UP/L+ and ZP/M of the CPU/bus module</td>
<td>Ca. 5 mA</td>
</tr>
<tr>
<td>Galvanic isolation</td>
<td>Yes, between the channels and the rest of the module</td>
</tr>
<tr>
<td>Isolated groups</td>
<td>1 (4 channels per group)</td>
</tr>
<tr>
<td>Surge-voltage (max.)</td>
<td>35 VDC for 0.5 s</td>
</tr>
<tr>
<td>Max. power dissipation within the module</td>
<td>2.6 W</td>
</tr>
<tr>
<td>Weight</td>
<td>Ca. 120 g</td>
</tr>
<tr>
<td>Mounting position</td>
<td>Horizontal or vertical</td>
</tr>
<tr>
<td>Cooling</td>
<td>The natural convection cooling must not be hindered by cable ducts or other parts in the switch-gear cabinet.</td>
</tr>
</tbody>
</table>

**NOTICE!**

**Attention:**

All I/O channels (digital and analog) are protected against reverse polarity, reverse supply, short circuit and continuous overvoltage up to 30 VDC.

### Technical data of the analog inputs

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of channels per module</td>
<td>4 configurable thermocouple (TC) inputs</td>
</tr>
<tr>
<td>Distribution of channels into groups</td>
<td>1 (4 channels per group)</td>
</tr>
<tr>
<td>Resolution</td>
<td></td>
</tr>
<tr>
<td>Temperature</td>
<td>0.1 °C</td>
</tr>
<tr>
<td>Voltage</td>
<td>15 bits plus sign</td>
</tr>
<tr>
<td>Connection of the signals I0+ to I3+</td>
<td>Terminals 1, 3, 5 and 7</td>
</tr>
<tr>
<td>Connection of the signals I0- to I3-</td>
<td>Terminals 2, 4, 6 and 8</td>
</tr>
<tr>
<td>Input type</td>
<td>Floating thermocouple</td>
</tr>
<tr>
<td>Galvanic isolation</td>
<td>Against internal power supply and other modules</td>
</tr>
<tr>
<td>Parameter</td>
<td>Value</td>
</tr>
<tr>
<td>-----------------------------------------</td>
<td>----------------------------------------------------------------------</td>
</tr>
<tr>
<td>Common mode rejection</td>
<td>&gt; 120 dB at 120 VAC</td>
</tr>
<tr>
<td>Indication of the input signals</td>
<td>No</td>
</tr>
<tr>
<td>Module update time</td>
<td>All channels: &lt; 1.6 s</td>
</tr>
<tr>
<td>Channel input resistance</td>
<td>On request</td>
</tr>
<tr>
<td>Input filter attenuation</td>
<td>-3 dB at 15 kHz</td>
</tr>
<tr>
<td>Cold junction error</td>
<td>±1.5 °C</td>
</tr>
<tr>
<td>Conversion error of the analog values</td>
<td>Typ. 0.1 % of full-scale (voltage)</td>
</tr>
<tr>
<td></td>
<td>Depending on thermocouple, see table</td>
</tr>
<tr>
<td></td>
<td>§ Chapter 1.5.3.6.2.1.3.9.1.1 “Accuracy of thermocouple ranges at 25 °C (with cold junction compensation)” on page 2575</td>
</tr>
<tr>
<td></td>
<td>at 25 °C</td>
</tr>
<tr>
<td></td>
<td>Max. ±2 % of full scale (T-Type: ±3 % for -240 °C...-270 °C)</td>
</tr>
<tr>
<td></td>
<td>at 0 °C...60 °C</td>
</tr>
<tr>
<td>Relationship between input signal and</td>
<td>§ Chapter 1.5.3.6.2.1.3.8 “Measuring ranges” on page 2572</td>
</tr>
<tr>
<td>hex code</td>
<td></td>
</tr>
<tr>
<td>Analog to digital conversion time</td>
<td>400 ms per channel</td>
</tr>
<tr>
<td>Unused inputs</td>
<td>Can be left open and should be configured as &quot;unused&quot;</td>
</tr>
<tr>
<td>Input data length</td>
<td>8 bytes</td>
</tr>
<tr>
<td>Overvoltage protection</td>
<td>Yes, up to 30 VDC</td>
</tr>
<tr>
<td>Repeatability</td>
<td>On request</td>
</tr>
<tr>
<td>Wire loop resistance</td>
<td>&lt; 100 Ω</td>
</tr>
<tr>
<td>Max. cable length (conductor cross</td>
<td>Unshielded wire 10 m</td>
</tr>
<tr>
<td>section &gt; 0.14 mm²)</td>
<td>Shielded wire 100 m</td>
</tr>
</tbody>
</table>

Accuracy of thermocouple ranges at 25 °C (with cold junction compensation)

<table>
<thead>
<tr>
<th>Thermocouple Type</th>
<th>Range</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>E</td>
<td>-270 °C...-220 °C</td>
<td>±2 %</td>
</tr>
<tr>
<td></td>
<td>-220 °C...+1000 °C</td>
<td>±0.6 %</td>
</tr>
<tr>
<td>J</td>
<td>-210 °C...+1200 °C</td>
<td>±0.6 %</td>
</tr>
<tr>
<td>K</td>
<td>-270 °C...-220 °C</td>
<td>±1.5 %</td>
</tr>
<tr>
<td></td>
<td>-220 °C...+1372 °C</td>
<td>±0.6 %</td>
</tr>
<tr>
<td>N</td>
<td>-270 °C...-150 °C</td>
<td>±2 %</td>
</tr>
<tr>
<td></td>
<td>-150 °C...+1300 °C</td>
<td>±0.6 %</td>
</tr>
<tr>
<td>R</td>
<td>-50 °C...+150 °C</td>
<td>±1.5 %</td>
</tr>
<tr>
<td></td>
<td>+150 °C...+1768 °C</td>
<td>±0.6 %</td>
</tr>
<tr>
<td>Thermocouple Type</td>
<td>Range</td>
<td>Accuracy</td>
</tr>
<tr>
<td>-------------------</td>
<td>---------------------------</td>
<td>----------------</td>
</tr>
<tr>
<td>S</td>
<td>-50 °C...+150 °C</td>
<td>±1.5 %</td>
</tr>
<tr>
<td></td>
<td>+150 °C...+1768 °C</td>
<td>±0.6 %</td>
</tr>
<tr>
<td>T</td>
<td>-270 °C...-240 °C</td>
<td>±3 %</td>
</tr>
<tr>
<td></td>
<td>-240 °C...0 °C</td>
<td>±2 %</td>
</tr>
<tr>
<td></td>
<td>0 °C...+400 °C</td>
<td>±0.6 %</td>
</tr>
</tbody>
</table>

These accuracy values are valid only for stable module temperatures.

Ordering data

<table>
<thead>
<tr>
<th>Part no.</th>
<th>Description</th>
<th>Product life cycle phase *)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1TNE 968 902 R1103</td>
<td>AI563, analog input module, 4 AI, thermocouple</td>
<td>Active</td>
</tr>
<tr>
<td>1TNE 968 901 R3101</td>
<td>Terminal block TA563-9, 9 pins, screw front, cable side, 6 pieces per unit</td>
<td>Active</td>
</tr>
<tr>
<td>1TNE 968 901 R3102</td>
<td>Terminal block TA563-11, 11 pins, screw front, cable side, 6 pieces per unit</td>
<td>Active</td>
</tr>
<tr>
<td>1TNE 968 901 R3103</td>
<td>Terminal block TA564-9, 9 pins, screw front, cable front, 6 pieces per unit</td>
<td>Active</td>
</tr>
<tr>
<td>1TNE 968 901 R3104</td>
<td>Terminal block TA564-11, 11 pins, screw front, cable front, 6 pieces per unit</td>
<td>Active</td>
</tr>
<tr>
<td>1TNE 968 901 R3105</td>
<td>Terminal block TA565-9, 9 pins, spring front, cable front, 6 pieces per unit</td>
<td>Active</td>
</tr>
<tr>
<td>1TNE 968 901 R3106</td>
<td>Terminal block TA565-11, 11 pins, spring front, cable front, 6 pieces per unit</td>
<td>Active</td>
</tr>
</tbody>
</table>

*) For planning and commissioning of new installations use modules in Active status only.

AO561 - Analog output module
- 2 configurable analog outputs (O0 and O1) in 1 group
- Resolution: 11 bits plus sign or 12 bit
1 I/O bus
2 1 green LED to display power supply, 1 red LED to display error
3 Terminal number
4 Allocation of signal name
5 Terminal block for output signals (11-pin)
6 2 holes for wall-mounting with screws
7 DIN rail

**Intended purpose**

The device can be used as a decentralized I/O extension module for S500 Communication Interface Modules (e.g. CI592-CS31, CI501-PNIO, CI541-DP, CI581-CN) or as a centralized extension module for AC500 CPUs (PM5xx).

The outputs are not electrically isolated from each other.

The other electronic circuitry of the module is not electrically isolated from the outputs or from the I/O bus.

*The I/O module must not be used as communication interface module at CI590-CS31-HA bus modules.*
Functionality

2 analog outputs, individually configurable for

- Not used (default setting)
- -10 V...+10 V
- 0 mA...20 mA
- 4 mA...20 mA

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Resolution of the analog channels</td>
<td></td>
</tr>
<tr>
<td>Voltage bipolar (-10 V...+10 V)</td>
<td>11 bits plus sign</td>
</tr>
<tr>
<td>Current (0 mA...20 mA; 4 mA...20 mA)</td>
<td>12 bits</td>
</tr>
<tr>
<td>LED displays</td>
<td>2 LEDs for process voltage and error messages</td>
</tr>
<tr>
<td>Internal supply</td>
<td>Via I/O bus</td>
</tr>
<tr>
<td>External supply</td>
<td>Via the terminals L+ (process voltage 24 VDC) and M (0 VDC); the M terminal is connected to the M terminal of the CPU via the I/O bus</td>
</tr>
</tbody>
</table>

Electrical connection

For a detailed description of the mounting, disassembly and electrical connection of the module, please refer to the System Assembly chapter.

If the output is configured as not used, the voltage and current output signals are undefined and must not be connected.

The electrical connection is carried out by using a removable 11-pin terminal block. These terminal blocks differ in their connection system (spring terminals or screw-type terminals, cable mounting from the front or from the side). The terminal blocks are not included in the module's scope of delivery and must be ordered separately.

The following block diagram shows the internal construction of the analog outputs:
The assignment of the terminals:

<table>
<thead>
<tr>
<th>Terminal</th>
<th>Signal</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>---</td>
<td>Reserved</td>
</tr>
<tr>
<td>11</td>
<td>---</td>
<td>Reserved</td>
</tr>
<tr>
<td>12</td>
<td>---</td>
<td>Reserved</td>
</tr>
<tr>
<td>13</td>
<td>O0U+</td>
<td>Voltage output of channel 0</td>
</tr>
<tr>
<td>14</td>
<td>O0I+</td>
<td>Current output of channel 0</td>
</tr>
<tr>
<td>15</td>
<td>O1U+</td>
<td>Voltage output of channel 1</td>
</tr>
<tr>
<td>16</td>
<td>O1I+</td>
<td>Current output of channel 1</td>
</tr>
<tr>
<td>17</td>
<td>O01-</td>
<td>Negative pole of channels O0 and O1</td>
</tr>
<tr>
<td>18</td>
<td>SG</td>
<td>Shield grounding</td>
</tr>
<tr>
<td>19</td>
<td>L+</td>
<td>Process voltage L+ (24 VDC)</td>
</tr>
<tr>
<td>20</td>
<td>M</td>
<td>Process voltage M (0 VDC)</td>
</tr>
</tbody>
</table>

The internal power supply voltage for the module's circuitry is carried out via the I/O bus (provided by a bus module or a CPU). Thus, the current consumption from 24 VDC power supply at the terminals UP/L+ and ZP/M of the CPU/bus module increases by 5 mA per AO561.

The external power supply connection is carried out via the L+ (+24 VDC) and the M (0 VDC) terminals. The M terminal is electrically interconnected to the M/ZP terminal of the CPU/bus module.

**NOTICE!**

**Risk of imprecise and faulty measurements!**

Analog signals may be distorted seriously by external electromagnetic influences.

Use shielded wires when wiring analog signal sources. The cable shield must be grounded at both ends of the cable. Provide a potential equalisation of a low resistance to avoid high potential differences between different parts of the plant.
**NOTICE!**

**Risk of damaging the PLC modules!**
The PLC modules must not be removed while the plant is connected to a power supply.

Make sure that all voltage sources (supply and process voltage) are switched off before you
– connect or disconnect any signal or terminal block
– remove or replace a module.

**NOTICE!**

**Risk of damaging the PLC modules!**
Overvoltages and short circuits might damage the PLC modules.

– Make sure that all voltage sources (supply voltage and process supply voltage) are switched off before you begin with operations at the system.
– Never connect any voltages or signals to reserved terminals (marked with ---). Reserved terminals may carry internal voltages.

The module provides several diagnosis functions ↳ Chapter 1.5.3.6.2.1.4.6 “Diagnosis” on page 2582.

The following figures show the electrical connection of analog actuators to the analog output module AO561.

---

The output signal is undefined if the supply voltage at the L+ terminal is below 10 V. This can, for example, occur if the supply voltage has a slow ramp-up / ramp-down behaviour and must be foreseen when planning the installation.

If the output is configured in current mode, the voltage output signal is undefined and must not be connected.

If the output is configured in voltage mode, the current output signal is undefined and must not be connected.
I/O configuration

The analog output module AO561 does not store configuration data itself.

Parameterization

The arrangement of the parameter data is performed with Automation Builder software.

The parameter data directly influences the functionality of modules.

For non-standard applications, it is necessary to adapt the parameters to your system configuration.

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>Internal value</th>
<th>Internal value, Type</th>
<th>Default</th>
<th>Min.</th>
<th>Max.</th>
<th>EDS Slot Index</th>
</tr>
</thead>
<tbody>
<tr>
<td>Module ID</td>
<td>Intern</td>
<td>6515 1)</td>
<td>WORD</td>
<td>0x1973</td>
<td>0</td>
<td>65535</td>
<td>xx01</td>
</tr>
<tr>
<td>Ignore module</td>
<td>No</td>
<td>0</td>
<td>BYTE</td>
<td>No</td>
<td>0x00</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Yes</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Parameter length</td>
<td>Intern</td>
<td>4</td>
<td>BYTE</td>
<td>0</td>
<td>0</td>
<td>255</td>
<td>xx02 2)</td>
</tr>
<tr>
<td>Check Supply</td>
<td>Off</td>
<td>0</td>
<td>BYTE</td>
<td>On</td>
<td>0x01</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>On</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Analog Data Format</td>
<td>Default</td>
<td>0</td>
<td>BYTE</td>
<td>Default</td>
<td>0x00</td>
<td>255</td>
<td></td>
</tr>
</tbody>
</table>

1) with CS31 and addresses less than 70, the value is increased by 1

2) Value is hexadecimal: HighByte is slot (xx: 0...7), LowByte is index (1...n)

GSD file:

```
Ext_User_Prm_Data.Len = Ext_User_Prm_Data.Const(0) =
0x07, 0x19, 0x04, \n0x01, 0x00, \n0x00, 0x00, 0x00, 0x00;
```

Output channel (2x)

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>Internal value</th>
<th>Internal value, Type</th>
<th>Default</th>
<th>Min.</th>
<th>Max.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Channel configuration</td>
<td>see table 2)</td>
<td>see table 2)</td>
<td>BYTE</td>
<td>0</td>
<td>0</td>
<td>65535</td>
</tr>
</tbody>
</table>
### Table 555: Channel configuration 2)

<table>
<thead>
<tr>
<th>Internal value</th>
<th>Operating modes for the analog outputs, individually configurable</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Not used (default)</td>
</tr>
<tr>
<td>128</td>
<td>-10 V...+10 V</td>
</tr>
<tr>
<td>129</td>
<td>0 mA...20 mA</td>
</tr>
<tr>
<td>130</td>
<td>4 mA...20 mA</td>
</tr>
</tbody>
</table>

### Diagnosis

<table>
<thead>
<tr>
<th>E1...E4</th>
<th>d1</th>
<th>d2</th>
<th>d3</th>
<th>d4</th>
<th>Identifier 000...063</th>
<th>AC500 display</th>
<th>&lt;- Display in</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class</td>
<td>Comp</td>
<td>Dev</td>
<td>Mod</td>
<td>Ch</td>
<td>Err</td>
<td>PS501 PLC Browser</td>
<td></td>
</tr>
<tr>
<td>Byte 6</td>
<td>Byte 3</td>
<td>Byte 4</td>
<td>Byte 5</td>
<td>Byte 6 Bit 0...5</td>
<td>PNIO diagnosis block</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bit 6...7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Class</td>
<td>Interface</td>
<td>Device</td>
<td>Module</td>
<td>Channel</td>
<td>Error Identifier</td>
<td>Error message</td>
<td>Remedy</td>
</tr>
<tr>
<td>1)</td>
<td>2)</td>
<td>3)</td>
<td>4)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### Module error

<table>
<thead>
<tr>
<th>3</th>
<th>14</th>
<th>1...10</th>
<th>31</th>
<th>31</th>
<th>19</th>
<th>Checksum error in the I/O module</th>
<th>Replace I/O module</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>11 / 12</td>
<td>ADR</td>
<td>1...10</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>14</td>
<td>1...10</td>
<td>31</td>
<td>31</td>
<td>9</td>
<td>Overflow diagnosis buffer</td>
<td>Restart</td>
</tr>
<tr>
<td></td>
<td>11 / 12</td>
<td>ADR</td>
<td>1...10</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>14</td>
<td>1...10</td>
<td>31</td>
<td>31</td>
<td>26</td>
<td>Parameter error</td>
<td>Check master</td>
</tr>
<tr>
<td></td>
<td>11 / 12</td>
<td>ADR</td>
<td>1...10</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>14</td>
<td>1...10</td>
<td>31</td>
<td>31</td>
<td>11</td>
<td>Process voltage too low</td>
<td>Check process voltage</td>
</tr>
<tr>
<td></td>
<td>11 / 12</td>
<td>ADR</td>
<td>1...10</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### Channel error

<table>
<thead>
<tr>
<th>4</th>
<th>14</th>
<th>1...10</th>
<th>3</th>
<th>0...1</th>
<th>48</th>
<th>Analog value overflow at an analog output</th>
<th>Check output value or terminal</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>11 / 12</td>
<td>ADR</td>
<td>1...10</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>14</td>
<td>1...10</td>
<td>3</td>
<td>0...1</td>
<td>7</td>
<td>Analog value underflow at an analog output</td>
<td>Check output value</td>
</tr>
<tr>
<td></td>
<td>11 / 12</td>
<td>ADR</td>
<td>1...10</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Remarks:
In AC500 the following interface identifier applies:
14 = I/O bus, 11 = COM1 (e.g. CS31 bus), 12 = COM2.
The PNIO diagnosis block does not contain this identifier.

With "Device" the following allocation applies:
31 = module itself, 1...10 = expansion module 1...10, ADR = hardware address (e.g. of the DC551-CS31)

With "Module" the following allocation applies depending on the master:
Module error: I/O bus or PNIO: 31 = module itself; COM1/COM2: 1...10 = expansion 1...10
Channel error: I/O bus or PNIO = module type (3 = AO); COM1/COM2: 1...10 = expansion 1...10

In case of module errors, with channel "31 = Module itself" is output.

### State LEDs

<table>
<thead>
<tr>
<th>LED</th>
<th>State</th>
<th>Color</th>
<th>LED = OFF</th>
<th>LED = ON</th>
</tr>
</thead>
<tbody>
<tr>
<td>PWR</td>
<td>Process voltage 24 VDC via terminal</td>
<td>Green</td>
<td>CPU module voltage or external 24 VDC supply voltage is missing</td>
<td>3.3 V system voltage (I/O bus) and external 24 VDC supply voltage are present</td>
</tr>
<tr>
<td>ERR</td>
<td>Channel or module error</td>
<td>Red</td>
<td>No error or process voltage is missing</td>
<td>Severe error in the module</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Error on 1 or more channels of the module</td>
</tr>
</tbody>
</table>

### Output ranges

<table>
<thead>
<tr>
<th>Range</th>
<th>-10 ... +10 V</th>
<th>0 ... 20 mA</th>
<th>4 ... 20 mA</th>
<th>Digital value</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Decimal</td>
</tr>
<tr>
<td>Overflow</td>
<td>&gt;11.7589</td>
<td>&gt;23.5178</td>
<td>&gt;22.8142</td>
<td>32767</td>
</tr>
<tr>
<td>Output value too high</td>
<td>11.7589</td>
<td>23.5178</td>
<td>22.8142</td>
<td>32511</td>
</tr>
<tr>
<td></td>
<td>10.0058</td>
<td>20.0058</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Output value too low</td>
<td>0.0000</td>
<td>4.0058</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Normal range</td>
<td>10.0000</td>
<td>20.0000</td>
<td>20.0000</td>
<td>27648</td>
</tr>
<tr>
<td>Normal range or output</td>
<td>0.0000</td>
<td>0</td>
<td>4</td>
<td>0</td>
</tr>
<tr>
<td>value too low</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Range

<table>
<thead>
<tr>
<th>Range</th>
<th>-10 ... +10 V</th>
<th>0 ... 20 mA</th>
<th>4 ... 20 mA</th>
<th>Digital value</th>
<th>Decimal</th>
<th>Hex.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-10.0058</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>-10</td>
<td>FFF6</td>
</tr>
<tr>
<td>-10.0000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>-10.0000</td>
<td>9400</td>
</tr>
<tr>
<td>0</td>
<td>3.9942</td>
<td></td>
<td></td>
<td>-10</td>
<td>FFF0</td>
<td></td>
</tr>
<tr>
<td>-16</td>
<td>-4864</td>
<td></td>
<td></td>
<td>E500</td>
<td></td>
<td></td>
</tr>
<tr>
<td>-6912</td>
<td>-27648</td>
<td></td>
<td></td>
<td>E500</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Output value too low**

<table>
<thead>
<tr>
<th>Output value too low</th>
<th>-10.0058</th>
<th>-27664</th>
<th>93F0</th>
</tr>
</thead>
<tbody>
<tr>
<td>-11.7589</td>
<td>-32512</td>
<td>8100</td>
<td></td>
</tr>
</tbody>
</table>

**Underflow**

| Underflow | <11.7589 | <0.0000 | -32768 | 8000 |

The represented resolution corresponds to 12 bit respectively 11 bit plus sign.

---

### Technical data

The System Data of AC500-eCo apply

Only additional details are therefore documented below.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Process supply voltage L+</td>
<td>Terminal 19 for L+ (+24 VDC) and terminal 20 for M (0 V)</td>
</tr>
<tr>
<td>Connections</td>
<td></td>
</tr>
<tr>
<td>Rated value</td>
<td>24 VDC</td>
</tr>
<tr>
<td>Current consumption</td>
<td>0.1 A + output load</td>
</tr>
<tr>
<td>Inrush current (at power-up)</td>
<td>0.05 A·s</td>
</tr>
<tr>
<td>Max. ripple</td>
<td>5 %</td>
</tr>
<tr>
<td>Protection against reversed voltage</td>
<td>Yes</td>
</tr>
<tr>
<td>Protection fuse for L+</td>
<td>Recommended</td>
</tr>
<tr>
<td>Current consumption from 24 VDC power supply at the terminals UP/L+ and ZP/M of the CPU/bus module</td>
<td>Ca. 5 mA</td>
</tr>
<tr>
<td>Galvanic isolation</td>
<td>No</td>
</tr>
<tr>
<td>Surge-voltage (max.)</td>
<td>35 VDC for 0.5 s</td>
</tr>
<tr>
<td>Max. power dissipation within the module</td>
<td>3.1 W</td>
</tr>
<tr>
<td>Weight</td>
<td>Ca. 120 g</td>
</tr>
<tr>
<td>Mounting position</td>
<td>Horizontal or vertical</td>
</tr>
<tr>
<td>Cooling</td>
<td>The natural convection cooling must not be hindered by cable ducts or other parts in the switch-gear cabinet.</td>
</tr>
</tbody>
</table>
NOTICE!
Attention:
All I/O channels (digital and analog) are protected against reverse polarity, reverse supply, short circuit and continuous overvoltage up to 30 VDC.

Technical data of the analog outputs

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of channels per module</td>
<td>2 configurable voltage or current outputs</td>
</tr>
<tr>
<td>Distribution of channels into groups</td>
<td>1 (2 channels per group)</td>
</tr>
<tr>
<td>Connection of the signals O0U- and O1U+</td>
<td>Terminals 13 and 15</td>
</tr>
<tr>
<td>Connection of the signals O0I+ and O1I+</td>
<td>Terminals 14 and 16</td>
</tr>
<tr>
<td>Output type</td>
<td>Bipolar with voltage, unipolar with current</td>
</tr>
<tr>
<td>Resolution</td>
<td>12 bits or 11 bits plus sign</td>
</tr>
<tr>
<td>Conversion error of the analog values</td>
<td>Typ. ±0.5 % of full scale at 25 °C</td>
</tr>
<tr>
<td>conversion caused by non-linearity, adjustment</td>
<td>Max. ±2 % of full scale at 0 °C...+60 °C or EMC disturbances</td>
</tr>
<tr>
<td>error at factory and resolution within the</td>
<td></td>
</tr>
<tr>
<td>normal range</td>
<td></td>
</tr>
<tr>
<td>Indication of the output signals</td>
<td>No</td>
</tr>
<tr>
<td>Output Resistance (load) as current output</td>
<td>0 Ω...500 Ω</td>
</tr>
<tr>
<td>Output load ability as voltage output</td>
<td>±2 mA max.</td>
</tr>
<tr>
<td>Output data length</td>
<td>4 bytes</td>
</tr>
<tr>
<td>Relationship between output signal and hex</td>
<td>% Chapter 1.5.3.6.2.1.4.8 &quot;Output ranges&quot; on page 2583</td>
</tr>
<tr>
<td>code</td>
<td></td>
</tr>
<tr>
<td>Unused outputs</td>
<td>Must not be connected and must be configured as &quot;unused&quot;</td>
</tr>
<tr>
<td>Overvoltage protection</td>
<td>Yes, up to 30 VDC</td>
</tr>
<tr>
<td>Max. cable length (conductor cross section &gt;</td>
<td>Unshielded wire 10 m</td>
</tr>
<tr>
<td>0.14 mm²)</td>
<td>Shielded wire 100 m</td>
</tr>
</tbody>
</table>

Ordering data

<table>
<thead>
<tr>
<th>Part no.</th>
<th>Description</th>
<th>Product life cycle phase *)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1TNE 968 902 R1201</td>
<td>AO561, analog output module, 2 AO, U/I</td>
<td>Active</td>
</tr>
<tr>
<td>1TNE 968 901 R3102</td>
<td>Terminal block TA563-11, 11 pins, screw front,</td>
<td>Active</td>
</tr>
<tr>
<td></td>
<td>cable side, 6 pieces per unit</td>
<td></td>
</tr>
<tr>
<td>Part no.</td>
<td>Description</td>
<td>Product life cycle phase *)</td>
</tr>
<tr>
<td>------------------</td>
<td>-------------------------------------------------------</td>
<td>-----------------------------</td>
</tr>
<tr>
<td>1TNE 968 901 R3104</td>
<td>Terminal block TA564-11, 11 pins, screw front, cable front, 6 pieces per unit</td>
<td>Active</td>
</tr>
<tr>
<td>1TNE 968 901 R3106</td>
<td>Terminal block TA565-11, 11 pins, spring front, cable front, 6 pieces per unit</td>
<td>Active</td>
</tr>
</tbody>
</table>

*) For planning and commissioning of new installations use modules in Active status only.

AX561 - Analog input/output module

- 4 configurable analog inputs (I0 to I3) in 1 group
- 2 configurable analog outputs (O0 and O1) in 1 group
- Resolution: 11 bits plus sign or 12 bits

1  I/O bus
2  1 green LED to display power supply, 1 red LED to display error
3 Terminal number
4 Allocation of signal name
5 Terminal block for input signals (9-pin)
6 Terminal block for output signals (11-pin)
7 2 holes for wall-mounting with screws
8 DIN rail

Intended purpose

The device can be used as a decentralized I/O extension module for S500 Communication Interface Modules (e.g., CI592-CS31, CI501-PNIO, CI541-DP, CI581-CN) or as a centralized extension module for AC500 CPUs (PM5xx).

The inputs are not electrically isolated from each other.
The outputs are not electrically isolated from each other.
All other circuitry of the module is not electrically isolated from the inputs/outputs or from the I/O bus.

The I/O module must not be used as a decentralized I/O module with CI590-CS31-HA bus modules.

Functionality

4 analog inputs, individually configurable for
- Not used (default)
- -2.5 V...+2.5 V
- -5 V...+5 V
- 0 V...+5 V
- 0 V...+10 V
- 0 mA...20 mA
- 4 mA...20 mA

2 analog outputs, individually configurable for
- Not used (default)
- -10 V...+10 V
- 0 mA...20 mA
- 4 mA...20 mA

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Resolution of the analog channels</td>
<td></td>
</tr>
<tr>
<td>Voltage bipolar (-2.5 V...+2.5 V; -5 V...+5 V)</td>
<td>11 bits plus sign</td>
</tr>
<tr>
<td>Voltage unipolar (0 V...5 V; 0 V...10 V)</td>
<td>12 bits</td>
</tr>
<tr>
<td>Current (0 mA...20 mA; 4 mA...20 mA)</td>
<td>12 bits</td>
</tr>
<tr>
<td>LED displays</td>
<td>2 LEDs for process voltage and error messages</td>
</tr>
</tbody>
</table>
### Electrical connection

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Internal supply</td>
<td>Via I/O bus</td>
</tr>
<tr>
<td>External supply</td>
<td>Via the terminals L+ (process voltage 24 VDC) and M (0 VDC); the M terminal is connected to the M terminal of the CPU via the I/O bus</td>
</tr>
</tbody>
</table>

*For a detailed description of the mounting, disassembly and electrical connection of the module, please refer to the System Assembly chapter.*

*If the output is configured as not used, the voltage and current output signals are undefined and must not be connected.*

The electrical connection is carried out by using a removable 9-pin and 11-pin terminal block. These terminal blocks differ in their connection system (spring terminals or screw terminals, cable mounting from the front or from the side). For more information, refer to terminal blocks for S500-eCo I/O modules. The terminal blocks are not included in the module's scope of delivery and must be ordered separately.

The following block diagram shows the internal construction of the analog inputs and outputs:
The assignment of the terminals:

<table>
<thead>
<tr>
<th>Terminal</th>
<th>Signal</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>R0</td>
<td>Burden resistor for input signal 0 for current sensing</td>
</tr>
<tr>
<td>2</td>
<td>I0+</td>
<td>Positive pole of input signal 0</td>
</tr>
<tr>
<td>3</td>
<td>I0-</td>
<td>Negative pole of input signal 0</td>
</tr>
<tr>
<td>4</td>
<td>R1</td>
<td>Burden resistor for input signal 1 for current sensing</td>
</tr>
<tr>
<td>5</td>
<td>I1+</td>
<td>Positive pole of input signal 1</td>
</tr>
<tr>
<td>6</td>
<td>I1-</td>
<td>Negative pole of input signal 1</td>
</tr>
<tr>
<td>7</td>
<td>R2</td>
<td>Burden resistor for input signal 2 for current sensing</td>
</tr>
<tr>
<td>8</td>
<td>I2+</td>
<td>Positive pole of input signal 2</td>
</tr>
<tr>
<td>9</td>
<td>I2-</td>
<td>Negative pole of input signal 2</td>
</tr>
<tr>
<td>10</td>
<td>R3</td>
<td>Burden resistor for input signal 3 for current sensing</td>
</tr>
<tr>
<td>11</td>
<td>I3+</td>
<td>Positive pole of input signal 3</td>
</tr>
<tr>
<td>12</td>
<td>I3-</td>
<td>Negative pole of input signal 3</td>
</tr>
<tr>
<td>13</td>
<td>O0U+</td>
<td>Voltage output of channel 0</td>
</tr>
<tr>
<td>14</td>
<td>O0I+</td>
<td>Current output of channel 0</td>
</tr>
<tr>
<td>Terminal</td>
<td>Signal</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td>15</td>
<td>O1U+</td>
<td>Voltage output of channel 1</td>
</tr>
<tr>
<td>16</td>
<td>O1I+</td>
<td>Current output of channel 1</td>
</tr>
<tr>
<td>17</td>
<td>001-</td>
<td>Negative pole of channels 00 and 01</td>
</tr>
<tr>
<td>18</td>
<td>SG</td>
<td>Shield grounding</td>
</tr>
<tr>
<td>19</td>
<td>L+</td>
<td>Process voltage L+ (24 VDC)</td>
</tr>
<tr>
<td>20</td>
<td>M</td>
<td>Process voltage M (0 VDC)</td>
</tr>
</tbody>
</table>

The internal power supply voltage for the module’s circuitry is carried out via the I/O bus (provided by a bus module or a CPU). Thus, the current consumption from 24 VDC power supply at the terminals UP/L+ and ZP/M of the CPU/bus module increases by 5 mA per AX561.

The external power supply connection is carried out via the L+ (+24 VDC) and the M (0 VDC) terminals. The M terminal is electrically interconnected to the M/ZP terminal of the CPU/bus module.

**NOTICE!**
Risk of imprecise and faulty measurements!
Analog signals may be distorted seriously by external electromagnetic influences.
Use shielded wires when wiring analog signal sources. The cable shield must be grounded at both ends of the cable. Provide a potential equalisation of a low resistance to avoid high potential differences between different parts of the plant.

**WARNING!**
Removal/Insertion under power
The devices are not designed for removal or insertion under power. Because of unforeseeable consequences, it is not allowed to plug or unplug devices with the power being ON.

Make sure that all voltage sources (supply and process voltage) are switched off before you

- connect or disconnect any signal or terminal block
- remove, mount or replace a module.

Disconnecting any powered devices while energized in a hazardous location could result in an electric arc, which could create a flammable ignition resulting in fire or explosion.

Make sure that power is removed and that the area has been thoroughly checked to ensure that flammable materials are not present prior to proceeding.

The devices must not be opened when in operation. The same applies to the network interfaces.

**NOTICE!**
Risk of damaging the PLC modules!
Overvoltages and short circuits might damage the PLC modules.

- Make sure that all voltage sources (supply voltage and process supply voltage) are switched off before you begin with operations at the system.
- Never connect any voltages or signals to reserved terminals (marked with ---). Reserved terminals may carry internal voltages.
The module provides several diagnosis functions \footnote{Chapter 1.5.3.6.2.1.5.6 “Diagnosis” on page 2594.}

The following figure is an example of the internal construction of the analog input AI0. The analog inputs AI1...AI3 are designed in the same way.

![Diagram of internal construction of analog input AI0](image_url)

CAUTION!
Risk of damaging the analog input!
The 250 Ω input resistor can be damaged by overcurrent.
Make sure that the current through the resistor never exceeds 30 mA.

The following figures are an example of the electrical connection of analog sensors (voltage) to the input I0 of the analog input/output module AX561. Proceed with the inputs I1 to I3 in the same way.

![Connection of active-type analog sensors (voltage)](image_url)

![Connection of passive-type analog sensors (voltage)](image_url)

The following figures are an example of the electrical connection of analog sensors (current) to the input I0 of the analog input/output module AX561. Proceed with the inputs I1 to I3 in the same way.

![Connection of active-type analog sensors (current)](image_url)

![Connection of passive-type analog sensors (current)](image_url)
The following figures are an example of the electrical connection of analog actuators to the analog input/output module AX561.

The output signal is undefined if the supply voltage at the L+ terminal is below 10 V. This can, for example, occur if the supply voltage has a slow ramp-up / ramp-down behaviour and must be foreseen when planning the installation.

If the output is configured in current mode, the voltage output signal is undefined and must not be connected.

If the output is configured in voltage mode, the current output signal is undefined and must not be connected.

The meaning of the LEDs is described in the displays chapter “State LEDs” on page 2595.

I/O configuration

The IO module does not store configuration data itself.

Parameterization

The arrangement of the parameter data is performed with Automation Builder software.

The parameter data directly influences the functionality of modules.

For non-standard applications, it is necessary to adapt the parameters to your system configuration.
### Module ID

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>Internal Value</th>
<th>Internal value, Type</th>
<th>Default</th>
<th>Min.</th>
<th>Max.</th>
<th>EDS Slot Index</th>
</tr>
</thead>
<tbody>
<tr>
<td>Module ID</td>
<td>Internal</td>
<td>6520 1)</td>
<td>WORD</td>
<td>0x1978</td>
<td>0</td>
<td>65535</td>
<td>xx01</td>
</tr>
</tbody>
</table>

1) With CS31 and addresses less than 70, the value is increased by 1

### Ignore module

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>Internal Value</th>
<th>Internal value, Type</th>
<th>Default</th>
<th>Min.</th>
<th>Max.</th>
<th>EDS Slot Index</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ignore module</td>
<td>No</td>
<td>0</td>
<td>BYTE</td>
<td>No</td>
<td>0x00</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Yes</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Parameter length

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>Internal Value</th>
<th>Internal value, Type</th>
<th>Default</th>
<th>Min.</th>
<th>Max.</th>
<th>EDS Slot Index</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parameter length</td>
<td>Internal</td>
<td>8</td>
<td>BYTE</td>
<td>0</td>
<td>0</td>
<td>255</td>
<td>xx02 2)</td>
</tr>
</tbody>
</table>

2) Value is hexadecimal: HighByte is slot (xx: 0...7), LowByte is index (1...n)

### Check Supply

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>Internal Value</th>
<th>Internal value, Type</th>
<th>Default</th>
<th>Min.</th>
<th>Max.</th>
<th>EDS Slot Index</th>
</tr>
</thead>
<tbody>
<tr>
<td>Check Supply</td>
<td>Off</td>
<td>0</td>
<td>BYTE</td>
<td>On</td>
<td>0x01</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>On</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Analog Data Format

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>Internal Value</th>
<th>Internal value, Type</th>
<th>Default</th>
<th>Min.</th>
<th>Max.</th>
<th>EDS Slot Index</th>
</tr>
</thead>
<tbody>
<tr>
<td>Analog Data Format</td>
<td>Default</td>
<td>0</td>
<td>BYTE</td>
<td>Default</td>
<td>0x00</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

GSD file:

```
Ext_User_Prm_Data_Len = 0x0B
Ext_User_Prm_Data_Const(0) = 0x79, 0x19, 0x08, \0x01, 0x00, \0x00, 0x00, 0x00, 0x00, 0x00, 0x00;
```

### Input channel (4x)

```
<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>Internal value</th>
<th>Internal value, Type</th>
<th>Default</th>
<th>Min.</th>
<th>Max.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Channel configuration</td>
<td>see table 2)</td>
<td>see table 2)</td>
<td>BYTE</td>
<td>0</td>
<td>0</td>
<td>65535</td>
</tr>
<tr>
<td></td>
<td>see table 2)</td>
<td>see table 2)</td>
<td></td>
<td>0x00 see table 2)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

1) With CS31 and addresses less than 70, the value is increased by 1

2) Value is hexadecimal: HighByte is slot (xx: 0...7), LowByte is index (1...n)

### Table 556: Channel configuration 2)

<table>
<thead>
<tr>
<th>Internal value</th>
<th>Operating modes for the analog inputs, individually configurable</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Not used (default)</td>
</tr>
<tr>
<td>1</td>
<td>0 V...+10 V</td>
</tr>
<tr>
<td>3</td>
<td>0 mA...20 mA</td>
</tr>
<tr>
<td>4</td>
<td>4 mA...20 mA</td>
</tr>
<tr>
<td>6</td>
<td>0 V...+5 V</td>
</tr>
<tr>
<td>7</td>
<td>-5 V...+5 V</td>
</tr>
<tr>
<td>20</td>
<td>-2.5 V...+2.5 V</td>
</tr>
</tbody>
</table>
Output channel (2x)

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>Internal value</th>
<th>Internal value, Type</th>
<th>Default</th>
<th>Min.</th>
<th>Max.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Channel configuration</td>
<td>see see table 2)</td>
<td>see see table 2)</td>
<td>BYTE</td>
<td>0</td>
<td>0x00</td>
<td>65535</td>
</tr>
</tbody>
</table>

Table 557: Channel configuration 2)

<table>
<thead>
<tr>
<th>Internal value</th>
<th>Operating modes for the analog outputs, individually configurable</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Not used (default)</td>
</tr>
<tr>
<td>128</td>
<td>-10 V...+ 10 V</td>
</tr>
<tr>
<td>129</td>
<td>0 mA...20 mA</td>
</tr>
<tr>
<td>130</td>
<td>4 mA...20 mA</td>
</tr>
</tbody>
</table>

Diagnosis

<table>
<thead>
<tr>
<th>E1...E4</th>
<th>d1</th>
<th>d2</th>
<th>d3</th>
<th>d4</th>
<th>Identifier 000...063</th>
<th>AC500 display</th>
<th>← Display in</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class</td>
<td>Comp</td>
<td>Dev</td>
<td>Mod</td>
<td>Ch</td>
<td>Err</td>
<td>PS501 PLC Browser</td>
<td>PNIO diagnosis block</td>
</tr>
<tr>
<td>Byte 6</td>
<td>Bit 6...7</td>
<td>Byte 3</td>
<td>Byte 4</td>
<td>Byte 5</td>
<td>Byte 6 Bit 0...5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Class</td>
<td>Interface</td>
<td>Device</td>
<td>Module</td>
<td>Channel</td>
<td>Error Identifier</td>
<td>Error message</td>
<td>Remedy</td>
</tr>
<tr>
<td></td>
<td>1)</td>
<td>2)</td>
<td>3)</td>
<td>4)</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Module error

<table>
<thead>
<tr>
<th>3</th>
<th>14</th>
<th>1...10</th>
<th>31</th>
<th>31</th>
<th>19</th>
<th>Checksum error in the I/O module</th>
<th>Replace I/O module</th>
</tr>
</thead>
<tbody>
<tr>
<td>11 / 12</td>
<td>ADR</td>
<td>1...10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>14</td>
<td>1...10</td>
<td>31</td>
<td>31</td>
<td>9</td>
<td>Overflow diagnosis buffer</td>
<td>Restart</td>
</tr>
<tr>
<td>11 / 12</td>
<td>ADR</td>
<td>1...10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>14</td>
<td>1...10</td>
<td>31</td>
<td>31</td>
<td>26</td>
<td>Parameter error</td>
<td>Check master</td>
</tr>
<tr>
<td>11 / 12</td>
<td>ADR</td>
<td>1...10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>14</td>
<td>1...10</td>
<td>31</td>
<td>31</td>
<td>11</td>
<td>Process voltage too low</td>
<td>Check process voltage</td>
</tr>
<tr>
<td>11 / 12</td>
<td>ADR</td>
<td>1...10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Channel error

<table>
<thead>
<tr>
<th>4</th>
<th>14</th>
<th>1...10</th>
<th>1</th>
<th>0...3</th>
<th>48</th>
<th>Analog value overflow at an analog input</th>
<th>Check input value or terminal</th>
</tr>
</thead>
<tbody>
<tr>
<td>11 / 12</td>
<td>ADR</td>
<td>1...10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>14</td>
<td>1...10</td>
<td>1</td>
<td>0...3</td>
<td>7</td>
<td>Analog value underflow at an analog input</td>
<td>Check input value</td>
</tr>
<tr>
<td>11 / 12</td>
<td>ADR</td>
<td>1...10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>E1...E4</td>
<td>d1</td>
<td>d2</td>
<td>d3</td>
<td>d4</td>
<td>Identifier 000...063</td>
<td>AC500 display</td>
<td>PS501 PLC Browser</td>
</tr>
<tr>
<td>---------</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>---------------------</td>
<td>----------------</td>
<td>-----------------</td>
</tr>
<tr>
<td>Class</td>
<td>Comp</td>
<td>Dev</td>
<td>Mod</td>
<td>Ch</td>
<td>Err</td>
<td>PNIO diagnosis block</td>
<td></td>
</tr>
<tr>
<td>Byte 6</td>
<td>-</td>
<td>Byte 3</td>
<td>Byte 4</td>
<td>Byte 5</td>
<td>Bit 0...5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bit 6...7</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Class</th>
<th>Interface</th>
<th>Device</th>
<th>Module</th>
<th>Channel</th>
<th>Error Identifier</th>
<th>Error message</th>
<th>Remedy</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>14</td>
<td>1...10</td>
<td>3</td>
<td>0...1</td>
<td>48</td>
<td>Analog value overflow at an analog output</td>
<td>Check output value or terminal</td>
</tr>
<tr>
<td></td>
<td>11 / 12</td>
<td>ADR</td>
<td>1...10</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>14</td>
<td>1...10</td>
<td>3</td>
<td>0...1</td>
<td>7</td>
<td>Analog value underflow at an analog output</td>
<td>Check output value</td>
</tr>
<tr>
<td></td>
<td>11 / 12</td>
<td>ADR</td>
<td>1...10</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Remarks:

1) In AC500 the following interface identifier applies:
   14 = I/O bus, 11 = COM1 (e.g. CS31 bus), 12 = COM2.
   The PNIO diagnosis block does not contain this identifier.

2) With "Device" the following allocation applies:
   31 = module itself, 1...10 = decentralized communication interface module 1...10, ADR = hardware address (e.g. of the DC551-CS31)

3) With "Module" the following allocation applies dependent of the master:
   Module error: I/O bus or PNIO: 31 = module itself; COM1/COM2: 1...10 = expansion 1...10
   Channel error: I/O bus or PNIO = module type (1 = AI, 3 = AO); COM1/COM2: 1...10 = expansion 1...10

4) In case of module errors, with channel "31 = Module itself" is output.

State LEDs

<table>
<thead>
<tr>
<th>LED</th>
<th>State</th>
<th>Color</th>
<th>LED = OFF</th>
<th>LED = ON</th>
<th>LED flashes</th>
</tr>
</thead>
<tbody>
<tr>
<td>PWR</td>
<td>Process voltage</td>
<td>Green</td>
<td>CPU module voltage or external 24 VDC supply voltage is missing</td>
<td>3.3 V system voltage (I/O bus) and external 24 VDC supply voltage are present</td>
<td>---</td>
</tr>
<tr>
<td></td>
<td>24 VDC via terminal</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ERR</td>
<td>Channel or module error</td>
<td>Red</td>
<td>No error or process voltage is missing</td>
<td>Severe error in the module</td>
<td>Error on 1 or more channels of the module</td>
</tr>
</tbody>
</table>
**CAUTION!**
**Risk of wrong analog input values!**
The analog input values may be wrong if the measuring range of the inputs are exceeded.

Make sure that the analog signal at the connection terminals is always within the signal range.

<table>
<thead>
<tr>
<th>Range</th>
<th>-2.5 ... +2.5 V</th>
<th>-5 ... +5 V</th>
<th>0 ... 5 V</th>
<th>0 ... 10 V</th>
<th>0 ... 20 mA</th>
<th>4 ... 20 mA</th>
<th>Digital value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Overflow</td>
<td>&gt;2.9397</td>
<td>&gt;5.8795</td>
<td>&gt;5.8795</td>
<td>&gt;11.7589</td>
<td>&gt;23.5178</td>
<td>&gt;22.8142</td>
<td>32767    7FFF</td>
</tr>
<tr>
<td>Measured value too high</td>
<td>2.9397</td>
<td>5.8795</td>
<td>5.8795</td>
<td>11.7589</td>
<td>23.5178</td>
<td>22.8142</td>
<td>32511    7EFF</td>
</tr>
<tr>
<td></td>
<td>2.5014</td>
<td>5.0029</td>
<td>5.0015</td>
<td>10.0029</td>
<td>20.0058</td>
<td></td>
<td>27664    6C10</td>
</tr>
<tr>
<td>Normal range</td>
<td>2.5000</td>
<td>5.0000</td>
<td>5.0000</td>
<td>10.0000</td>
<td>20.0000</td>
<td>20.0000</td>
<td>27648    6C00</td>
</tr>
<tr>
<td>Normal range or measured value too low</td>
<td>0.0014</td>
<td>0.0029</td>
<td>0.0015</td>
<td>0.0029</td>
<td>0.0058</td>
<td></td>
<td>16       0010</td>
</tr>
<tr>
<td></td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0</td>
<td>4</td>
<td>0</td>
<td>8        0008</td>
</tr>
<tr>
<td>Measured value too low</td>
<td>-2.5014</td>
<td>-5.0029</td>
<td>-5.0000</td>
<td></td>
<td></td>
<td>3.9942</td>
<td>-10      FFF6</td>
</tr>
<tr>
<td></td>
<td>-2.9398</td>
<td>-5.8795</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>-16      FFF0</td>
</tr>
<tr>
<td>Underflow</td>
<td>&lt;-2.9398</td>
<td>&lt;-5.8795</td>
<td>&lt;-0.0300</td>
<td>&lt;-0.0600</td>
<td>&lt;-0.1200</td>
<td>&lt;-0.1200</td>
<td>-27664    93F0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>-32512    8100</td>
</tr>
</tbody>
</table>

The represented resolution corresponds to 12 bits respectively 11 bits plus sign.
### Output ranges

<table>
<thead>
<tr>
<th>Range</th>
<th>-10 ... +10 V</th>
<th>0 ... 20 mA</th>
<th>4 ... 20 mA</th>
<th>Digital value</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><em>Decimal</em></td>
<td><em>Hex.</em></td>
<td><em>Decimal</em></td>
<td><em>Hex.</em></td>
</tr>
<tr>
<td><strong>Overflow</strong></td>
<td>&gt; 11.7589</td>
<td>&gt; 23.5178</td>
<td>&gt; 22.8142</td>
<td>32767</td>
</tr>
<tr>
<td>Output value too high</td>
<td>11.7589</td>
<td>23.5178</td>
<td>22.8142</td>
<td>32511</td>
</tr>
<tr>
<td></td>
<td>10.0058</td>
<td></td>
<td>20.0058</td>
<td>27664</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>20.0058</td>
<td>27658</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>20.0058</td>
<td>27656</td>
</tr>
<tr>
<td>Normal range</td>
<td>10.0000</td>
<td>20.0000</td>
<td>20.0000</td>
<td>27648</td>
</tr>
<tr>
<td>or output value too low</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.0058</td>
<td></td>
<td>4.0058</td>
<td>16</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>10</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>8</td>
</tr>
<tr>
<td></td>
<td>0.0000</td>
<td>0</td>
<td>4</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>3.9942</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>-16</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>-4864</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>-6912</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>-27648</td>
</tr>
<tr>
<td>Output value too low</td>
<td>-10.0058</td>
<td></td>
<td></td>
<td>-27664</td>
</tr>
<tr>
<td></td>
<td>-11.7589</td>
<td></td>
<td></td>
<td>-32512</td>
</tr>
<tr>
<td><strong>Underflow</strong></td>
<td>&lt; -11.7589</td>
<td>&lt;0.0000</td>
<td></td>
<td>-32768</td>
</tr>
</tbody>
</table>

The represented resolution corresponds to 12 bits respectively 11 bits plus sign.

### Technical data

The System Data of AC500-eCo apply

Only additional details are therefore documented below.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Process supply voltage L+</strong></td>
<td></td>
</tr>
<tr>
<td>Connections</td>
<td>Terminal 19 for L+ (+24 VDC) and terminal 20 for M (0 V)</td>
</tr>
<tr>
<td>Rated value</td>
<td>24 VDC</td>
</tr>
<tr>
<td>Current consumption via L+ terminal</td>
<td>0.14 A + output load</td>
</tr>
<tr>
<td>Inrush current (at power-up)</td>
<td>0.05 A</td>
</tr>
<tr>
<td>Max. ripple</td>
<td>5 %</td>
</tr>
<tr>
<td>Protection against reversed voltage</td>
<td>Yes</td>
</tr>
<tr>
<td>Protection fuse for L+</td>
<td>Recommended</td>
</tr>
</tbody>
</table>
### Technical data of the analog inputs

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of channels per module</td>
<td>4 individually configurable voltage or current inputs</td>
</tr>
<tr>
<td>Distribution of channels into groups</td>
<td>1 (4 channels per group)</td>
</tr>
<tr>
<td>Resolution</td>
<td></td>
</tr>
<tr>
<td>Unipolar Voltage:</td>
<td></td>
</tr>
<tr>
<td>Voltage: 0 V...+5 V; 0 V...+10 V: 12 bits</td>
<td></td>
</tr>
<tr>
<td>Current 0 mA...20 mA; 4 mA...20 mA: 12 bits</td>
<td></td>
</tr>
<tr>
<td>Bipolar Voltage:</td>
<td></td>
</tr>
<tr>
<td>Voltage -2.5 V...+2.5 V; -5 V...+5 V: 11 bits plus sign</td>
<td></td>
</tr>
<tr>
<td>Connection of the signals I0- to I3-</td>
<td>Terminals 3, 6, 9, 12</td>
</tr>
<tr>
<td>Connection of the signals I0+ to I3+</td>
<td>Terminals 2, 5, 8, 11</td>
</tr>
<tr>
<td>Input type</td>
<td>Differential</td>
</tr>
<tr>
<td>Galvanic isolation</td>
<td>No galvanic isolation between the inputs and the I/O bus</td>
</tr>
<tr>
<td>Common mode input range</td>
<td>Signal voltage plus common mode voltage must be within ±12 V</td>
</tr>
<tr>
<td>Indication of the input signals</td>
<td>No</td>
</tr>
<tr>
<td>Channel input resistance</td>
<td>Voltage: &gt;1 MΩ</td>
</tr>
<tr>
<td></td>
<td>Current: ca. 250 Ω</td>
</tr>
<tr>
<td>Conversion error of the analog values</td>
<td>Typ.</td>
</tr>
<tr>
<td>caused by non-linearity, adjustment error at</td>
<td>±0.5 % of full scale (voltage)</td>
</tr>
<tr>
<td>factory and resolution within the normal range</td>
<td>±0.5 % of full scale (current 0 mA...20 mA)</td>
</tr>
<tr>
<td></td>
<td>±0.7 % of full scale (current 4 mA...20 mA) at 25 °C</td>
</tr>
</tbody>
</table>
Technical data of the analog outputs

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of channels per module</td>
<td>2 configurable voltage or current outputs</td>
</tr>
<tr>
<td>Distribution of channels into groups</td>
<td>1 (2 channels per group)</td>
</tr>
<tr>
<td>Connection of the signals O0U- and O1U+</td>
<td>Terminals 13 and 15</td>
</tr>
<tr>
<td>Connection of the signals O0I+ and O1I+</td>
<td>Terminals 14 and 16</td>
</tr>
<tr>
<td>Output type</td>
<td>Bipolar with voltage, unipolar with current</td>
</tr>
<tr>
<td>Resolution</td>
<td>12 bits or 11 bits plus sign</td>
</tr>
<tr>
<td>Indication of the output signals</td>
<td>No</td>
</tr>
<tr>
<td>Output resistance (load) as current output</td>
<td>0 Ω...500 Ω</td>
</tr>
<tr>
<td>Output load ability as voltage output</td>
<td>2 mA max.</td>
</tr>
<tr>
<td>Relationship between input signal and hex code</td>
<td>Table Output Ranges ( \text{Table on page 2597} )</td>
</tr>
<tr>
<td>Conversion error of the analog values caused by non-linearity, adjustment error at factory and resolution within the normal range</td>
<td>Typ. ( \pm 0.5 % ) of full scale (voltage) ( \pm 0.5 % ) of full scale (current 0 mA...20 mA) ( \pm 0.7 % ) of full scale (current 4 mA...20 mA) ( \pm 2 % ) of full scale (all ranges) at 25°C at 0 °C...60 °C or EMC disturbance</td>
</tr>
<tr>
<td>Unused outputs</td>
<td>Can be left open and should be configured as &quot;unused&quot;</td>
</tr>
<tr>
<td>Output data length</td>
<td>4 bytes</td>
</tr>
<tr>
<td>Overvoltage protection</td>
<td>Yes, up to 30 VDC</td>
</tr>
</tbody>
</table>
### Parameter

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Max. cable length (conductor cross section &gt; 0.14 mm²)</td>
<td></td>
</tr>
<tr>
<td>Unshielded wire</td>
<td>10 m</td>
</tr>
<tr>
<td>Shielded wire</td>
<td>100 m</td>
</tr>
</tbody>
</table>

### Ordering data

<table>
<thead>
<tr>
<th>Part no.</th>
<th>Description</th>
<th>Product life cycle phase *)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1TNE 968 902 R1301</td>
<td>AX561, analog input/output module, 4 AI, 2 AO, U/I</td>
<td>Active</td>
</tr>
<tr>
<td>1TNE 968 901 R3101</td>
<td>Terminal block TA563-9, 9 pins, screw front, cable side, 6 pieces per unit</td>
<td>Active</td>
</tr>
<tr>
<td>1TNE 968 901 R3102</td>
<td>Terminal block TA563-11, 11 pins, screw front, cable side, 6 pieces per unit</td>
<td>Active</td>
</tr>
<tr>
<td>1TNE 968 901 R3103</td>
<td>Terminal block TA564-9, 9 pins, screw front, cable side, 6 pieces per unit</td>
<td>Active</td>
</tr>
<tr>
<td>1TNE 968 901 R3104</td>
<td>Terminal block TA564-11, 11 pins, screw front, cable front, 6 pieces per unit</td>
<td>Active</td>
</tr>
<tr>
<td>1TNE 968 901 R3105</td>
<td>Terminal block TA565-9, 9 pins, spring front, cable front, 6 pieces per unit</td>
<td>Active</td>
</tr>
<tr>
<td>1TNE 968 901 R3106</td>
<td>Terminal block TA565-11, 11 pins, spring front, cable front, 6 pieces per unit</td>
<td>Active</td>
</tr>
</tbody>
</table>

*) For planning and commissioning of new installations use modules in Active status only.

### S500

**AI523 - Analog input module**

- 16 configurable analog inputs (I0 to I15) in 2 groups (1.0...2.7 and 3.0...4.7)
- Module-wise electrically isolated
- XC version for use in extreme ambient conditions available
1 I/O bus
2 Allocation between terminal number and signal name
3 16 yellow LEDs to display the signal states at the analog inputs (I0 - I15)
4 1 green LED to display the state of the process supply voltage UP
5 2 red LEDs to display errors
6 Label
7 Terminal unit
8 DIN rail
9 Sign for XC version

**Intended purpose**

The device can be used as a decentralized I/O extension module for S500 Communication Interface Modules (e.g., CI592-CS31, CI501-PNIO, CI541-DP, CI581-CN) or as a centralized extension module for AC500 CPUs (PM5xx).

**Functionality**

16 analog inputs, individually configurable for

- Unused (default setting)
- 0 V...10 V
- -10 V...+10 V
- 0 mA...20 mA
- 4 mA...20 mA

2020/12/10
3ADR010583, 1, en_US
2601
- Pt100, -50 °C...+400 °C (2-wire)
- Pt100, -50 °C...+400 °C (3-wire), requires 2 channels
- Pt100, -50 °C...+70 °C (2-wire)
- Pt100, -50 °C...+70 °C (3-wire), requires 2 channels
- Pt1000, -50 °C...+400 °C (2-wire)
- Pt1000, -50 °C...+400 °C (3-wire), requires 2 channels
- Ni1000, -50 °C...+150 °C (2-wire)
- Ni1000, -50 °C...+150 °C (3-wire), requires 2 channels
- 0 V...10 V with differential inputs, requires 2 channels
- -10 V...+10 V with differential inputs, requires 2 channels
- Digital signals (digital input)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Resolution of the analog channels</td>
<td></td>
</tr>
<tr>
<td>Voltage -10 V... +10 V</td>
<td>12 bits plus sign</td>
</tr>
<tr>
<td>Voltage 0 V...10 V</td>
<td>12 bits</td>
</tr>
<tr>
<td>Current 0 mA...20 mA, 4 mA...20 mA</td>
<td>12 bits</td>
</tr>
<tr>
<td>Temperature</td>
<td>0.1 °C</td>
</tr>
<tr>
<td>LED displays</td>
<td>19 LEDs for signals and error messages</td>
</tr>
<tr>
<td>Internal power supply</td>
<td>Via the expansion bus interface (I/O bus)</td>
</tr>
<tr>
<td>External power supply</td>
<td>Via the terminals ZP and UP (process voltage 24 VDC)</td>
</tr>
<tr>
<td>Required terminal unit</td>
<td>TU515 or TU516 “Chapter 1.5.3.5.2 “TU515, TU516, TU541 and TU542 for I/O modules” on page 2311</td>
</tr>
</tbody>
</table>

**Electrical connection**

The modules are plugged on an I/O terminal unit “Chapter 1.5.3.5.2 “TU515, TU516, TU541 and TU542 for I/O modules” on page 2311. Properly position the modules and press until they lock in place. The terminal units are mounted on a DIN rail or with 2 screws plus the additional accessory for wall mounting (TA526 “Chapter 1.5.3.8.1.4 “TA526 - Wall mounting accessory” on page 3035).

The electrical connection of the I/O channels is carried out using the 40 terminals of the I/O terminal unit. I/O modules can be replaced without re-wiring the terminal units.

For a detailed description of the mounting, disassembly and electrical connection of the module, please refer to the System Assembly, Construction and Connection chapter “Chapter 1.5.4.5 “AC500 (Standard)” on page 3058.

The terminals 1.8 to 4.8 and 1.9 to 4.9 are electrically interconnected within the I/O terminal units and have always the same assignment, independent of the inserted module:

- Terminals 1.8 to 4.8: process voltage UP = +24 VDC
- Terminals 1.9 to 4.9: process voltage ZP = 0 V

The assignment of the other terminals:
<table>
<thead>
<tr>
<th>Terminals</th>
<th>Signal</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0 to 1.7</td>
<td>I0- to I7-</td>
<td>Negative poles of the first 8 analog inputs</td>
</tr>
<tr>
<td>2.0 to 2.7</td>
<td>I0+ to I7+</td>
<td>Positive poles of the first 8 analog inputs</td>
</tr>
<tr>
<td>3.0 to 3.7</td>
<td>I8- to I15-</td>
<td>Negative poles of the following 8 analog inputs</td>
</tr>
<tr>
<td>4.0 to 4.7</td>
<td>I8+ to I15+</td>
<td>Positive poles of the following 8 analog inputs</td>
</tr>
</tbody>
</table>

**CAUTION!**
The negative poles of the analog inputs are electrically connected to each other. They form an "Analog Ground" signal for the module. The negative poles of the analog outputs are also electrically connected to each other to form an "Analog Ground" signal.

**CAUTION!**
There is no galvanic isolation between the analog circuitry and ZP/UP. Therefore, the analog sensors must be electrically isolated in order to avoid loops via the earth potential or the supply voltage.

**CAUTION!**
Because of their common reference potential, analog current inputs cannot be circuitled in series, neither within the module nor with channels of other modules.

*For the open-circuit detection (cut wire), each analog input channel is pulled up to "plus" by a high-resistance resistor. If nothing is connected, the maximum voltage will be read in then.*

The internal power supply voltage for the module's circuitry is carried out via the I/O bus (provided by a bus module or a CPU). Thus, the current consumption from 24 VDC power supply at the terminals L+/UP and M/ZP of the CPU/bus module increases by 2 mA per AI523.

The external power supply connection is carried out via the UP (+24 VDC) and the ZP (0 VDC) terminals.

**NOTICE!**
Risk of damaging the PLC modules!
- Overvoltages and short circuits might damage the PLC modules.
- Make sure that all voltage sources (supply voltage and process supply voltage) are switched off before you begin with operations at the system.
- Never connect any voltages or signals to reserved terminals (marked with --). Reserved terminals may carry internal voltages.
Generally, analog signals must be laid in shielded cables. The cable shields must be earthed at both sides of the cables. In order to avoid unacceptable potential differences between different parts of the installation, low resistance equipotential bonding conductors must be laid.

Only for simple applications (low electromagnetic disturbances, no high requirement on precision), the shielding can also be omitted.

The following figure shows the electrical connection of the module:

The modules provide several diagnosis functions Chapter 1.5.3.6.2.1.7 “Diagnosis” on page 2615.

Connection of resistance thermometers in 2-wire configuration

When resistance thermometers (Pt100, Pt1000, Ni1000) are used, a constant current must flow through them to build the necessary voltage drop for the evaluation. For this, the module AI523 provides a constant current source which is multiplexed over the 8 analog channels.
The following measuring ranges can be configured. Chapter 1.5.3.6.2.1.6 “Parameterization” on page 2612.

<table>
<thead>
<tr>
<th>Sensor</th>
<th>Measuring Range</th>
<th>Configuration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pt100</td>
<td>-50 °C...+70 °C</td>
<td>2-wire config. one channel used</td>
</tr>
<tr>
<td>Pt100</td>
<td>-50 °C...+400 °C</td>
<td>2-wire config. one channel used</td>
</tr>
<tr>
<td>Pt1000</td>
<td>-50 °C...+400 °C</td>
<td>2-wire config. one channel used</td>
</tr>
<tr>
<td>Ni1000</td>
<td>-50 °C...+150 °C</td>
<td>2-wire config. one channel used</td>
</tr>
</tbody>
</table>

The function of the LEDs is described under Displays. Chapter 1.5.3.6.2.1.7 “Diagnosis” on page 2615.

The module AI523 performs a linearization of the resistance characteristic.

In order to avoid error messages from unused analog input channels, it is useful to configure them as “unused”.

**Connection of resistance thermometers in 3-wire configuration**

When resistance thermometers (Pt100, Pt1000, Ni1000) are used, a constant current must flow through them to build the necessary voltage drop for the evaluation. For this, the module AI523 provides a constant current source which is multiplexed over the max. 8 (depending on the configuration) analog channels.
If several measuring points are adjacent to each other, the return line is necessary only once. This saves wiring costs.

With 3-wire configuration, two adjacent analog channels belong together (e.g. the channels 0 and 1). In this case, both channels are configured according to the desired operating mode. The lower address must be the even address (channel 0), the next higher address must be the odd address (channel 1).

The constant current of one channel flows through the resistance thermometer. The constant current of the other channel flows through one of the cores. The module calculates the measured value from the two voltage drops and stores it under the input with the higher channel number (e.g. I1).

In order to keep measuring errors as small as possible, it is necessary to have all the involved conductors in the same cable. All the conductors must have the same cross section.

The following measuring ranges can be configured

<table>
<thead>
<tr>
<th>Temperature Range</th>
<th>Configuration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pt100 -50 °C...+70 °C</td>
<td>3-wire configuration, two channels used</td>
</tr>
<tr>
<td>Pt1000 -50 °C...+400 °C</td>
<td>3-wire configuration, two channels used</td>
</tr>
<tr>
<td>Pt1000 -50 °C...+400 °C</td>
<td>3-wire configuration, two channels used</td>
</tr>
<tr>
<td>Ni1000 -50 °C...+150 °C</td>
<td>3-wire configuration, two channels used</td>
</tr>
</tbody>
</table>

The function of the LEDs is described under Displays

The module AI523 performs a linearization of the resistance characteristic.

In order to avoid error messages from unused analog input channels, it is useful to configure them as "unused".
Connection of active-type analog sensors (Voltage) with electrically isolated power supply

![Connection diagram](image)

**Fig. 120: Connection example**

By connecting the sensor’s negative pole of the output voltage to AGND, the electrically isolated voltage source of the sensor is referred to ZP.

The following measuring ranges can be configured: Chapter 1.5.3.6.2.2.1.6 “Parameterization” on page 2612 or Chapter 1.5.3.6.2.2.1.9 “Measuring ranges” on page 2617.

<table>
<thead>
<tr>
<th>Voltage</th>
<th>0 V...10 V</th>
<th>1 channel used</th>
</tr>
</thead>
<tbody>
<tr>
<td>Voltage</td>
<td>-10 V...+10 V</td>
<td>1 channel used</td>
</tr>
</tbody>
</table>

The function of the LEDs is described under Displays Chapter 1.5.3.6.2.2.1.7 “Diagnosis” on page 2615.

In order to avoid error messages or long processing times, it is useful to configure unused analog input channels as “unused”.

Connection of active-type analog sensors (Current) with electrically isolated power supply

![Connection diagram](image)

**Fig. 121: Connection example**

The following measuring ranges can be configured:

<table>
<thead>
<tr>
<th>Voltage</th>
<th>0 ... +20 mA</th>
<th>1 channel used</th>
</tr>
</thead>
<tbody>
<tr>
<td>Voltage</td>
<td>+4 ... +20 mA</td>
<td>1 channel used</td>
</tr>
</tbody>
</table>

In order to avoid error messages or long processing times, it is useful to configure unused analog input channels as “unused”. 
The following measuring ranges can be configured \( \text{Chapter 1.5.3.6.2.2.1.6 "Parameterization" on page 2612 } \text{ Chapter 1.5.3.6.2.1.9 "Measuring ranges" on page 2617} \\

<table>
<thead>
<tr>
<th>Measure</th>
<th>Range</th>
<th>Channels used</th>
</tr>
</thead>
<tbody>
<tr>
<td>Current</td>
<td>0 mA...20 mA</td>
<td>1 channel</td>
</tr>
<tr>
<td>Current</td>
<td>4 mA...20 mA</td>
<td>1 channel</td>
</tr>
</tbody>
</table>

The function of the LEDs is described under Displays \( \text{Chapter 1.5.3.6.2.2.1.7 "Diagnosis" on page 2615} \\

Unused input channels can be left open-circuited, because they are of low resistance.

Connection of active-type analog sensors (Voltage) with no electrically isolated power supply

![Connection diagram](image)

**Fig. 122: Connection example**

**CAUTION!**

The potential difference between AGND and ZP at the module must not be greater than 1 V, not even in case of long lines.

If AGND does not get connected to ZP, the sensor current flows to ZP via the AGND line. The measuring signal is distorted, as a very low current flows over the voltage line. The total current through the PTC should not exceed 50 mA.

This measuring method is therefore only suitable for short lines and small sensor currents. If there are bigger distances, the difference measuring method has to be preferred.

The following measuring ranges can be configured \( \text{Chapter 1.5.3.6.2.1.9 "Measuring ranges" on page 2617} \\

<table>
<thead>
<tr>
<th>Measure</th>
<th>Range</th>
<th>Channels used</th>
</tr>
</thead>
<tbody>
<tr>
<td>Voltage</td>
<td>0 V...10 V</td>
<td>1 channel</td>
</tr>
<tr>
<td>Voltage</td>
<td>-10 V...+10 V *)</td>
<td>1 channel</td>
</tr>
</tbody>
</table>

*) if the sensor can provide this signal range

The function of the LEDs is described under Displays \( \text{Chapter 1.5.3.6.2.1.7 "Diagnosis" on page 2615} \\

In order to avoid error messages or long processing times, it is useful to configure unused analog input channels as "unused".
Connection of passive-type analog sensors (Current)

The following measuring ranges can be configured "Parameterization" on page 2612 "Measuring ranges" on page 2617

| Current | 4 mA...20 mA | 1 channel used |

The function of the LEDs is described under Displays "Diagnosis" on page 2615.

CAUTION!
If, during initialization, an analog current sensor supplies more than 25 mA for more than 1 second into an analog input, this input is switched off by the module (input protection). In such cases, it is recommended to protect the analog input by a 10 volt Zener diode (in parallel to I+ and I-). But, in general, it is a better solution to use sensors with fast initialization or without current peaks higher than 25 mA.

Unused input channels can be left open-circuited, because they are of low resistance.

Connection of active-type analog sensors (Voltage) to differential inputs

Differential inputs are very useful if analog sensors which are remotely non-isolated (e.g. the negative terminal is remotely earthed) are used.

The evaluation using differential inputs helps to considerably increase the measuring accuracy and to avoid earthing loops.

With differential input configurations, two adjacent analog channels belong together (e.g. channels 0 and 1). In this case, both channels are configured according to the desired operating mode. The lower address must be the even address (channel 0), the next higher address must be the odd address (channel 1). The converted analog value is available at the higher address (channel 1).

The converted analog value is available at the odd channel (higher address).
CAUTION!
The earthing potential at the sensors must not have a too big potential difference with respect to ZP (max. ±1 V within the full signal range). Otherwise problems can occur concerning the common-mode input voltages of the involved analog inputs.

![Connection Diagram]

**Fig. 124: Connection example**

The negative pole of the sensor must be earthed next to the sensor.

The following measuring ranges can be configured:

- **Voltage**: 0 V...10 V with differential inputs, 2 channels used
- **Voltage**: -10 V...+10 V with differential inputs, 2 channels used

The function of the LEDs is described under Displays.

In order to avoid error messages or long processing times, it is useful to configure unused analog input channels as "unused".

**Use of analog inputs as digital inputs**

Several (or all) analog inputs can be configured as digital inputs. The inputs are not electrically isolated against the other analog channels.
Fig. 125: Connection example

The following operating mode can be configured Chapter 1.5.3.6.2.2.1.6 “Parameterization” on page 2612 Chapter 1.5.3.6.2.2.1.9 “Measuring ranges” on page 2617

<table>
<thead>
<tr>
<th>Digital input</th>
<th>24 V</th>
<th>1 channel used</th>
</tr>
</thead>
<tbody>
<tr>
<td>Effect of incorrect input terminal connection</td>
<td>Wrong or no signal detected, no damage up to 35 V</td>
<td></td>
</tr>
</tbody>
</table>

The function of the LEDs is described under Displays.

**Internal data exchange**

<table>
<thead>
<tr>
<th>Digital inputs (bytes)</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Digital outputs (bytes)</td>
<td>0</td>
</tr>
<tr>
<td>Counter input data (words)</td>
<td>16</td>
</tr>
<tr>
<td>Counter output data (words)</td>
<td>0</td>
</tr>
</tbody>
</table>

**I/O configuration**

The module does not store configuration data itself. It gets its parameterization data from the master device of the I/O bus (CPU or bus module) during power-up of the system.

That means replacing I/O modules is possible without any re-parameterization via software.

*If the external power supply voltage via UP/ZP terminals fails, the I/O module loses its configuration data. The whole station has to be switched off and on again to re-configure the module.*
Parameterization

<table>
<thead>
<tr>
<th>No.</th>
<th>Name</th>
<th>Value</th>
<th>Internal value</th>
<th>Internal value, type</th>
<th>Default</th>
<th>Min.</th>
<th>Max.</th>
<th>EDS Slot/Index</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Module ID</td>
<td>Internal</td>
<td>1515 ¹)</td>
<td>Word</td>
<td>1515 0x05eb</td>
<td>0</td>
<td>65535</td>
<td>0x0Y01</td>
</tr>
<tr>
<td>2</td>
<td>Ignore module</td>
<td>No Yes</td>
<td>0 1</td>
<td>Byte</td>
<td>No 0x00</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Parameter length in bytes</td>
<td>Internal</td>
<td>34</td>
<td>Byte</td>
<td>34-CPU 34-FBP</td>
<td>0</td>
<td>255</td>
<td>0x0Y02</td>
</tr>
<tr>
<td>4</td>
<td>Check supply</td>
<td>Off On</td>
<td>0 1</td>
<td>Byte</td>
<td>On 0x01</td>
<td>0</td>
<td>1</td>
<td>0x0Y03</td>
</tr>
<tr>
<td>5</td>
<td>Analog data format</td>
<td>Default</td>
<td>0</td>
<td>Byte</td>
<td>Default 0x00</td>
<td>0</td>
<td>19</td>
<td>0x0Y04</td>
</tr>
<tr>
<td>6</td>
<td>Channel configuration Input channel 0</td>
<td>see table</td>
<td>&quot;Channel configuration&quot;</td>
<td>Byte</td>
<td>Default 0x00</td>
<td>0</td>
<td>19</td>
<td>0x0Y05</td>
</tr>
<tr>
<td>7</td>
<td>Channel monitoring Input channel 0</td>
<td>see table</td>
<td>&quot;Channel monitoring&quot;</td>
<td>Byte</td>
<td>Default 0x00</td>
<td>0</td>
<td>3</td>
<td>0x0Y06</td>
</tr>
<tr>
<td>8 to 35</td>
<td>Channel configuration and channel monitoring of the input channels 1 to 14</td>
<td>see tables</td>
<td>&quot;Channel configuration&quot; and &quot;Channel monitoring&quot;</td>
<td>Byte</td>
<td>Default 0x00</td>
<td>0</td>
<td>19 to 3</td>
<td>0x0Y07 to 0x0Y22</td>
</tr>
</tbody>
</table>

The parameter data directly influences the functionality of modules.

For non-standard applications, it is necessary to adapt the parameters to your system configuration.

Module: Module slot address: Y = 1...10
<table>
<thead>
<tr>
<th>No.</th>
<th>Name</th>
<th>Value</th>
<th>Internal value</th>
<th>Internal value, type</th>
<th>Default</th>
<th>Min.</th>
<th>Max.</th>
<th>EDS Slot/Index</th>
</tr>
</thead>
<tbody>
<tr>
<td>36</td>
<td>Channel configuration</td>
<td>see table &quot;Channel configura-</td>
<td>Byte</td>
<td></td>
<td>0x00</td>
<td>0</td>
<td>19</td>
<td>0x0Y23</td>
</tr>
<tr>
<td></td>
<td>Internal value, type</td>
<td>&quot;tion&quot;</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>37</td>
<td>Channel monitoring</td>
<td>see table &quot;Channel monitor-</td>
<td>Byte</td>
<td></td>
<td>0x00</td>
<td>0</td>
<td>3</td>
<td>0x0Y24</td>
</tr>
<tr>
<td></td>
<td>Internal value, type</td>
<td>&quot;ing&quot;</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

1) With CS31 and addresses less than 70 and FBP, the value is increased by 1
2) Not with FBP

GSD file:

```
Ext_User_Prm_Data.Len = 37
Ext_User_Prm_Data.Const(0) =
0x05, 0xec, 0x22, \
0x01, 0x00, \
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, \
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, \
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, \
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, \
```

Input channel (16 x with AI523)

<table>
<thead>
<tr>
<th>No.</th>
<th>Name</th>
<th>Value</th>
<th>Internal value</th>
<th>Internal value, type</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Channel configuration</td>
<td>see table 2)</td>
<td>see table 2)</td>
<td>Byte</td>
<td>0 0x00 see 3)</td>
</tr>
<tr>
<td>2</td>
<td>Channel monitoring</td>
<td>see table 4)</td>
<td>see table 4)</td>
<td>Byte</td>
<td>0 0x00 see 5)</td>
</tr>
</tbody>
</table>

Table 558: Channel configuration 2)

<table>
<thead>
<tr>
<th>Internal value</th>
<th>Operating modes of the analog inputs, individually configurable</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Unused (default)</td>
</tr>
<tr>
<td>1</td>
<td>Analog input 0 V...10 V</td>
</tr>
</tbody>
</table>

2020/12/10
### Operating modes of the analog inputs, individually configurable

<table>
<thead>
<tr>
<th>Internal value</th>
<th>Monitoring</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>Digital input</td>
</tr>
<tr>
<td>3</td>
<td>Analog input 0 mA...20 mA</td>
</tr>
<tr>
<td>4</td>
<td>Analog input 4 mA...20 mA</td>
</tr>
<tr>
<td>5</td>
<td>Analog input -10 V...+10 V</td>
</tr>
<tr>
<td>8</td>
<td>Analog input Pt100, -50 °C...+400 °C (2-wire)</td>
</tr>
<tr>
<td>9</td>
<td>Analog input Pt100, -50 °C...+400 °C (3-wire), requires 2 channels *)</td>
</tr>
<tr>
<td>10</td>
<td>Analog input 0...10 V via differential inputs, requires 2 channels *)</td>
</tr>
<tr>
<td>11</td>
<td>Analog input -10 V...+10 V via differential inputs, requires 2 channels *)</td>
</tr>
<tr>
<td>14</td>
<td>Analog input Pt100, -50 °C...+70 °C (2-wire)</td>
</tr>
<tr>
<td>15</td>
<td>Analog input Pt100, -50 °C...+70 °C (3-wire), requires 2 channels *)</td>
</tr>
<tr>
<td>16</td>
<td>Analog input Pt1000, -50 °C...+400 °C (2-wire)</td>
</tr>
<tr>
<td>17</td>
<td>Analog input Pt1000, -50 °C...+400 °C (3-wire), requires 2 channels *)</td>
</tr>
<tr>
<td>18</td>
<td>Analog input Ni1000, -50 °C...+150 °C (2-wire)</td>
</tr>
<tr>
<td>19</td>
<td>Analog input Ni1000, -50 °C...+150 °C (3-wire), requires 2 channels *)</td>
</tr>
</tbody>
</table>

*) In the operating modes with 3-wire configuration or with differential inputs, two adjacent analog inputs belong together (e.g. the channels 0 and 1). In these cases, both channels are configured in the desired operating mode. The lower address must be the even address (channel 0). The next higher address must be the odd address (channel 1). The converted analog value is available at the higher address (channel 1).

### Table 559: Channel monitoring 4)

<table>
<thead>
<tr>
<th>Internal value</th>
<th>Monitoring</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Plausibility, open-circuit (broken wire) and short circuit 5)</td>
</tr>
<tr>
<td>1</td>
<td>Open-circuit and short circuit</td>
</tr>
<tr>
<td>2</td>
<td>Plausibility</td>
</tr>
<tr>
<td>3</td>
<td>No monitoring</td>
</tr>
</tbody>
</table>
## Diagnosis

<table>
<thead>
<tr>
<th>Class</th>
<th>Interface</th>
<th>Device</th>
<th>Module</th>
<th>Channel</th>
<th>Error identifier</th>
<th>Error message</th>
<th>Remedy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Byte 6 Bit 6...7</td>
<td>1)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Byte 3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Byte 4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Byte 5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Byte 6 Bit 0...5</td>
<td>FBP diagnosis block</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Module error

<table>
<thead>
<tr>
<th>Class</th>
<th>Interface</th>
<th>Device</th>
<th>Module</th>
<th>Channel</th>
<th>Error identifier</th>
<th>Error message</th>
<th>Remedy</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>14</td>
<td>1...10</td>
<td>31</td>
<td>31</td>
<td>19</td>
<td>Checksum error in the I/O module</td>
<td>Replace I/O module</td>
</tr>
<tr>
<td>11 / 12 ADR 1...10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>14</td>
<td>1...10</td>
<td>31</td>
<td>31</td>
<td>3</td>
<td>Timeout in the I/O module</td>
<td>Replace I/O module</td>
</tr>
<tr>
<td>11 / 12 ADR 1...10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>14</td>
<td>1...10</td>
<td>31</td>
<td>31</td>
<td>40</td>
<td>Different hard-/firmware versions in the module</td>
<td>Replace I/O module</td>
</tr>
<tr>
<td>11 / 12 ADR 1...10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>14</td>
<td>1...10</td>
<td>31</td>
<td>31</td>
<td>43</td>
<td>Internal error in the module</td>
<td>Replace I/O module</td>
</tr>
<tr>
<td>11 / 12 ADR 1...10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>14</td>
<td>1...10</td>
<td>31</td>
<td>31</td>
<td>36</td>
<td>Internal data exchange failure</td>
<td>Replace I/O module</td>
</tr>
<tr>
<td>11 / 12 ADR 1...10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>14</td>
<td>1...10</td>
<td>31</td>
<td>31</td>
<td>9</td>
<td>Overflow diagnosis buffer</td>
<td>New start</td>
</tr>
<tr>
<td>11 / 12 ADR 1...10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>14</td>
<td>1...10</td>
<td>31</td>
<td>31</td>
<td>26</td>
<td>Parameter error</td>
<td>Check master</td>
</tr>
<tr>
<td>11 / 12 ADR 1...10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>14</td>
<td>1...10</td>
<td>31</td>
<td>31</td>
<td>11</td>
<td>Process voltage too low</td>
<td>Check process voltage</td>
</tr>
<tr>
<td>11 / 12 ADR 1...10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>14</td>
<td>1...10</td>
<td>31</td>
<td>31</td>
<td>45</td>
<td>Process voltage is switched off (ON -&gt; OFF)</td>
<td>Process voltage ON</td>
</tr>
<tr>
<td>11 / 12 ADR 1...10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Channel error

<table>
<thead>
<tr>
<th>Class</th>
<th>Interface</th>
<th>Device</th>
<th>Module</th>
<th>Channel</th>
<th>Error identifier</th>
<th>Error message</th>
<th>Remedy</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>14</td>
<td>1...10</td>
<td>1</td>
<td>0...15</td>
<td>48</td>
<td>Analog value overflow or broken wire at an analog input</td>
<td>Check input value or terminal</td>
</tr>
<tr>
<td>11 / 12 ADR 1...10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>14</td>
<td>1...10</td>
<td>1</td>
<td>0...15</td>
<td>7</td>
<td>Analog value underflow at an analog input</td>
<td>Check input value</td>
</tr>
<tr>
<td>11 / 12 ADR 1...10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>14</td>
<td>1...10</td>
<td>1</td>
<td>0...15</td>
<td>47</td>
<td>Short circuit at an analog input</td>
<td>Check terminal</td>
</tr>
<tr>
<td>11 / 12 ADR 1...10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Remarks:
1) In AC500, the following interface identifier applies:
   14 = I/O bus, 11 = COM1 (e.g. CS31 bus), 12 = COM2.
   The FBP diagnosis block does not contain this identifier.

2) With "Device" the following allocation applies:
   31 = module itself, 1..10 = decentralized communication interface module 1...10, ADR = hardware address (e.g. of the DC551)

3) With "Module" the following allocation applies depending on the master:
   Module error: I/O bus or FBP: 31 = module itself; COM1/COM2: 1..10 = expansion 1...10
   Channel error: I/O bus or FBP = module type (1 = AI); COM1/COM2: 1..10 = expansion 1...10

4) In case of module errors, with channel "31 = Module itself" is output.

State LEDs

During the power ON procedure, the module initializes automatically. All LEDs (except the channel LEDs) are ON during this time.

<table>
<thead>
<tr>
<th>LED</th>
<th>State</th>
<th>Color</th>
<th>LED = OFF</th>
<th>LED = ON</th>
<th>LED flashes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inputs I0...I7 and I8...I15</td>
<td>Analog input</td>
<td>Yellow</td>
<td>Input is OFF</td>
<td>Input is ON (brightness depends on the value of the analog signal)</td>
<td>--</td>
</tr>
<tr>
<td>UP</td>
<td>Process voltage 24 VDC via terminal</td>
<td>Green</td>
<td>Process voltage is missing</td>
<td>Process voltage OK</td>
<td>--</td>
</tr>
<tr>
<td>CH-ERR2</td>
<td>Channel error, error messages in groups (analog inputs or outputs combined into the groups 2 and 4)</td>
<td>Red</td>
<td>No error or process voltage is missing</td>
<td>Severe error within the corresponding group</td>
<td>Error on one channel of the group</td>
</tr>
<tr>
<td>CH-ERR4</td>
<td></td>
<td>Red</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CH-ERR *)</td>
<td>Module error</td>
<td>Red</td>
<td>--</td>
<td>Internal error</td>
<td>--</td>
</tr>
</tbody>
</table>

*) Both LEDs (CH-ERR2 and CH-ERR4) light up together
# Measuring ranges

## Input ranges of voltage, current and digital input

<table>
<thead>
<tr>
<th>Range</th>
<th>0...10 V</th>
<th>-10...+10 V</th>
<th>0...20 mA</th>
<th>4...20 mA</th>
<th>Digital input</th>
<th>Digital value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Overflow</td>
<td>&gt;11.7589</td>
<td>&gt;11.7589</td>
<td>&gt;23.5178</td>
<td>&gt;22.8142</td>
<td></td>
<td>32767</td>
</tr>
<tr>
<td>Normal range</td>
<td>10.0000 :</td>
<td>0.0004</td>
<td>20.0000   :</td>
<td>4.0006 : OFF</td>
<td>27648 :</td>
<td>6C01</td>
</tr>
<tr>
<td>Normal range or measured value too low</td>
<td>-0.0004 :</td>
<td>-1.7593 : 3.9994 :</td>
<td>-1 : FFFF : -4864 : E000</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Measured value too low</td>
<td>-10.0004 :</td>
<td>-11.7589 : -27649 : 93FF :</td>
<td>-32512 : 8100 :</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Underflow</td>
<td>&lt; -1.7593</td>
<td>&lt; -11.7589</td>
<td>&lt; 0.0000</td>
<td>&lt; 1.1858</td>
<td>-32768 :</td>
<td>8000</td>
</tr>
</tbody>
</table>

The represented resolution corresponds to 16 bits.

## Input ranges resistance temperature detector

The resolution corresponds to 16 bits.

<table>
<thead>
<tr>
<th>Range</th>
<th>Pt100 / Pt1000 -50...70 °C</th>
<th>Pt100 / Pt1000 -50...400 °C</th>
<th>Ni1000 -50...150 °C</th>
<th>Digital value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Overflow</td>
<td>&gt; 80.0 °C</td>
<td>&gt; 450.0 °C</td>
<td>&gt; 160.0 °C</td>
<td>32767</td>
</tr>
<tr>
<td>Measured value too high</td>
<td>450.0 °C : 400.1 °C</td>
<td>4500 : 4001</td>
<td>1194 : 0FA1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>160.0 °C : 150.1 °C</td>
<td>1600 : 1501</td>
<td>0640 : 05DD</td>
<td></td>
</tr>
<tr>
<td></td>
<td>80.0 °C : 70.1 °C</td>
<td>800 : 701</td>
<td>0320 : 02BD</td>
<td></td>
</tr>
</tbody>
</table>
### Technical data

The system data of AC500 and S500 \( \text{Chapter 1.5.4.5.1 “System data AC500” on page 3058} \) are valid for standard version.

The system data of AC500-XC \( \text{Chapter 1.5.4.6.1 “System data AC500-XC” on page 3099} \) are valid for the XC version.

Only additional details are therefore documented below.

The technical data are also valid for the XC version.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Process voltage</strong></td>
<td></td>
</tr>
<tr>
<td>Connections</td>
<td>Terminals 1.8, 2.8, 3.8 and 4.8 for +24 V (UP) as well as 1.9, 2.9, 3.9 and 4.9 for 0 V (ZP)</td>
</tr>
<tr>
<td>Rated value</td>
<td>24 VDC</td>
</tr>
<tr>
<td>Max. ripple</td>
<td>5 %</td>
</tr>
<tr>
<td>Protection against reversed voltage</td>
<td>Yes</td>
</tr>
<tr>
<td>Rated protection fuse on UP</td>
<td>10 A fast</td>
</tr>
<tr>
<td>Galvanic isolation</td>
<td>Yes, per module</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Current consumption</strong></td>
<td></td>
</tr>
<tr>
<td>From 24 VDC power supply at the terminals UP/L+ and ZP/M of the CPU/bus module</td>
<td>Ca. 2 mA</td>
</tr>
<tr>
<td>From UP at normal operation / with outputs</td>
<td>0.15 A + output loads</td>
</tr>
<tr>
<td>Inrush current from UP (at power up)</td>
<td>0.050 A²s</td>
</tr>
<tr>
<td>Max. length of analog cables, conductor cross section &gt; 0.14 mm²</td>
<td>100 m</td>
</tr>
</tbody>
</table>
### Technical data of the analog inputs

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of channels per module</td>
<td>16</td>
</tr>
<tr>
<td>Distribution of channels into groups</td>
<td>2 groups of 8 channels each</td>
</tr>
<tr>
<td>Connections of the channels I0- to I7-</td>
<td>Terminals 1.0 to 1.7</td>
</tr>
<tr>
<td>Connections of the channels I0+ to I7+</td>
<td>Terminals 2.0 to 2.7</td>
</tr>
<tr>
<td>Connections of the channels I8- to I15-</td>
<td>Terminals 3.0 to 3.7 Terminals 4.0 to 4.7</td>
</tr>
<tr>
<td>Connections of the channels I8+ to I15+</td>
<td>Terminals 4.0 to 4.7</td>
</tr>
<tr>
<td>Input type</td>
<td>Bipolar (not with current or Pt100/ Pt1000/ Ni1000)</td>
</tr>
<tr>
<td>Galvanic isolation</td>
<td>Against internal supply and other modules</td>
</tr>
<tr>
<td>Configurability</td>
<td>0 V...10 V, -10 V...+10 V, 0/4 mA...20 mA, Pt100/1000, Ni1000 (each input can be configured individually)</td>
</tr>
<tr>
<td>Channel input resistance</td>
<td>Voltage: &gt; 100 kΩ</td>
</tr>
<tr>
<td></td>
<td>Current: ca. 330 Ω</td>
</tr>
<tr>
<td>Time constant of the input filter</td>
<td>Voltage: 100 μs</td>
</tr>
<tr>
<td></td>
<td>Current: 100 μs</td>
</tr>
<tr>
<td>Indication of the input signals</td>
<td>1 LED per channel</td>
</tr>
<tr>
<td>Conversion cycle</td>
<td>2 ms (for 16 inputs), with Pt/Ni... 1 s</td>
</tr>
<tr>
<td>Resolution</td>
<td>Range 0 V...10 V: 12 bits</td>
</tr>
<tr>
<td></td>
<td>Range -10 V...+10 V: 12 bits + sign</td>
</tr>
<tr>
<td></td>
<td>Range 0 mA...20 mA: 12 bits</td>
</tr>
<tr>
<td></td>
<td>Range 4 mA...20 mA: 12 bits</td>
</tr>
<tr>
<td>Conversion error of the analog values caused by non-linearity, adjustment error at factory and resolution within the normal range</td>
<td>Typ. ±0.5 % of full scale at 25 °C</td>
</tr>
<tr>
<td></td>
<td>Max. ±1 % of full scale (all ranges)</td>
</tr>
<tr>
<td></td>
<td>at 0 °C...60 °C or EMC disturbance</td>
</tr>
</tbody>
</table>

**NOTICE!**
**Attention:**
All I/O channels (digital and analog) are protected against reverse polarity, reverse supply, short circuit and continuous overvoltage up to 30 VDC.
### Technical data of the analog inputs, if used as digital inputs

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of channels per module</td>
<td>Max. 16</td>
</tr>
<tr>
<td>Distribution of channels into groups</td>
<td>2 groups of 8 channels each</td>
</tr>
<tr>
<td>Connections of the channels I0+ to I7+</td>
<td>Terminals 2.0 to 2.7</td>
</tr>
<tr>
<td>Connections of the channels I8+ to I15+</td>
<td>Terminals 4.0 to 4.7</td>
</tr>
<tr>
<td>Reference potential for the inputs</td>
<td>Terminals 1.9, 2.9, 3.9 and 4.9 (ZP)</td>
</tr>
<tr>
<td>Input signal delay</td>
<td>Typ. 8 ms, configurable from 0.1 to 32 ms</td>
</tr>
<tr>
<td>Indication of the input signals</td>
<td>1 LED per channel</td>
</tr>
<tr>
<td>Input signal voltage</td>
<td>24 VDC</td>
</tr>
<tr>
<td>Signal 0</td>
<td>-30 V...+5 V</td>
</tr>
<tr>
<td>Undefined signal</td>
<td>+5 V...+13 V</td>
</tr>
<tr>
<td>Signal 1</td>
<td>+13 V...+30 V</td>
</tr>
<tr>
<td>Input current per channel</td>
<td></td>
</tr>
<tr>
<td>Input voltage +24 V</td>
<td>Typ. 7 mA</td>
</tr>
<tr>
<td>Input voltage +5 V</td>
<td>Typ. 1.4 mA</td>
</tr>
<tr>
<td>Input voltage +15 V</td>
<td>Typ. 4.3 mA</td>
</tr>
<tr>
<td>Input voltage +30 V</td>
<td>&lt; 9 mA</td>
</tr>
<tr>
<td>Input resistance</td>
<td>Ca. 3.5 kΩ</td>
</tr>
</tbody>
</table>

### Ordering data

<table>
<thead>
<tr>
<th>Part no.</th>
<th>Description</th>
<th>Product life cycle phase *)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1SAP 250 300 R0001</td>
<td>AI523, analog input module, 16 Al, U/I/Pt100, 12 bits + sign, 2-wires</td>
<td>Active</td>
</tr>
<tr>
<td>1SAP 450 300 R0001</td>
<td>AI523-XC, analog input module, 16 Al, U/I/Pt100, 12 bits + sign, 2-wires, XC version</td>
<td>Active</td>
</tr>
</tbody>
</table>

*) For planning and commissioning of new installations use modules in Active status only.
AI531 - Analog input module

- 8 configurable analog inputs (I0 to I7) in 2 groups (1.0...1.7 and 2.0...2.7 as well as 3.0...3.7 and 4.0...4.7)
  Resolution 15 bits plus sign
- Module-wise electrically isolated
- XC version for use in extreme ambient conditions available

### Intended purpose

The device can be used as a decentralized I/O extension module for S500 Communication Interface Modules (e.g., CI592-CS31, CI501-PNIO, CI541-DP, CI581-CN) or as a centralized extension module for AC500 CPUs (PM5xx).
Functionality

8 analog inputs, individually configurable for
- Unused (default setting)
- 0 V...5 V, 0 V...10 V
- -50 mV...+50 mV, -500 mV...+500 mV
- -1 V...+1 V, -5 V...+5 V, -10 V...+10 V
- 0 mA...20 mA
- 4 mA...20 mA
- -20 mA...20 mA
- Pt100, -50 °C...+70 °C or 400 °C (2-, 3- and 4-wire)
- Pt100, -200 °C...+850 °C (2-, 3- and 4-wire)
- Pt1000, -50 °C...+400 °C (2-, 3- and 4-wire)
- Ni1000, -50 °C...+150 °C (2-, 3- and 4-wire)
- Cu50 (1.426): -50 °C...+200 °C (2-, 3- and 4-wire)
- Cu50 (1.428): -200 °C...+200 °C (2-, 3- and 4-wire)
- 0 Ω...50 kΩ
- Thermocouples of types J, K, T, N, S
- Resistance measuring bridge
- Digital signals (digital input)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Resolution of the analog channels</td>
<td></td>
</tr>
<tr>
<td>Voltage and current, bipolar</td>
<td>15 bits plus sign</td>
</tr>
<tr>
<td>Voltage and current, unipolar</td>
<td>15 bits</td>
</tr>
<tr>
<td>Temperature</td>
<td>0.1 °C (0,01°C at Pt100 -50 °C...+70 °C)</td>
</tr>
<tr>
<td>LED displays</td>
<td>11 LEDs for signals and error messages</td>
</tr>
<tr>
<td>Internal power supply</td>
<td>through the expansion bus interface (I/O bus)</td>
</tr>
<tr>
<td>External power supply</td>
<td>via terminals (process voltage UP = 24 VDC)</td>
</tr>
<tr>
<td>Required terminal unit</td>
<td>TU515 or TU516  ¤  Chapter 1.5.3.5.2 “TU515, TU516, TU541 and TU542 for I/O modules” on page 2311</td>
</tr>
</tbody>
</table>

Electrical connection

For a detailed description of the mounting, disassembly and electrical connection of the module, please refer to the System Assembly, Construction and Connection chapter ¤ Chapter 1.5.4.5 “AC500 (Standard)” on page 3058.

The modules are plugged on an I/O terminal unit ¤ Chapter 1.5.3.5.2 “TU515, TU516, TU541 and TU542 for I/O modules” on page 2311. Properly position the modules and press until they lock in place. The terminal units are mounted on a DIN rail or with 2 screws plus the additional accessory for wall mounting (TA526 ¤ Chapter 1.5.3.8.1.4 “TA526 - Wall mounting accessory” on page 3035).

The electrical connection of the I/O channels is carried out using the 40 terminals of the I/O terminal unit. I/O modules can be replaced without re-wiring the terminal units.
The terminals 1.8, 2.8, 3.8, 4.8, 1.9, 2.9, 3.9 and 4.9 are electrically interconnected within the I/O terminal units and always have the same assignment, independent of the inserted module:
Terminals 1.8, 2.8, 3.8 and 4.8: process voltage UP = +24 VDC
Terminals 1.9, 2.9, 3.9 and 4.9: process voltage ZP = 0 V
The assignment of the other terminals:

<table>
<thead>
<tr>
<th>Terminals</th>
<th>Signal</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.0, 2.2, 2.4, 2.6</td>
<td>I0+ to I3+</td>
<td>Positive poles of the first 4 analog inputs</td>
</tr>
<tr>
<td>1.0, 1.2, 1.4, 1.6</td>
<td>I0- to I3-</td>
<td>Negative poles of the first 4 analog inputs</td>
</tr>
<tr>
<td>2.1, 2.3, 2.5, 2.7</td>
<td>I0A to I3A</td>
<td>Connections A (supply) of the first 4 analog inputs</td>
</tr>
<tr>
<td>1.1, 1.3, 1.5, 1.7</td>
<td>I0B to I3B</td>
<td>Connections B (analog ground) of the first 4 analog inputs</td>
</tr>
<tr>
<td>4.0, 4.2, 4.4, 4.6</td>
<td>I4+ to I7+</td>
<td>Positive poles of the following 4 analog inputs</td>
</tr>
<tr>
<td>3.0, 3.2, 3.4, 3.6</td>
<td>I4- to I7-</td>
<td>Negative poles of the following 4 analog inputs</td>
</tr>
<tr>
<td>4.1, 4.3, 4.5, 4.7</td>
<td>I4A to I7A</td>
<td>Connections A (supply) of the following 4 analog inputs</td>
</tr>
<tr>
<td>3.1, 3.3, 3.5, 3.7</td>
<td>I4B to I7B</td>
<td>Connections B (analog ground) of the following 4 analog inputs</td>
</tr>
</tbody>
</table>

**CAUTION!**
Analog sensors must be electrically isolated against the earth. In order to avoid inaccuracy with the measuring results, the analog sensors should also be isolated against the power supply.

The "IxB" clamps (x=0..7) of the analog inputs are electrically connected to each other. They form an "Analog Ground Signal" (AGND) for the module.

The negative poles of the analog inputs Ix- may accept a potential difference up to ±20 VDC with regard to the common reference potential IxB (AGND, ZP). Observing this maximum voltage difference, analog current inputs of one module can be switched in series to each other and also with current inputs of other modules.

For the open-circuit detection (cut wire), each positive analog input channel Ix+ is pulled up to "plus" by a high-resistance resistor and each negative analog input channel Ix- is pulled down to "minus" by a resistor. If cut wire occurs, a maximum voltage (overflow or underflow) will be read in then.

The internal power supply voltage for the module’s circuitry is carried out via the I/O bus (provided by a bus module or a CPU). Thus, the current consumption from 24 VDC power supply at the terminals L+/UP and M/ZP of the CPU/bus module increases by 2 mA per AI531.
The external power supply connection is carried out via the UP (+24 VDC) and the ZP (0 VDC) terminals.

**NOTICE!**

**Risk of damaging the PLC modules!**

Overvoltages and short circuits might damage the PLC modules.

- Make sure that all voltage sources (supply voltage and process supply voltage) are switched off before you begin with operations at the system.
- Never connect any voltages or signals to reserved terminals (marked with ---). Reserved terminals may carry internal voltages.

Generally, analog signals must be laid in shielded cables. The cable shields must be earthed at both sides of the cables. In order to avoid unacceptable potential differences between different parts of the installation, low resistance equipotential bonding conductors must be laid.

*Only for simple applications (low electromagnetic disturbances, no high requirement on precision), the shielding can also be omitted.*

---

**Fig. 126: Terminal assignment of the module**

The module provides several diagnosis functions © Chapter 1.5.3.6.2.2.2.7 “Diagnosis” on page 2642.
Connection of active-type analog sensors (Voltage) with electrically isolated power supply

**Standard ranges**

![Diagram of sensor connection]

- Voltage: -50 mV...+50 mV
- Voltage: -500 mV...+500 mV
- Voltage: -1 V...+1 V
- Voltage: -5 V...+5 V
- Voltage: -10 V...+10 V
- Voltage: 0 V...+5 V
- Voltage: 0 V...+10 V

**Common mode range (+/-20 V)**

![Diagram of sensor connection]

- Voltage: -20 V...+20 V

**Fig. 127: Connection example**

The measuring ranges can be configured in **Chapter 1.5.3.6.2.2.2.6 “Parameterization” on page 2638**:

<table>
<thead>
<tr>
<th>Voltage</th>
<th>Range</th>
<th>Channels used</th>
</tr>
</thead>
<tbody>
<tr>
<td>Voltage</td>
<td>-50 mV...+50 mV</td>
<td>1 channel used</td>
</tr>
<tr>
<td>Voltage</td>
<td>-500 mV...+500 mV</td>
<td>1 channel used</td>
</tr>
<tr>
<td>Voltage</td>
<td>-1 V...+1 V</td>
<td>1 channel used</td>
</tr>
<tr>
<td>Voltage</td>
<td>-5 V...+5 V</td>
<td>1 channel used</td>
</tr>
<tr>
<td>Voltage</td>
<td>-10 V...+10 V</td>
<td>1 channel used</td>
</tr>
<tr>
<td>Voltage</td>
<td>0 V...+5 V</td>
<td>1 channel used</td>
</tr>
<tr>
<td>Voltage</td>
<td>0 V...+10 V</td>
<td>1 channel used</td>
</tr>
</tbody>
</table>

**Fig. 128: Connection example**

The measuring range can be configured in **Chapter 1.5.3.6.2.2.2.6 “Parameterization” on page 2638**:
The function of the LEDs is described under Diagnosis and displays \( \text{Chapter 1.5.3.6.2.2.2.7 "Diagnosis" on page 2642.} \)

In order to avoid error messages or long processing times, it is useful to configure unused analog input channels as "unused".

---

**Connection of active-type analog sensors (Voltage) with no electrically isolated power supply**

**Standard ranges**

<table>
<thead>
<tr>
<th>Voltage</th>
<th>Common mode voltage</th>
<th>1 channel used</th>
</tr>
</thead>
<tbody>
<tr>
<td>-50 mV...+50 mV</td>
<td>-500 mV...+500 mV</td>
<td>1 channel used</td>
</tr>
<tr>
<td>-1 V...+1 V</td>
<td>-5 V...+5 V</td>
<td>1 channel used</td>
</tr>
<tr>
<td>0 V...+5 V</td>
<td>0 V...+10 V</td>
<td>1 channel used</td>
</tr>
</tbody>
</table>

---

**CAUTION!**

If GND is not directly connected to ZP at the sensor, the supply current flows via the GND line to ZP. Measuring errors can only occur caused by voltage differences higher than \( \pm 20 \text{VDC} \) between GND and ZP.

The measuring ranges can be configured \( \text{Chapter 1.5.3.6.2.2.2.6 "Parameterization" on page 2638} \)
CAUTION!
If GND is not directly connected to ZP at the sensor, the supply current flows via the GND line to ZP. Measuring errors can only occur caused by voltage differences higher than ±20 VDC between GND and ZP.

The measuring range can be configured "Chapter 1.5.3.6.2.2.2.6 “Parameterization” on page 2638:

<table>
<thead>
<tr>
<th>Voltage</th>
<th>Common mode voltage</th>
<th>1 channel used</th>
</tr>
</thead>
</table>

The function of the LEDs is described under Diagnosis and displays / displays "Chapter 1.5.3.6.2.2.2.7 “Diagnosis” on page 2642.

In order to avoid error messages or long processing times, it is useful to configure unused analog input channels as “unused”.

Connection of active-type analog sensors (Current) with electrically isolated power supply

*Fig. 131: Connection example*
The following measuring ranges can be configured "Parameterization" on page 2638:

<table>
<thead>
<tr>
<th>Current</th>
<th>-20 mA...20 mA</th>
<th>1 channel used</th>
</tr>
</thead>
<tbody>
<tr>
<td>Current</td>
<td>0 mA...20 mA</td>
<td>1 channel used</td>
</tr>
<tr>
<td>Current</td>
<td>4 mA...20 mA</td>
<td>1 channel used</td>
</tr>
</tbody>
</table>

The function of the LEDs is described under Diagnosis and displays / displays "Diagnosis" on page 2642.

Unused input channels can be left open, because they are of low resistance.

Connection of active-type analog sensors (Current) with electrically isolated power supply and series-connection of an additional input

![Connection Diagram](image)

*Fig. 132: Connection example*

1 Analog input of the second device

If series-connection of an additional input is used, the input resistance of the module (ca. 330 Ω) must be added to the input resistance of the second device. Make sure that the maximum permitted load resistance of the analog sensor is not exceeded (see the data sheet of the analog sensor).

The input of the module is not related to ZP. If the input of the second device is related to ZP, the order of sequence in the series-connection must be observed by all means (from the sensor to the module and then to the input of the second device).

The following measuring ranges can be configured "Parameterization" on page 2638:

<table>
<thead>
<tr>
<th>Current</th>
<th>-20 mA...20 mA</th>
<th>1 channel used</th>
</tr>
</thead>
<tbody>
<tr>
<td>Current</td>
<td>0 mA...20 mA</td>
<td>1 channel used</td>
</tr>
<tr>
<td>Current</td>
<td>4 mA...20 mA</td>
<td>1 channel used</td>
</tr>
</tbody>
</table>
For a description of the functions of the LEDs, please refer to Diagnosis and displays / displays Chapter 1.5.3.6.2.2.2.7 “Diagnosis” on page 2642.

Unused input channels can be left open, because they are of low resistance.

Connection of passive-type analog sensors (Current)

Fig. 133: Connection example

The following measuring ranges can be configured Chapter 1.5.3.6.2.2.6 “Parameterization” on page 2638:

<table>
<thead>
<tr>
<th>Current</th>
<th>Measuring Range</th>
<th>Channels Used</th>
</tr>
</thead>
<tbody>
<tr>
<td>Current</td>
<td>-20 mA... 20 mA *)</td>
<td>1 channel used</td>
</tr>
<tr>
<td>Current</td>
<td>0 mA... 20 mA *)</td>
<td>1 channel used</td>
</tr>
<tr>
<td>Current</td>
<td>4 mA... 20 mA</td>
<td>1 channel used</td>
</tr>
</tbody>
</table>

*) This setting is not applicable with passive-type analog sensors (current).

The function of the LEDs is described under Diagnosis and displays / displays Chapter 1.5.3.6.2.2.2.7 “Diagnosis” on page 2642.

Unused input channels can be left open, because they are of low resistance.
Connection of passive-type analog sensors (Current) and series-connection of an additional analog sensor

![Connection diagram](image)

*Fig. 134: Connection example*

1. Analog input of the second device

---

If series-connection of an additional input is used, the input resistance of the module (ca. 330 Ω) must be added to the input resistance of the second device. Make sure that the maximum permitted load resistance of the analog sensor is not exceeded (see the data sheet of the analog sensor).

---

The input of the module is not related to ZP. If the input of the second device is related to ZP, the order of sequence in the series-connection must be observed by all means (from the sensor to the module and then to the input of the second device).

---

The following measuring ranges can be configured

<table>
<thead>
<tr>
<th>Measuring Range</th>
<th>Channels Used</th>
</tr>
</thead>
<tbody>
<tr>
<td>Current -20 mA...20 mA (*)</td>
<td>1 channel used</td>
</tr>
<tr>
<td>Current 0 mA...20 mA (*)</td>
<td>1 channel used</td>
</tr>
<tr>
<td>Current 4 mA...20 mA</td>
<td>1 channel used</td>
</tr>
</tbody>
</table>

*) This setting is not applicable with passive-type analog sensors (current).

---

The function of the LEDs is described under Diagnosis and displays / displays

---

Unused input channels can be left open, because they are of low resistance.

---

Connection of digital signal sources at analog inputs

Several (or all) analog inputs can be configured as digital inputs. The inputs are not electrically isolated against the other analog channels.
The following operating mode can be configured (Chapter 1.5.3.6.2.2.2.6 “Parameterization” on page 2638):

<table>
<thead>
<tr>
<th>Digital input</th>
<th>24 V</th>
<th>1 channel used</th>
</tr>
</thead>
<tbody>
<tr>
<td>Effect of incorrect input terminal connection</td>
<td>Wrong or no signal detected, no damage up to 35 V</td>
<td></td>
</tr>
</tbody>
</table>

For a description of the function of the LEDs, please refer to Diagnosis and displays (Chapter 1.5.3.6.2.2.2.7 “Diagnosis” on page 2642).

**Connection of resistance thermometers in 2-wire configuration**

When resistance thermometers (Pt100, Pt1000, Ni1000, Cu50) are used, a constant current must flow through them to build the necessary voltage drop for the evaluation. For this, the module AI531 provides a constant current source which is multiplexed over the 4 analog channels.

The following measuring ranges can be configured (Chapter 1.5.3.6.2.2.2.6 “Parameterization” on page 2638):
Connection of resistance thermometers in 3-wire configuration

When resistance thermometers (Pt100, Pt1000, Ni1000, Cu50) are used, a constant current must flow through them to build the necessary voltage drop for the evaluation. For this, the module AI531 provides a constant current source which is multiplexed over the 4 analog channels.

![Connection example](image)

The following measuring ranges can be configured:

<table>
<thead>
<tr>
<th>Thermometer</th>
<th>Measuring Range</th>
<th>Channels Used</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pt100</td>
<td>-50 °C...+70 °C / +400 °C; -200 °C...+850 °C</td>
<td>1 channel used</td>
</tr>
<tr>
<td>Pt1000</td>
<td>-50 °C...+400 °C</td>
<td>1 channel used</td>
</tr>
<tr>
<td>Ni1000</td>
<td>-50 °C...+150 °C</td>
<td>1 channel used</td>
</tr>
<tr>
<td>Cu50</td>
<td>-50 °C...+200 °C (1.426); -200 °C...+200 °C (1.428)</td>
<td>1 channel used</td>
</tr>
</tbody>
</table>

For a description of the function of the LEDs, please refer to Diagnosis and displays / displays “Chapter 1.5.3.6.2.2.2.7 “Diagnosis” on page 2642.

The module linearizes the resistance thermometer characteristics. In order to avoid error messages from unused analog input channels, it is useful to configure them as "unused".
In order to avoid error messages from unused analog input channels, it is useful to configure them as "unused".

Connection of resistance thermometers in 4-wire configuration

When resistance thermometers (Pt100, Pt1000, Ni1000, Cu50) are used, a constant current must flow through them to build the necessary voltage drop for the evaluation. For this, the module AI531 provides a constant current source which is multiplexed over the 4 analog channels.

![Connection example](image)

**Fig. 138: Connection example**

The following measuring ranges can be configured [Chapter 1.5.3.6.2.2.2.6 “Parameterization” on page 2638]:

<table>
<thead>
<tr>
<th>Thermometer</th>
<th>Measuring Range</th>
<th>Channels Used</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pt100</td>
<td>-50 °C...+70 °C / +400 °C; -200 °C...+850 °C</td>
<td>1 channel used</td>
</tr>
<tr>
<td>Pt1000</td>
<td>-50 °C...+400 °C</td>
<td>1 channel used</td>
</tr>
<tr>
<td>Ni1000</td>
<td>-50 °C...+150 °C</td>
<td>1 channel used</td>
</tr>
<tr>
<td>Cu50</td>
<td>-50 °C...+200 °C (1.426); -200 °C...+200 °C (1.428)</td>
<td>1 channel used</td>
</tr>
</tbody>
</table>

For a description of the function of the LEDs, please refer to Diagnosis and displays / displays [Chapter 1.5.3.6.2.2.2.7 “Diagnosis” on page 2642].

The module linearizes the resistance thermometer characteristics. In order to keep measuring errors as small as possible, it is necessary by all means, to have all the involved conductors in the same cable.

In order to avoid error messages from unused analog input channels, it is useful to configure them as "unused".

Connection of resistors in 2-wire configuration

For evaluating resistors, a constant current must flow through them to build the necessary voltage drop. For this, the module AI531 provides a constant current source which is multiplexed over the 4 analog channels.
Fig. 139: Connection example

The following measuring ranges can be configured  \( \text{Chapter 1.5.3.6.2.2.2.6 “Parameterization” on page 2638} \):

<table>
<thead>
<tr>
<th>Resistor</th>
<th>50 kΩ</th>
<th>1 channel used</th>
</tr>
</thead>
</table>

For a description of the function of the LEDs, please refer to Diagnosis and displays / displays  \( \text{Chapter 1.5.3.6.2.2.2.7 “Diagnosis” on page 2642} \).

In order to avoid error messages from unused analog input channels, it is useful to configure them as “unused”.

**Connection of a resistance measuring bridge with internal supply**

When resistance measuring bridges are connected, the short-circuit-proof voltage output (internal supply) at pin I0A (or I2A, I4A, I6A) must be used. This supply voltage is activated as soon as "Voltage Measurement" is configured for the relevant channel.
Fig. 140: Connection example

1 Internal supply

All voltage measuring ranges can be configured \(\text{\textcopyright} \) Chapter 1.5.3.6.2.2.2.6 "Parameterization" on page 2638.

The calculation of the resistor deviation must be performed via the bridge voltage by the PLC user program.

Connection of a resistance measuring bridge with external supply

With the connection of a resistance measuring bridge with external supply, the supply voltage is provided separately.
Fig. 141: Connection example

1 Bridge to IxB necessary with electrically isolated supply

All voltage measuring ranges can be configured ≈ Chapter 1.5.3.6.2.2.2.6 "Parameterization" on page 2638:

The calculation of the resistor deviation must be performed via the bridge voltage by the PLC user program.

Connection of thermocouples

Fig. 142: Connection example

The following measuring ranges can be configured ≈ Chapter 1.5.3.6.2.2.2.6 "Parameterization" on page 2638:

<table>
<thead>
<tr>
<th>Type</th>
<th>Temperature Range</th>
<th>Material</th>
<th>Channels Used</th>
</tr>
</thead>
<tbody>
<tr>
<td>J type</td>
<td>-210 °C...1200 °C</td>
<td>Fe-CuNi</td>
<td>1 channel used</td>
</tr>
<tr>
<td>K type</td>
<td>-270 °C...1372 °C</td>
<td>Ni-CrNi</td>
<td>1 channel used</td>
</tr>
<tr>
<td>Type</td>
<td>Temperature</td>
<td>Thermocouple</td>
<td>Channels Used</td>
</tr>
<tr>
<td>--------</td>
<td>-------------------</td>
<td>--------------------</td>
<td>---------------</td>
</tr>
<tr>
<td>N type</td>
<td>-270 °C...1300 °C</td>
<td>NiCrSi-NiSi</td>
<td>1</td>
</tr>
<tr>
<td>S type</td>
<td>-50 °C...1768 °C</td>
<td>Pt10Rh-Pt</td>
<td>1</td>
</tr>
<tr>
<td>T type</td>
<td>-270 °C...400 °C</td>
<td>Cu-CuNi</td>
<td>1</td>
</tr>
</tbody>
</table>

For a description of the function of the LEDs, please refer to Diagnosis and displays / displays Chapter 1.5.6.2.2.7 “Diagnosis” on page 2642.

The module linearizes the thermocouple characteristics. It supports the following possibilities of temperature compensation and handling with cold junctions:

**Internal compensation**

An internal temperature sensor which is located next to the terminal unit is used to detect the temperature of the cold junction. So the compensating cables must be connected directly to the terminal unit, where the cold junction is located.

The setting "Internal compensation (default)" for the parameter "Compensation channel" should be selected.

> To get more precise temperature measurements, the use of an external compensation method is recommended.

**External compensation with temperature input**

The temperature for the cold junction can be determined externally.

A measured or known temperature value (e.g. ambient temperature in the cabinet) is transferred to the module via the output data word to all required channels. The possible temperature range is from -25 °C to +60 °C and is monitored by the AI531.

The setting "External with temperature value" for the parameter "Compensation channel" should be selected.

**External compensation with compensation box**

A compensation box balances the temperature difference between the cold junction and the reference temperature by generating a bridge voltage. The reference temperature is transferred via the output data word.

The compensation box must fit to the type of thermocouple and is located at the end of the compensating cables, where the cold junction is located. The cabling to the AI531 can be carried out with normal cables. The operating manual of the compensation box also has to be considered.

The setting "External with temperature value" for the parameter "Compensation channel" should be selected.

**External compensation with flanking channel**

A flanking channel of the same input group can be used for compensation, e.g. for channel 3, the channels 0, 1 and 2 can be selected as reference channels. The type of sensor for the reference channel can be selected in the parameters for the flanking channel. For example, a RTD sensor which is located next to the thermocouple terminal can be used as reference point for other channels.
The setting "Channel x" for the parameter "Compensation channel" should be selected. Refer to Channel configuration § Chapter 1.5.3.6.2.2.2.6 “Parameterization” on page 2638 for possible settings.

In order to avoid error messages from unused analog input channels, it is useful to configure them as "unused".

Internal data exchange

<table>
<thead>
<tr>
<th>Digital inputs (bytes)</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Digital outputs (bytes)</td>
<td>0</td>
</tr>
<tr>
<td>Analog inputs (words)</td>
<td>8</td>
</tr>
<tr>
<td>Analog outputs (words)</td>
<td>1</td>
</tr>
</tbody>
</table>

I/O configuration

The module does not store configuration data itself. It gets its parameterization data from the master device of the I/O bus (CPU or bus module) during power-up of the system.

This means that replacing I/O modules is possible without any re-parameterization via software.

If the external power supply voltage via UP/ZP terminals fails, the I/O module loses its configuration data. The whole station has to be switched off and on again to re-configure the module.

Parameterization

<table>
<thead>
<tr>
<th>Firmware version</th>
<th>Configuration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Firmware version &gt; V2.0.0</td>
<td>The arrangement of the parameter data is performed by Control Builder Plus/ Automation Builder software.</td>
</tr>
</tbody>
</table>

The parameter data directly influences the functionality of modules.

For non-standard applications, it is necessary to adapt the parameters to your system configuration.

Module: Module slot address: Y = 1...10

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>Internal value</th>
<th>Internal value, Type</th>
<th>Default</th>
<th>Min.</th>
<th>Max.</th>
<th>EDS Slot/Index</th>
</tr>
</thead>
<tbody>
<tr>
<td>Module ID</td>
<td>Internal</td>
<td>1535 1)</td>
<td>Word</td>
<td>1535 0x05ff</td>
<td>0</td>
<td>65535</td>
<td>0x0Y01</td>
</tr>
<tr>
<td>Ignore module 2)</td>
<td>No</td>
<td>0</td>
<td>Byte</td>
<td>No 0x00</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Parameter length in bytes</td>
<td>Internal</td>
<td>36</td>
<td>Byte</td>
<td>36 0</td>
<td></td>
<td>255</td>
<td>0x0Y02</td>
</tr>
<tr>
<td>Name</td>
<td>Value</td>
<td>Internal value</td>
<td>Internal value, Type</td>
<td>Default</td>
<td>Min.</td>
<td>Max.</td>
<td>EDS Slot/ Index</td>
</tr>
<tr>
<td>-----------------------------------</td>
<td>-------</td>
<td>----------------</td>
<td>----------------------</td>
<td>---------</td>
<td>------</td>
<td>------</td>
<td>-----------------</td>
</tr>
<tr>
<td>Check supply</td>
<td>Off</td>
<td>0</td>
<td>Byte</td>
<td>On</td>
<td>0x01</td>
<td></td>
<td>0x0Y03</td>
</tr>
<tr>
<td></td>
<td>On</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Analog data format</td>
<td>Default</td>
<td>0</td>
<td>Byte</td>
<td>Default</td>
<td>0x00</td>
<td></td>
<td>0x0Y04</td>
</tr>
</tbody>
</table>

1) With CS31 and addresses smaller than 70 and FBP, the value is increased by 1

2) Not with FBP

GSD file:

```plaintext
Ext_User_Prm_Data_Len = 39
Ext_User_Prm_Data_Const(0) =
0x05, 0xff, 0x24, \0x01, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00;```

Input channel (8x)

<table>
<thead>
<tr>
<th>No.</th>
<th>Name</th>
<th>Value</th>
<th>Internal value</th>
<th>Internal value, Type</th>
<th>Default</th>
<th>EDS Slot/ Index</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Channel configuration</td>
<td>see Table 560 &quot;Channel configuration&quot; on page 2640</td>
<td>see Table 560 &quot;Channel configuration&quot; on page 2640</td>
<td>Byte</td>
<td>0</td>
<td>0x00</td>
</tr>
<tr>
<td>2</td>
<td>Channel monitoring</td>
<td>see Table 561 &quot;Channel monitoring&quot; on page 2640</td>
<td>see Table 561 &quot;Channel monitoring&quot; on page 2640</td>
<td>Byte</td>
<td>0</td>
<td>0x03</td>
</tr>
<tr>
<td>3</td>
<td>Line frequency suppression</td>
<td>see Further information on page 2641</td>
<td>see Further information on page 2641</td>
<td>Byte</td>
<td>0</td>
<td>0x00</td>
</tr>
<tr>
<td>4</td>
<td>Compensation channel</td>
<td>see Further information on page 2641</td>
<td>see Further information on page 2641</td>
<td>Byte</td>
<td>0</td>
<td>0x00</td>
</tr>
</tbody>
</table>
### Table 560: Channel configuration

<table>
<thead>
<tr>
<th>Internal value</th>
<th>Operating modes for the analog inputs, individually configurable</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Unused (default)</td>
</tr>
<tr>
<td>2</td>
<td>Digital input</td>
</tr>
<tr>
<td>34</td>
<td>Analog input -50 mV...+50 mV</td>
</tr>
<tr>
<td>35</td>
<td>Analog input -500 mV...+500 mV</td>
</tr>
<tr>
<td>36</td>
<td>Analog input -1 V...+1 V</td>
</tr>
<tr>
<td>7</td>
<td>Analog input -5 V...+5 V</td>
</tr>
<tr>
<td>5</td>
<td>Analog input -10 V...+10 V</td>
</tr>
<tr>
<td>6</td>
<td>Analog input 0 V...+5 V</td>
</tr>
<tr>
<td>1</td>
<td>Analog input 0 V...+10 V</td>
</tr>
<tr>
<td>37</td>
<td>Analog input -20 mA...+20 mA</td>
</tr>
<tr>
<td>3</td>
<td>Analog input 0 mA...20 mA</td>
</tr>
<tr>
<td>4</td>
<td>Analog input 4 mA...20 mA</td>
</tr>
<tr>
<td>14</td>
<td>Analog input Pt100 (2-wire), -50 °C...+70 °C</td>
</tr>
<tr>
<td>15</td>
<td>Analog input Pt100 (3-wire), -50 °C...+70 °C</td>
</tr>
<tr>
<td>48</td>
<td>Analog input Pt100 (4-wire), -50 °C...+70 °C</td>
</tr>
<tr>
<td>57</td>
<td>Analog input Pt100 (2-wire), -50 °C...+70 °C (resolution: 0,01 K)</td>
</tr>
<tr>
<td>58</td>
<td>Analog input Pt100 (3-wire), -50 °C...+70 °C (resolution: 0,01 K)</td>
</tr>
<tr>
<td>59</td>
<td>Analog input Pt100 (4-wire), -50 °C...+70 °C (resolution: 0,01 K)</td>
</tr>
<tr>
<td>8</td>
<td>Analog input Pt100 (2-wire), -50 °C...+400 °C</td>
</tr>
<tr>
<td>9</td>
<td>Analog input Pt100 (3-wire), -50 °C...+400 °C</td>
</tr>
<tr>
<td>49</td>
<td>Analog input Pt100 (4-wire), -50 °C...+400 °C</td>
</tr>
<tr>
<td>45</td>
<td>Analog input Pt100 (2-wire), -200 °C...+850 °C</td>
</tr>
<tr>
<td>46</td>
<td>Analog input Pt100 (3-wire), -200 °C...+850 °C</td>
</tr>
<tr>
<td>47</td>
<td>Analog input Pt100 (4-wire), -200 °C...+850 °C</td>
</tr>
<tr>
<td>16</td>
<td>Analog input Pt1000 (2-wire), -50 °C...+400 °C</td>
</tr>
<tr>
<td>17</td>
<td>Analog input Pt1000 (3-wire), -50 °C...+400 °C</td>
</tr>
<tr>
<td>50</td>
<td>Analog input Pt1000 (4-wire), -50 °C...+400 °C</td>
</tr>
<tr>
<td>18</td>
<td>Analog input Ni1000 (2-wire), -50 °C...+150 °C</td>
</tr>
<tr>
<td>19</td>
<td>Analog input Ni1000 (3-wire), -50 °C...+150 °C</td>
</tr>
<tr>
<td>51</td>
<td>Analog input Ni1000 (4-wire), -50 °C...+150 °C</td>
</tr>
<tr>
<td>39</td>
<td>Analog input Cu50 1.426 (2-wire) -50 °C...+200 °C</td>
</tr>
<tr>
<td>40</td>
<td>Analog input Cu50 1.426 (3-wire) -50 °C...+200 °C</td>
</tr>
<tr>
<td>41</td>
<td>Analog input Cu50 1.426 (4-wire) -50 °C...+200 °C</td>
</tr>
<tr>
<td>42</td>
<td>Analog input Cu50 1.428 (2-wire) -200 °C...+200 °C</td>
</tr>
<tr>
<td>43</td>
<td>Analog input Cu50 1.428 (3-wire) -200 °C...+200 °C</td>
</tr>
<tr>
<td>44</td>
<td>Analog input Cu50 1.428 (4-wire) -200 °C...+200 °C</td>
</tr>
<tr>
<td>24</td>
<td>Analog input J-type thermocouple -210 °C...+1200 °C</td>
</tr>
<tr>
<td>25</td>
<td>Analog input K-type thermocouple -270 °C...+1372 °C</td>
</tr>
<tr>
<td>30</td>
<td>Analog input N-type thermocouple -270 °C...+1300 °C</td>
</tr>
</tbody>
</table>
Internal value | Operating modes for the analog inputs, individually configurable
---|---
27 | Analog input S-type thermocouple -50 °C...+1768 °C
28 | Analog input T-type thermocouple -270 °C...+400 °C
38 | Analog input resistor 50 kΩ
52 | Temperature-internal reference point
53 | Common mode voltage

Table 561: Channel monitoring

<table>
<thead>
<tr>
<th>Internal value</th>
<th>Monitoring</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Plausibility, open-circuit (cut wire) and short circuit (default)</td>
</tr>
<tr>
<td>3</td>
<td>No monitoring</td>
</tr>
</tbody>
</table>

Table 562: Line frequency suppression

<table>
<thead>
<tr>
<th>Internal value</th>
<th>Line frequency suppression</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>50 Hz</td>
</tr>
<tr>
<td>1</td>
<td>60 Hz</td>
</tr>
<tr>
<td>2</td>
<td>No line frequency suppression</td>
</tr>
</tbody>
</table>

Table 563: Compensation channel

<table>
<thead>
<tr>
<th>Internal value</th>
<th>Compensation channel</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Internal compensation (default)</td>
</tr>
<tr>
<td>1</td>
<td>Channel 0 (possible with channels 1, 2, 3)</td>
</tr>
<tr>
<td>2</td>
<td>Channel 1 (possible with channels 0, 2, 3)</td>
</tr>
<tr>
<td>3</td>
<td>Channel 2 (possible with channels 0, 1, 3)</td>
</tr>
<tr>
<td>4</td>
<td>Channel 3 (possible with channels 0, 1, 2)</td>
</tr>
<tr>
<td>5</td>
<td>Channel 4 (possible with channels 5, 6, 7)</td>
</tr>
<tr>
<td>6</td>
<td>Channel 5 (possible with channels 4, 6, 7)</td>
</tr>
<tr>
<td>7</td>
<td>Channel 6 (possible with channels 4, 5, 7)</td>
</tr>
<tr>
<td>8</td>
<td>Channel 7 (possible with channels 4, 5, 6)</td>
</tr>
<tr>
<td>9</td>
<td>External with temperature value</td>
</tr>
</tbody>
</table>
## Diagnosis

<table>
<thead>
<tr>
<th>E1...E4</th>
<th>d1</th>
<th>d2</th>
<th>d3</th>
<th>d4</th>
<th>Identifier 000...063</th>
<th>AC500 display</th>
<th>← Display in</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class</td>
<td>Comp</td>
<td>Dev</td>
<td>Mod</td>
<td>Ch</td>
<td>Err</td>
<td>PS501 PLC browser</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Byte 6 Bit 6...7</th>
<th>Byte 3</th>
<th>Byte 4</th>
<th>Byte 5</th>
<th>Byte 6 Bit 0...5</th>
<th>FBP diagnosis block</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class</td>
<td>Interface</td>
<td>Device</td>
<td>Module</td>
<td>Channel</td>
<td>Error identifier</td>
</tr>
<tr>
<td>1)</td>
<td>2)</td>
<td>3)</td>
<td>4)</td>
<td></td>
<td>1</td>
</tr>
</tbody>
</table>

### Module error

<table>
<thead>
<tr>
<th>3</th>
<th>14</th>
<th>1...10</th>
<th>31</th>
<th>31</th>
<th>19</th>
<th>Checksum error in the I/O module</th>
<th>Replace I/O module</th>
</tr>
</thead>
<tbody>
<tr>
<td>11 / 12</td>
<td>ADR</td>
<td>1...10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>14</td>
<td>1...10</td>
<td>31</td>
<td>31</td>
<td>3</td>
<td>Timeout in the I/O module</td>
<td>Replace I/O module</td>
</tr>
<tr>
<td>11 / 12</td>
<td>ADR</td>
<td>1...10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>14</td>
<td>1...10</td>
<td>31</td>
<td>31</td>
<td>40</td>
<td>Different hard-/firmware versions in the module</td>
<td>Replace I/O module</td>
</tr>
<tr>
<td>11 / 12</td>
<td>ADR</td>
<td>1...10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>14</td>
<td>1...10</td>
<td>31</td>
<td>31</td>
<td>43</td>
<td>Internal error in the module, e.g. internal analog voltage is not correct</td>
<td>Replace I/O module</td>
</tr>
<tr>
<td>11 / 12</td>
<td>ADR</td>
<td>1...10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>14</td>
<td>1...10</td>
<td>31</td>
<td>31</td>
<td>36</td>
<td>Internal data exchange failure</td>
<td>Replace I/O module</td>
</tr>
<tr>
<td>11 / 12</td>
<td>ADR</td>
<td>1...10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>14</td>
<td>1...10</td>
<td>31</td>
<td>31</td>
<td>9</td>
<td>Overflow diagnosis buffer</td>
<td>Restart</td>
</tr>
<tr>
<td>11 / 12</td>
<td>ADR</td>
<td>1...10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>14</td>
<td>1...10</td>
<td>31</td>
<td>31</td>
<td>26</td>
<td>Parameter error</td>
<td>Check master</td>
</tr>
<tr>
<td>11 / 12</td>
<td>ADR</td>
<td>1...10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>14</td>
<td>1...10</td>
<td>31</td>
<td>31</td>
<td>11</td>
<td>Process voltage too low</td>
<td>Check process voltage</td>
</tr>
<tr>
<td>11 / 12</td>
<td>ADR</td>
<td>1...10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>14</td>
<td>1...10</td>
<td>31</td>
<td>31</td>
<td>45</td>
<td>Process voltage is switched OFF (ON −&gt; OFF)</td>
<td>Process voltage ON</td>
</tr>
<tr>
<td>11 / 12</td>
<td>ADR</td>
<td>1...10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Channel error

<table>
<thead>
<tr>
<th>4</th>
<th>14</th>
<th>1...10</th>
<th>1</th>
<th>0...7</th>
<th>48</th>
<th>Analog value overflow or broken wire at an analog input</th>
<th>Check input value or terminal</th>
</tr>
</thead>
<tbody>
<tr>
<td>11 / 12</td>
<td>ADR</td>
<td>1...10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>14</td>
<td>1...10</td>
<td>1</td>
<td>0...7</td>
<td>7</td>
<td>Analog value underflow at an analog input</td>
<td>Check input value</td>
</tr>
<tr>
<td>11 / 12</td>
<td>ADR</td>
<td>1...10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>14</td>
<td>1...10</td>
<td>1</td>
<td>0...7</td>
<td>47</td>
<td>Short circuit at an analog input</td>
<td>Check terminal</td>
</tr>
<tr>
<td>11 / 12</td>
<td>ADR</td>
<td>1...10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>14</td>
<td>1...10</td>
<td>1</td>
<td>0...7</td>
<td>1</td>
<td>Possibly wrong measured value caused by inadmissible temperature of the compensation channel</td>
<td>Check the temperature compensation channel</td>
</tr>
<tr>
<td>11 / 12</td>
<td>ADR</td>
<td>1...10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>E1...E4</td>
<td>d1</td>
<td>d2</td>
<td>d3</td>
<td>d4</td>
<td>Identifier 000...063</td>
<td>AC500 display</td>
<td>&lt;- Display in</td>
</tr>
<tr>
<td>--------</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>---------------------</td>
<td>---------------</td>
<td>--------------</td>
</tr>
<tr>
<td>Class</td>
<td>Comp</td>
<td>Dev</td>
<td>Mod</td>
<td>Ch</td>
<td>Err</td>
<td>PS501 PLC browser</td>
<td></td>
</tr>
<tr>
<td>Byte 6</td>
<td>-</td>
<td>Byte 3</td>
<td>Byte 4</td>
<td>Byte 5</td>
<td>Byte 6 Bit 0...5</td>
<td>FBP diagnosis block</td>
<td></td>
</tr>
<tr>
<td>Bit 6...7</td>
<td>4 14 1...10</td>
<td>0...7</td>
<td>2</td>
<td>Invalid measured value of the channel caused by overly high voltage difference</td>
<td>Check voltage difference; install equalizing conductors if necessary</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Class</td>
<td>Interface</td>
<td>Device</td>
<td>Module</td>
<td>Channel</td>
<td>Error identifier</td>
<td>Error message</td>
<td>Remedy</td>
</tr>
<tr>
<td>4</td>
<td>14</td>
<td>1...10</td>
<td>1</td>
<td>0...7</td>
<td>2</td>
<td>Invalid measured value of the channel caused by overly high voltage difference</td>
<td>Check voltage difference; install equalizing conductors if necessary</td>
</tr>
<tr>
<td></td>
<td>11 / 12</td>
<td>ADR</td>
<td>1...10</td>
<td>11</td>
<td>Output voltage 10 V faulty</td>
<td>Check output load</td>
<td></td>
</tr>
</tbody>
</table>

Remarks:

1) In AC500, the following interface identifier applies:
   14 = I/O bus, 11 = COM1 (e.g. CS31 bus), 12 = COM2.
   The FBP diagnosis block does not contain this identifier.

2) With "Device" the following allocation applies:
   31 = module itself, 1...10 expansion module 1...10, ADR = hardware address (e.g. of the DC551)

3) With "Module" the following allocation applies dependent of the master:
   Module error: I/O bus or FBP: 31 = module itself; COM1/COM2: 1...10 = expansion 1...10
   Channel error: I/O bus or FBP = module type (1 = AI); COM1/COM2: 1...10 = expansion 1...10

4) In case of module errors, with channel "31 = Module itself" is output.

State LEDs

During the power ON procedure, the module initializes automatically. All LEDs (except the channel LEDs) are ON during this time.

States of the LEDs (see also section Diagnosis LEDs in the S500 system data):
### LED States

<table>
<thead>
<tr>
<th>LED</th>
<th>State</th>
<th>Color</th>
<th>LED = OFF</th>
<th>LED = ON</th>
<th>LED flashes</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Inputs I0...I3 and I4...I7</td>
<td>Analog input</td>
<td>Yellow</td>
<td>Input is OFF</td>
<td>Input is ON (brightness depends on the value of the analog signal)</td>
</tr>
<tr>
<td></td>
<td>UP</td>
<td>Process voltage 24 VDC via terminal</td>
<td>Green</td>
<td>Process voltage is missing</td>
<td>Process voltage OK</td>
</tr>
<tr>
<td></td>
<td>CH-ERR2</td>
<td>Channel error, messages in groups (analog inputs combined into the groups 2 and 4)</td>
<td>Red</td>
<td>No error, or process voltage is missing</td>
<td>Severe error within the corresponding group</td>
</tr>
<tr>
<td></td>
<td>CH-ERR4</td>
<td>Module error</td>
<td>Red</td>
<td>--</td>
<td>Internal error</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Measuring Ranges

**Voltage Input Ranges**

**Bipolar Voltage Input Range, Measuring Bridge**

<table>
<thead>
<tr>
<th>Range</th>
<th>-50 ... +50 mV</th>
<th>-500 ... +500 mV</th>
<th>-1 ... +1 V</th>
<th>-5 ... +5 V</th>
<th>-10 ... +10 V</th>
<th>Common Mode Voltage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Overflow</td>
<td>&gt; 58.7945</td>
<td>&gt; 587.9449</td>
<td>&gt; 1.17589</td>
<td>&gt; 5.8794</td>
<td>&gt; 11.7589</td>
<td>&gt; 20.0000</td>
</tr>
<tr>
<td>Measured value too high</td>
<td>58.7945 : 50.0018</td>
<td>587.9449 : 500.0181</td>
<td>1.17589 : 1.00004</td>
<td>5.8794 : 5.0002</td>
<td>11.7589 : 10.0004</td>
<td></td>
</tr>
<tr>
<td>Normal range</td>
<td>50.0000 : 0.0018</td>
<td>500.0000 : 0.0004</td>
<td>1.0000 : 0.0002</td>
<td>5.0000 : 0.004</td>
<td>10.0000 : 0.0004</td>
<td>20.0000 : 0.0008</td>
</tr>
<tr>
<td>Normal range or Measured value too low</td>
<td>0.0000 : 0.0000</td>
<td>0.0000 : 0.0000</td>
<td>0.0000 : 0.0000</td>
<td>0.0000 : 0.0000</td>
<td>0.0000 : 0.0000</td>
<td>0.0000 : 0.0000</td>
</tr>
<tr>
<td>Underflow</td>
<td>&lt; -58.7945</td>
<td>&lt; -587.9449</td>
<td>&lt; -1.17589</td>
<td>&lt; -5.8794</td>
<td>&lt; -11.7589</td>
<td>&lt; -20.0000</td>
</tr>
</tbody>
</table>

The represented resolution corresponds to 16 bits.
## Range

<table>
<thead>
<tr>
<th>Range</th>
<th>Decimal</th>
<th>Hex.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Overflow</strong></td>
<td>32767</td>
<td>7FFF</td>
</tr>
<tr>
<td>Measured value too high</td>
<td>32511</td>
<td>7EFF</td>
</tr>
<tr>
<td>:</td>
<td>27649</td>
<td>6C01</td>
</tr>
<tr>
<td>Normal range</td>
<td>27648</td>
<td>6C00</td>
</tr>
<tr>
<td>Normal range or Measured value too low</td>
<td>0</td>
<td>0000</td>
</tr>
<tr>
<td>:</td>
<td>-1</td>
<td>FFFF</td>
</tr>
<tr>
<td>:</td>
<td>-27648</td>
<td>9400</td>
</tr>
<tr>
<td>Measured value too low</td>
<td>-27649</td>
<td>93FF</td>
</tr>
<tr>
<td>:</td>
<td>-32512</td>
<td>8100</td>
</tr>
<tr>
<td>Underflow</td>
<td>-32768</td>
<td>8000</td>
</tr>
</tbody>
</table>

## Unipolar voltage input range, measuring bridge, digital input

<table>
<thead>
<tr>
<th>Range</th>
<th>0 ... +5 V</th>
<th>0 ... +10 V</th>
<th>Digital input</th>
</tr>
</thead>
<tbody>
<tr>
<td>Measured value too high</td>
<td>5.8794</td>
<td>11.7589</td>
<td></td>
</tr>
<tr>
<td>:</td>
<td>5.0002</td>
<td>10.0004</td>
<td></td>
</tr>
<tr>
<td>Normal range</td>
<td>5.0000</td>
<td>10.0000</td>
<td>ON</td>
</tr>
<tr>
<td>:</td>
<td>0.0002</td>
<td>0.0004</td>
<td></td>
</tr>
<tr>
<td>:</td>
<td>0.0000</td>
<td>0.0000</td>
<td>OFF</td>
</tr>
<tr>
<td>Measured value too low</td>
<td>-0.0002</td>
<td>-0.0004</td>
<td></td>
</tr>
<tr>
<td>:</td>
<td>-0.8794</td>
<td>-1.1759</td>
<td></td>
</tr>
<tr>
<td>Underflow</td>
<td>&lt; -0.8794</td>
<td>&lt; -1.1759</td>
<td></td>
</tr>
</tbody>
</table>

## Range

<table>
<thead>
<tr>
<th>Range</th>
<th>Decimal</th>
<th>Hex.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Overflow</strong></td>
<td>32767</td>
<td>7FFF</td>
</tr>
<tr>
<td>Measured value too high</td>
<td>32511</td>
<td>7EFF</td>
</tr>
<tr>
<td>:</td>
<td>27649</td>
<td>6C01</td>
</tr>
</tbody>
</table>
### Range

<table>
<thead>
<tr>
<th>Range</th>
<th>Digital value</th>
<th>Hex.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Decimal</strong></td>
<td><strong>Hex.</strong></td>
<td></td>
</tr>
<tr>
<td>Normal range</td>
<td>27648</td>
<td>6C00</td>
</tr>
<tr>
<td>:</td>
<td>:</td>
<td>:</td>
</tr>
<tr>
<td>:</td>
<td>1</td>
<td>0001</td>
</tr>
<tr>
<td>:</td>
<td>0</td>
<td>0000</td>
</tr>
<tr>
<td>Measured value too low</td>
<td>-1</td>
<td>FFFF</td>
</tr>
<tr>
<td>:</td>
<td>:</td>
<td>:</td>
</tr>
<tr>
<td>:</td>
<td>-4864</td>
<td>ED00</td>
</tr>
<tr>
<td>Underflow</td>
<td>-32768</td>
<td>8000</td>
</tr>
</tbody>
</table>

### Current input ranges

<table>
<thead>
<tr>
<th>Range</th>
<th>-20 ... +20 mA</th>
<th>0 ... +20 mA</th>
<th>4 ... 20 mA</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Overflow</strong></td>
<td>&gt; 23.5178</td>
<td>&gt; 23.5178</td>
<td>&gt; 22.8142</td>
</tr>
<tr>
<td>Measured value too high</td>
<td>23.5178</td>
<td>23.5178</td>
<td>22.8142</td>
</tr>
<tr>
<td>:</td>
<td>:</td>
<td>20.0007</td>
<td>20.0006</td>
</tr>
<tr>
<td>Normal range</td>
<td>20.0000</td>
<td>20.0000</td>
<td>20.0000</td>
</tr>
<tr>
<td>:</td>
<td>:</td>
<td>0.0007</td>
<td>4.0006</td>
</tr>
<tr>
<td>:</td>
<td>0.0000</td>
<td>0.0000</td>
<td>4.0000</td>
</tr>
<tr>
<td>:</td>
<td>-0.0007</td>
<td></td>
<td></td>
</tr>
<tr>
<td>:</td>
<td>-20.0000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Measured value too low</td>
<td>-0.0007</td>
<td>-3.5178</td>
<td>3.9994</td>
</tr>
<tr>
<td>:</td>
<td>:</td>
<td></td>
<td>1.1852</td>
</tr>
<tr>
<td>:</td>
<td>-20.0007</td>
<td>-3.5178</td>
<td></td>
</tr>
<tr>
<td>:</td>
<td>-23.5178</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Underflow</td>
<td>&lt; -23.5178</td>
<td>&lt; -3.5178</td>
<td>&lt; 1.1852</td>
</tr>
</tbody>
</table>

### Range

<table>
<thead>
<tr>
<th>Range</th>
<th>Digital value</th>
<th>Hex.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Decimal</strong></td>
<td><strong>Hex.</strong></td>
<td></td>
</tr>
<tr>
<td>Overflow</td>
<td>32767</td>
<td>7FFF</td>
</tr>
<tr>
<td>Measured value too high</td>
<td>32511</td>
<td>7EFF</td>
</tr>
<tr>
<td>:</td>
<td>:</td>
<td>:</td>
</tr>
<tr>
<td>:</td>
<td>27649</td>
<td>6C01</td>
</tr>
<tr>
<td>Range</td>
<td>Digital value</td>
<td>Hex.</td>
</tr>
<tr>
<td>---------------------------</td>
<td>---------------</td>
<td>------</td>
</tr>
<tr>
<td>Normal range</td>
<td>27648</td>
<td>6C00</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>0001</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>0000</td>
</tr>
<tr>
<td></td>
<td>-1</td>
<td>FFFF</td>
</tr>
<tr>
<td></td>
<td>-27648</td>
<td>9400</td>
</tr>
<tr>
<td>Measured value too low</td>
<td>-1</td>
<td>FFFF</td>
</tr>
<tr>
<td></td>
<td>-4864</td>
<td>ED00</td>
</tr>
<tr>
<td></td>
<td>-27649</td>
<td>93FF</td>
</tr>
<tr>
<td></td>
<td>-32512</td>
<td>8100</td>
</tr>
<tr>
<td>Underflow</td>
<td>-32768</td>
<td>8000</td>
</tr>
</tbody>
</table>

**Resistance thermometer input ranges**

<table>
<thead>
<tr>
<th>Range</th>
<th>Pt100 -50 ... +70 °C</th>
<th>Pt100 / Pt1000 -50 ... +400 °C</th>
<th>Pt100 -200 ... +850 °C</th>
<th>Ni1000 -50 ... +150 °C</th>
<th>Cu50 -200 ... +200 °C</th>
</tr>
</thead>
<tbody>
<tr>
<td>Overflow</td>
<td>&gt; 80.0 °C</td>
<td>&gt; 450.0 °C</td>
<td>&gt; 850 °C</td>
<td>&gt; 160.0 °C</td>
<td>&gt; 200 °C</td>
</tr>
<tr>
<td>Measured value too high</td>
<td></td>
<td>450.0 °C</td>
<td>400.1 °C</td>
<td>160.0 °C</td>
<td>150.1 °C</td>
</tr>
<tr>
<td></td>
<td>80.0 °C</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>70.1 °C</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Normal range</td>
<td>:</td>
<td>:</td>
<td>:</td>
<td>:</td>
<td>:</td>
</tr>
<tr>
<td></td>
<td>:</td>
<td>400.0 °C</td>
<td>:</td>
<td>:</td>
<td>: 200.0 °C</td>
</tr>
<tr>
<td></td>
<td>:</td>
<td>:</td>
<td>:</td>
<td>:</td>
<td>: 150.0 °C</td>
</tr>
<tr>
<td></td>
<td>:</td>
<td>70.0 °C</td>
<td>:</td>
<td>:</td>
<td>:</td>
</tr>
<tr>
<td></td>
<td>:</td>
<td>0.1 °C</td>
<td>:</td>
<td>:</td>
<td>:</td>
</tr>
<tr>
<td></td>
<td>:</td>
<td>0.0 °C</td>
<td>:</td>
<td>:</td>
<td>:</td>
</tr>
<tr>
<td>Range</td>
<td>Pt100 / Pt1000</td>
<td>Pt100 / Pt1000</td>
<td>Pt100 / Pt1000</td>
<td>Ni1000</td>
<td>Cu50</td>
</tr>
<tr>
<td>-----------</td>
<td>----------------</td>
<td>----------------</td>
<td>----------------</td>
<td>--------</td>
<td>------</td>
</tr>
<tr>
<td></td>
<td>-50 °C ... +70 °C</td>
<td>-50 °C ... +400 °C</td>
<td>-50 °C ... +850 °C</td>
<td>-50 °C ... +150 °C</td>
<td>-200 °C ... +200 °C</td>
</tr>
<tr>
<td>-0.1 °C</td>
<td>-0.1 °C</td>
<td>-0.1 °C</td>
<td>-0.1 °C</td>
<td>-0.1 °C</td>
<td>-0.1 °C</td>
</tr>
<tr>
<td>-50.0 °C</td>
<td>:</td>
<td>:</td>
<td>:</td>
<td>:</td>
<td>:</td>
</tr>
<tr>
<td>-50.1 °C</td>
<td>-50.1 °C</td>
<td>-50.1 °C</td>
<td>-50.1 °C</td>
<td>-50.1 °C</td>
<td>-50.1 °C</td>
</tr>
<tr>
<td>-60.0 °C</td>
<td>:</td>
<td>:</td>
<td>:</td>
<td>:</td>
<td>:</td>
</tr>
<tr>
<td>Measured value too low</td>
<td>-50.1 °C</td>
<td>-50.1 °C</td>
<td>-50.1 °C</td>
<td>-50.1 °C</td>
<td>-50.1 °C</td>
</tr>
<tr>
<td>&lt; -60.0 °C</td>
<td>:</td>
<td>:</td>
<td>:</td>
<td>:</td>
<td>:</td>
</tr>
<tr>
<td>Underflow</td>
<td>&lt; -60.0 °C</td>
<td>&lt; -60.0 °C</td>
<td>&lt; -200 °C</td>
<td>&lt; -60.0 °C</td>
<td>&lt; -200 °C</td>
</tr>
<tr>
<td></td>
<td>&lt; -60.0 °C</td>
<td>&lt; -60.0 °C</td>
<td>&lt; -200 °C</td>
<td>&lt; -60.0 °C</td>
<td>&lt; -200 °C</td>
</tr>
</tbody>
</table>

1) also possible with resolution 0.01 K
2) if Cu50 with 1.426, -50 °C is valid; if Cu50 with 1.428, -200.0 °C is valid

The represented resolution corresponds to 16 bits.

<table>
<thead>
<tr>
<th>Range</th>
<th>Digital value</th>
<th>Hex.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Overflow</td>
<td>32767</td>
<td>7FFF</td>
</tr>
<tr>
<td>Measured value too high</td>
<td>4500</td>
<td>1194</td>
</tr>
<tr>
<td></td>
<td>4001</td>
<td>0FA1</td>
</tr>
<tr>
<td></td>
<td>1600</td>
<td>0640</td>
</tr>
<tr>
<td></td>
<td>1501</td>
<td>05DD</td>
</tr>
<tr>
<td></td>
<td>800</td>
<td>0320</td>
</tr>
<tr>
<td></td>
<td>701</td>
<td>02BD</td>
</tr>
<tr>
<td>Normal range</td>
<td>8500</td>
<td>2134</td>
</tr>
<tr>
<td></td>
<td>4000</td>
<td>0FA0</td>
</tr>
<tr>
<td></td>
<td>2000</td>
<td>07D0</td>
</tr>
<tr>
<td></td>
<td>1500</td>
<td>05DC</td>
</tr>
<tr>
<td></td>
<td>700</td>
<td>02BC</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>0000</td>
</tr>
<tr>
<td></td>
<td>-1</td>
<td>FFFF</td>
</tr>
<tr>
<td></td>
<td>-500</td>
<td>FE0C</td>
</tr>
<tr>
<td></td>
<td>-2000</td>
<td>F830</td>
</tr>
</tbody>
</table>
### Range

<table>
<thead>
<tr>
<th>Range</th>
<th>Decal</th>
<th>Hex.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Measured value too low</td>
<td>-501</td>
<td>FE0B</td>
</tr>
<tr>
<td></td>
<td>-600</td>
<td>FDA8</td>
</tr>
<tr>
<td>Underflow</td>
<td>-32768</td>
<td>8000</td>
</tr>
</tbody>
</table>

### Resistor input range

<table>
<thead>
<tr>
<th>Range</th>
<th>Resistor [Ω]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Overflow</td>
<td>&gt; 55000</td>
</tr>
<tr>
<td>Measured value too high</td>
<td>55000</td>
</tr>
<tr>
<td></td>
<td>:</td>
</tr>
<tr>
<td></td>
<td>50001</td>
</tr>
<tr>
<td>Normal range</td>
<td>50000</td>
</tr>
<tr>
<td></td>
<td>:</td>
</tr>
<tr>
<td></td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>0</td>
</tr>
</tbody>
</table>

The represented resolution corresponds to 16 bits.

### Thermocouple input ranges

<table>
<thead>
<tr>
<th>Range</th>
<th>Typ J -210 ... +1200 °C</th>
<th>Typ K -270 ... +1372 °C</th>
<th>Typ N -270 ... +1300 °C</th>
<th>Typ S -50 ... +1768 °C</th>
<th>Typ T -270 ... +400 °C</th>
</tr>
</thead>
<tbody>
<tr>
<td>Overflow</td>
<td>&gt; 1200.0 °C</td>
<td>&gt; 1372.0 °C</td>
<td>&gt; 1300.0 °C</td>
<td>&gt; 1768.0 °C</td>
<td>&gt; 400.0 °C</td>
</tr>
<tr>
<td>Normal range</td>
<td></td>
<td></td>
<td></td>
<td>1768.0 °C</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1372.0 °C</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Temperature-internal reference point ranges

<table>
<thead>
<tr>
<th>Range</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Overflow</td>
<td>&gt; +85 °C</td>
</tr>
<tr>
<td>Normal range</td>
<td>+85 °C</td>
</tr>
<tr>
<td></td>
<td>0 °C</td>
</tr>
</tbody>
</table>
### Technical data

The system data of AC500 and S500 (Chapter 1.5.4.5.1 “System data AC500” on page 3058) are valid for standard version.

The system data of AC500-XC (Chapter 1.5.4.6.1 “System data AC500-XC” on page 3099) are valid for the XC version.

Only additional details are therefore documented below.

The technical data are also valid for the XC version.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Process voltage</td>
<td></td>
</tr>
<tr>
<td>Connections</td>
<td>Terminals 1.8, 2.8, 3.8 and 4.8 for +24 V (UP) as well as 1.9, 2.9, 3.9 and 4.9 for 0 V (ZP)</td>
</tr>
<tr>
<td>Rated value</td>
<td>24 VDC</td>
</tr>
<tr>
<td>Max. ripple</td>
<td>5 %</td>
</tr>
<tr>
<td>Protection against reversed voltage</td>
<td>Yes</td>
</tr>
<tr>
<td>Rated protection fuse on UP</td>
<td>10 A fast</td>
</tr>
<tr>
<td>Galvanic isolation</td>
<td>Yes, per module</td>
</tr>
<tr>
<td>Current consumption</td>
<td></td>
</tr>
<tr>
<td>From 24 VDC power supply at the terminals UP/L+ and ZP/M of the CPU/bus module</td>
<td>Ca. 2 mA</td>
</tr>
<tr>
<td>Current consumption from UP in normal operation</td>
<td>130 mA</td>
</tr>
<tr>
<td>Inrush current from UP (at power up)</td>
<td>On request</td>
</tr>
<tr>
<td>Max. length of analog cables, conductor cross section &gt; 0.14 mm²</td>
<td>100 m</td>
</tr>
<tr>
<td>Weight</td>
<td>130 g</td>
</tr>
</tbody>
</table>

---

<table>
<thead>
<tr>
<th>Range</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>-40 °C</td>
<td></td>
</tr>
<tr>
<td>Underflow</td>
<td>&lt; -40 °C</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Range</th>
<th>Digital value</th>
<th>Hex.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Overflow</td>
<td>32767</td>
<td>7FFF</td>
</tr>
<tr>
<td>Normal range</td>
<td>850</td>
<td>0352</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>0000</td>
</tr>
<tr>
<td></td>
<td>-400</td>
<td>FE70</td>
</tr>
<tr>
<td>Underflow</td>
<td>-32768</td>
<td>8000</td>
</tr>
</tbody>
</table>
Parameter | Value
---|---
Mounting position | Horizontal or vertical with derating (max. temperature 40 °C)
Cooling | The natural convection cooling must not be hindered by cable ducts or other parts in the switch-gear cabinet.

**NOTICE!**
**Attention:**
All I/O channels (digital and analog) are protected against reverse polarity, reverse supply, short circuit and continuous overvoltage up to 30 VDC.

### Technical data of the analog inputs

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of channels per module</td>
<td>8</td>
</tr>
<tr>
<td>Distribution of channels into groups</td>
<td>2 groups of 4 channels each</td>
</tr>
<tr>
<td>Connections of the channels I0 to I3</td>
<td>Terminals 1.0 to 1.7 and terminals 2.0 to 2.7</td>
</tr>
<tr>
<td>Connections of the channels I4 to I7</td>
<td>Terminals 3.0 to 3.7 and terminals 4.0 to 4.7</td>
</tr>
<tr>
<td>Input type</td>
<td>Bipolar (not with current or Pt100/ Pt1000/ Ni1000/ Cu50/ resistor)</td>
</tr>
<tr>
<td>Galvanic isolation</td>
<td>Against internal supply and other modules</td>
</tr>
<tr>
<td>Configurability</td>
<td>Digital input, -50 mV...+50 mV, -500 mV...+500 mV, -1 V...+1 V, -5 V...+5 V, -10 V...+10 V, 0 V...+5 V, 0 V...+10 V, -20 mA...+20 mA, 0 mA...20 mA, 4 mA...20 mA, Pt100, Pt1000, Ni1000, Cu50, resistor, thermocouple types J, K, N, S, T (each input can be configured individually)</td>
</tr>
<tr>
<td>Channel input resistance</td>
<td>Voltage: &gt; 100 kΩ, current: ca. 330 Ω</td>
</tr>
<tr>
<td>Time constant of the input filter</td>
<td>Line-frequency suppression 50 Hz, 60 Hz, none</td>
</tr>
<tr>
<td>Indication of the input signals</td>
<td>1 yellow LED per channel, the brightness depends on the value of the analog signal</td>
</tr>
<tr>
<td>Conversion time</td>
<td>1 ms (none), 100 ms (50 Hz / 60 Hz) per channel</td>
</tr>
<tr>
<td>Resolution</td>
<td>Range</td>
</tr>
<tr>
<td></td>
<td>unipolar</td>
</tr>
<tr>
<td></td>
<td>bipolar</td>
</tr>
<tr>
<td>Conversion error of the analog values caused by non-linearity, adjustment error at factory and resolution within the normal range</td>
<td>Typ.</td>
</tr>
<tr>
<td></td>
<td>±0.1 % (voltage)</td>
</tr>
<tr>
<td></td>
<td>±0.3 % (current, resistor) at 25 °C</td>
</tr>
</tbody>
</table>
### Technical data of the analog inputs if used as digital inputs

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of channels per module</td>
<td>Max. 8</td>
</tr>
<tr>
<td>Distribution of channels into groups</td>
<td>2 groups of 4 channels each</td>
</tr>
<tr>
<td>Connections of the channels I0+ to I3+</td>
<td>Terminals 2.0, 2.2, 2.4, 2.6</td>
</tr>
<tr>
<td>Connections of the channels I4+ to I7+</td>
<td>Terminals 4.0, 4.2, 4.4, 4.6</td>
</tr>
<tr>
<td>Reference potential for the inputs</td>
<td>Terminals 1.9, 2.9, 3.9 and 4.9 (ZP)</td>
</tr>
<tr>
<td>Input delay</td>
<td>Typ. 2 ms</td>
</tr>
<tr>
<td>Indication of the input signals</td>
<td>1 LED per channel</td>
</tr>
<tr>
<td>Input signal voltage</td>
<td>24 VDC</td>
</tr>
<tr>
<td>Signal 0</td>
<td>-30 V...+5 V</td>
</tr>
<tr>
<td>Undefined signal</td>
<td>+5 V...+13 V</td>
</tr>
<tr>
<td>Signal 1</td>
<td>+13 V...+30 V</td>
</tr>
<tr>
<td>Input current per channel</td>
<td></td>
</tr>
<tr>
<td>Input voltage +24 V</td>
<td>Typ. 5 mA</td>
</tr>
</tbody>
</table>

### Maximum permanent allowed overload (no damage)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Current input</td>
<td>When the input current exceeds the overflow value of the measurement range, the input impedance is switched to high impedance for protection. The maximum allowed overload is then 30 V. The digital value corresponds to the overflow value. Periodically, the input impedance is switched to the normal value and the input current is measured. If the input current is within the measurement range, the input impedance remains at the normal level and the digital value corresponds to the measured current.</td>
</tr>
<tr>
<td>Voltage input</td>
<td>30 V</td>
</tr>
</tbody>
</table>

### Table 561 “Channel monitoring” on page 2641

<table>
<thead>
<tr>
<th>Relationship between input signal and hex code</th>
<th>% Table 561 “Channel monitoring” on page 2641</th>
<th>Unused voltage inputs</th>
<th>Are configured as &quot;unused&quot;</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unused current inputs</td>
<td>Have a low resistance, can be left open-circuited</td>
<td>Overvoltage protection</td>
<td>Yes</td>
</tr>
</tbody>
</table>
### Input voltage

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input voltage +5 V</td>
<td>Typ. 1 mA</td>
</tr>
<tr>
<td>Input voltage +15 V</td>
<td>Typ. 3.1 mA</td>
</tr>
<tr>
<td>Input voltage +30 V</td>
<td>&lt; 7 mA</td>
</tr>
</tbody>
</table>

| Input resistance   | Ca. 4.8 kΩ |

### Ordering data

<table>
<thead>
<tr>
<th>Part no.</th>
<th>Description</th>
<th>Product life cycle phase *)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1SAP 250 600 R0001</td>
<td>AI531, analog input module, 8 AI, U/I/Pt100, TC, 15 bits + sign, 4-wires</td>
<td>Active</td>
</tr>
<tr>
<td>1SAP 450 600 R0001</td>
<td>AI531-XC, analog input module, 8 AI, U/I/Pt100, TC, 15 bits + sign, 4-wires, XC version</td>
<td>Active</td>
</tr>
</tbody>
</table>

*) For planning and commissioning of new installations use modules in Active status only.

### AO523 - Analog output module

- 16 analog outputs in two groups:
  - 8 channels configurable for voltage or current output
  - 8 channels for voltage output
  - Resolution 12 bits plus sign

- Module-wise electrically isolated
- XC version for use in extreme ambient conditions available
1 I/O bus
2 Allocation between terminal number and signal name
3 16 yellow LEDs to display the signal states at the analog outputs (O0 - O15)
4 1 green LED to display the state of the process supply voltage UP
5 2 red LEDs to display errors
6 Label
7 Terminal unit
8 DIN rail
9 Sign for XC version

**Intended purpose**

The device can be used as a decentralized I/O extension module for S500 Communication Interface Modules (e.g., CI592-CS31, CI501-PNIO, CI541-DP, CI581-CN) or as a centralized extension module for AC500 CPUs (PM5xx).

**Functionality**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Resolution of the analog channels</td>
<td>12 bits plus sign</td>
</tr>
<tr>
<td>Voltage -10 V...+10 V</td>
<td>12 bits plus sign</td>
</tr>
<tr>
<td>Current 0 mA...20 mA, 4 mA...20 mA</td>
<td>12 bits</td>
</tr>
</tbody>
</table>
### Parameter Table

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>LED displays</td>
<td>19 LEDs for signals and error messages</td>
</tr>
<tr>
<td>Internal power supply</td>
<td>Through the expansion bus interface (I/O bus)</td>
</tr>
<tr>
<td>External power supply</td>
<td>Via the terminals ZP and UP (process voltage 24 VDC)</td>
</tr>
<tr>
<td>Required terminal unit</td>
<td>TU515 or TU516 *Chapter 1.5.3.5.2 “TU515, TU516, TU541 and TU542 for I/O modules” on page 2311</td>
</tr>
</tbody>
</table>

### Electrical connection

*For a detailed description of the mounting, disassembly and electrical connection of the module, please refer to the System Assembly, Construction and Connection chapter *Chapter 1.5.4.5 “AC500 (Standard)” on page 3058.*

The modules are plugged on an I/O terminal unit *Chapter 1.5.3.5.2 “TU515, TU516, TU541 and TU542 for I/O modules” on page 2311*. Properly position the modules and press until they lock in place. The terminal units are mounted on a DIN rail or with 2 screws plus the additional accessory for wall mounting (TA526 *Chapter 1.5.3.8.1.4 “TA526 - Wall mounting accessory” on page 3035).

The electrical connection of the I/O channels is carried out using the 40 terminals of the I/O terminal unit. I/O modules can be replaced without re-wiring the terminal units.

The terminals 1.8 to 4.8 and 1.9 to 4.9 are electrically interconnected within the I/O terminal units and have always the same assignment, independent of the inserted module:

- Terminals 1.8 to 4.8: process voltage UP = +24 VDC
- Terminals 1.9 to 4.9: process voltage ZP = 0 VDC

The assignment of the other terminals:

<table>
<thead>
<tr>
<th>Terminals</th>
<th>Signal</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0 to 1.7</td>
<td>O0- to O7-</td>
<td>Negative poles of the first 8 analog outputs</td>
</tr>
<tr>
<td>2.0 to 2.7</td>
<td>O0+ to O7+</td>
<td>Positive poles of the first 8 analog outputs</td>
</tr>
<tr>
<td>3.0 to 3.7</td>
<td>O8- to O15-</td>
<td>Negative poles of the following 8 analog outputs</td>
</tr>
<tr>
<td>4.0 to 4.7</td>
<td>O8+ to O15+</td>
<td>Positive poles of the following 8 analog outputs</td>
</tr>
</tbody>
</table>

*For the open-circuit detection (cut wire), each analog input channel is pulled up to “plus” by a high-resistance resistor. If nothing is connected, the maximum voltage will be read in then.*

The internal power supply voltage for the module's circuitry is carried out via the I/O bus (provided by a bus module or a CPU). Thus, the current consumption from 24 VDC power supply at the terminals L+/UP and M/ZP of the CPU/bus module increases by 2 mA per AO523.
The external power supply connection is carried out via the UP (+24 VDC) and the ZP (0 VDC) terminals.

**NOTICE!**

Risk of damaging the PLC modules!

- Overvoltages and short circuits might damage the PLC modules.
- Make sure that all voltage sources (supply voltage and process supply voltage) are switched off before you begin with operations at the system.
- Never connect any voltages or signals to reserved terminals (marked with ---). Reserved terminals may carry internal voltages.

**Info**

Generally, analog signals must be laid in shielded cables. The cable shields must be earthed at both sides of the cables. In order to avoid unacceptable potential differences between different parts of the installation, low resistance equipotential bonding conductors must be laid.

*Only for simple applications (low electromagnetic disturbances, no high requirement on precision), the shielding can also be omitted.*

The following figure shows the electrical connection of the module:

---

Attention:

By installing equipotential bonding conductors between the different parts of the system, it must be made sure that the potential difference between ZP and AGND never can exceed 1 V.

---

The modules provide several diagnosis functions ⇤ Chapter 1.5.3.6.2.2.3.7 “Diagnosis” on page 2663.
Connection of analog output loads (Voltage, current)

![Connection diagram]

**Fig. 143: Connection example**

The following measuring ranges can be configured

<table>
<thead>
<tr>
<th>Voltage</th>
<th>-10 V...+10 V</th>
<th>Load max. ±10 mA</th>
<th>1 channel used</th>
</tr>
</thead>
<tbody>
<tr>
<td>Current</td>
<td>0 mA...20 mA</td>
<td>Load 0 Ω...500 Ω</td>
<td>1 channel used</td>
</tr>
<tr>
<td>Current</td>
<td>4...20 mA</td>
<td>Load 0 Ω...500 Ω</td>
<td>1 channel used</td>
</tr>
</tbody>
</table>

Only the channels 0...3 and 8...11 can be configured as current output (0 mA...20 mA or 4 mA...20 mA).

The function of the LEDs is described under Displays. Unused analog outputs can be left open-circuited.

### Internal data exchange

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Digital inputs</td>
<td>0</td>
</tr>
<tr>
<td>Digital outputs</td>
<td>0</td>
</tr>
<tr>
<td>Counter input</td>
<td>0</td>
</tr>
<tr>
<td>Counter output</td>
<td>16</td>
</tr>
</tbody>
</table>

### I/O configuration

The module does not store configuration data itself. It gets its parameterization data from the master device of the I/O bus (CPU or bus module) during power-up of the system.

That means replacing I/O modules is possible without any re-parameterization via software.

*If the external power supply voltage via UP/ZP terminals fails, the I/O module loses its configuration data. The whole station has to be switched off and on again to re-configure the module.*
The arrangement of the parameter data is performed by Control Builder Plus/ Automation Builder software.

The parameter data directly influences the functionality of modules.

For non-standard applications, it is necessary to adapt the parameters to your system configuration.

Module: Module slot address: Y = 1...10

<table>
<thead>
<tr>
<th>No.</th>
<th>Name</th>
<th>Value</th>
<th>Internal value</th>
<th>Internal value, type</th>
<th>Default</th>
<th>Min.</th>
<th>Max.</th>
<th>EDS Slot/Index</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Module ID</td>
<td>Internal</td>
<td>1510 1)</td>
<td>Word</td>
<td>1510</td>
<td>0</td>
<td>65535</td>
<td>0x0Y01</td>
</tr>
<tr>
<td>2</td>
<td>Ignore module</td>
<td>No</td>
<td>0</td>
<td>Byte</td>
<td>No</td>
<td>0</td>
<td>1</td>
<td>Not for FBP</td>
</tr>
<tr>
<td>3</td>
<td>Parameter length in bytes</td>
<td>Internal</td>
<td>39</td>
<td>Byte</td>
<td>39-CPU 39-FBP</td>
<td>0</td>
<td>255</td>
<td>0x0Y02</td>
</tr>
<tr>
<td>4</td>
<td>Check supply</td>
<td>Off</td>
<td>0</td>
<td>Byte</td>
<td>On</td>
<td>0</td>
<td>1</td>
<td>0x0Y03</td>
</tr>
<tr>
<td>5</td>
<td>Analog data format</td>
<td>Default</td>
<td>0</td>
<td>Byte</td>
<td>Default</td>
<td>0</td>
<td>1</td>
<td>0x0Y04</td>
</tr>
<tr>
<td>6</td>
<td>Behaviour of outputs at comm-</td>
<td>Off</td>
<td>0</td>
<td>Byte</td>
<td>Off</td>
<td>0</td>
<td>2</td>
<td>0x0Y05</td>
</tr>
<tr>
<td></td>
<td>uniication errors</td>
<td>Last value, Substitute value</td>
<td>1+(n<em>5) 2+(n</em>5), n ≤ 2</td>
<td>0x00</td>
<td>0x00</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>Channel configuration</td>
<td>see table &quot;Channel configuration&quot;</td>
<td>Byte</td>
<td>Default</td>
<td>0</td>
<td>130</td>
<td>0x0Y06</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Output channel 0</td>
<td></td>
<td></td>
<td></td>
<td>0x00</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>Channel monitoring</td>
<td>see table &quot;Channel monitoring&quot;</td>
<td>Byte</td>
<td>Default</td>
<td>0</td>
<td>3</td>
<td>0x0Y07</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Output channel 0</td>
<td></td>
<td></td>
<td></td>
<td>0x00</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>No.</td>
<td>Name</td>
<td>Value</td>
<td>Internal Value</td>
<td>Internal Value, type</td>
<td>Default</td>
<td>Min.</td>
<td>Max.</td>
<td>EDS Slot/Index</td>
</tr>
<tr>
<td>-----</td>
<td>-------------------------------</td>
<td>----------------</td>
<td>------------------------</td>
<td>----------------------</td>
<td>---------------</td>
<td>------</td>
<td>------</td>
<td>----------------</td>
</tr>
<tr>
<td>9</td>
<td>Substitute value</td>
<td>Output channel 0!</td>
<td>0...0xffff</td>
<td>Word</td>
<td>Default 0x0000</td>
<td>0</td>
<td>65535</td>
<td>0x0Y08</td>
</tr>
<tr>
<td></td>
<td>Output channel 0!</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10 to 15</td>
<td>Channel configuration and channel monitoring of the output channels 1 to 3</td>
<td>see tables &quot;Channel configuration&quot; and &quot;Channel monitoring&quot;</td>
<td>Byte</td>
<td>Default 0x0000</td>
<td>0</td>
<td>130</td>
<td>0x0Y09 to 0x0Y0E</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Output channel 0!</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>16 to 23</td>
<td>Channel configuration and channel monitoring of the output channels 4 to 7</td>
<td>see tables &quot;Channel configuration&quot; and &quot;Channel monitoring&quot;</td>
<td>Byte</td>
<td>Default 0x0000</td>
<td>0</td>
<td>128</td>
<td>0x0Y0F to 0x0Y16</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Output channel 0!</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>24</td>
<td>Channel configuration</td>
<td>see table &quot;Channel configuration&quot;</td>
<td>Byte</td>
<td>Default 0x0000</td>
<td>0</td>
<td>130</td>
<td>0x0Y17</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Output channel 8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>25</td>
<td>Channel monitoring</td>
<td>see table &quot;Channel monitoring&quot;</td>
<td>Byte</td>
<td>Default 0x0000</td>
<td>0</td>
<td>3</td>
<td>0x0Y18</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Output channel 8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>26</td>
<td>Substitute value</td>
<td>Output channel 8!</td>
<td>0...0xffff</td>
<td>Word</td>
<td>Default 0x0000</td>
<td>0</td>
<td>65535</td>
<td>0x0Y19</td>
</tr>
<tr>
<td></td>
<td>Output channel 8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>No.</td>
<td>Name</td>
<td>Value</td>
<td>Internal value, type</td>
<td>Default</td>
<td>Min.</td>
<td>Max.</td>
<td>EDS Slot/Index</td>
<td></td>
</tr>
<tr>
<td>-----</td>
<td>----------------------------------------------------------------------</td>
<td>----------------------------------------------------------------------</td>
<td>---------------------</td>
<td>---------</td>
<td>------</td>
<td>------</td>
<td>----------------</td>
<td></td>
</tr>
<tr>
<td>27 to 32</td>
<td>Channel configuration and channel monitoring of the output channels 9 to 11</td>
<td>see tables &quot;Channel configuration&quot; and &quot;Channel monitoring&quot;</td>
<td>Byte</td>
<td>Default 0x00 0x00</td>
<td>0</td>
<td>130</td>
<td>0x0Y1A to 0x0Y1F</td>
<td></td>
</tr>
<tr>
<td>33 to 40</td>
<td>Channel configuration and channel monitoring of the output channels 12 to 15</td>
<td>see tables &quot;Channel configuration&quot; and &quot;Channel monitoring&quot;</td>
<td>Byte</td>
<td>Default 0x00 0x00</td>
<td>0</td>
<td>128</td>
<td>0x0Y20 to 0x0Y27</td>
<td></td>
</tr>
</tbody>
</table>

1) With CS31 and addresses less than 70 and FBP, the value is increased by 1
2) Not with FBP

GSD file:

```plaintext
Ext_User_Prm_Data_Len = 42
Ext_User_Prm_Data_Const(0) =
0x05, 0xe7, 0x27, \n0x01, 0x00, 0x00, \n0x00, 0x00, 0x00, 0x00, \n0x00, 0x00, 0x00, 0x00, 0x00, \n0x00, 0x00, 0x00, 0x00, 0x00, 0x00, \n0x00, 0x00, 0x00, 0x00, 0x00, \n0x00, 0x00, 0x00, 0x00, 0x00, \n0x00, 0x00, 0x00, 0x00, 0x00, 0x00, \n0x00, 0x00, 0x00, 0x00, 0x00, 0x00;```

2020/12/10
### Output channels 0 and 8 (2 channels, AO523)

<table>
<thead>
<tr>
<th>No.</th>
<th>Name</th>
<th>Value</th>
<th>Internal value</th>
<th>Internal value, type</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Channel configuration</td>
<td>see below Table 564 “</td>
<td>see below Table 564 “</td>
<td>Byte</td>
<td>see below Table 564 “</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Channel configuration 3)&quot;</td>
<td>Channel configuration 3)&quot;</td>
<td></td>
<td>Channel configuration 3)&quot;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>on page 2662</td>
<td>on page 2662</td>
<td></td>
<td>on page 2662</td>
</tr>
<tr>
<td>2</td>
<td>Channel monitoring</td>
<td>see below Table 565 “</td>
<td>see below Table 565 “</td>
<td>Byte</td>
<td>see below Table 565 “</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Channel monitoring 4)&quot;</td>
<td>Channel monitoring 4)&quot;</td>
<td></td>
<td>Channel monitoring 4)&quot;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>on page 2662</td>
<td>on page 2662</td>
<td></td>
<td>on page 2662</td>
</tr>
<tr>
<td>3</td>
<td>Substitute value</td>
<td>0...65535</td>
<td>0...0xffff</td>
<td>Word</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>see Table 566 “Substitute</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>value” on page 2663</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Output channels 1...7 and 9...15 (14 channels, AO523)

<table>
<thead>
<tr>
<th>No.</th>
<th>Name</th>
<th>Internal value, type</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Channel configuration</td>
<td>Byte</td>
</tr>
<tr>
<td></td>
<td>see table 3)</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Channel monitoring</td>
<td>Byte</td>
</tr>
<tr>
<td></td>
<td>see table 4)</td>
<td></td>
</tr>
</tbody>
</table>

---

**Table 564: Channel configuration 3)**

<table>
<thead>
<tr>
<th>Internal value</th>
<th>Operating modes of the analog outputs, individually configurable</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Unused (default)</td>
</tr>
<tr>
<td>128</td>
<td>Analog output -10 V...+10 V</td>
</tr>
<tr>
<td>129</td>
<td>Analog output 0 mA...20 mA (not with the channels 4...7 and 12...15)</td>
</tr>
<tr>
<td>130</td>
<td>Analog output 4 mA...20 mA (not with the channels 4...7 and 12...15)</td>
</tr>
</tbody>
</table>

**Table 565: Channel monitoring 4)**

<table>
<thead>
<tr>
<th>Internal value</th>
<th>Monitoring</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Plausibility, open-circuit (broken wire) and short circuit (default)</td>
</tr>
<tr>
<td>1</td>
<td>Open-circuit (broken wire) and short circuit</td>
</tr>
<tr>
<td>2</td>
<td>Plausibility</td>
</tr>
<tr>
<td>3</td>
<td>No monitoring</td>
</tr>
</tbody>
</table>
### Table 566: Substitute value

<table>
<thead>
<tr>
<th>Intended behaviour of channel 0 when the control system stops</th>
<th>Required setting of the module parameter &quot;Behaviour of outputs in case of a communication error&quot;</th>
<th>Required setting of the channel parameter &quot;Substitute value&quot;</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output OFF</td>
<td>OFF</td>
<td>0</td>
</tr>
<tr>
<td>Last value</td>
<td>Last value</td>
<td>0</td>
</tr>
<tr>
<td>Substitute value</td>
<td>OFF or Last value</td>
<td>1...65535</td>
</tr>
</tbody>
</table>

### Diagnosis

<table>
<thead>
<tr>
<th>E1...E4</th>
<th>d1</th>
<th>d2</th>
<th>d3</th>
<th>d4</th>
<th>Identifier 000...063</th>
<th>AC500 display</th>
<th>&lt;- Display in</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>PS501 PLC browser</td>
<td></td>
</tr>
<tr>
<td>Byte 6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bit 6...7</td>
<td>-</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Class</th>
<th>Interface</th>
<th>Device</th>
<th>Module</th>
<th>Channel</th>
<th>Error Identifier</th>
<th>Error message</th>
<th>Remedy</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1)</td>
<td>2)</td>
<td>3)</td>
<td>4)</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### Module error

<table>
<thead>
<tr>
<th>3</th>
<th>14</th>
<th>1...10</th>
<th>31</th>
<th>31</th>
<th>19</th>
<th>Checksum error in the I/O module</th>
<th>Replace I/O module</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>11 / 12</td>
<td>ADR</td>
<td>1...10</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>14</td>
<td>1...10</td>
<td>31</td>
<td>31</td>
<td>3</td>
<td>Timeout in the I/O module</td>
<td>Replace I/O module</td>
</tr>
<tr>
<td></td>
<td>11 / 12</td>
<td>ADR</td>
<td>1...10</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>14</td>
<td>1...10</td>
<td>31</td>
<td>31</td>
<td>40</td>
<td>Different hard-/firmware versions in the module</td>
<td>Replace I/O module</td>
</tr>
<tr>
<td></td>
<td>11 / 12</td>
<td>ADR</td>
<td>1...10</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>14</td>
<td>1...10</td>
<td>31</td>
<td>31</td>
<td>43</td>
<td>Internal error in the module</td>
<td>Replace I/O module</td>
</tr>
<tr>
<td></td>
<td>11 / 12</td>
<td>ADR</td>
<td>1...10</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>14</td>
<td>1...10</td>
<td>31</td>
<td>31</td>
<td>36</td>
<td>Internal data exchange failure</td>
<td>Replace I/O module</td>
</tr>
<tr>
<td></td>
<td>11 / 12</td>
<td>ADR</td>
<td>1...10</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>14</td>
<td>1...10</td>
<td>31</td>
<td>31</td>
<td>9</td>
<td>Overflow diagnosis buffer</td>
<td>New start</td>
</tr>
<tr>
<td></td>
<td>11 / 12</td>
<td>ADR</td>
<td>1...10</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>14</td>
<td>1...10</td>
<td>31</td>
<td>31</td>
<td>26</td>
<td>Parameter error</td>
<td>Check master</td>
</tr>
<tr>
<td></td>
<td>11 / 12</td>
<td>ADR</td>
<td>1...10</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>14</td>
<td>1...10</td>
<td>31</td>
<td>31</td>
<td>11</td>
<td>Process voltage too low</td>
<td>Check process voltage</td>
</tr>
<tr>
<td></td>
<td>11 / 12</td>
<td>ADR</td>
<td>1...10</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>14</td>
<td>1...10</td>
<td>31</td>
<td>31</td>
<td>45</td>
<td>Process voltage is switched off (ON -&gt; OFF)</td>
<td>Process voltage ON</td>
</tr>
<tr>
<td></td>
<td>11 / 12</td>
<td>ADR</td>
<td>1...10</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### Channel error

<table>
<thead>
<tr>
<th>4</th>
<th>14</th>
<th>1...10</th>
<th>3</th>
<th>0...15</th>
<th>48</th>
<th>Analog value overflow at an analog output</th>
<th>Check output value</th>
</tr>
</thead>
<tbody>
<tr>
<td>E1...E4</td>
<td>d1</td>
<td>d2</td>
<td>d3</td>
<td>d4</td>
<td>Identifier 000...063</td>
<td>AC500 display</td>
<td>&lt;- Display in</td>
</tr>
<tr>
<td>--------</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----------------------</td>
<td>---------------</td>
<td>-------------</td>
</tr>
<tr>
<td>Class</td>
<td>Comp</td>
<td>Dev</td>
<td>Mod</td>
<td>Ch</td>
<td>Err</td>
<td>PS501 PLC browser</td>
<td></td>
</tr>
</tbody>
</table>

| Byte 6 | Bit 6...7 | Byte 3 | Byte 4 | Byte 5 | Byte 6 Bit 0...5 | FBP diagnosis block |

<table>
<thead>
<tr>
<th>Class</th>
<th>Interface</th>
<th>Device</th>
<th>Module</th>
<th>Channel</th>
<th>Error Identifier</th>
<th>Error message</th>
<th>Remedy</th>
</tr>
</thead>
<tbody>
<tr>
<td>1)</td>
<td>2)</td>
<td>3)</td>
<td>4)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11 / 12</td>
<td>ADR</td>
<td>1...10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>14</td>
<td>1...10</td>
<td>3</td>
<td>0...15</td>
<td>7</td>
<td>Analog value underflow at an analog output</td>
<td>Check output value</td>
</tr>
<tr>
<td>11 / 12</td>
<td>ADR</td>
<td>1...10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Remarks:

1) In AC500, the following interface identifier applies:
   14 = I/O bus, 11 = COM1 (e.g. CS31 bus), 12 = COM2.
   The FBP diagnosis block does not contain this identifier.

2) With "Device" the following allocation applies:
   31 = module itself, 1...10 = decentralized communication interface module 1...10, ADR = hardware address (e.g. of the DC551)

3) With "Module" the following allocation applies dependent of the master:
   Module error: I/O bus or FBP: 31 = module itself; COM1/COM2: 1...10 = expansion 1...10
   Channel error: I/O bus or FBP = module type (3 = AO); COM1/COM2: 1...10 = expansion 1...10

4) In case of module errors, with channel "31 = Module itself" is output.

State LEDs

During the power ON procedure, the module initializes automatically. All LEDs (except the channel LEDs) are ON during this time.
### LED States and Color

<table>
<thead>
<tr>
<th>Outputs O0...O7 and O8...O15</th>
<th>Analog output</th>
<th>LED = OFF</th>
<th>LED = ON</th>
<th>LED flashes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Outputs O0...O7 and O8...O15</td>
<td>Analog output</td>
<td>Yellow</td>
<td>Output is OFF</td>
<td>Output is ON (brightness depends on the value of the analog signal)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>UP</th>
<th>Process voltage 24 VDC via terminal</th>
<th>Green</th>
<th>Process voltage is missing</th>
<th>Process voltage OK</th>
<th>--</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>CH-ERR2</th>
<th>Channel error, error messages in groups (analog inputs or outputs combined into the groups 2 and 4)</th>
<th>Red</th>
<th>No error or process voltage is missing</th>
<th>Severe error within the corresponding group</th>
<th>Error on one channel of the group</th>
</tr>
</thead>
<tbody>
<tr>
<td>CH-ERR4</td>
<td>-------------------------------------------------------------------------------------------------</td>
<td>Red</td>
<td>--</td>
<td>Internal error</td>
<td>--</td>
</tr>
<tr>
<td>CH-ERR *)</td>
<td>Module error</td>
<td>Red</td>
<td>--</td>
<td></td>
<td>--</td>
</tr>
</tbody>
</table>

*) Both LEDs (CH-ERR2 and CH-ERR4) light up together

### Output ranges

#### Output ranges voltage and current

<table>
<thead>
<tr>
<th>Range</th>
<th>-10...+10 V</th>
<th>0...20 mA</th>
<th>4...20 mA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Overflow</td>
<td>&gt; 11.7589 V</td>
<td>&gt; 23.5178 mA</td>
<td>&gt; 22.8142 mA</td>
</tr>
<tr>
<td>Measured value too high</td>
<td>11.7589 V</td>
<td>23.5178 mA</td>
<td>22.8142 mA</td>
</tr>
<tr>
<td></td>
<td>: 10.0004 V</td>
<td>: 20.0007 mA</td>
<td>: 20.0006 mA</td>
</tr>
<tr>
<td>Normal range</td>
<td>10.0000 V</td>
<td>20.0000 mA</td>
<td>20.0000 mA</td>
</tr>
<tr>
<td></td>
<td>: 0.0004 V</td>
<td>: 0.0007 mA</td>
<td>: 4.0006 mA</td>
</tr>
<tr>
<td></td>
<td>: 0.0000 V</td>
<td>: 0.0000 mA</td>
<td>: 4.0000 mA</td>
</tr>
<tr>
<td></td>
<td>-0.0004 V</td>
<td>0 mA</td>
<td>3.9994 mA</td>
</tr>
<tr>
<td></td>
<td>: -10.0000 V</td>
<td>: 0 mA</td>
<td>0 mA</td>
</tr>
<tr>
<td>Measured value too low</td>
<td>-10.0004 V</td>
<td>0 mA</td>
<td>0 mA</td>
</tr>
<tr>
<td></td>
<td>: -11.7589 V</td>
<td>: 0 mA</td>
<td>0 mA</td>
</tr>
<tr>
<td>Underflow</td>
<td>0 V</td>
<td>0 mA</td>
<td>0 mA</td>
</tr>
</tbody>
</table>
### Range

<table>
<thead>
<tr>
<th>Range</th>
<th>Decimal</th>
<th>Hex.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Overflow</td>
<td>&gt; 32511</td>
<td>&gt; 7EFF</td>
</tr>
<tr>
<td>Measured value too high</td>
<td>32511</td>
<td>7EFF</td>
</tr>
<tr>
<td></td>
<td>27649</td>
<td>6C01</td>
</tr>
<tr>
<td>Normal range</td>
<td>27648</td>
<td>6C00</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>0000</td>
</tr>
<tr>
<td></td>
<td>-1</td>
<td>FFFF</td>
</tr>
<tr>
<td></td>
<td>-6912</td>
<td>E500</td>
</tr>
<tr>
<td></td>
<td>-27648</td>
<td>9400</td>
</tr>
<tr>
<td>Measured value too low</td>
<td>-27649</td>
<td>93FF</td>
</tr>
<tr>
<td></td>
<td>-32512</td>
<td>8100</td>
</tr>
<tr>
<td>Underflow</td>
<td>&lt; -32512</td>
<td>&lt; 8100</td>
</tr>
</tbody>
</table>

The represented resolution corresponds to 16 bits.

### Technical data

The system data of AC500 and S500 \(\text{©}\) Chapter 1.5.4.5.1 “System data AC500” on page 3058 are valid for standard version.

The system data of AC500-XC \(\text{©}\) Chapter 1.5.4.6.1 “System data AC500-XC” on page 3099 are valid for the XC version.

Only additional details are therefore documented below.

The technical data are also valid for the XC version.

### Parameter

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Process voltage</td>
<td></td>
</tr>
<tr>
<td>Connections</td>
<td>Terminals 1.8, 2.8, 3.8 and 4.8 for +24 V (UP) as well as 1.9, 2.9, 3.9 and 4.9 for 0 V (ZP)</td>
</tr>
<tr>
<td>Rated value</td>
<td>24 VDC</td>
</tr>
<tr>
<td>Max. ripple</td>
<td>5 %</td>
</tr>
<tr>
<td>Protection against reversed voltage</td>
<td>Yes</td>
</tr>
<tr>
<td>Rated protection fuse on UP</td>
<td>10 A fast</td>
</tr>
<tr>
<td>Galvanic isolation</td>
<td>Yes, per module</td>
</tr>
<tr>
<td>Current consumption</td>
<td></td>
</tr>
<tr>
<td>From 24 VDC power supply at the terminals UP/L+ and ZP/M of the CPU/bus module</td>
<td>Ca. 2 mA</td>
</tr>
<tr>
<td>Current consumption from UP at normal operation</td>
<td>0.15 A + output loads</td>
</tr>
</tbody>
</table>
### Technical data of the analog outputs

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inrush current from UP (at power up)</td>
<td>0.040 A²s</td>
</tr>
<tr>
<td>Max. length of analog cables, conductor cross section &gt; 0.14 mm²</td>
<td>100 m</td>
</tr>
<tr>
<td>Weight</td>
<td>300 g</td>
</tr>
<tr>
<td>Mounting position</td>
<td>Horizontal or vertical with derating (output load reduced to 50 % at 40 °C per group)</td>
</tr>
<tr>
<td>Cooling</td>
<td>The natural convection cooling must not be hindered by cable ducts or other parts in the switch-gear cabinet.</td>
</tr>
</tbody>
</table>

**NOTICE!**
**Attention:**
All I/O channels (digital and analog) are protected against reverse polarity, reverse supply, short circuit and continuous overvoltage up to 30 VDC.

---

#### Technical data of the analog outputs

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of channels per module</td>
<td>16, of which channels O0...O3 and O8...O11 for voltage and current, and channels O4...7 and O12...15 only for voltage</td>
</tr>
<tr>
<td>Distribution of channels into groups</td>
<td>2 groups of 8 channels each</td>
</tr>
<tr>
<td>Channels O0-...O7-</td>
<td>Terminals 1.0...1.7</td>
</tr>
<tr>
<td>Channels O0+...O7+</td>
<td>Terminals 2.0...2.7</td>
</tr>
<tr>
<td>Channels O8-...O15-</td>
<td>Terminals 3.0...3.7</td>
</tr>
<tr>
<td>Channels O8+...O15+</td>
<td>Terminals 4.0...4.7</td>
</tr>
<tr>
<td>Output type</td>
<td>Bipolar with voltage, unipolar with current</td>
</tr>
<tr>
<td>Galvanic isolation</td>
<td>Against internal supply and other modules</td>
</tr>
<tr>
<td>Configurability</td>
<td>-10 V...+10 V, 0 mA...20 mA, 4 mA...20 mA (each output can be configured individually), current outputs only channels 0...3 and 8...11</td>
</tr>
<tr>
<td>Output resistance (load), as current output</td>
<td>0 Ω...500 Ω</td>
</tr>
<tr>
<td>Output loadability, as voltage output</td>
<td>Max. ±10 mA</td>
</tr>
<tr>
<td>Indication of the output signals</td>
<td>One LED per channel</td>
</tr>
<tr>
<td>Resolution</td>
<td>12 bits (+ sign)</td>
</tr>
<tr>
<td>Conversion error of the analog values caused by non-linearity, adjustment error at factory and resolution within the normal range</td>
<td>Typ. ±0.5 % of full scale at 25 °C</td>
</tr>
<tr>
<td></td>
<td>Max. ±1 % of full scale (all ranges) at 0 °C...60 °C or EMC disturbance</td>
</tr>
<tr>
<td>Relationship between output signal and hex code</td>
<td>% Chapter 1.5.3.6.2.2.3.9 “Output ranges” on page 2665</td>
</tr>
<tr>
<td>Unused outputs</td>
<td>Can be left open-circuited</td>
</tr>
</tbody>
</table>
Ordering data

<table>
<thead>
<tr>
<th>Part no.</th>
<th>Description</th>
<th>Product life cycle phase *)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1SAP 250 200 R0001</td>
<td>AO523, analog output module, 16 AO, U/I, 12 bits + sign, 2-wires</td>
<td>Active</td>
</tr>
<tr>
<td>1SAP 450 200 R0001</td>
<td>AO523-XC, analog output module, 16 AO, U/I, 12 bits + sign, 2-wires, XC version</td>
<td>Active</td>
</tr>
</tbody>
</table>

*) For planning and commissioning of new installations use modules in Active status only.

AX521 - Analog input/output module
- 4 configurable analog inputs (I0 to I3) in 1 group (1.0...2.3)
  Resolution 12 bits plus sign
- 4 configurable analog outputs (O0 to O3) in 1 group (3.0...4.3)
  Resolution 12 bits plus sign
- Module-wise electrically isolated
- XC version for use in extreme ambient conditions available
1 I/O bus
2 Allocation between terminal number and signal name
3 4 yellow LEDs to display the signal states at the analog inputs (I0 - I3)
4 4 yellow LEDs to display the signal states at the analog outputs (O0 - O3)
5 1 green LED to display the state of the process supply voltage UP
6 2 red LEDs to display errors
7 Label
8 Terminal unit
9 DIN rail
10 Sign for XC version

Intended purpose

The device can be used as a decentralized I/O extension module for S500 Communication Interface Modules (e.g. CI592-CS31, CI501-PNIO, CI541-DP, CI581-CN) or as a centralized extension module for AC500 CPUs (PM5xx).

Functionality

AX521 4 analog inputs, individually configurable for
- Unused (default setting)
- 0 V...10 V
- -10 V...+10 V
- 0 mA...20 mA
- 4 mA...20 mA
- Pt100, -50 °C...+400 °C (2-wire)
- Pt100, -50 °C...+400 °C (3-wire), requires 2 channels
- Pt100, -50 °C...+70 °C (2-wire)
- Pt100, -50 °C...+70 °C (3-wire), requires 2 channels
- Pt1000, -50 °C...+400 °C (2-wire)
- Pt1000, -50 °C...+400 °C (3-wire), requires 2 channels
- Ni1000, -50 °C...+150 °C (2-wire)
- Ni1000, -50 °C...+150 °C (3-wire), requires 2 channels
- 0 V...10 V with differential inputs, requires 2 channels
- -10 V...+10 V with differential inputs, requires 2 channels
- Digital signals (digital input)

4 analog outputs, individually configurable for

- Unused (default setting)
- -10 V...+10 V
- 0 mA...20 mA
- 4 mA...20 mA

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Resolution of the analog channels</td>
<td></td>
</tr>
<tr>
<td>Voltage -10 V...+10 V</td>
<td>12 bits plus sign</td>
</tr>
<tr>
<td>Voltage 0 V...10 V</td>
<td>12 bits</td>
</tr>
<tr>
<td>Current 0 mA...20 mA, 4 mA...20 mA</td>
<td>12 bits</td>
</tr>
<tr>
<td>Temperature</td>
<td>0.1 °C</td>
</tr>
<tr>
<td>LED displays</td>
<td>11 LEDs for signals and error messages</td>
</tr>
<tr>
<td>Internal power supply</td>
<td>Via the expansion bus interface (I/O bus)</td>
</tr>
<tr>
<td>External power supply</td>
<td>Via the terminals ZP and UP (process voltage 24 VDC)</td>
</tr>
<tr>
<td>Required terminal unit</td>
<td>TU515 or TU516, Chapter 1.5.3.5.2 “TU515, TU516, TU541 and TU542 for I/O modules” on page 2311</td>
</tr>
</tbody>
</table>

Electrical Connection

For a detailed description of the mounting, disassembly and electrical connection of the module, please refer to the System Assembly, Construction and Connection chapter, Chapter 1.5.4.5 “AC500 (Standard)” on page 3058.

The modules are plugged on an I/O terminal unit, Chapter 1.5.3.5.2 “TU515, TU516, TU541 and TU542 for I/O modules” on page 2311. Properly position the modules and press until they lock in place. The terminal units are mounted on a DIN rail or with 2 screws plus the additional accessory for wall mounting (TA526, Chapter 1.5.3.8.1.4 “TA526 - Wall mounting accessory” on page 3035).

The electrical connection of the I/O channels is carried out using the 40 terminals of the I/O terminal unit. I/O modules can be replaced without re-wiring the terminal units.
The terminals 1.8, 2.8, 3.8 and 4.8 as well as 1.9, 2.9, 3.9 and 4.9 are electrically interconnected within the I/O terminal units and have always the same assignment, irrespective of the inserted module:

Terminals 1.8, 2.8, 3.8 and 4.8: process voltage UP = +24 VDC
Terminals 1.9, 2.9, 3.9 and 4.9: process voltage ZP = 0 VDC

The assignment of the other terminals:

<table>
<thead>
<tr>
<th>Terminals</th>
<th>Signal</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0 to 1.3</td>
<td>I0- to I3-</td>
<td>Negative poles of the 4 analog inputs</td>
</tr>
<tr>
<td>2.0 to 2.3</td>
<td>I0+ to I3+</td>
<td>Positive poles of the 4 analog inputs</td>
</tr>
<tr>
<td>3.0 to 3.3</td>
<td>O0- to O3-</td>
<td>Negative poles of the 4 analog outputs</td>
</tr>
<tr>
<td>4.0 to 4.3</td>
<td>O0+ to O3+</td>
<td>Positive poles of the 4 analog outputs</td>
</tr>
</tbody>
</table>

The negative poles of the analog inputs are electrically connected to each other to form an "Analog Ground" signal for the module.

The negative poles of the analog outputs are electrically connected to each other to form an "Analog Ground" signal for the module.

There is no galvanic isolation between the analog circuitry and ZP/UP. Therefore, the analog sensors must be galvanically isolated in order to avoid loops via the earth potential or the supply voltage.

Because of their common reference potential, analog current inputs cannot be circuitied in series, neither within the module nor with channels of other modules.

For the open-circuit detection (cut wire), each analog input channel is pulled up to "plus" by a high-resistance resistor. If nothing is connected, the maximum voltage will be read in then.

The internal power supply voltage for the module's circuitry is carried out via the I/O bus (provided by a bus module or a CPU). Thus, the current consumption from 24 VDC power supply at the terminals L+/UP and M/ZP of the CPU/bus module increases by 2 mA per I/O module.

The external power supply connection is carried out via the UP (+24 VDC) and the ZP (0 VDC) terminals.
**NOTICE!**

**Risk of damaging the PLC modules!**

Overvoltages and short circuits might damage the PLC modules.

- Make sure that all voltage sources (supply voltage and process supply voltage) are switched off before you begin with operations at the system.
- Never connect any voltages or signals to reserved terminals (marked with ---). Reserved terminals may carry internal voltages.

Generally, analog signals must be laid in shielded cables. The cable shields must be earthed at both sides of the cables. In order to avoid unacceptable potential differences between different parts of the installation, low resistance equipotential bonding conductors must be laid.

Only for simple applications (low electromagnetic disturbances, no high requirement on precision), the shielding can also be omitted.

The following figure shows the electrical connection of the I/O module.

Fig. 144: Terminal assignment

Attention:
By installing equipotential bonding conductors between the different parts of the system, it must be made sure that the potential difference between ZP and AGND never can exceed 1 V.

Attention:
The process voltage must be included in the earthing concept of the control system (e.g. earthing the minus pole).
Connection of resistance thermometers in 2-wire configuration

When resistance thermometers (Pt100, Pt1000, Ni1000) are used, a constant current must flow through them to build the necessary voltage drop for the evaluation. For this, the I/O module provides a constant current source which is multiplexed over the 8 analog channels.

![Diagram of 2-wire configuration](image)

**Fig. 145: Connection example**

<table>
<thead>
<tr>
<th>Thermometer</th>
<th>Temperature Range</th>
<th>Configuration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pt100</td>
<td>-50 °C...+70 °C</td>
<td>2-wire used</td>
</tr>
<tr>
<td>Pt1000</td>
<td>-50 °C...+400 °C</td>
<td>2-wire used</td>
</tr>
<tr>
<td>Ni1000</td>
<td>-50 °C...+150 °C</td>
<td>2-wire used</td>
</tr>
</tbody>
</table>

The I/O module performs a linearization of the resistance characteristic.

In order to avoid error messages from unused analog input channels, it is useful to configure them as "unused".

Connection of resistance thermometers in 3-wire configuration

When resistance thermometers (Pt100, Pt1000, Ni1000) are used, a constant current must flow through them to build the necessary voltage drop for the evaluation. For this, the I/O module provides a constant current source which is multiplexed over the max. 8 (depending on the configuration) analog channels.
Fig. 146: Connection example

If several measuring points are adjacent to each other, only one return line is necessary. This saves wiring costs.

With the 3-wire configuration, two adjacent analog channels belong together (e.g. the channels 0 and 1). In this case, both channels are configured according to the desired operating mode. The lower address must be the even address (channel 0), the next higher address must be the odd address (channel 1).

The constant current of one channel flows through the resistance thermometer. The constant current of the other channel flows through one of the cores. The module calculates the measured value from the two voltage drops and stores it under the input with the higher channel number (e.g. I1).

In order to keep measuring errors as small as possible, it is necessary to have all the involved conductors in the same cable. All the conductors must have the same cross section.

<table>
<thead>
<tr>
<th>Sensor</th>
<th>Temperature Range</th>
<th>Configuration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pt100</td>
<td>-50 °C...+70 °C</td>
<td>3-wire configuration, two channels used</td>
</tr>
<tr>
<td>Pt100</td>
<td>-50 °C...+400 °C</td>
<td>3-wire configuration, two channels used</td>
</tr>
<tr>
<td>Pt1000</td>
<td>-50 °C...+400 °C</td>
<td>3-wire configuration, two channels used</td>
</tr>
<tr>
<td>Ni1000</td>
<td>-50 °C...+150 °C</td>
<td>3-wire configuration, two channels used</td>
</tr>
</tbody>
</table>

The I/O module performs a linearization of the resistance characteristic.

In order to avoid error messages from unused analog input channels, it is useful to configure them as "unused".
Connection of active-type analog sensors (Voltage) with electrically isolated power supply

By connecting the sensor's negative pole of the output voltage to AGND, the electrically isolated voltage source of the sensor is referred to ZP.

The following measuring ranges can be configured for AX521 "Parameterization" on page 2680 and for AX522 "Parameterization" on page 2705:

<table>
<thead>
<tr>
<th>Voltage</th>
<th>0 V...10 V</th>
<th>1 channel used</th>
</tr>
</thead>
<tbody>
<tr>
<td>Voltage</td>
<td>-10 V...+10 V</td>
<td>1 channel used</td>
</tr>
</tbody>
</table>

In order to avoid error messages or long processing times, it is useful to configure unused analog input channels as "unused".

Connection of active-type analog sensors (Current) with electrically isolated power supply

Fig. 147: Connection example

Fig. 148: Connection example
Unused input channels can be left open-circuited, because they are of low resistance.

Connection of active-type analog sensors (Voltage) with no electrically isolated power supply

![Connection diagram](image)

**Fig. 149: Connection example**

**CAUTION!**
The potential difference between AGND and ZP at the module must not be greater than 1V, not even in case of long lines (see figure Terminal Assignment).

If AGND does not get connected to ZP, the sensor current flows to ZP via the AGND line. The measuring signal is distorted, as a very small current flows through the voltage line. The total current through the PTC should not exceed 50 mA. This measuring method is therefore only suitable for short lines and small sensor currents. If there are bigger distances, the difference measuring method should be applied.

<table>
<thead>
<tr>
<th>Voltage</th>
<th>0 V...10 V</th>
<th>1 channel used</th>
</tr>
</thead>
<tbody>
<tr>
<td>Voltage</td>
<td>-10 V...+10 V *)</td>
<td>1 channel used</td>
</tr>
</tbody>
</table>

*) if the sensor can provide this signal range

In order to avoid error messages or long processing times, it is useful to configure unused analog input channels as "unused".
Connection of passive-type analog sensors (Current)

Fig. 150: Connection example

<table>
<thead>
<tr>
<th>Current</th>
<th>4 mA...20 mA</th>
<th>1 channel used</th>
</tr>
</thead>
</table>

**CAUTION!**

If, during initialization, an analog current sensor supplies more than 25 mA for more than 1 second to an analog input, this input is switched off by the module (input protection). In such cases, it is recommended to protect the analog input by a 10-volt Zener diode (in parallel to I+ and I-). But, in general, sensors with fast initialization or without current peaks higher than 25 mA are preferrable.

Unused input channels can be left open-circuited because they are of low resistance.

Connection of active-type analog sensors (Voltage) to differential inputs

Differential inputs are very useful if analog sensors are used which are remotely non-isolated (e.g. the minus terminal is remotely earthed).

The use of differential inputs helps to considerably increase the measuring accuracy and to avoid earthing loops.

With differential input configurations, two adjacent analog channels belong together (e.g. the channels 0 and 1). In this case, both channels are configured according to the desired operating mode. The lower address must be the even address (channel 0), the next higher address must be the odd address (channel 1). The converted analog value is available at the higher address (channel 1).

The analog value is calculated by subtraction of the input value with the higher address from the input value of the lower address.

The converted analog value is available at the odd channel (higher address).

**CAUTION!**

The earthing potential at the sensors must not have too large a potential difference with respect to ZP (max. ±1 V within the full signal range). Otherwise, problems may occur concerning the common-mode input voltages of the involved analog inputs.
The negative pole of the sensor must be earthed next to the sensor.

<table>
<thead>
<tr>
<th>Voltage</th>
<th>0 V...10 V</th>
<th>-10 V...+10 V</th>
</tr>
</thead>
<tbody>
<tr>
<td>with differential inputs, 2 channels used</td>
<td>with differential inputs, 2 channels used</td>
<td></td>
</tr>
</tbody>
</table>

In order to avoid error messages or long processing times, it is useful to configure unused analog input channels as "unused".

**Use of analog inputs as digital inputs**

Several (or all) analog inputs can be configured as digital inputs. The inputs are not electrically isolated against the other analog channels.
**Connection of analog output loads (Voltage, current)**

<table>
<thead>
<tr>
<th>Voltage</th>
<th>[-10 V...+10 V]</th>
<th>Load max. ±10 mA</th>
<th>1 channel used</th>
</tr>
</thead>
<tbody>
<tr>
<td>Current</td>
<td>0 mA...20 mA</td>
<td>Load 0 Ω...500 Ω</td>
<td>1 channel used</td>
</tr>
<tr>
<td>Current</td>
<td>4 mA...20 mA</td>
<td>Load 0 Ω...500 Ω</td>
<td>1 channel used</td>
</tr>
</tbody>
</table>

Only the channels 0...3 can be configured as current output (0 mA...20 mA or 4 mA...20 mA). Unused analog outputs can be left open-circuited.

**Internal data exchange**

<table>
<thead>
<tr>
<th>Digital inputs (bytes)</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Digital outputs (bytes)</td>
<td>0</td>
</tr>
<tr>
<td>Counter input data (words)</td>
<td>4</td>
</tr>
<tr>
<td>Counter output data (words)</td>
<td>4</td>
</tr>
</tbody>
</table>

**I/O configuration**

The module does not store configuration data itself. It gets its parameterization data from the master device of the I/O bus (CPU or bus module) during power-up of the system. Hence, replacing I/O modules is possible without any re-parameterization via software.
Parameterization

<table>
<thead>
<tr>
<th>No.</th>
<th>Name</th>
<th>Value</th>
<th>Internal value</th>
<th>Internal value, type</th>
<th>Default</th>
<th>Min.</th>
<th>Max.</th>
<th>EDS Slot/Index</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Module ID</td>
<td>Internal value</td>
<td>1505</td>
<td>Word</td>
<td>1505</td>
<td>0</td>
<td>65535</td>
<td>0x0Y01</td>
</tr>
<tr>
<td>2</td>
<td>Ignore module</td>
<td>No</td>
<td>0</td>
<td>Byte</td>
<td>No</td>
<td>0x00</td>
<td>Not for FBP</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Parameter length in bytes</td>
<td>Internal value</td>
<td>21</td>
<td>Byte</td>
<td>21-CPU 21-FBP</td>
<td>0</td>
<td>255</td>
<td>0x0Y02</td>
</tr>
<tr>
<td>4</td>
<td>Check supply</td>
<td>Off</td>
<td>0</td>
<td>Byte</td>
<td>On</td>
<td>0x01</td>
<td>1</td>
<td>0x0Y03</td>
</tr>
<tr>
<td>5</td>
<td>Analog data format</td>
<td>Default</td>
<td>0</td>
<td>Byte</td>
<td>Default 0x00</td>
<td>0</td>
<td>19</td>
<td>0x0Y04</td>
</tr>
<tr>
<td>6</td>
<td>Behaviour of outputs at com-</td>
<td>Off</td>
<td>0</td>
<td>Byte</td>
<td>Off 0x00</td>
<td>0</td>
<td>2</td>
<td>0x0Y05</td>
</tr>
<tr>
<td></td>
<td>unication errors</td>
<td>Last value</td>
<td>1+(n*5)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Substitute value</td>
<td>2+(n*5), n ≤ 2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>Channel configuration</td>
<td>see table</td>
<td>Byte</td>
<td>Default 0x00</td>
<td>0</td>
<td>17</td>
<td>19</td>
<td>0x0Y06</td>
</tr>
<tr>
<td>8</td>
<td>Channel monitoring</td>
<td>see table</td>
<td>Byte</td>
<td>Default 0x00</td>
<td>0</td>
<td>3</td>
<td>3</td>
<td>0x0Y07</td>
</tr>
<tr>
<td>No.</td>
<td>Name</td>
<td>Value</td>
<td>Internal value, type</td>
<td>Internal value</td>
<td>Default</td>
<td>Min.</td>
<td>Max.</td>
<td>EDS Slot/Index</td>
</tr>
<tr>
<td>-----</td>
<td>--------------------------------------------------</td>
<td>-----------------------------------------------------------------------</td>
<td>----------------------</td>
<td>----------------</td>
<td>---------</td>
<td>------</td>
<td>------</td>
<td>----------------</td>
</tr>
<tr>
<td>9 to 14</td>
<td>Channel configuration and channel monitoring of the input channels 1 to 3</td>
<td>see tables &quot;Channel configuration&quot; and &quot;Channel monitoring&quot;</td>
<td>Byte</td>
<td>Default</td>
<td>0x00</td>
<td>0</td>
<td>19</td>
<td>0x0Y08 to 0x0Y0D</td>
</tr>
<tr>
<td>15</td>
<td>Channel configuration Output channel 0</td>
<td>see table &quot;Channel configuration&quot;</td>
<td>Byte</td>
<td>Default</td>
<td>0x00</td>
<td>0</td>
<td>130</td>
<td>0x0Y0E</td>
</tr>
<tr>
<td>16</td>
<td>Channel monitoring Output channel 0</td>
<td>see table &quot;Channel monitoring&quot;</td>
<td>Byte</td>
<td>Default</td>
<td>0x00</td>
<td>0</td>
<td>3</td>
<td>0x0Y0F</td>
</tr>
<tr>
<td>17</td>
<td>Substitute value Output channel 0</td>
<td>only valid for output channel 0</td>
<td>Word</td>
<td>Default</td>
<td>0x0000</td>
<td>0</td>
<td>65535</td>
<td>0x0Y10</td>
</tr>
<tr>
<td>18 to 21</td>
<td>Channel configuration and channel monitoring of the output channels 1 to 2</td>
<td>see tables &quot;Channel configuration&quot; and &quot;Channel monitoring&quot;</td>
<td>Byte</td>
<td>Default</td>
<td>0x00</td>
<td>0</td>
<td>130</td>
<td>0x0Y11 to 0x0Y14</td>
</tr>
<tr>
<td>22</td>
<td>Channel configuration Output channel 3</td>
<td>see table &quot;Channel configuration&quot;</td>
<td>Byte</td>
<td>Default</td>
<td>0x00</td>
<td>0</td>
<td>130</td>
<td>0x0Y15</td>
</tr>
<tr>
<td>23</td>
<td>Channel monitoring Output channel 3</td>
<td>see table &quot;Channel monitoring&quot;</td>
<td>Byte</td>
<td>Default</td>
<td>0x00</td>
<td>0</td>
<td>3</td>
<td>0x0Y16</td>
</tr>
</tbody>
</table>

1) With CS31 and addresses less than 70 and FBP, the value is increased by 1
2) Not with FBP
GSD file:

```
Ext_User_Prm_Data_Len = 24
Ext_User_Prm_Data_Const(0) = 0x05, 0xe2, 0x15, \
0x01, 0x00, 0x00 \
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00; 
```

Table 567: Input channel (4x)

<table>
<thead>
<tr>
<th>No.</th>
<th>Name</th>
<th>Internal value, type</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Channel configuration see table 2)</td>
<td>Byte</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0x00 see table 2)</td>
</tr>
<tr>
<td>2</td>
<td>Channel monitoring see table 3)</td>
<td>Byte</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0x00 see table 3)</td>
</tr>
</tbody>
</table>

Table 568: Channel configuration 2)

<table>
<thead>
<tr>
<th>Internal value</th>
<th>Operating modes of the analog inputs, individually configurable</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Unused (default)</td>
</tr>
<tr>
<td>1</td>
<td>Analog input 0 V...10 V</td>
</tr>
<tr>
<td>2</td>
<td>Digital input</td>
</tr>
<tr>
<td>3</td>
<td>Analog input 0 mA...20 mA</td>
</tr>
<tr>
<td>4</td>
<td>Analog input 4 mA...20 mA</td>
</tr>
<tr>
<td>5</td>
<td>Analog input -10 V...+10 V</td>
</tr>
<tr>
<td>8</td>
<td>Analog input Pt100, -50 °C...+400 °C (2-wire)</td>
</tr>
<tr>
<td>9</td>
<td>Analog input Pt100, -50 °C...+400 °C (3-wire), requires 2 channels *)</td>
</tr>
<tr>
<td>10</td>
<td>Analog input 0...10 V via differential inputs, requires 2 channels *)</td>
</tr>
<tr>
<td>11</td>
<td>Analog input -10 V...+10 V via differential inputs, requires 2 channels *)</td>
</tr>
<tr>
<td>14</td>
<td>Analog input Pt100, -50 °C...+70 °C (2-wire)</td>
</tr>
<tr>
<td>15</td>
<td>Analog input Pt100, -50 °C...+70 °C (3-wire), requires 2 channels *)</td>
</tr>
<tr>
<td>16</td>
<td>Analog input Pt1000, -50 °C...+400 °C (2-wire)</td>
</tr>
<tr>
<td>17</td>
<td>Analog input Pt1000, -50 °C...+400 °C (3-wire), requires 2 channels *)</td>
</tr>
<tr>
<td>18</td>
<td>Analog input Ni1000, -50 °C...+150 °C (2-wire)</td>
</tr>
<tr>
<td>19</td>
<td>Analog input Ni1000, -50 °C...+150 °C (3-wire), requires 2 channels *)</td>
</tr>
</tbody>
</table>

*) In the operating modes with 3-wire configuration or with differential inputs, two adjacent analog inputs belong together (e.g. the channels 4 and 1). In these cases, both channels are configured in the desired operating mode. The lower address must be the even address (channel 0). The next higher address must be the odd address (channel 1). The converted analog value is available at the higher address (channel 1).
### Table 569: Channel monitoring

<table>
<thead>
<tr>
<th>Internal value</th>
<th>Monitoring</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Plausibility, open-circuit (broken wire) and short circuit</td>
</tr>
<tr>
<td>3</td>
<td>No monitoring</td>
</tr>
</tbody>
</table>

### Table 570: Output channel 0 (1 channel)

<table>
<thead>
<tr>
<th>No.</th>
<th>Name</th>
<th>Value</th>
<th>Internal value</th>
<th>Internal value, type</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Channel configuration</td>
<td>see table 4)</td>
<td>see table 4)</td>
<td>Byte</td>
<td>see table 4)</td>
</tr>
<tr>
<td>2</td>
<td>Channel monitoring</td>
<td>see table 5)</td>
<td>see table 5)</td>
<td>Byte</td>
<td>see table 5)</td>
</tr>
<tr>
<td>3</td>
<td>Substitute value</td>
<td>0...65535</td>
<td>0...0xffff</td>
<td>Word</td>
<td>0</td>
</tr>
</tbody>
</table>

### Table 571: Output channels 1...3 (3x)

<table>
<thead>
<tr>
<th>No.</th>
<th>Name</th>
<th>Internal value, type</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Channel configuration</td>
<td>Byte</td>
</tr>
<tr>
<td>2</td>
<td>Channel monitoring</td>
<td>Byte</td>
</tr>
</tbody>
</table>

### Table 572: Channel configuration

<table>
<thead>
<tr>
<th>Internal value</th>
<th>Operating modes of the analog outputs, individually configurable</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Unused (default)</td>
</tr>
<tr>
<td>128</td>
<td>Analog output -10 V...+10 V</td>
</tr>
<tr>
<td>129</td>
<td>Analog output 0 mA...20 mA (not with the channels 4...7 and 12...15)</td>
</tr>
<tr>
<td>130</td>
<td>Analog output 4 mA...20 mA (not with the channels 4...7 and 12...15)</td>
</tr>
</tbody>
</table>

### Table 573: Channel monitoring

<table>
<thead>
<tr>
<th>Internal value</th>
<th>Monitoring</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Plausibility, open circuit (broken wire) and short circuit (default)</td>
</tr>
<tr>
<td>3</td>
<td>No monitoring</td>
</tr>
</tbody>
</table>

### Table 574: Substitute value

<table>
<thead>
<tr>
<th>Intended behaviour of output channel when the control system stops</th>
<th>Required setting of the module parameter &quot;Behaviour of outputs in case of a communication error&quot;</th>
<th>Required setting of the channel parameter &quot;Substitute value&quot;</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output OFF</td>
<td>Off</td>
<td>0</td>
</tr>
<tr>
<td>Last value infinite</td>
<td>Last value</td>
<td>0</td>
</tr>
</tbody>
</table>

---

2020/12/10

3ADR010583, 1, en_US

2683
## Intended behaviour of output channel when the control system stops

<table>
<thead>
<tr>
<th></th>
<th>Required setting of the module parameter &quot;Behaviour of outputs in case of a communication error&quot;</th>
<th>Required setting of the channel parameter &quot;Substitute value&quot;</th>
</tr>
</thead>
<tbody>
<tr>
<td>Last value for 5 s and then turn off</td>
<td>Last value 5 sec</td>
<td>0</td>
</tr>
<tr>
<td>Last value for 10 s and then turn off</td>
<td>Last value 10 sec</td>
<td>0</td>
</tr>
<tr>
<td>Substitute value infinite</td>
<td>Substitute value</td>
<td>Depending on configuration</td>
</tr>
<tr>
<td>Substitute value for 5 s and then turn off</td>
<td>Substitute value 5 sec</td>
<td>Depending on configuration</td>
</tr>
<tr>
<td>Substitute value for 10 s and then turn off</td>
<td>Substitute value 10 sec</td>
<td>Depending on configuration</td>
</tr>
</tbody>
</table>

## Diagnosis

<table>
<thead>
<tr>
<th>E1...E4</th>
<th>d1</th>
<th>d2</th>
<th>d3</th>
<th>d4</th>
<th>Identifier 000...063</th>
<th>AC500 display</th>
<th>&lt;- Display in</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class</td>
<td>Comp</td>
<td>Dev</td>
<td>Mod</td>
<td>Ch</td>
<td>Err</td>
<td>PS501 PLC browser</td>
<td></td>
</tr>
<tr>
<td>Byte 6</td>
<td>Bit 6...7</td>
<td>Byte 3</td>
<td>Byte 4</td>
<td>Byte 5</td>
<td>Byte 6 Bit 0...5</td>
<td>FBP diagnosis block</td>
<td></td>
</tr>
<tr>
<td>Class</td>
<td>Interface</td>
<td>Device</td>
<td>Module</td>
<td>Channel</td>
<td>Error Identifier</td>
<td>Error message</td>
<td>Remedy</td>
</tr>
<tr>
<td>---------</td>
<td>-------------</td>
<td>--------</td>
<td>--------</td>
<td>--------</td>
<td>----------------</td>
<td>-------------</td>
<td>---------</td>
</tr>
<tr>
<td>1)</td>
<td>2)</td>
<td>3)</td>
<td>4)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Module error

<table>
<thead>
<tr>
<th>3</th>
<th>14</th>
<th>1...10</th>
<th>31</th>
<th>31</th>
<th>19</th>
<th>Checksum error in the I/O module</th>
<th>Replace I/O module</th>
</tr>
</thead>
<tbody>
<tr>
<td>11 / 12</td>
<td>ADR</td>
<td>1...10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>14</td>
<td>1...10</td>
<td>31</td>
<td>31</td>
<td>3</td>
<td>Timeout in the I/O module</td>
<td>Replace I/O module</td>
</tr>
<tr>
<td>11 / 12</td>
<td>ADR</td>
<td>1...10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>14</td>
<td>1...10</td>
<td>31</td>
<td>31</td>
<td>40</td>
<td>Different hard-/firmware versions in the module</td>
<td>Replace I/O module</td>
</tr>
<tr>
<td>11 / 12</td>
<td>ADR</td>
<td>1...10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>14</td>
<td>1...10</td>
<td>31</td>
<td>31</td>
<td>43</td>
<td>Internal error in the module</td>
<td>Replace I/O module</td>
</tr>
<tr>
<td>11 / 12</td>
<td>ADR</td>
<td>1...10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>14</td>
<td>1...10</td>
<td>31</td>
<td>31</td>
<td>36</td>
<td>Internal data exchange failure</td>
<td>Replace I/O module</td>
</tr>
<tr>
<td>11 / 12</td>
<td>ADR</td>
<td>1...10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>14</td>
<td>1...10</td>
<td>31</td>
<td>31</td>
<td>9</td>
<td>Overflow diagnosis buffer</td>
<td>New start</td>
</tr>
<tr>
<td>11 / 12</td>
<td>ADR</td>
<td>1...10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>14</td>
<td>1...10</td>
<td>31</td>
<td>31</td>
<td>26</td>
<td>Parameter error</td>
<td>Check master</td>
</tr>
<tr>
<td>11 / 12</td>
<td>ADR</td>
<td>1...10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>14</td>
<td>1...10</td>
<td>31</td>
<td>31</td>
<td>11</td>
<td>Process voltage too low</td>
<td>Check process voltage</td>
</tr>
</tbody>
</table>
## Channel error

<table>
<thead>
<tr>
<th>Class</th>
<th>Interface</th>
<th>Device</th>
<th>Module</th>
<th>Channel</th>
<th>Error Identifier</th>
<th>Error message</th>
<th>Remedy</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>14</td>
<td>1...10</td>
<td>31</td>
<td>31</td>
<td>45</td>
<td>Process voltage is switched off (ON −&gt; OFF)</td>
<td>Process voltage ON</td>
</tr>
<tr>
<td></td>
<td>11 / 12</td>
<td>ADR</td>
<td>1...10</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Channel error rules:

1. **In AC500, the following interface identifier applies:**
   - 14 = I/O bus, 11 = COM1 (e.g. CS31 bus), 12 = COM2.
   - The FBP diagnosis block does not contain this identifier.

2. **With "Device" the following allocation applies:**
   - 31 = module itself, 1...10 = decentralized communication interface module 1...10, ADR = hardware address (e.g. of the DC551)

3. **With "Module" the following allocation applies depending on the master:**
   - Module error: I/O bus or FBP: 31 = module itself; COM1/COM2: 1...10 = expansion 1...10
   - Channel error: I/O bus or FBP = module type (1 = AI, 3 = AO); COM1/COM2: 1...10 = expansion 1...10

4. **In case of module errors, with channel "31 = Module itself" is output.**

## State LEDs

During the power ON procedure, the module initializes automatically. All LEDs (except the channel LEDs) are ON during this time.
### Measuring ranges

**Input ranges of voltage, current and digital input**

The represented resolution corresponds to 16 bits.

<table>
<thead>
<tr>
<th>Range</th>
<th>0...10 V</th>
<th>-10...+10 V</th>
<th>0...20 mA</th>
<th>4...20 mA</th>
<th>Digital input</th>
</tr>
</thead>
<tbody>
<tr>
<td>Overflow</td>
<td>&gt;11.7589</td>
<td>&gt;11.7589</td>
<td>&gt;23.5178</td>
<td>&gt;22.8142</td>
<td></td>
</tr>
<tr>
<td>Measured value too high</td>
<td>11.7589</td>
<td>11.7589</td>
<td>23.5178</td>
<td>22.8142</td>
<td></td>
</tr>
<tr>
<td>.</td>
<td>.</td>
<td>10.0004</td>
<td>20.0007</td>
<td>20.0006</td>
<td></td>
</tr>
<tr>
<td>10.0004</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Normal range</td>
<td>10.0000</td>
<td>10.0000</td>
<td>20.0000</td>
<td>20.0000</td>
<td>ON</td>
</tr>
<tr>
<td>Normal range or measured value too low</td>
<td>0.0004</td>
<td>0.0004</td>
<td>0.0007</td>
<td>4.0006</td>
<td></td>
</tr>
<tr>
<td>.</td>
<td>.</td>
<td>0.0007</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.0004</td>
<td>0.0000</td>
<td>0</td>
<td>4</td>
<td>OFF</td>
<td></td>
</tr>
</tbody>
</table>
### Input ranges resistance temperature detector

<table>
<thead>
<tr>
<th>Range</th>
<th>Pt100 / Pt 1000 -50...70 °C</th>
<th>Pt100 / Pt1000 -50...400 °C</th>
<th>Ni1000 -50...150 °C</th>
</tr>
</thead>
<tbody>
<tr>
<td>Overflow</td>
<td>&gt; 80.0 °C</td>
<td>&gt; 450.0 °C</td>
<td>&gt; 160.0 °C</td>
</tr>
<tr>
<td>Measured value too high</td>
<td>450.0 °C</td>
<td>400.1 °C</td>
<td></td>
</tr>
</tbody>
</table>

### Digital value

<table>
<thead>
<tr>
<th>Range</th>
<th>Digital value</th>
<th>Decimal</th>
<th>Hex.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Overflow</td>
<td>32767</td>
<td>32767</td>
<td>7FFF</td>
</tr>
<tr>
<td>Measured value too high</td>
<td>32511</td>
<td>32511</td>
<td>7EFF</td>
</tr>
<tr>
<td>Normal range</td>
<td>27648</td>
<td>27648</td>
<td>6C00</td>
</tr>
<tr>
<td>Normal range or measured value too low</td>
<td>0</td>
<td>0</td>
<td>0000</td>
</tr>
<tr>
<td></td>
<td>-1</td>
<td>-1</td>
<td>FFFF</td>
</tr>
<tr>
<td></td>
<td>-4864</td>
<td>-4864</td>
<td>ED00</td>
</tr>
<tr>
<td></td>
<td>-6912</td>
<td>-6912</td>
<td>E500</td>
</tr>
<tr>
<td></td>
<td>-27648</td>
<td>-27648</td>
<td>9400</td>
</tr>
<tr>
<td>Measured value too low</td>
<td>-27649</td>
<td>-27649</td>
<td>93FF</td>
</tr>
<tr>
<td></td>
<td>-32512</td>
<td>-32512</td>
<td>8100</td>
</tr>
<tr>
<td>Underflow</td>
<td>-32768</td>
<td>-32768</td>
<td>8000</td>
</tr>
</tbody>
</table>

### Range

<table>
<thead>
<tr>
<th>Range</th>
<th>0...10 V</th>
<th>-10...+10 V</th>
<th>0...20 mA</th>
<th>4...20 mA</th>
<th>Digital input</th>
</tr>
</thead>
<tbody>
<tr>
<td>Digital input</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>3.9994</td>
</tr>
<tr>
<td>Measured value too low</td>
<td></td>
<td>-10.0004</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Underflow</td>
<td>&lt;-1.7593</td>
<td>&lt;-11.7589</td>
<td>&lt;0.0000</td>
<td>&lt;1.1858</td>
<td></td>
</tr>
<tr>
<td>Range</td>
<td>Pt100 / Pt 1000 -50...70 °C</td>
<td>Pt100 / Pt1000 -50...400 °C</td>
<td>Ni1000 -50...150 °C</td>
<td></td>
<td></td>
</tr>
<tr>
<td>------------------------</td>
<td>----------------------------</td>
<td>----------------------------</td>
<td>---------------------</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>160.0 °C</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>:</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>150.1 °C</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Normal range</td>
<td></td>
<td>80.0 °C</td>
<td>:</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>70.1 °C</td>
<td>:</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>400.0 °C</td>
<td>:</td>
<td>150.0 °C</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>:</td>
<td>:</td>
<td>:</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.1 °C</td>
<td>0.1 °C</td>
<td>0.1 °C</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Normal range</td>
<td></td>
<td>0.0 °C</td>
<td>0.0 °C</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>-0.1 °C</td>
<td>-0.1 °C</td>
<td>-0.1 °C</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>-50.0 °C</td>
<td>-50.0 °C</td>
<td>-50.0 °C</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Measured value too low</td>
<td>-50.1 °C</td>
<td>-50.1 °C</td>
<td>-50.1 °C</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>-60.0 °C</td>
<td>-60.0 °C</td>
<td>-60.0 °C</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Underflow</td>
<td>&lt; -60.0 °C</td>
<td>&lt; -60.0 °C</td>
<td>&lt; -60.0 °C</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Range</th>
<th>Digital value</th>
<th>Hex.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Decimal</td>
<td></td>
</tr>
<tr>
<td>Overflow</td>
<td>32767</td>
<td>7FFF</td>
</tr>
<tr>
<td>Measured value too high</td>
<td>4500</td>
<td>1194</td>
</tr>
<tr>
<td></td>
<td>4001</td>
<td>0FA1</td>
</tr>
<tr>
<td></td>
<td>1600</td>
<td>0640</td>
</tr>
<tr>
<td></td>
<td>1501</td>
<td>05DD</td>
</tr>
<tr>
<td></td>
<td>800</td>
<td>0320</td>
</tr>
<tr>
<td></td>
<td>701</td>
<td>02BD</td>
</tr>
<tr>
<td>Normal range</td>
<td>4000</td>
<td>0FA0</td>
</tr>
<tr>
<td></td>
<td>1500</td>
<td>05DC</td>
</tr>
<tr>
<td></td>
<td>700</td>
<td>02BC</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>0001</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>0000</td>
</tr>
</tbody>
</table>
### Output ranges voltage and current

The represented resolution corresponds to 16 bits.

<table>
<thead>
<tr>
<th>Range</th>
<th>Decimal Value</th>
<th>Hex.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Measured value too low</td>
<td>-501</td>
<td>FE0B</td>
</tr>
<tr>
<td></td>
<td>:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>-600</td>
<td>FDA8</td>
</tr>
<tr>
<td>Underflow</td>
<td>-32768</td>
<td>8000</td>
</tr>
</tbody>
</table>

#### Range

<table>
<thead>
<tr>
<th>Range</th>
<th>Digital value</th>
<th>Hex.</th>
</tr>
</thead>
<tbody>
<tr>
<td>-1</td>
<td>:</td>
<td>FFFF</td>
</tr>
<tr>
<td>:</td>
<td>-500</td>
<td>FE0C</td>
</tr>
<tr>
<td>Measured value too low</td>
<td>-501</td>
<td>FE0B</td>
</tr>
<tr>
<td>:</td>
<td>-600</td>
<td>FDA8</td>
</tr>
<tr>
<td>Underflow</td>
<td>-32768</td>
<td>8000</td>
</tr>
</tbody>
</table>

#### Output ranges voltage and current

<table>
<thead>
<tr>
<th>Range</th>
<th>-10...+10 V</th>
<th>0...20 mA</th>
<th>4...20 mA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Overflow</td>
<td>0 V</td>
<td>0 mA</td>
<td>0 mA</td>
</tr>
<tr>
<td>Measured value too high</td>
<td>11.7589 V</td>
<td>23.5178 mA</td>
<td>22.8142 mA</td>
</tr>
<tr>
<td></td>
<td>:</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>10.0004 V</td>
<td>20.0007 mA</td>
<td>20.0006 mA</td>
</tr>
<tr>
<td>Normal range</td>
<td>10.0000 V</td>
<td>20.0000 mA</td>
<td>20.0000 mA</td>
</tr>
<tr>
<td></td>
<td>:</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.0004 V</td>
<td>0.0007 mA</td>
<td>4.0006 mA</td>
</tr>
<tr>
<td></td>
<td>0.0000 V</td>
<td>0.0000 mA</td>
<td>4.0000 mA</td>
</tr>
<tr>
<td></td>
<td>-0.0004 V</td>
<td>0 mA</td>
<td>3.9994 mA</td>
</tr>
<tr>
<td></td>
<td>:</td>
<td></td>
<td>0 mA</td>
</tr>
<tr>
<td></td>
<td>-10.0000 V</td>
<td>0 mA</td>
<td>0 mA</td>
</tr>
<tr>
<td>Measured value too low</td>
<td>-10.0004 V</td>
<td>0 mA</td>
<td>0 mA</td>
</tr>
<tr>
<td></td>
<td>:</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>-11.7589 V</td>
<td>0 mA</td>
<td>0 mA</td>
</tr>
<tr>
<td>Underflow</td>
<td>0 V</td>
<td>0 mA</td>
<td>0 mA</td>
</tr>
</tbody>
</table>

#### Range

<table>
<thead>
<tr>
<th>Range</th>
<th>Digital value</th>
<th>Hex.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Overflow</td>
<td>&gt; 32511</td>
<td>&gt; 7EFF</td>
</tr>
<tr>
<td>Measured value too high</td>
<td>32511</td>
<td>7EFF</td>
</tr>
<tr>
<td></td>
<td>:</td>
<td>6C01</td>
</tr>
<tr>
<td></td>
<td>27649</td>
<td></td>
</tr>
<tr>
<td>Normal range</td>
<td>27648</td>
<td>6C00</td>
</tr>
<tr>
<td></td>
<td>:</td>
<td>0001</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>0000</td>
</tr>
</tbody>
</table>
### Range

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Digital value</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Decimal</strong></td>
<td><strong>Hex.</strong></td>
</tr>
<tr>
<td>-1</td>
<td>FFFF</td>
</tr>
<tr>
<td>-6912</td>
<td>E500</td>
</tr>
<tr>
<td>-27648</td>
<td>9400</td>
</tr>
<tr>
<td><strong>Measured value too low</strong></td>
<td></td>
</tr>
<tr>
<td>-27649</td>
<td>93FF</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>8100</td>
</tr>
<tr>
<td><strong>Underflow</strong></td>
<td></td>
</tr>
<tr>
<td>&lt; -32512</td>
<td>&lt; 8100</td>
</tr>
</tbody>
</table>

### Technical data

The system data of AC500 and S500 \(\text{\textcopyright} Chapter 1.5.4.5.1 \) “System data AC500” on page 3058 are valid for standard version.

The system data of AC500-XC \(\text{\textcopyright} Chapter 1.5.4.6.1 \) “System data AC500-XC” on page 3099 are valid for the XC version.

Only additional details are therefore documented below.

The technical data are also valid for the XC version.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Process voltage</strong></td>
<td></td>
</tr>
<tr>
<td>Connections</td>
<td>Terminals 1.8, 2.8, 3.8 and 4.8 for +24 V (UP) as well as 1.9, 2.9, 3.9 and 4.9 for 0 V (ZP)</td>
</tr>
<tr>
<td>Rated value</td>
<td>24 VDC</td>
</tr>
<tr>
<td>Max. ripple</td>
<td>5 %</td>
</tr>
<tr>
<td>Protection against reversed voltage</td>
<td>Yes</td>
</tr>
<tr>
<td>Rated protection fuse on UP</td>
<td>10 A fast</td>
</tr>
<tr>
<td>Galvanic isolation</td>
<td>Yes, per module</td>
</tr>
<tr>
<td><strong>Current consumption</strong></td>
<td></td>
</tr>
<tr>
<td>From 24 VDC power supply at the terminals UP/L+ and ZP/M of the CPU/bus module</td>
<td>Ca. 2 mA</td>
</tr>
<tr>
<td>From UP at normal operation</td>
<td>0.15 A + output loads</td>
</tr>
<tr>
<td>Inrush current from UP (at power up)</td>
<td>0.020 A^2s</td>
</tr>
<tr>
<td>Max. length of analog cables, conductor cross section &gt; 0.14 mm²</td>
<td>100 m</td>
</tr>
<tr>
<td>Weight</td>
<td>300 g</td>
</tr>
<tr>
<td>Mounting position</td>
<td>Horizontal or vertical with derating (output load reduced to 50 % at 40 °C per group)</td>
</tr>
<tr>
<td>Cooling</td>
<td>The natural convection cooling must not be hindered by cable ducts or other parts in the switch-gear cabinet.</td>
</tr>
</tbody>
</table>
NOTICE!
Attention:
All I/O channels (digital and analog) are protected against reverse polarity, reverse supply, short circuit and continuous overvoltage up to 30 VDC.

Technical data of the analog inputs

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of channels per module</td>
<td>4</td>
</tr>
<tr>
<td>Distribution of channels into groups</td>
<td>1 group of 4 channels</td>
</tr>
<tr>
<td>Connections of the channels I0- to I3-</td>
<td>Terminals 1.0 to 1.3</td>
</tr>
<tr>
<td>Connections of the channels I0+ to I3+</td>
<td>Terminals 2.0 to 2.3</td>
</tr>
<tr>
<td>Input type</td>
<td>Bipolar (not with current or Pt100/Pt1000/Ni1000)</td>
</tr>
<tr>
<td>Galvanic isolation</td>
<td>Against internal supply and other modules</td>
</tr>
<tr>
<td>Configurability</td>
<td>0 V...10 V, -10 V...+10 V, 0 mA...20 mA, 4 mA...20 mA, Pt100/1000, Ni1000 (each input can be configured individually)</td>
</tr>
<tr>
<td>Channel input resistance</td>
<td>Voltage: &gt; 100 kΩ</td>
</tr>
<tr>
<td></td>
<td>Current: ca. 330 Ω</td>
</tr>
<tr>
<td>Time constant of the input filter</td>
<td>Voltage: 100 µs</td>
</tr>
<tr>
<td></td>
<td>Current: 100 µs</td>
</tr>
<tr>
<td>Indication of the input signals</td>
<td>One LED per channel</td>
</tr>
<tr>
<td>Conversion cycle</td>
<td>2 ms (for 8 inputs + 8 outputs), with Pt/Ni... 1 s</td>
</tr>
<tr>
<td>Resolution</td>
<td>Range 0 V...10 V: 12 bits</td>
</tr>
<tr>
<td></td>
<td>Range -10 V...+10 V: 12 bits + sign</td>
</tr>
<tr>
<td></td>
<td>Range 0 mA...20 mA: 12 bits</td>
</tr>
<tr>
<td></td>
<td>Range 4 mA...20 mA: 12 bits</td>
</tr>
<tr>
<td>Conversion error of the analog values caused by non-linearity, adjustment error at factory and resolution within the normal range</td>
<td>Typ. ±0.5 % of full scale at 25 °C</td>
</tr>
<tr>
<td></td>
<td>Max. ±1 % of full scale (all ranges) at 0 °C...60 °C or EMC disturbance</td>
</tr>
<tr>
<td>Relationship between input signal and hex code</td>
<td>See tables Chapter 1.5.3.6.2.2.4.9.1 “Input ranges of voltage, current and digital input” on page 2686</td>
</tr>
<tr>
<td>Unused voltage inputs</td>
<td>Are configured as &quot;unused&quot;</td>
</tr>
<tr>
<td>Unused current inputs</td>
<td>Have a low resistance, can be left open-circuited</td>
</tr>
<tr>
<td>Overvoltage protection</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Technical data of the analog inputs, if used as digital inputs

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of channels per module</td>
<td>Max. 4</td>
</tr>
<tr>
<td>Distribution of channels into groups</td>
<td>1 group of 4 channels</td>
</tr>
<tr>
<td>Parameter</td>
<td>Value</td>
</tr>
<tr>
<td>-----------------------------------------------</td>
<td>--------------------------------------------</td>
</tr>
<tr>
<td>Connections of the channels I0+ to I3+</td>
<td>Terminals 2.0 to 2.3</td>
</tr>
<tr>
<td>Reference potential for the inputs</td>
<td>Terminals 1.9, 2.9, 3.9 and 4.9 (ZP)</td>
</tr>
<tr>
<td>Input signal delay</td>
<td>Typ. 8 ms, configurable from 0.1 to 32 ms</td>
</tr>
<tr>
<td>Indication of the input signals</td>
<td>1 LED per channel</td>
</tr>
<tr>
<td>Input signal voltage</td>
<td>24 VDC</td>
</tr>
<tr>
<td>Signal 0</td>
<td>-30 V...+5 V</td>
</tr>
<tr>
<td>Undefined signal</td>
<td>+5 V...+13 V</td>
</tr>
<tr>
<td>Signal 1</td>
<td>+13 V...+30 V</td>
</tr>
<tr>
<td>Input current per channel</td>
<td></td>
</tr>
<tr>
<td>Input voltage +24 V</td>
<td>Typ. 7 mA</td>
</tr>
<tr>
<td>Input voltage +5 V</td>
<td>Typ. 1.4 mA</td>
</tr>
<tr>
<td>Input voltage +15 V</td>
<td>Typ. 4.3 mA</td>
</tr>
<tr>
<td>Input voltage +30 V</td>
<td>&lt; 9 mA</td>
</tr>
<tr>
<td>Input resistance</td>
<td>ca. 3.5 kΩ</td>
</tr>
</tbody>
</table>

Technical data of the analog outputs

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of channels per module</td>
<td>4, all channels for voltage and current</td>
</tr>
<tr>
<td>Distribution of channels into groups</td>
<td>1 group of 4 channels</td>
</tr>
<tr>
<td>Channels O0-...O3-</td>
<td>Terminals 3.0...3.3</td>
</tr>
<tr>
<td>Channels O0+...O3+</td>
<td>Terminals 4.0...4.3</td>
</tr>
<tr>
<td>Output type</td>
<td>Bipolar with voltage, unipolar with current</td>
</tr>
<tr>
<td>Galvanic isolation</td>
<td>Against internal supply and other modules</td>
</tr>
<tr>
<td>Configurability</td>
<td>-10 V...+10 V, 0 mA...20 mA, 4 mA...20 mA (each output can be configured individually), current outputs only channels 0...3</td>
</tr>
<tr>
<td>Output resistance (load), as current output</td>
<td>0 Ω...500 Ω</td>
</tr>
<tr>
<td>Output loadability, as voltage output</td>
<td>Max. ±10 mA</td>
</tr>
<tr>
<td>Indication of the output signals</td>
<td>One LED per channel</td>
</tr>
<tr>
<td>Resolution</td>
<td>12 bits (+ sign)</td>
</tr>
<tr>
<td>Conversion error of the analog values caused by non-linearity, adjustment error at factory and resolution within the normal range</td>
<td>Typ. ±0.5 % of full scale at 25 °C</td>
</tr>
<tr>
<td></td>
<td>Max. ±1 % of full scale (all ranges) at 0 °C...60 °C or EMC disturbance</td>
</tr>
<tr>
<td>Relationship between output signal and hex code</td>
<td>See table \ Chapter 1.5.3.6.2.2.4.9.3 “Output ranges voltage and current” on page 2689</td>
</tr>
<tr>
<td>Unused outputs</td>
<td>Can be left open-circuited</td>
</tr>
</tbody>
</table>
Ordering Data

<table>
<thead>
<tr>
<th>Part no.</th>
<th>Description</th>
<th>Product life cycle phase *)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1SAP 250 100 R0001</td>
<td>AX521, analog input/output module, 4 AI, 4 AO, U/I/Pt100, 12 bits + sign, 2-wires</td>
<td>Active</td>
</tr>
<tr>
<td>1SAP 450 100 R0001</td>
<td>AX521-XC, analog input/output module, 4 AI, 4 AO, U/I/Pt100, 12 bits + sign, 2-wires, XC version</td>
<td>Active</td>
</tr>
</tbody>
</table>

*) For planning and commissioning of new installations use modules in Active status only.

AX522 - Analog input/output module

- 8 configurable analog inputs (I0 to I7) in 1 group (1.0...2.7)
  Resolution 12 bits plus sign
- 8 configurable analog outputs (O0 to O7) in 1 group (3.0...4.7)
  Resolution 12 bits plus sign
- Module-wise electrically isolated
- XC version for use in extreme ambient conditions available
1. I/O bus
2. Allocation between terminal number and signal name
3. 8 yellow LEDs to display the signal states at the analog inputs (I0 - I7)
4. 8 yellow LEDs to display the signal states at the analog outputs (O0 - O7)
5. 1 green LED to display the state of the process supply voltage UP
6. 2 red LEDs to display errors
7. Label
8. Terminal unit
9. DIN rail
10. Sign for XC version

**Intended purpose**

The device can be used as a decentralized I/O extension module for S500 Communication Interface Modules (e.g. CI592-CS31, CI501-PNIO, CI541-DP, CI581-CN) or as a centralized extension module for AC500 CPUs (PM5xx).

**Functionality**

8 analog inputs, individually configurable for
- Unused (default setting)
- 0 V...10 V
- -10 V...+10 V
- 0 mA...20 mA
- 4 mA...20 mA
- Pt100, -50 °C...+400 °C (2-wire)
- Pt100, -50 °C...+400 °C (3-wire), requires 2 channels
- Pt100, -50 °C...+70 °C (2-wire)
- Pt100, -50 °C...+70 °C (3-wire), requires 2 channels
- Pt1000, -50 °C...+400 °C (2-wire)
- Pt1000, -50 °C...+400 °C (3-wire), requires 2 channels
- Ni1000, -50 °C...+150 °C (2-wire)
- Ni1000, -50 °C...+150 °C (3-wire), requires 2 channels
- 0 V...+10 V with differential inputs, requires 2 channels
- -10 V...+10 V with differential inputs, requires 2 channels
- Digital signals (digital input)

4 analog outputs, individually configurable for
- Unused (default setting)
- -10 V...+10 V
- 0 mA...20 mA
- 4 mA...20 mA

4 analog outputs, individually configurable for
- Unused (default setting)
- -10 V...+10 V

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Resolution of the analog channels</td>
<td></td>
</tr>
<tr>
<td>Voltage -10 V...+10 V</td>
<td>12 bits plus sign</td>
</tr>
<tr>
<td>Voltage 0 V...10 V</td>
<td>12 bits</td>
</tr>
<tr>
<td>Current 0 mA...20 mA, 4 mA...20 mA</td>
<td>12 bits</td>
</tr>
<tr>
<td>Temperature</td>
<td>0.1 °C</td>
</tr>
<tr>
<td>LED displays</td>
<td>19 LEDs for signals and error messages</td>
</tr>
<tr>
<td>Internal power supply</td>
<td>Via the expansion bus interface (I/O bus)</td>
</tr>
<tr>
<td>External power supply</td>
<td>Via the terminals ZP and UP (process voltage 24 VDC)</td>
</tr>
<tr>
<td>Required terminal unit</td>
<td>TU515 or TU516 &quot;Chapter 1.5.3.5.2 “TU515, TU516, TU541 and TU542 for I/O modules” on page 2311&quot;</td>
</tr>
</tbody>
</table>

Electrical Connection

For a detailed description of the mounting, disassembly and electrical connection of the module, please refer to the System Assembly, Construction and Connection chapter "Chapter 1.5.4.5 “AC500 (Standard)” on page 3058.

The modules are plugged on an I/O terminal unit "Chapter 1.5.3.5.2 “TU515, TU516, TU541 and TU542 for I/O modules” on page 2311. Properly position the modules and press until they lock in place. The terminal units are mounted on a DIN rail or with 2 screws plus the additional accessory for wall mounting (TA526 "Chapter 1.5.3.8.1.4 “TA526 - Wall mounting accessory” on page 3035).
The electrical connection of the I/O channels is carried out using the 40 terminals of the I/O terminal unit. I/O modules can be replaced without re-wiring the terminal units.

The terminals 1.8, 2.8, 3.8 and 4.8 as well as 1.9, 2.9, 3.9 and 4.9 are electrically interconnected within the I/O terminal units and always have the same assignment, independent of the inserted module:

Terminals 1.8, 2.8, 3.8 and 4.8: process voltage UP = +24 VDC
Terminals 1.9, 2.9, 3.9 and 4.9: process voltage ZP = 0 VDC

The assignment of the other terminals:

<table>
<thead>
<tr>
<th>Terminals</th>
<th>Signal</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0 to 1.7</td>
<td>I0- to I7-</td>
<td>Negative poles of the 8 analog inputs</td>
</tr>
<tr>
<td>2.0 to 2.7</td>
<td>I0+ to I7+</td>
<td>Positive poles of the 8 analog inputs</td>
</tr>
<tr>
<td>3.0 to 3.7</td>
<td>O0- to O7-</td>
<td>Negative poles of the 8 analog outputs</td>
</tr>
<tr>
<td>4.0 to 4.7</td>
<td>O0+ to O7+</td>
<td>Positive poles of the 8 analog outputs</td>
</tr>
</tbody>
</table>

The negative poles of the analog inputs are electrically connected to each other to form an "Analog Ground" signal for the module.

The negative poles of the analog outputs are electrically connected to each other to form an "Analog Ground" signal for the module.

There is no galvanic isolation between the analog circuitry and ZP/UP. Therefore, the analog sensors must be galvanically isolated in order to avoid loops via the earth potential or the supply voltage.

Because of their common reference potential, analog current inputs cannot be circuited in series, neither within the module nor with channels of other modules.

For the open-circuit detection (cut wire), each analog input channel is pulled up to "plus" by a high-resistance resistor. If nothing is connected, the maximum voltage will be read in then.

The internal power supply voltage for the module's circuitry is carried out via the I/O bus (provided by a bus module or a CPU). Thus, the current consumption from 24 VDC power supply at the terminals L+/UP and M/ZP of the CPU/bus module increases by 2 mA per I/O module.

The external power supply connection is carried out via the UP (+24 VDC) and the ZP (0 VDC) terminals.
NOTICE!
Risk of damaging the PLC modules!
Overvoltages and short circuits might damage the PLC modules.
- Make sure that all voltage sources (supply voltage and process supply voltage) are switched off before you begin with operations at the system.
- Never connect any voltages or signals to reserved terminals (marked with ---). Reserved terminals may carry internal voltages.

Generally, analog signals must be laid in shielded cables. The cable shields must be earthed at both sides of the cables. In order to avoid unacceptable potential differences between different parts of the installation, low resistance equipotential bonding conductors must be laid.

Only for simple applications (low electromagnetic disturbances, no high requirement on precision), the shielding can also be omitted.

The following figure shows the electrical connection of the I/O module.

Fig. 154: Terminal assignment

Attention:
By installing equipotential bonding conductors between the different parts of the system, it must be made sure that the potential difference between ZP and AGND never can exceed 1 V.

Attention:
The process voltage must be included in the earthing concept of the control system (e.g. earthing the minus pole).
Connection of resistance thermometers in 2-wire configuration

When resistance thermometers (Pt100, Pt1000, Ni1000) are used, a constant current must flow through them to build the necessary voltage drop for the evaluation. For this, the I/O module provides a constant current source which is multiplexed over the 8 analog channels.

![Connection Example Diagram]

*Fig. 155: Connection example*

<table>
<thead>
<tr>
<th>Thermometer</th>
<th>Temperature Range</th>
<th>Configuration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pt100</td>
<td>-50 °C...+70 °C</td>
<td>2-wire used</td>
</tr>
<tr>
<td>Pt100</td>
<td>-50 °C...+400 °C</td>
<td>2-wire used</td>
</tr>
<tr>
<td>Pt1000</td>
<td>-50 °C...+400 °C</td>
<td>2-wire used</td>
</tr>
<tr>
<td>Ni1000</td>
<td>-50 °C...+150 °C</td>
<td>2-wire used</td>
</tr>
</tbody>
</table>

The I/O module performs a linearization of the resistance characteristic.

In order to avoid error messages from unused analog input channels, it is useful to configure them as "unused".

Connection of resistance thermometers in 3-wire configuration

When resistance thermometers (Pt100, Pt1000, Ni1000) are used, a constant current must flow through them to build the necessary voltage drop for the evaluation. For this, the I/O module provides a constant current source which is multiplexed over the max. 8 (depending on the configuration) analog channels.
If several measuring points are adjacent to each other, only one return line is necessary. This saves wiring costs.

With the 3-wire configuration, two adjacent analog channels belong together (e.g. the channels 0 and 1). In this case, both channels are configured according to the desired operating mode. The lower address must be the even address (channel 0), the next higher address must be the odd address (channel 1).

The constant current of one channel flows through the resistance thermometer. The constant current of the other channel flows through one of the cores. The module calculates the measured value from the two voltage drops and stores it under the input with the higher channel number (e.g. I1).

In order to keep measuring errors as small as possible, it is necessary to have all the involved conductors in the same cable. All the conductors must have the same cross section.

<table>
<thead>
<tr>
<th>Sensor</th>
<th>Temperature Range</th>
<th>Configuration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pt100</td>
<td>-50 °C...+70 °C</td>
<td>3-wire configuration, two channels used</td>
</tr>
<tr>
<td>Pt100</td>
<td>-50 °C...+400 °C</td>
<td>3-wire configuration, two channels used</td>
</tr>
<tr>
<td>Pt1000</td>
<td>-50 °C...+400 °C</td>
<td>3-wire configuration, two channels used</td>
</tr>
<tr>
<td>Ni1000</td>
<td>-50 °C...+150 °C</td>
<td>3-wire configuration, two channels used</td>
</tr>
</tbody>
</table>

The I/O module performs a linearization of the resistance characteristic.

In order to avoid error messages from unused analog input channels, it is useful to configure them as "unused".
Connection of active-type analog sensors (Voltage) with electrically isolated power supply

![Connection diagram](image)

Fig. 157: Connection example

By connecting the sensor's negative pole of the output voltage to AGND, the electrically isolated voltage source of the sensor is referred to ZP.

The following measuring ranges can be configured for AX521 “Parameterization” on page 2680 and for AX522 “Parameterization” on page 2705:

<table>
<thead>
<tr>
<th>Voltage</th>
<th>0 V...10 V</th>
<th>1 channel used</th>
</tr>
</thead>
<tbody>
<tr>
<td>Voltage</td>
<td>-10 V...+10 V</td>
<td>1 channel used</td>
</tr>
</tbody>
</table>

In order to avoid error messages or long processing times, it is useful to configure unused analog input channels as "unused".

Connection of active-type analog sensors (Current) with electrically isolated power supply

![Connection diagram](image)

Fig. 158: Connection example
Current
0 mA...20 mA 1 channel used
Current
4 mA...20 mA 1 channel used

Unused input channels can be left open-circuited, because they are of low resistance.

Connection of active-type analog sensors (Voltage) with no electrically isolated power supply

![Connection diagram](image)

**Fig. 159: Connection example**

**CAUTION!**
The potential difference between AGND and ZP at the module must not be greater than 1V, not even in case of long lines (see figure Terminal Assignment).

---

*If AGND does not get connected to ZP, the sensor current flows to ZP via the AGND line. The measuring signal is distorted, as a very small current flows through the voltage line. The total current through the PTC should not exceed 50 mA. This measuring method is therefore only suitable for short lines and small sensor currents. If there are bigger distances, the difference measuring method should be applied.*

---

<table>
<thead>
<tr>
<th>Voltage</th>
<th>0 V...10 V</th>
<th>1 channel used</th>
</tr>
</thead>
<tbody>
<tr>
<td>Voltage</td>
<td>-10 V...+10 V *)</td>
<td>1 channel used</td>
</tr>
</tbody>
</table>

*) if the sensor can provide this signal range

In order to avoid error messages or long processing times, it is useful to configure unused analog input channels as "unused".
Connection of passive-type analog sensors (Current)

![Connection example diagram]

**Fig. 160: Connection example**

<table>
<thead>
<tr>
<th>Current</th>
<th>4 mA...20 mA</th>
<th>1 channel used</th>
</tr>
</thead>
</table>

**CAUTION!**
If, during initialization, an analog current sensor supplies more than 25 mA for more than 1 second to an analog input, this input is switched off by the module (input protection). In such cases, it is recommended to protect the analog input by a 10-volt Zener diode (in parallel to I+ and I-). But, in general, sensors with fast initialization or without current peaks higher than 25 mA are preferrable.

Unused input channels can be left open-circuited because they are of low resistance.

Connection of active-type analog sensors (Voltage) to differential inputs

Differential inputs are very useful if analog sensors are used which are remotely non-isolated (e.g. the minus terminal is remotely earthed).

The use of differential inputs helps to considerably increase the measuring accuracy and to avoid earthing loops.

With differential input configurations, two adjacent analog channels belong together (e.g. the channels 0 and 1). In this case, both channels are configured according to the desired operating mode. The lower address must be the even address (channel 0), the next higher address must be the odd address (channel 1). The converted analog value is available at the higher address (channel 1).

The analog value is calculated by subtraction of the input value with the higher address from the input value of the lower address.

The converted analog value is available at the odd channel (higher address).

**CAUTION!**
The earthing potential at the sensors must not have too large a potential difference with respect to ZP (max. ±1 V within the full signal range). Otherwise, problems may occur concerning the common-mode input voltages of the involved analog inputs.
The negative pole of the sensor must be earthed next to the sensor.

<table>
<thead>
<tr>
<th>Voltage</th>
<th>0 V...10 V</th>
<th>with differential inputs, 2 channels used</th>
</tr>
</thead>
<tbody>
<tr>
<td>Voltage</td>
<td>-10 V...+10 V</td>
<td>with differential inputs, 2 channels used</td>
</tr>
</tbody>
</table>

In order to avoid error messages or long processing times, it is useful to configure unused analog input channels as "unused".

**Use of analog inputs as digital inputs**

Several (or all) analog inputs can be configured as digital inputs. The inputs are not electrically isolated against the other analog channels.
<table>
<thead>
<tr>
<th>Digital input</th>
<th>24 V</th>
<th>1 channel used</th>
</tr>
</thead>
<tbody>
<tr>
<td>Effect of incorrect input terminal connection</td>
<td>Wrong or no signal detected, no damage up to 35 V</td>
<td></td>
</tr>
</tbody>
</table>

**Connection of analog output loads (Voltage, current)**

**Fig. 163: Connection example**

<table>
<thead>
<tr>
<th>Voltage</th>
<th>-10 V...+10 V</th>
<th>Load max. ±10 mA</th>
<th>1 channel used</th>
</tr>
</thead>
<tbody>
<tr>
<td>Current</td>
<td>0 mA...20 mA</td>
<td>Load 0 Ω...500 Ω</td>
<td>1 channel used</td>
</tr>
<tr>
<td>Current</td>
<td>4 mA...20 mA</td>
<td>Load 0 Ω...500 Ω</td>
<td>1 channel used</td>
</tr>
</tbody>
</table>

Only the channels 0...3 can be configured as current output (0 mA...20 mA or 4 mA...20 mA). Unused analog outputs can be left open-circuited.

**Internal data exchange**

<table>
<thead>
<tr>
<th>Internal data exchange</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Digital inputs (bytes)</td>
<td>0</td>
</tr>
<tr>
<td>Digital outputs (bytes)</td>
<td>0</td>
</tr>
<tr>
<td>Counter input data (words)</td>
<td>8</td>
</tr>
<tr>
<td>Counter output data (words)</td>
<td>8</td>
</tr>
</tbody>
</table>

**I/O configuration**

The module does not store configuration data itself. It gets its parameterization data from the master device of the I/O bus (CPU or bus module) during power-up of the system.

Hence, replacing I/O modules is possible without any re-parameterization via software.
Parameterization

The parameter data directly influences the functionality of modules.
For non-standard applications, it is necessary to adapt the parameters to your system configuration.

Module: Module slot address: Y = 1...10

Module slot address: Y = 1...7

<table>
<thead>
<tr>
<th>No.</th>
<th>Name</th>
<th>Value</th>
<th>Internal value</th>
<th>Internal value, type</th>
<th>Default</th>
<th>Min.</th>
<th>Max.</th>
<th>EDS Slot/Index</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Module ID</td>
<td>Internal</td>
<td>1500</td>
<td>Word</td>
<td>1500</td>
<td>0</td>
<td>65535</td>
<td>0x0Y01</td>
</tr>
<tr>
<td>2</td>
<td>Ignore module</td>
<td>No</td>
<td>0</td>
<td>Byte</td>
<td>No</td>
<td>0</td>
<td>not for FBP</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Yes</td>
<td>1</td>
<td>Byte</td>
<td>0</td>
<td>1</td>
<td>0x00</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Parameter length in</td>
<td>Internal</td>
<td>37</td>
<td>Byte</td>
<td>37-CPU</td>
<td>0</td>
<td>255</td>
<td>0x0Y02</td>
</tr>
<tr>
<td></td>
<td>bytes</td>
<td></td>
<td></td>
<td></td>
<td>37-FBP</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Check supply</td>
<td>Off</td>
<td>0</td>
<td>Byte</td>
<td>On</td>
<td>1</td>
<td>0x01</td>
<td>0x0Y03</td>
</tr>
<tr>
<td></td>
<td></td>
<td>On</td>
<td>1</td>
<td>Byte</td>
<td>0</td>
<td>1</td>
<td>0x00</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Analog data format</td>
<td>Default</td>
<td>0</td>
<td>Byte</td>
<td>Default</td>
<td>0</td>
<td>3</td>
<td>0x0Y04</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0x00</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>Behaviour of outputs</td>
<td>Off</td>
<td>0</td>
<td>Byte</td>
<td>Off</td>
<td>0</td>
<td>2</td>
<td>0x0Y05</td>
</tr>
<tr>
<td></td>
<td>at communication</td>
<td></td>
<td>1</td>
<td>Byte</td>
<td>0x00</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>errors</td>
<td></td>
<td>1+(n*5)</td>
<td></td>
<td>n ≤ 2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>Channel configuration</td>
<td>see table</td>
<td>Channel configuration</td>
<td>Byte</td>
<td>Default 0x00</td>
<td>0</td>
<td>19</td>
<td>0x0Y06</td>
</tr>
<tr>
<td></td>
<td>Input channel 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>Channel monitoring</td>
<td>see table</td>
<td>Channel monitoring</td>
<td>Byte</td>
<td>Default 0x00</td>
<td>0</td>
<td>3</td>
<td>0x0Y07</td>
</tr>
<tr>
<td></td>
<td>Input channel 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>No.</td>
<td>Name</td>
<td>Value</td>
<td>Internal value, type</td>
<td>Default</td>
<td>Min.</td>
<td>Max.</td>
<td>EDS Slot/Index</td>
<td></td>
</tr>
<tr>
<td>-----</td>
<td>-----------------------------------------------------------</td>
<td>--------------------------------</td>
<td>----------------------</td>
<td>---------</td>
<td>------</td>
<td>------</td>
<td>-----------------</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>Channel configuration and channel monitoring of the input</td>
<td>see tables</td>
<td>Byte</td>
<td>Default</td>
<td>0</td>
<td>19</td>
<td>0x0Y08</td>
<td></td>
</tr>
<tr>
<td></td>
<td>channels 1 to 7</td>
<td>channel configuration and channel monitoring</td>
<td>Byte</td>
<td>0x00</td>
<td>0</td>
<td>3</td>
<td>to</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0x00</td>
<td></td>
<td></td>
<td>0x0Y15</td>
<td></td>
</tr>
<tr>
<td>23</td>
<td>Channel configuration</td>
<td>see table</td>
<td>Byte</td>
<td>Default</td>
<td>0</td>
<td>130</td>
<td>0x0Y16</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Output channel 0</td>
<td>Channel configuration</td>
<td></td>
<td>0x00</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>24</td>
<td>Channel monitoring</td>
<td>see table</td>
<td>Byte</td>
<td>Default</td>
<td>0</td>
<td>3</td>
<td>0x0Y17</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Output channel 0</td>
<td>Channel monitoring</td>
<td></td>
<td>0x00</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>25</td>
<td>Substitute value</td>
<td>only valid for output channel 0</td>
<td>Word</td>
<td>Default</td>
<td>0</td>
<td>65535</td>
<td>0x0Y18</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Output channel 0</td>
<td></td>
<td></td>
<td>0x0000</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>26</td>
<td>Channel configuration</td>
<td>see tables</td>
<td>Byte</td>
<td>Default</td>
<td>0</td>
<td>130</td>
<td>0x0Y19</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Output channel 1 to 3</td>
<td>channel configuration and channel monitoring</td>
<td>Byte</td>
<td>0x00</td>
<td>0</td>
<td>3</td>
<td>to</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0x00</td>
<td></td>
<td></td>
<td>0x0Y1E</td>
<td></td>
</tr>
<tr>
<td>32</td>
<td>Channel configuration</td>
<td>see table</td>
<td>Byte</td>
<td>Default</td>
<td>0</td>
<td>128</td>
<td>0x0Y1F</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Output channel 4</td>
<td>Channel configuration</td>
<td></td>
<td>0x00</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>No.</td>
<td>Name</td>
<td>Value</td>
<td>Internal value, type</td>
<td>Default</td>
<td>Min.</td>
<td>Max.</td>
<td>EDS Slot/Index</td>
<td></td>
</tr>
<tr>
<td>-----</td>
<td>-------------------------------------------</td>
<td>----------------</td>
<td>---------------------</td>
<td>---------</td>
<td>------</td>
<td>------</td>
<td>----------------</td>
<td></td>
</tr>
<tr>
<td>33</td>
<td>Channel monitoring Output channel 4</td>
<td>see table</td>
<td>Channel monitoring</td>
<td>Byte</td>
<td>0x00</td>
<td>3</td>
<td>0x0Y20</td>
<td></td>
</tr>
<tr>
<td>34</td>
<td>Channel configuration and channel</td>
<td>see tables</td>
<td>channel configuration and channel monitoring</td>
<td>Byte</td>
<td>0x00</td>
<td>128</td>
<td>0x0Y21</td>
<td></td>
</tr>
<tr>
<td></td>
<td>monitoring of the output channels 5 to 7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0x0Y26</td>
<td></td>
</tr>
</tbody>
</table>

1) With CS31 and addresses less than 70 and FBP, the value is increased by 1
2) Not with FBP

GSD file:

```plaintext
Ext_User_Prm_Data_Len = 24
Ext_User_Prm_Data_Const(0) = 0x05, 0xe2, 0x15, 0x01, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00;```

Table 575: Input channel (4x)

<table>
<thead>
<tr>
<th>No.</th>
<th>Name</th>
<th>Internal value, type</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Channel configuration see table 2)</td>
<td>Byte</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0x00 see table 2)</td>
</tr>
<tr>
<td>2</td>
<td>Channel monitoring see table 3)</td>
<td>Byte</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0x00 see table 3)</td>
</tr>
</tbody>
</table>

Table 576: Channel configuration 2)

<table>
<thead>
<tr>
<th>Internal value</th>
<th>Operating modes of the analog inputs, individually configurable</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Unused (default)</td>
</tr>
<tr>
<td>1</td>
<td>Analog input 0 V...10 V</td>
</tr>
<tr>
<td>2</td>
<td>Digital input</td>
</tr>
<tr>
<td>3</td>
<td>Analog input 0 mA...20 mA</td>
</tr>
</tbody>
</table>
Operating modes of the analog inputs, individually configurable

<table>
<thead>
<tr>
<th>Internal value</th>
<th>Analog input 4 mA...20 mA</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>Analog input -10 V...+10 V</td>
</tr>
<tr>
<td>8</td>
<td>Analog input Pt100, -50 °C...+400 °C (2-wire)</td>
</tr>
<tr>
<td>9</td>
<td>Analog input Pt100, -50 °C...+400 °C (3-wire), requires 2 channels *)</td>
</tr>
<tr>
<td>10</td>
<td>Analog input 0...10 V via differential inputs, requires 2 channels *)</td>
</tr>
<tr>
<td>11</td>
<td>Analog input -10 V...+10 V via differential inputs, requires 2 channels *)</td>
</tr>
<tr>
<td>14</td>
<td>Analog input Pt100, -50 °C...+70 °C (2-wire)</td>
</tr>
<tr>
<td>15</td>
<td>Analog input Pt100, -50 °C...+70 °C (3-wire), requires 2 channels *)</td>
</tr>
<tr>
<td>16</td>
<td>Analog input Pt1000, -50 °C...+400 °C (2-wire)</td>
</tr>
<tr>
<td>17</td>
<td>Analog input Pt1000, -50 °C...+400 °C (3-wire), requires 2 channels *)</td>
</tr>
<tr>
<td>18</td>
<td>Analog input Ni1000, -50 °C...+150 °C (2-wire)</td>
</tr>
<tr>
<td>19</td>
<td>Analog input Ni1000, -50 °C...+150 °C (3-wire), requires 2 channels *)</td>
</tr>
</tbody>
</table>

*) In the operating modes with 3-wire configuration or with differential inputs, two adjacent analog inputs belong together (e.g. the channels 0 and 1). In these cases, both channels are configured in the desired operating mode. The lower address must be the even address (channel 0). The next higher address must be the odd address (channel 1). The converted analog value is available at the higher address (channel 1).

Table 577: Channel monitoring 3)

<table>
<thead>
<tr>
<th>Internal value</th>
<th>Monitoring</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Plausibility, open-circuit (broken wire) and short circuit</td>
</tr>
<tr>
<td>3</td>
<td>No monitoring</td>
</tr>
</tbody>
</table>

Table 578: Output channel 0 (1 channel)

<table>
<thead>
<tr>
<th>No.</th>
<th>Name</th>
<th>Value</th>
<th>Internal value, type</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Channel configuration</td>
<td>see table 4)</td>
<td>see table 4)</td>
<td>Byte</td>
</tr>
<tr>
<td>2</td>
<td>Channel monitoring</td>
<td>see table 5)</td>
<td>see table 5)</td>
<td>Byte</td>
</tr>
<tr>
<td>3</td>
<td>Substitute value</td>
<td>0...65535</td>
<td>0...0xffff</td>
<td>Word</td>
</tr>
</tbody>
</table>

Table 579: Output channels 1...3 (3x)

<table>
<thead>
<tr>
<th>No.</th>
<th>Name</th>
<th>Internal value, type</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Channel configuration</td>
<td>Byte</td>
</tr>
<tr>
<td>2</td>
<td>Channel monitoring</td>
<td>Byte</td>
</tr>
</tbody>
</table>
### Table 580: Channel configuration

<table>
<thead>
<tr>
<th>Internal value</th>
<th>Operating modes of the analog outputs, individually configurable</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Unused (default)</td>
</tr>
<tr>
<td>128</td>
<td>Analog output -10 V...+10 V</td>
</tr>
<tr>
<td>129</td>
<td>Analog output 0 mA...20 mA (not with the channels 4...7 and 12...15)</td>
</tr>
<tr>
<td>130</td>
<td>Analog output 4 mA...20 mA (not with the channels 4...7 and 12...15)</td>
</tr>
</tbody>
</table>

### Table 581: Channel monitoring

<table>
<thead>
<tr>
<th>Internal value</th>
<th>Monitoring</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Plausibility, open circuit (broken wire) and short circuit (default)</td>
</tr>
<tr>
<td>3</td>
<td>No monitoring</td>
</tr>
</tbody>
</table>

### Table 582: Substitute value

<table>
<thead>
<tr>
<th>Intended behaviour of output channel when the control system stops</th>
<th>Required setting of the module parameter &quot;Behaviour of outputs in case of a communication error&quot;</th>
<th>Required setting of the channel parameter &quot;Substitute value&quot;</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output OFF</td>
<td>Off</td>
<td>0</td>
</tr>
<tr>
<td>Last value infinite</td>
<td>Last value</td>
<td>0</td>
</tr>
<tr>
<td>Last value for 5 s and then turn off</td>
<td>Last value 5 sec</td>
<td>0</td>
</tr>
<tr>
<td>Last value for 10 s and then turn off</td>
<td>Last value 10 sec</td>
<td>0</td>
</tr>
<tr>
<td>Substitute value infinite</td>
<td>Substitute value</td>
<td>Depending on configuration</td>
</tr>
<tr>
<td>Substitute value for 5 s and then turn off</td>
<td>Substitute value 5 sec</td>
<td>Depending on configuration</td>
</tr>
<tr>
<td>Substitute value for 10 s and then turn off</td>
<td>Substitute value 10 sec</td>
<td>Depending on configuration</td>
</tr>
</tbody>
</table>

### Diagnosis

<table>
<thead>
<tr>
<th>E1...E4</th>
<th>d1</th>
<th>d2</th>
<th>d3</th>
<th>d4</th>
<th>Identifier</th>
<th>AC500 display</th>
<th>← Display in</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Class</th>
<th>Comp</th>
<th>Dev</th>
<th>Mod</th>
<th>Ch</th>
<th>Err</th>
<th>PS501 PLC browser</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Byte 6 Bit 6...7</th>
<th>Interface</th>
<th>Device</th>
<th>Module</th>
<th>Channel</th>
<th>Error Identifier</th>
<th>Error message</th>
<th>Remedy</th>
</tr>
</thead>
<tbody>
<tr>
<td>-</td>
<td></td>
<td>Byte 3</td>
<td>Byte 4</td>
<td>Byte 5</td>
<td>Byte 6 Bit 0...5</td>
<td>FBP diagnosis block</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Class</th>
<th>Error</th>
<th>Identifier</th>
<th>Error message</th>
<th>Remedy</th>
</tr>
</thead>
<tbody>
<tr>
<td>1)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Module error</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>14</td>
<td>1...10</td>
<td>31</td>
<td>31</td>
<td>19</td>
<td>Checksum error in the I/O module</td>
</tr>
<tr>
<td>11 / 12</td>
<td>ADR</td>
<td>1...10</td>
<td></td>
<td></td>
<td></td>
<td>Replace I/O module</td>
</tr>
<tr>
<td>Class</td>
<td>Interface</td>
<td>Device</td>
<td>Module</td>
<td>Channel</td>
<td>Error Identifier</td>
<td>Error message</td>
</tr>
<tr>
<td>-------</td>
<td>-----------</td>
<td>--------</td>
<td>--------</td>
<td>---------</td>
<td>-----------------</td>
<td>---------------</td>
</tr>
<tr>
<td>1)</td>
<td>2)</td>
<td>3)</td>
<td>4)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>14</td>
<td>1...10</td>
<td>31</td>
<td>31</td>
<td>3</td>
<td>Timeout in the I/O module</td>
</tr>
<tr>
<td></td>
<td>11 / 12</td>
<td>ADR</td>
<td>1...10</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>14</td>
<td>1...10</td>
<td>31</td>
<td>31</td>
<td>40</td>
<td>Different hard-/firmware versions in the module</td>
</tr>
<tr>
<td></td>
<td>11 / 12</td>
<td>ADR</td>
<td>1...10</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>14</td>
<td>1...10</td>
<td>31</td>
<td>31</td>
<td>43</td>
<td>Internal error in the module</td>
</tr>
<tr>
<td></td>
<td>11 / 12</td>
<td>ADR</td>
<td>1...10</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>14</td>
<td>1...10</td>
<td>31</td>
<td>31</td>
<td>36</td>
<td>Internal data exchange failure</td>
</tr>
<tr>
<td></td>
<td>11 / 12</td>
<td>ADR</td>
<td>1...10</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>14</td>
<td>1...10</td>
<td>31</td>
<td>31</td>
<td>9</td>
<td>Overflow diagnosis buffer</td>
</tr>
<tr>
<td></td>
<td>11 / 12</td>
<td>ADR</td>
<td>1...10</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>14</td>
<td>1...10</td>
<td>31</td>
<td>31</td>
<td>26</td>
<td>Parameter error</td>
</tr>
<tr>
<td></td>
<td>11 / 12</td>
<td>ADR</td>
<td>1...10</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>14</td>
<td>1...10</td>
<td>31</td>
<td>31</td>
<td>11</td>
<td>Process voltage too low</td>
</tr>
<tr>
<td></td>
<td>11 / 12</td>
<td>ADR</td>
<td>1...10</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>14</td>
<td>1...10</td>
<td>31</td>
<td>31</td>
<td>45</td>
<td>Process voltage is switched off (ON -&gt; OFF)</td>
</tr>
<tr>
<td></td>
<td>11 / 12</td>
<td>ADR</td>
<td>1...10</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Remarks:**

Analog value overflow or broken wire at an analog input
Check input value or terminal

Analog value underflow at an analog input
Check input value

Short circuit at an analog input
Check terminal

Analog value overflow at an analog output
Check output value

Analog value underflow at an analog output
Check output value
In AC500, the following interface identifier applies:
14 = I/O bus, 11 = COM1 (e.g. CS31 bus), 12 = COM2.
The FBP diagnosis block does not contain this identifier.

With "Device" the following allocation applies:
31 = module itself, 1...10 = decentralized communication interface module 1...10, ADR = hardware address (e.g. of the DC551)

With "Module" the following allocation applies depending on the master:
Module error: I/O bus or FBP: 31 = module itself; COM1/COM2: 1...10 = expansion 1...10
Channel error: I/O bus or FBP = module type (1 = AI, 3 = AO); COM1/COM2: 1...10 = expansion 1...10

In case of module errors, with channel "31 = Module itself" is output.

State LEDs
During the power ON procedure, the module initializes automatically. All LEDs (except the channel LEDs) are ON during this time.

<table>
<thead>
<tr>
<th>LED</th>
<th>State</th>
<th>Color</th>
<th>LED = OFF</th>
<th>LED = ON</th>
<th>LED flashes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inputs I0...I7</td>
<td>Analog input</td>
<td>Yellow</td>
<td>Input is OFF</td>
<td>Input is ON (brightness depends on the value of the analog signal)</td>
<td>--</td>
</tr>
<tr>
<td>Outputs O0...O7</td>
<td>Analog output</td>
<td>Yellow</td>
<td>Output is OFF</td>
<td>Output is ON (brightness depends on the value of the analog signal)</td>
<td>--</td>
</tr>
<tr>
<td>UP</td>
<td>Process voltage 24 VDC via terminal</td>
<td>Green</td>
<td>Process voltage is missing</td>
<td>Process voltage OK</td>
<td>--</td>
</tr>
<tr>
<td>CH-ERR2</td>
<td>Channel error, error messages in groups (analog inputs or outputs combined into the groups 2 and 4)</td>
<td>Red</td>
<td>No error or process voltage is missing</td>
<td>Severe error within the corresponding group</td>
<td>Error on one channel of the group</td>
</tr>
<tr>
<td>CH-ERR4</td>
<td></td>
<td>Red</td>
<td>--</td>
<td>--</td>
<td></td>
</tr>
<tr>
<td>CH-ERR *)</td>
<td>Module error</td>
<td>Red</td>
<td>--</td>
<td>Internal error</td>
<td>--</td>
</tr>
</tbody>
</table>

*) Both LEDs (CH-ERR2 and CH-ERR4) light up together
### Measuring ranges

#### Input ranges of voltage, current and digital input

The represented resolution corresponds to 16 bits.

<table>
<thead>
<tr>
<th>Range</th>
<th>0...10 V</th>
<th>-10...+10 V</th>
<th>0...20 mA</th>
<th>4...20 mA</th>
<th>Digital input</th>
</tr>
</thead>
<tbody>
<tr>
<td>Overflow</td>
<td>&gt;11.7589</td>
<td>&gt;11.7589</td>
<td>&gt;23.5178</td>
<td>&gt;22.8142</td>
<td></td>
</tr>
<tr>
<td>Measured value too high</td>
<td>11.7589</td>
<td>11.7589</td>
<td>23.5178</td>
<td>22.8142</td>
<td></td>
</tr>
<tr>
<td></td>
<td>10.0004</td>
<td>10.0004</td>
<td>20.0007</td>
<td>20.0006</td>
<td></td>
</tr>
<tr>
<td>Normal range</td>
<td>10.0000</td>
<td>10.0000</td>
<td>20.0000</td>
<td>20.0000</td>
<td>ON</td>
</tr>
<tr>
<td>Normal range or measured value too low</td>
<td>0.0004</td>
<td>0.0004</td>
<td>0.0007</td>
<td>4.0006</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.0000</td>
<td>0.0000</td>
<td>0</td>
<td>4</td>
<td>OFF</td>
</tr>
<tr>
<td>Measured value too low</td>
<td>-10.0000</td>
<td>-10.0000</td>
<td>-10.0000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Underflow</td>
<td>&lt;-1.7593</td>
<td>&lt;-11.7589</td>
<td>&lt;0.0000</td>
<td>&lt;1.1858</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Range</th>
<th>Digital value</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Decimal</td>
</tr>
<tr>
<td>Overflow</td>
<td>32767</td>
</tr>
<tr>
<td>Measured value too high</td>
<td>32511</td>
</tr>
<tr>
<td></td>
<td>27649</td>
</tr>
<tr>
<td>Normal range</td>
<td>27648</td>
</tr>
<tr>
<td>Normal range or measured value too low</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>-1</td>
</tr>
<tr>
<td></td>
<td>-4864</td>
</tr>
<tr>
<td></td>
<td>-6912</td>
</tr>
<tr>
<td></td>
<td>-27648</td>
</tr>
</tbody>
</table>
## Input ranges resistance temperature detector

<table>
<thead>
<tr>
<th>Range</th>
<th>Pt100 / Pt 1000 -50...70 °C</th>
<th>Pt100 / Pt1000 -50...400 °C</th>
<th>Ni1000 -50...150 °C</th>
</tr>
</thead>
<tbody>
<tr>
<td>Overflow</td>
<td>&gt; 80.0 °C</td>
<td>&gt; 450.0 °C</td>
<td>&gt; 160.0 °C</td>
</tr>
<tr>
<td>Measured value too high</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>450.0 °C</td>
<td></td>
<td>160.0 °C</td>
</tr>
<tr>
<td></td>
<td>:</td>
<td></td>
<td>:</td>
</tr>
<tr>
<td></td>
<td>400.1 °C</td>
<td></td>
<td>:</td>
</tr>
<tr>
<td></td>
<td>80.0 °C</td>
<td></td>
<td>150.1 °C</td>
</tr>
<tr>
<td></td>
<td>:</td>
<td></td>
<td>:</td>
</tr>
<tr>
<td></td>
<td>70.1 °C</td>
<td></td>
<td>:</td>
</tr>
<tr>
<td>Normal range</td>
<td></td>
<td>400.0 °C</td>
<td>150.0 °C</td>
</tr>
<tr>
<td></td>
<td>:</td>
<td>:</td>
<td>:</td>
</tr>
<tr>
<td></td>
<td>70.0 °C</td>
<td>:</td>
<td>:</td>
</tr>
<tr>
<td></td>
<td>:</td>
<td>0.1 °C</td>
<td>:</td>
</tr>
<tr>
<td></td>
<td>0.1 °C</td>
<td>0.1 °C</td>
<td>0.1 °C</td>
</tr>
<tr>
<td></td>
<td>0.0 °C</td>
<td>0.0 °C</td>
<td>0.0 °C</td>
</tr>
<tr>
<td></td>
<td>-0.1 °C</td>
<td>-0.1 °C</td>
<td>-0.1 °C</td>
</tr>
<tr>
<td></td>
<td>:</td>
<td>-50.0 °C</td>
<td>:</td>
</tr>
<tr>
<td></td>
<td>-50.0 °C</td>
<td>-50.0 °C</td>
<td>-50.0 °C</td>
</tr>
<tr>
<td>Measured value too low</td>
<td>-50.1 °C</td>
<td>-50.1 °C</td>
<td>-50.1 °C</td>
</tr>
<tr>
<td></td>
<td>:</td>
<td>-60.0 °C</td>
<td>:</td>
</tr>
<tr>
<td></td>
<td>-60.0 °C</td>
<td>-60.0 °C</td>
<td>-60.0 °C</td>
</tr>
<tr>
<td>Underflow</td>
<td>&lt; -60.0 °C</td>
<td>&lt; -60.0 °C</td>
<td>&lt; -60.0 °C</td>
</tr>
</tbody>
</table>

## Range values

<table>
<thead>
<tr>
<th>Range</th>
<th>Digital value</th>
<th>Hex.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Measured value too low</td>
<td>-27649</td>
<td>93FF</td>
</tr>
<tr>
<td></td>
<td>-32512</td>
<td></td>
</tr>
<tr>
<td>Underflow</td>
<td>-32768</td>
<td>8000</td>
</tr>
</tbody>
</table>
### Output ranges voltage and current

The represented resolution corresponds to 16 bits.

<table>
<thead>
<tr>
<th>Range</th>
<th>-10...+10 V</th>
<th>0...20 mA</th>
<th>4...20 mA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Overflow</td>
<td>0 V</td>
<td>0 mA</td>
<td>0 mA</td>
</tr>
<tr>
<td>Measured value too high</td>
<td>11.7589 V</td>
<td>23.5178 mA</td>
<td>22.8142 mA</td>
</tr>
<tr>
<td></td>
<td>10.0004 V</td>
<td>20.0007 mA</td>
<td>20.0006 mA</td>
</tr>
<tr>
<td>Normal range</td>
<td>10.0000 V</td>
<td>20.0000 mA</td>
<td>20.0000 mA</td>
</tr>
<tr>
<td></td>
<td>0.0004 V</td>
<td>0.0007 mA</td>
<td>4.0006 mA</td>
</tr>
<tr>
<td></td>
<td>0.0000 V</td>
<td>0.0000 mA</td>
<td>4.0000 mA</td>
</tr>
<tr>
<td>-0.0004 V</td>
<td>0 mA</td>
<td>3.9994 mA</td>
<td>0 mA</td>
</tr>
<tr>
<td>-10.0000 V</td>
<td>0 mA</td>
<td>0 mA</td>
<td>0 mA</td>
</tr>
<tr>
<td>Range</td>
<td>-10...+10 V</td>
<td>0...20 mA</td>
<td>4...20 mA</td>
</tr>
<tr>
<td>-----------------------</td>
<td>--------------</td>
<td>-----------</td>
<td>-----------</td>
</tr>
<tr>
<td>Measured value too low</td>
<td>-10.0004 V</td>
<td>0 mA</td>
<td>0 mA</td>
</tr>
<tr>
<td></td>
<td>-11.7589 V</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Underflow</td>
<td>0 V</td>
<td>0 mA</td>
<td>0 mA</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Range</th>
<th>Digital value</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Decimal</td>
</tr>
<tr>
<td>Overflow</td>
<td>&gt; 32511</td>
</tr>
<tr>
<td>Measured value too high</td>
<td>32511</td>
</tr>
<tr>
<td></td>
<td>27649</td>
</tr>
<tr>
<td>Normal range</td>
<td>27648</td>
</tr>
<tr>
<td></td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>-1</td>
</tr>
<tr>
<td></td>
<td>-6912</td>
</tr>
<tr>
<td></td>
<td>-27648</td>
</tr>
<tr>
<td>Measured value too low</td>
<td>-27649</td>
</tr>
<tr>
<td></td>
<td>-32512</td>
</tr>
<tr>
<td>Underflow</td>
<td>&lt; -32512</td>
</tr>
</tbody>
</table>

Technical data

The system data of AC500 and S500 refers to Chapter 1.5.4.5.1 “System data AC500” on page 3058 and are valid for the standard version.

The system data of AC500-XC refers to Chapter 1.5.4.6.1 “System data AC500-XC” on page 3099 and are valid for the XC version.

Only additional details are therefore documented below.

The technical data are also valid for the XC version.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Process voltage</td>
<td></td>
</tr>
<tr>
<td>Connections</td>
<td>Terminals 1.8, 2.8, 3.8 and 4.8 for +24 V (UP) as well as 1.9, 2.9, 3.9 and 4.9 for 0 V (ZP)</td>
</tr>
<tr>
<td>Rated value</td>
<td>24 VDC</td>
</tr>
<tr>
<td>Max. ripple</td>
<td>5 %</td>
</tr>
<tr>
<td>Protection against reversed voltage</td>
<td>Yes</td>
</tr>
<tr>
<td>Rated protection fuse on UP</td>
<td>10 A fast</td>
</tr>
</tbody>
</table>
### Parameter | Value
---|---
Galvanic isolation | Yes, per module

**Current consumption**
- From 24 VDC power supply at the terminals UP/L+ and ZP/M of the CPU/bus module: Ca. 2 mA
- From UP at normal operation: 0.15 A + output loads
- Inrush current from UP (at power up): 0.020 A²s

**Max. length of analog cables, conductor cross section > 0.14 mm²**: 100 m

**Weight**: 300 g

**Mounting position**: Horizontal or vertical with derating (output load reduced to 50 % at 40 °C per group)

**Cooling**: The natural convection cooling must not be hindered by cable ducts or other parts in the switch-gear cabinet.

---

**NOTICE!**

**Attention:**

All I/O channels (digital and analog) are protected against reverse polarity, reverse supply, short circuit and continuous overvoltage up to 30 VDC.

---

**Technical data of the analog inputs**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of channels per module</td>
<td>8</td>
</tr>
<tr>
<td>Distribution of channels into groups</td>
<td>1 group of 8 channels</td>
</tr>
<tr>
<td>Connections of the channels I0- to I7-</td>
<td>Terminals 1.0 to 1.7</td>
</tr>
<tr>
<td>Connections of the channels I0+ to I7+</td>
<td>Terminals 2.0 to 2.3</td>
</tr>
<tr>
<td>Input type</td>
<td>Bipolar (not with current or Pt100/Pt1000/Ni1000)</td>
</tr>
<tr>
<td>Galvanic isolation</td>
<td>Against internal supply and other modules</td>
</tr>
<tr>
<td>Configurability</td>
<td>0 V...10 V, -10 V...+10 V, 0 mA...20 mA, 4 mA...20 mA, Pt100/1000, Ni1000 (each input can be configured individually)</td>
</tr>
</tbody>
</table>
| Channel input resistance | Voltage: > 100 kΩ  
Current: ca. 330 Ω |
| Time constant of the input filter | Voltage: 100 μs  
current: 100 μs |
<p>| Indication of the input signals | One LED per channel |
| Conversion cycle | 2 ms (for 8 inputs + 8 outputs), with Pt/Ni... 1 s |</p>
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
</table>
| Resolution | Range 0 V...10 V: 12 bits  
Range -10 V...+10 V: 12 bits + sign  
Range 0 mA...20 mA: 12 bits  
Range 4 mA...20 mA: 12 bits |
| Conversion error of the analog values caused by non-linearity, adjustment error at factory and resolution within the normal range | Typ. ±0.5 % of full scale at 25 °C  
Max. ±1 % of full scale (all ranges) at 0 °C...60 °C or EMC disturbance |
| Unused voltage inputs | Are configured as "unused" |
| Unused current inputs | Have a low resistance, can be left open-circuited |
| Overvoltage protection | Yes |

### Technical data of the analog inputs, if used as digital Inputs

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of channels per module</td>
<td>Max. 8</td>
</tr>
<tr>
<td>Distribution of channels into groups</td>
<td>1 group of 8 channels</td>
</tr>
<tr>
<td>Connections of the channels I0+ to I7+</td>
<td>Terminals 2.0 to 2.7</td>
</tr>
<tr>
<td>Reference potential for the inputs</td>
<td>Terminals 1.9, 2.9, 3.9 and 4.9 (ZP)</td>
</tr>
<tr>
<td>Input signal delay</td>
<td>Typ. 8 ms, configurable from 0.1 to 32 ms</td>
</tr>
<tr>
<td>Indication of the input signals</td>
<td>1 LED per channel</td>
</tr>
</tbody>
</table>
| Input signal voltage | 24 VDC  
Signal 0 | -30 V...+5 V  
Undefined signal | +5 V...+13 V  
Signal 1 | +13 V...+30 V |
| Input current per channel |  
Input voltage +24 V | Typ. 7 mA |
| Input voltage +5 V | Typ. 1.4 mA |
| Input voltage +15 V | Typ. 4.3 mA |
| Input voltage +30 V | < 9 mA |
| Input resistance | Ca. 3.5 kΩ |

### Technical data of the analog outputs

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of channels per module</td>
<td>8, all channels for voltage, the first 4 channels also for current</td>
</tr>
<tr>
<td>Distribution of channels into groups</td>
<td>1 group of 8 channels</td>
</tr>
<tr>
<td>Channels O0-...O7-</td>
<td>Terminals 3.0...3.7</td>
</tr>
<tr>
<td>Channels O0+...O7+</td>
<td>Terminals 4.0...4.7</td>
</tr>
</tbody>
</table>
### Parameter | Value
---|---
Output type | Bipolar with voltage, unipolar with current
Galvanic isolation | Against internal supply and other modules
Configurability | -10 V...+10 V, 0 mA...20 mA, 4 mA...20 mA (each output can be configured individually), current outputs only channels 0...3
Output resistance (load), as current output | 0 Ω...500 Ω
Output loadability, as voltage output | Max. ±10 mA
Indication of the output signals | One LED per channel
Resolution | 12 bits (+ sign)
Conversion error of the analog values caused by non-linearity, adjustment error at factory and resolution within the normal range | Typ. ±0.5 % of full scale at 25 °C
Max. ±1 % of full scale (all ranges) at 0 °C...60 °C or EMC disturbance
Relationship between output signal and hex code | See table, Chapter 1.5.3.6.2.2.4.9.3 “Output ranges voltage and current” on page 2689
Unused outputs | Can be left open-circuited

### Ordering data

<table>
<thead>
<tr>
<th>Part no.</th>
<th>Description</th>
<th>Product life cycle phase *)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1SAP 250 000 R0001</td>
<td>AX522, analog input/output module, 8 AI, 8 AO, U/I/Pt100, 12 bits + sign, 2-wires</td>
<td>Active</td>
</tr>
<tr>
<td>1SAP 450 000 R0001</td>
<td>AX522-XC, analog input/output module, 8 AI, 8 AO, U/I/Pt100, 12 bits + sign, 2-wires, XC version</td>
<td>Active</td>
</tr>
</tbody>
</table>

*) For planning and commissioning of new installations use modules in Active status only.

#### 1.5.3.6.3 Digital/Analog I/O modules

**S500**

**DA501 - Digital/Analog input/output module**
- 16 digital inputs 24 VDC
- 8 configurable digital inputs/outputs 24 VDC, 0.5 A max.
- 4 analog inputs, voltage, current and RTD.
  - Resolution 12 bits plus sign
- 2 analog outputs, voltage and current
  - Resolution 12 bits plus sign
- Fast counter
- Module-wise electrically isolated
- XC version for use in extreme ambient conditions available

Intended purpose

The device can be used as a decentralized I/O extension module for S500 Communication Interface Modules (e.g., CI592-CS31, CI501-PNIO, CI541-DP, CI581-CN) or as a centralized extension module for AC500 CPUs (PM5xx).
**Functionality**

- 16 digital inputs 24 VDC
- 8 configurable digital inputs/outputs 24 VDC, 0.5 A max.
- 4 analog inputs, voltage, current and RTD.
  
  Resolution 12 bits plus sign
- 2 analog outputs, voltage and current
  
  Resolution 12 bits plus sign
- Fast counter

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fast Counter</td>
<td>Integrated, many configurable operating modes</td>
</tr>
<tr>
<td>Power supply</td>
<td>From the process supply voltage UP</td>
</tr>
<tr>
<td>LED displays</td>
<td>For system displays, signal states, errors and power supply</td>
</tr>
<tr>
<td>Internal supply voltage</td>
<td>Via the expansion bus interface (I/O bus)</td>
</tr>
<tr>
<td>External supply voltage</td>
<td>Via terminals UP and ZP (process supply voltage 24 VDC)</td>
</tr>
<tr>
<td>Effect of incorrect input terminal connection</td>
<td>Wrong or no signal detected, no damage up to 35 V</td>
</tr>
<tr>
<td>Required terminal unit</td>
<td>TU515 or TU516 ☄ Chapter 1.5.3.5.2 “TU515, TU516, TU541 and TU542 for I/O modules” on page 2311</td>
</tr>
</tbody>
</table>

**Electrical connection**

For a detailed description of the mounting, disassembly and electrical connection of the module, please refer to the System Assembly, Construction and Connection chapter ☄ Chapter 1.5.4.5 “AC500 (Standard)” on page 3058.

The electrical connection is carried out by using the 40 terminals of the terminal unit TU515/ TU516 ☄ Chapter 1.5.3.5.2 “TU515, TU516, TU541 and TU542 for I/O modules” on page 2311.

The assignment of the terminals:

<table>
<thead>
<tr>
<th>Terminal</th>
<th>Signal</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0</td>
<td>DI0</td>
<td>Signal of the digital input DI0</td>
</tr>
<tr>
<td>1.1</td>
<td>DI1</td>
<td>Signal of the digital input DI1</td>
</tr>
<tr>
<td>1.2</td>
<td>DI2</td>
<td>Signal of the digital input DI2</td>
</tr>
<tr>
<td>1.3</td>
<td>DI3</td>
<td>Signal of the digital input DI3</td>
</tr>
<tr>
<td>1.4</td>
<td>DI4</td>
<td>Signal of the digital input DI4</td>
</tr>
<tr>
<td>1.5</td>
<td>DI5</td>
<td>Signal of the digital input DI5</td>
</tr>
<tr>
<td>1.6</td>
<td>DI6</td>
<td>Signal of the digital input DI6</td>
</tr>
<tr>
<td>1.7</td>
<td>DI7</td>
<td>Signal of the digital input DI7</td>
</tr>
<tr>
<td>1.8</td>
<td>UP</td>
<td>Process voltage UP (24 VDC)</td>
</tr>
<tr>
<td>1.9</td>
<td>ZP</td>
<td>Process voltage ZP (0 VDC)</td>
</tr>
<tr>
<td>Terminal</td>
<td>Signal</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td>2.0</td>
<td>DI8</td>
<td>Signal of the digital input DI8</td>
</tr>
<tr>
<td>2.1</td>
<td>DI9</td>
<td>Signal of the digital input DI9</td>
</tr>
<tr>
<td>2.2</td>
<td>DI10</td>
<td>Signal of the digital input DI10</td>
</tr>
<tr>
<td>2.3</td>
<td>DI11</td>
<td>Signal of the digital input DI11</td>
</tr>
<tr>
<td>2.4</td>
<td>DI12</td>
<td>Signal of the digital input DI12</td>
</tr>
<tr>
<td>2.5</td>
<td>DI13</td>
<td>Signal of the digital input DI13</td>
</tr>
<tr>
<td>2.6</td>
<td>DI14</td>
<td>Signal of the digital input DI14</td>
</tr>
<tr>
<td>2.7</td>
<td>DI15</td>
<td>Signal of the digital input DI15</td>
</tr>
<tr>
<td>2.8</td>
<td>UP</td>
<td>Process voltage UP (24 VDC)</td>
</tr>
<tr>
<td>2.9</td>
<td>ZP</td>
<td>Process voltage ZP (0 VDC)</td>
</tr>
<tr>
<td>3.0</td>
<td>AI0+</td>
<td>Positive pole of analog input signal 0</td>
</tr>
<tr>
<td>3.1</td>
<td>AI1+</td>
<td>Positive pole of analog input signal 1</td>
</tr>
<tr>
<td>3.2</td>
<td>AI2+</td>
<td>Positive pole of analog input signal 2</td>
</tr>
<tr>
<td>3.3</td>
<td>AI3+</td>
<td>Positive pole of analog input signal 3</td>
</tr>
<tr>
<td>3.4</td>
<td>AI-</td>
<td>Negative pole of analog input signals 0 to 3</td>
</tr>
<tr>
<td>3.5</td>
<td>AO0+</td>
<td>Positive pole of analog output signal 0</td>
</tr>
<tr>
<td>3.6</td>
<td>AO1+</td>
<td>Positive pole of analog output signal 1</td>
</tr>
<tr>
<td>3.7</td>
<td>AO-</td>
<td>Negative pole of analog output signals 0 and 1</td>
</tr>
<tr>
<td>3.8</td>
<td>UP</td>
<td>Process voltage UP (24 VDC)</td>
</tr>
<tr>
<td>3.9</td>
<td>ZP</td>
<td>Process voltage ZP (0 VDC)</td>
</tr>
<tr>
<td>4.0</td>
<td>C16</td>
<td>Signal of the configurable digital input/ output C16</td>
</tr>
<tr>
<td>4.1</td>
<td>C17</td>
<td>Signal of the configurable digital input/ output C17</td>
</tr>
<tr>
<td>4.2</td>
<td>C18</td>
<td>Signal of the configurable digital input/ output C18</td>
</tr>
<tr>
<td>4.3</td>
<td>C19</td>
<td>Signal of the configurable digital input/ output C19</td>
</tr>
<tr>
<td>4.4</td>
<td>C20</td>
<td>Signal of the configurable digital input/ output C20</td>
</tr>
<tr>
<td>4.5</td>
<td>C21</td>
<td>Signal of the configurable digital input/ output C21</td>
</tr>
<tr>
<td>4.6</td>
<td>C22</td>
<td>Signal of the configurable digital input/ output C22</td>
</tr>
<tr>
<td>4.7</td>
<td>C23</td>
<td>Signal of the configurable digital input/ output C23</td>
</tr>
<tr>
<td>4.8</td>
<td>UP</td>
<td>Process voltage UP (24 VDC)</td>
</tr>
<tr>
<td>4.9</td>
<td>ZP</td>
<td>Process voltage ZP (0 VDC)</td>
</tr>
</tbody>
</table>

The internal power supply voltage for the module's circuitry is carried out via the I/O bus (provided by a bus module or a CPU). Thus, the current consumption from 24 VDC power supply at the terminals L+/UP and M/ZP of the CPU/bus module increases by 2 mA per DA501.

The external power supply connection is carried out via the UP (+24 VDC) and the ZP (0 VDC) terminals.
NOTICE!
Risk of damaging the PLC modules!
Overvoltages and short circuits might damage the PLC modules.
- Make sure that all voltage sources (supply voltage and process supply voltage) are switched off before you begin with operations at the system.
- Never connect any voltages or signals to reserved terminals (marked with ---). Reserved terminals may carry internal voltages.

NOTICE!
Risk of damaging the PLC modules!
The PLC modules must not be removed while the plant is connected to a power supply.
Make sure that all voltage sources (supply and process voltage) are switched off before you
- connect or disconnect any signal or terminal block
- remove or replace a module.

CAUTION!
Risk of imprecise and faulty measurements!
Analog signals may be distorted seriously by external electromagnetic influences.
Use shielded wires when wiring analog signal sources. The cable shield must be grounded at both ends of the cable. Provide a potential equalization of a low resistance to avoid high potential differences between different parts of the plant.

\[ \text{Fig. 164: Terminal assignment of the module} \]

The module provides several diagnosis functions \$\text{Chapter 1.5.3.6.3.1.1.7 “Diagnosis” on page 2739.} \]
Connection of the digital inputs

The following figure shows the electrical connection of the digital input DI0. Proceed with the digital inputs DI1 to DI15 in the same way.

Fig. 165: Connection of the module

The meaning of the LEDs is described in the Displays Chapter 1.5.3.6.3.1.1.8 “State LEDs” on page 2742 chapter.

Connection of the configurable digital inputs/outputs

The following figure shows the electrical connection of the configurable digital input/output DC16 and DC17. DC16 is connected as an input and DC17 is connected as an output. Proceed with the configurable digital inputs/outputs DC18 to DC23 in the same way.
Fig. 166: Connection of configurable digital inputs/outputs to the module

**CAUTION!**
**Risk of influences to the connected sensors!**
Some sensors may be influenced by the deactivated module outputs of DA501.
If the inputs are used as fast counter inputs, connect a 470 Ω / 1 W resistor in series to inputs DC16/DC17.

Connection of resistance thermometers in 2-wire configuration to the analog inputs

When resistance thermometers (Pt100, Pt1000, Ni1000) are used, a constant current must flow through them to build the necessary voltage drop for the evaluation. For this, the module DA501 provides a constant current source which is multiplexed over the max. 4 analog input channels.

The following figure shows the connection of resistance thermometers in 2-wire configuration to the analog input AI0. Proceed with the analog inputs AI1 to AI3 in the same way.
The following measuring ranges can be configured:

<table>
<thead>
<tr>
<th>Thermometer</th>
<th>Measuring Range</th>
<th>Configuration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pt100</td>
<td>-50 °C...+400 °C</td>
<td>2-wire configuration, 1 channel used</td>
</tr>
<tr>
<td>Pt1000</td>
<td>-50 °C...+400 °C</td>
<td>2-wire configuration, 1 channel used</td>
</tr>
<tr>
<td>Ni1000</td>
<td>-50 °C...+150 °C</td>
<td>2-wire configuration, 1 channel used</td>
</tr>
</tbody>
</table>

The function of the LEDs is described under Diagnosis and displays / Displays:

The module DA501 performs a linearization of the resistance characteristic.

To avoid error messages from unused analog input channels, configure them as "unused".

Connection of resistance thermometers in 3-wire configuration to the analog inputs

When resistance thermometers (Pt100, Pt1000, Ni1000) are used, a constant current must flow through them to build the necessary voltage drop for the evaluation. For this, the module DA501 provides a constant current source which is multiplexed over the max. 4 analog input channels.
Fig. 168: Connection of resistance thermometers in 3-wire configuration to the analog inputs

With 3-wire configuration, 2 adjacent analog channels belong together (e.g. the channels 0 and 1). In this case, both channels are configured according to the desired operating mode. The lower address must be the even address (channel 0), the next higher address must be the odd address (channel 1).

The constant current of one channel flows through the resistance thermometer. The constant current of the other channel flows through one of the cores. The module calculates the measured value from the two voltage drops and stores it under the input with the higher channel number (e.g. I1).

In order to keep measuring errors as small as possible, it is necessary to have all the involved conductors in the same cable. All the conductors must have the same cross section.

The following measuring ranges can be configured (Chapter 1.5.3.6.3.1.1.6 “Parameterization” on page 2735):

<table>
<thead>
<tr>
<th>Sensor</th>
<th>Measuring Range</th>
<th>Configuration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pt100</td>
<td>-50 °C...+400 °C</td>
<td>3-wire configuration, 2 channels used</td>
</tr>
<tr>
<td>Pt1000</td>
<td>-50 °C...+400 °C</td>
<td>3-wire configuration, 2 channels used</td>
</tr>
<tr>
<td>Ni1000</td>
<td>-50 °C...+150 °C</td>
<td>3-wire configuration, 2 channels used</td>
</tr>
</tbody>
</table>

The function of the LEDs is described under Diagnosis and displays / Displays (Chapter 1.5.3.6.3.1.1.7 “Diagnosis” on page 2739).

The module DA501 performs a linearization of the resistance characteristic.

To avoid error messages from unused analog input channels, configure them as "unused".

### Connection of active-type analog sensors (Voltage) with electrically isolated power supply to the analog inputs

The following figure shows the connection of active-type analog sensors (voltage) with electrically isolated power supply to the analog input AI0. Proceed with the analog inputs AI1 to AI3 in the same way.
The following measuring ranges can be configured ➥ Chapter 1.5.3.6.3.1.1.6 “Parameterization” on page 2735:

<table>
<thead>
<tr>
<th>Voltage</th>
<th>0 V...10 V</th>
<th>1 channel used</th>
</tr>
</thead>
<tbody>
<tr>
<td>Voltage</td>
<td>-10 V...+10 V</td>
<td>1 channel used</td>
</tr>
</tbody>
</table>

The function of the LEDs is described under Diagnosis and displays / Displays ➥ Chapter 1.5.3.6.3.1.8 “State LEDs” on page 2742.

To avoid error messages from unused analog input channels, configure them as "unused".

Connection of active-type analog sensors (Current) with electrically isolated power supply to the analog inputs

The following figure shows the connection of active-type analog sensors (current) with electrically isolated power supply to the analog input AI0. Proceed with the analog inputs AI1 to AI3 in the same way.
Fig. 170: Connection of active-type analog sensors (current) with electrically isolated power supply to the analog inputs

The following measuring ranges can be configured in Chapter 1.5.3.6.3.1.1.6 “Parameterization” on page 2735:

<table>
<thead>
<tr>
<th>Current</th>
<th>0 mA...20 mA</th>
<th>1 channel used</th>
</tr>
</thead>
<tbody>
<tr>
<td>Current</td>
<td>4 mA...20 mA</td>
<td>1 channel used</td>
</tr>
</tbody>
</table>

The function of the LEDs is described under Diagnosis and displays / Displays in Chapter 1.5.3.6.3.1.1.8 “State LEDs” on page 2742.

Unused input channels can be left open-circuited, because they are of low resistance.

Connection of active-type analog sensors (Voltage) with no electrically isolated power supply to the analog inputs

The following figure shows the connection of active-type analog sensors (voltage) with no electrically isolated power supply to the analog input AI0. Proceed with the analog inputs AI1 to AI3 in the same way.
Fig. 171: Connection of active-type sensors (voltage) with no electrically isolated power supply to the analog inputs

CAUTION!
Risk of faulty measurements!
The negative pole at the sensors must not have a too big potential difference with respect to ZP (max. ±1 V within the full signal range).
Make sure that the potential difference never exceeds ±1 V.

The following measuring ranges can be configured "Parameterization" on page 2735:

<table>
<thead>
<tr>
<th>Voltage</th>
<th>0 V...10 V</th>
<th>1 channel used</th>
</tr>
</thead>
<tbody>
<tr>
<td>Voltage</td>
<td>-10 V...+10 V</td>
<td>1 channel used</td>
</tr>
</tbody>
</table>

For a description of the function of the LEDs, please refer to the Diagnosis and displays / Displays chapter "State LEDs" on page 2742.
To avoid error messages from unused analog input channels, configure them as "unused".

Connection of passive-type analog sensors (Current) to the analog inputs
The following figure shows the connection of passive-type analog sensors (current) to the analog input AI0. Proceed with the analog inputs AI1 to AI3 in the same way.
Fig. 172: Connection of passive-type analog sensors (current) to the analog inputs

The following measuring ranges can be configured 

| Current | 4 mA...20 mA | 1 channel used |

For a description of function of the LEDs, please refer to the Diagnosis and displays / Displays chapter

**CAUTION!**
Risk of overloading the analog input!

If an analog current sensor supplies more than 25 mA for more than 1 second during initialization, this input is switched off by the module (input protection).

Only use sensors with fast initialization or without current peaks higher than 25 mA. If not possible, connect a 10-volt Zener diode in parallel to I+ and I-.

Unused input channels can be left open-circuited, because they are of low resistance.

**Connection of active-type analog sensors (Voltage) to differential analog inputs**

Differential inputs are very useful if analog sensors which are remotely non-isolated (e.g. the negative terminal is remotely earthed) are used.

Using differential inputs helps to considerably increase the measuring accuracy and to avoid earthing loops.

With differential input configurations, two adjacent analog channels belong together (e.g. the channels 0 and 1). In this case, both channels are configured according to the desired operating mode. The lower address must be the even address (channel 0), the next higher address must be the odd address (channel 1). The converted analog value is available at the higher address (channel 1).

The analog value is calculated by subtraction of the input value with the higher address from the input value of the lower address.

The converted analog value is available at the odd channel (higher address).
CAUTION!
Risk of faulty measurements!
The negative pole at the sensors must not have too large a potential difference with respect to ZP (max. ±1 V within the full signal range).
Make sure that the potential difference never exceeds ±1 V.

The following figure shows the connection of active-type analog sensors (voltage) to differential analog inputs AI0 and AI1. Proceed with AI2 and AI3 in the same way.

**Fig. 173: Connection of active-type analog sensors (voltage) to differential analog inputs**

The following measuring ranges can be configured (Chapter 1.5.3.6.3.1.1.6 “Parameterization” on page 2735):

<table>
<thead>
<tr>
<th>Voltage</th>
<th>0 V...10 V</th>
<th>with differential inputs, 2 channels used</th>
</tr>
</thead>
<tbody>
<tr>
<td>Voltage</td>
<td>-10 V...+10 V</td>
<td>with differential inputs, 2 channels used</td>
</tr>
</tbody>
</table>

For a description of the function of the LEDs, please refer to the Diagnosis and displays chapter (Chapter 1.5.3.6.3.1.1.8 “State LEDs” on page 2742).

To avoid error messages from unused analog input channels, configure them as "unused".

**Use of analog inputs as digital inputs**

Several (or all) analog inputs can be configured as digital inputs. The inputs are not electrically isolated against the other analog channels.

The following figure shows the connection of digital sensors to the analog input AI0. Proceed with the analog inputs AI1 to AI3 in the same way.
The following measuring ranges can be configured: "Parameterization" on page 2735:

<table>
<thead>
<tr>
<th>Digital input</th>
<th>24 V</th>
<th>1 channel used</th>
</tr>
</thead>
<tbody>
<tr>
<td>24 V DC</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

For a description of the function of the LEDs, please refer to the Diagnosis and displays chapter: "State LEDs" on page 2742.

**Connection of analog output loads (Voltage)**

The following figure shows the connection of output loads to the analog output AO0. Proceed with the analog output AO1 in the same way.
Connection of analog output loads (voltage)

The following measuring ranges can be configured “Chapter 1.5.3.6.3.1.1.6 “Parameterization” on page 2735:

<table>
<thead>
<tr>
<th>Voltage</th>
<th>-10 V...+10 V</th>
<th>Load ±10 mA max.</th>
<th>1 channel used</th>
</tr>
</thead>
</table>

For a description of the function of the LEDs, please refer to the Diagnosis and displays / Displays chapter “Chapter 1.5.3.6.3.1.1.8 “State LEDs“ on page 2742.

Unused analog outputs can be left open-circuited.

Connection of analog output loads (Current)

The following figure shows the connection of output loads to the analog output AO0. Proceed with the analog output AO1 in the same way.
The following measuring ranges can be configured on page 2735:

- **Current:** 0 mA...20 mA
  - **Load:** 0 Ω...500 Ω
  - **1 channel used**

- **Current:** 4 mA...20 mA
  - **Load:** 0 Ω...500 Ω
  - **1 channel used**

For a description of the function of the LEDs, please refer to the Diagnosis and displays chapter on page 2742.

Unused analog outputs can be left open-circuited.

### Internal data exchange

<table>
<thead>
<tr>
<th></th>
<th>Without the Fast Counter</th>
<th>With the Fast Counter (only with AC500)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Digital inputs (bytes)</td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>Digital outputs (bytes)</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>Analog inputs (words)</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>Digital outputs (words)</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>Counter input data (words)</td>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td>Counter output data (words)</td>
<td>0</td>
<td>8</td>
</tr>
</tbody>
</table>

### I/O configuration

The module does not store configuration data itself. It gets its parameterization data from the master device of the I/O bus (CPU or bus module) during power-up of the system.

Hence, replacing I/O modules is possible without any re-parameterization via software.
Parameterization

<table>
<thead>
<tr>
<th>Firmware version</th>
<th>Configuration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Firmware version &gt; V2.0.0</td>
<td>The arrangement of the parameter data is performed by Control Builder Plus/ Automation Builder software.</td>
</tr>
</tbody>
</table>

The parameter data directly influences the functionality of modules.

For non-standard applications, it is necessary to adapt the parameters to your system configuration.

Module: Module slot address: Y = 1...10

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>Internal value</th>
<th>Internal value, type</th>
<th>Default</th>
<th>EDS Slot / Index</th>
</tr>
</thead>
<tbody>
<tr>
<td>Module ID 1)</td>
<td>Internal</td>
<td>1810</td>
<td>WORD</td>
<td>1810</td>
<td>0x0Y01</td>
</tr>
<tr>
<td>Ignore module see table 2)</td>
<td>Internal</td>
<td>Yes</td>
<td>BYTE</td>
<td>No</td>
<td>not for FBP</td>
</tr>
<tr>
<td>Parameter length</td>
<td>Internal</td>
<td>8</td>
<td>BYTE</td>
<td>8</td>
<td>0xY02</td>
</tr>
<tr>
<td>Check supply</td>
<td>off</td>
<td>0</td>
<td>BYTE</td>
<td>1</td>
<td>0xY03</td>
</tr>
<tr>
<td></td>
<td>on</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Fast counter 3)</td>
<td>0</td>
<td>0</td>
<td>BYTE</td>
<td>0</td>
<td>not for FBP</td>
</tr>
<tr>
<td></td>
<td>:</td>
<td>:</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>10</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>4)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Behavior outputs at comm. error 5)</td>
<td>Off Last value Last value 5 sec Last value 10 sec Substitute value Substitute value 5 sec Substitute value 10 sec</td>
<td>0 1 6 11 2 7 12</td>
<td>BYTE</td>
<td>Off 0x00</td>
<td>0x0Y07</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>On</td>
<td>Error LED lights up at errors of all error classes, Failsafe mode off</td>
</tr>
<tr>
<td>Off by E4</td>
<td>Error LED lights up at errors of error classes E1, E2 and E3, Failsafe mode off</td>
</tr>
<tr>
<td>Off by E3</td>
<td>Error LED lights up at errors of error classes E1 and E2, Failsafe mode off</td>
</tr>
<tr>
<td>On +Failsafe</td>
<td>Error LED lights up at errors of all error classes, Failsafe mode on *)</td>
</tr>
<tr>
<td>Setting</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Off by E4 + Failsafe</td>
<td>Error LED lights up at errors of error classes E1, E2 and E3, Failsafe mode on *)</td>
</tr>
<tr>
<td>Off by E3 + Failsafe</td>
<td>Error LED lights up at errors of error classes E1 and E2, Failsafe mode on *)</td>
</tr>
</tbody>
</table>

Remarks:

1) With a faulty ID, the Modules reports a "parameter error" and does not perform cyclic process data transmission

2) Not for FBP

3) With FBP or CS31 without the parameter "Fast Counter"

The fast counter of the module does not work if the module is connected to an FBP interface module or CS31 bus module.

4) For counter operating modes, please refer to the description of the fast counter ☞ Chapter 1.5.3.6.1.2.9 “Fast counter” on page 2543

5) The parameter Behavior outputs at comm. error is only analyzed if the Failsafe-mode is ON.

---

**Group parameters for the digital part**

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>Internal value</th>
<th>Internal value, type</th>
<th>Default</th>
<th>EDS Slot / Index</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input delay</td>
<td>0.1 ms</td>
<td>0</td>
<td>BYTE</td>
<td>0.1 ms</td>
<td>0x0Y05</td>
</tr>
<tr>
<td></td>
<td>1 ms</td>
<td>1</td>
<td></td>
<td>0x00</td>
<td></td>
</tr>
<tr>
<td></td>
<td>8 ms</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>32 ms</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Detect short circuit at out-</td>
<td>Off</td>
<td>0</td>
<td>BYTE</td>
<td>On</td>
<td>0x0Y06</td>
</tr>
<tr>
<td>puts</td>
<td>On</td>
<td>1</td>
<td></td>
<td>0x01</td>
<td></td>
</tr>
<tr>
<td>Substitute value at output</td>
<td>0...255</td>
<td>00h...FFh</td>
<td>BYTE</td>
<td>0</td>
<td>0x0Y08</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0x0000</td>
<td></td>
</tr>
</tbody>
</table>

*) The parameters Behavior DO at comm. error is only analyzed if the Failsafe mode is ON.

**Group parameters for the analog part**

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>Internal value</th>
<th>Internal value, type</th>
<th>Default</th>
<th>EDS Slot / Index</th>
</tr>
</thead>
<tbody>
<tr>
<td>Analog data format</td>
<td>Standard</td>
<td>0</td>
<td>BYTE</td>
<td>0</td>
<td>0x0Y04</td>
</tr>
<tr>
<td></td>
<td>Reserved</td>
<td>255</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

*) The parameter Behavior AO at comm. error is only analyzed if the Failsafe mode is ON.
Channel parameters for the analog inputs (4x)

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>Internal value</th>
<th>Internal value, type</th>
<th>Default</th>
<th>EDS Slot / Index</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input 0, Channel configuration</td>
<td>see Table 583 “Channel configuration” on page 2737</td>
<td>see Table 583 “Channel configuration” on page 2737</td>
<td>BYTE</td>
<td>0</td>
<td>0x0Y09</td>
</tr>
<tr>
<td>Input 0, Check channel</td>
<td>see Table 584 “Channel monitoring” on page 2738</td>
<td>see Table 584 “Channel monitoring” on page 2738</td>
<td>BYTE</td>
<td>0</td>
<td>0x0Y0A</td>
</tr>
<tr>
<td>Input 3, Channel configuration</td>
<td>see Table 583 “Channel configuration” on page 2737</td>
<td>see Table 583 “Channel configuration” on page 2737</td>
<td>BYTE</td>
<td>0</td>
<td>0x0Y0F</td>
</tr>
<tr>
<td>Input 3, Check channel</td>
<td>see Table 584 “Channel monitoring” on page 2738</td>
<td>see Table 584 “Channel monitoring” on page 2738</td>
<td>BYTE</td>
<td>0</td>
<td>0x0Y10</td>
</tr>
</tbody>
</table>

Table 583: Channel configuration

<table>
<thead>
<tr>
<th>Internal value</th>
<th>Operating modes of the analog inputs, individually configurable</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 (default)</td>
<td>Not used</td>
</tr>
<tr>
<td>1</td>
<td>0 V...10 V</td>
</tr>
<tr>
<td>2</td>
<td>Digital input</td>
</tr>
<tr>
<td>3</td>
<td>0 mA…20 mA</td>
</tr>
<tr>
<td>4</td>
<td>4 mA...20 mA</td>
</tr>
<tr>
<td>5</td>
<td>-10 V...+10 V</td>
</tr>
<tr>
<td>8</td>
<td>2-wire Pt100 -50 °C...+400 °C</td>
</tr>
<tr>
<td>9</td>
<td>3-wire Pt100 -50 °C...+400 °C *)</td>
</tr>
<tr>
<td>10</td>
<td>0 V...10 V (voltage diff.) *)</td>
</tr>
<tr>
<td>11</td>
<td>-10 V...+10 V (voltage diff.) *)</td>
</tr>
<tr>
<td>14</td>
<td>2-wire Pt100 -50 °C...+70 °C</td>
</tr>
<tr>
<td>15</td>
<td>3-wire Pt100 -50 °C...+70 °C *)</td>
</tr>
<tr>
<td>16</td>
<td>2-wire Pt1000 -50 °C...+400 °C</td>
</tr>
<tr>
<td>17</td>
<td>3-wire Pt1000 -50 °C...+400 °C *)</td>
</tr>
<tr>
<td>18</td>
<td>2-wire Ni1000 -50 °C...+150 °C</td>
</tr>
</tbody>
</table>
Internal value | Operating modes of the analog inputs, individually configurable
---|---
19 | 3-wire Ni1000 -50 °C...+150 °C *)

*) In the operating modes with 3-wire configuration or with differential inputs, two adjacent analog inputs belong together (e.g. the channels 0 and 1). In these cases, both channels are configured in the desired operating mode. The lower address must be the even address (channel 0). The next higher address must be the odd address (channel 1). The converted analog value is available at the higher address (channel 1).

**Table 584: Channel monitoring**

<table>
<thead>
<tr>
<th>Internal Value</th>
<th>Check Channel</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 (default)</td>
<td>Plausibility, cut wire, short circuit</td>
</tr>
<tr>
<td>3</td>
<td>Not used</td>
</tr>
</tbody>
</table>

**Channel parameters for the analog outputs (2x)**

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>Internal value</th>
<th>Internal value, type</th>
<th>Default</th>
<th>EDS Slot / Index</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Output 0, Channel configuration</td>
<td>see Table 585 “Channel configuration” on page 2739</td>
<td>see Table 585 “Channel configuration” on page 2739</td>
<td>BYTE</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>Output 0, Check channel</td>
<td>see Table 586 “Channel monitoring” on page 2739</td>
<td>see Table 586 “Channel monitoring” on page 2739</td>
<td>BYTE</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>Output 0, Substitute value</td>
<td>see Table 587 “Substitute value” on page 2739</td>
<td>see Table 587 “Substitute value” on page 2739</td>
<td>WORD</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>Output 1, Channel configuration</td>
<td>see Table 585 “Channel configuration” on page 2739</td>
<td>see Table 585 “Channel configuration” on page 2739</td>
<td>BYTE</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>Output 1, Check channel</td>
<td>see Table 586 “Channel monitoring” on page 2739</td>
<td>see Table 586 “Channel monitoring” on page 2739</td>
<td>BYTE</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>Output 1, Substitute value</td>
<td>see Table 587 “Substitute value” on page 2739</td>
<td>see Table 587 “Substitute value” on page 2739</td>
<td>WORD</td>
<td>0</td>
</tr>
</tbody>
</table>
Table 585: Channel configuration

<table>
<thead>
<tr>
<th>Internal value</th>
<th>Operating modes of the analog outputs, individually configurable</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 (default)</td>
<td>Not used</td>
</tr>
<tr>
<td>128</td>
<td>-10 V...+10 V</td>
</tr>
<tr>
<td>129</td>
<td>0 mA...20 mA</td>
</tr>
<tr>
<td>130</td>
<td>4 mA...20 mA</td>
</tr>
</tbody>
</table>

Table 586: Channel monitoring

<table>
<thead>
<tr>
<th>Internal value</th>
<th>Check channel</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Plausibility, cut wire, short circuit</td>
</tr>
<tr>
<td>3</td>
<td>None</td>
</tr>
</tbody>
</table>

Table 587: Substitute value

<table>
<thead>
<tr>
<th>Intended behavior of output channel when the control system stops</th>
<th>Required setting of the module parameter &quot;Behavior of outputs in case of a communication error&quot;</th>
<th>Required setting of the channel parameter &quot;Substitute value&quot;</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output OFF</td>
<td>Off</td>
<td>0</td>
</tr>
<tr>
<td>Last value infinite</td>
<td>Last value</td>
<td>0</td>
</tr>
<tr>
<td>Last value for 5 s and then turn off</td>
<td>Last value 5 sec</td>
<td>0</td>
</tr>
<tr>
<td>Last value for 10 s and then turn off</td>
<td>Last value 10 sec</td>
<td>0</td>
</tr>
<tr>
<td>Substitute value infinite</td>
<td>Substitute value</td>
<td>Depending on configuration</td>
</tr>
<tr>
<td>Substitute value for 5 s and then turn off</td>
<td>Substitute value 5 sec</td>
<td>Depending on configuration</td>
</tr>
<tr>
<td>Substitute value for 10 s and then turn off</td>
<td>Substitute value 10 sec</td>
<td>Depending on configuration</td>
</tr>
</tbody>
</table>

Diagnosis

In cases of short circuit or overload, the digital outputs are turned off. The module performs reactivation automatically. Thus an acknowledgement of the errors is not necessary. The error message is stored via the LED.
<table>
<thead>
<tr>
<th>Class</th>
<th>Interface</th>
<th>Device</th>
<th>Module</th>
<th>Channel</th>
<th>Error Identifier</th>
<th>Error message</th>
<th>Remedy</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>14</td>
<td>1...10</td>
<td>31</td>
<td>31</td>
<td>19</td>
<td>Checksum error in the I/O module</td>
<td>Replace I/O module</td>
</tr>
<tr>
<td>3</td>
<td>14</td>
<td>1...10</td>
<td>31</td>
<td>31</td>
<td>3</td>
<td>Timeout in the I/O module</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>14</td>
<td>1...10</td>
<td>31</td>
<td>31</td>
<td>40</td>
<td>Different hard-/firmware versions in the module</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>14</td>
<td>1...10</td>
<td>31</td>
<td>31</td>
<td>43</td>
<td>Internal error in the module</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>14</td>
<td>1...10</td>
<td>31</td>
<td>31</td>
<td>36</td>
<td>Internal data exchange failure</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>14</td>
<td>1...10</td>
<td>31</td>
<td>31</td>
<td>9</td>
<td>Overflow diagnosis buffer</td>
<td>New start</td>
</tr>
<tr>
<td>3</td>
<td>14</td>
<td>1...10</td>
<td>31</td>
<td>31</td>
<td>26</td>
<td>Parameter error</td>
<td>Check master</td>
</tr>
<tr>
<td>3</td>
<td>14</td>
<td>1...10</td>
<td>31</td>
<td>31</td>
<td>11</td>
<td>Process voltage too low</td>
<td>Check process voltage</td>
</tr>
<tr>
<td>4</td>
<td>14</td>
<td>1...10</td>
<td>31</td>
<td>31</td>
<td>45</td>
<td>Process voltage is switched off (ON -&gt; OFF)</td>
<td>Process voltage ON</td>
</tr>
</tbody>
</table>

**Module error**

**Channel error DA501**

| 4     | 14        | 1...10 | 2      | 22...29 | 47               | Short circuit at a digital output | Check connection |

**Channel error DA501**

| 4     | 14        | 1...10 | 1      | 16...19 | 48               | Analog value overflow or broken wire at an analog input | Check input value or terminal |
| 4     | 14        | 1...10 | 1      | 16...19 | 7                | Analog value underflow at an analog input | Check input value |
| 4     | 14        | 1...10 | 1      | 16...19 | 47               | Short circuit at an analog input | Check terminal |
| 4     | 14        | 1...10 | 3      | 20...21 | 4                | Analog value overflow at an analog output | Check output value |
### PLC Automation with V3 CPUs

PLC integration > Device specifications

#### Table: Display in AC500

<table>
<thead>
<tr>
<th>Class</th>
<th>Interface</th>
<th>Device</th>
<th>Module</th>
<th>Channel</th>
<th>Error Identifier</th>
<th>Error message</th>
<th>Remedy</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>14</td>
<td>1...10</td>
<td>3</td>
<td>20...21</td>
<td>7</td>
<td>Analog value underflow at an analog output</td>
<td>Check output value</td>
</tr>
<tr>
<td></td>
<td>11 / 12</td>
<td>ADR</td>
<td>1...10</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Remarks:

1) In AC500, the following interface identifier applies:
   14 = I/O bus, 11 = COM1 (e.g. CS31 bus), 12 = COM2.
   The FBP diagnosis block does not contain this identifier.

2) With "Device" the following allocation applies:
   31 = module itself,
   1...10 = decentralized communication interface module 1...10,
   ADR = hardware address (e.g. of the DC551)

3) With "Module" the following allocation depending on the master:
   Module error: I/O bus or FBP: 31 = module itself; COM1/COM2: 1...10 = expansion 1...10
   Channel error: I/O bus or FBP = module type (1 = AI, 3 = AO, 4 = DC); COM1/COM2: 1...10 = expansion 1...10

4) In case of module errors, with channel "31 = module itself" is output.

5) Ch = 22...29 indicates the digital inputs/outputs DC16...DC23

6) Ch = 16...19 indicates the analog inputs AI0...AI3

7) Ch = 20...21 indicates the analog outputs AO0...AO1
State LEDs

<table>
<thead>
<tr>
<th>LED</th>
<th>State</th>
<th>Color</th>
<th>LED = OFF</th>
<th>LED = ON</th>
<th>LED flashes</th>
</tr>
</thead>
<tbody>
<tr>
<td>DI0 to DI15</td>
<td>Digital input</td>
<td>Yellow</td>
<td>Input is OFF</td>
<td>Input is ON 1)</td>
<td>--</td>
</tr>
<tr>
<td>DC16 to DC23</td>
<td>Digital input/output</td>
<td>Yellow</td>
<td>Input/output is OFF</td>
<td>Input/output is ON 1)</td>
<td>--</td>
</tr>
<tr>
<td>AI0 to AI3</td>
<td>Analog input</td>
<td>Yellow</td>
<td>Input is OFF</td>
<td>Input is ON 2)</td>
<td>--</td>
</tr>
<tr>
<td>AO0 to AO1</td>
<td>Analog output</td>
<td>Yellow</td>
<td>Output is OFF</td>
<td>Output is ON 2)</td>
<td>--</td>
</tr>
<tr>
<td>UP</td>
<td>Process supply voltage 24 VDC via terminal</td>
<td>Green</td>
<td>Process supply voltage is missing</td>
<td>Process supply voltage OK</td>
<td>--</td>
</tr>
<tr>
<td>CH-ERR1</td>
<td>Channel error, error messages in groups (digital inputs/outputs combined into the groups 1, 2, 3, 4)</td>
<td>Red</td>
<td>No error or process supply voltage is missing</td>
<td>Severe error within the corresponding group</td>
<td>Severe error within the corresponding group (e.g. short circuit at an output)</td>
</tr>
<tr>
<td>CH-ERR2</td>
<td></td>
<td>Red</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CH-ERR3</td>
<td></td>
<td>Red</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CH-ERR4</td>
<td></td>
<td>Red</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CH-ERR 3)</td>
<td>Module error</td>
<td>Red</td>
<td>--</td>
<td>Internal error</td>
<td>--</td>
</tr>
</tbody>
</table>

1) Indication LED is ON even if an input signal is applied to the channel and the supply voltage is off. In this case the module is not operating and does not generate an input signal.

2) Brightness depends on the value of the analog signal

3) All of the LEDs CH-ERR1 to CH-ERR4 light up together

Measuring ranges

Input ranges voltage, current and digital input

<table>
<thead>
<tr>
<th>Range</th>
<th>0...10 V</th>
<th>-10...+10 V</th>
<th>0...20 mA</th>
<th>4...20 mA</th>
<th>Digital input</th>
</tr>
</thead>
<tbody>
<tr>
<td>Overflow</td>
<td>&gt; 11.7589</td>
<td>&gt; 11.7589</td>
<td>&gt; 23.5178</td>
<td>&gt; 22.8142</td>
<td></td>
</tr>
<tr>
<td>Measured value too high</td>
<td>11.7589</td>
<td>11.7589</td>
<td>23.5178</td>
<td>22.8142</td>
<td></td>
</tr>
<tr>
<td>Normal range</td>
<td>10.0000</td>
<td>10.0000</td>
<td>20.0000</td>
<td>20.0000</td>
<td>on</td>
</tr>
<tr>
<td>Normal range or measured value too low</td>
<td>10.0000</td>
<td>10.0000</td>
<td>20.0000</td>
<td>20.0000</td>
<td></td>
</tr>
<tr>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0004</td>
<td>0.0007</td>
<td>4.0006</td>
<td></td>
</tr>
<tr>
<td>-0.0004</td>
<td>-0.0004</td>
<td>-10.0000</td>
<td>-10.0000</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Digital input on/off

2742
3ADR010583, 1, en_US
2020/12/10
### Range

<table>
<thead>
<tr>
<th>Range</th>
<th>0...10 V</th>
<th>-10...+10 V</th>
<th>0...20 mA</th>
<th>4...20 mA</th>
<th>Digital input</th>
</tr>
</thead>
<tbody>
<tr>
<td>Measured value too low</td>
<td></td>
<td>-10.0004</td>
<td>-11.7589</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Underflow</td>
<td>&lt; 0.0000</td>
<td>&lt; -11.7589</td>
<td>&lt; 0.0000</td>
<td>&lt; 0.0000</td>
<td></td>
</tr>
</tbody>
</table>

### Range

<table>
<thead>
<tr>
<th>Range</th>
<th>Digital value</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Decimal</td>
</tr>
<tr>
<td>Overflow</td>
<td>32767</td>
</tr>
<tr>
<td>Measured value too high</td>
<td>32511</td>
</tr>
<tr>
<td></td>
<td>27649</td>
</tr>
<tr>
<td>Normal range Normal range or measured value too low</td>
<td>27648</td>
</tr>
<tr>
<td></td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>-1</td>
</tr>
<tr>
<td></td>
<td>-4864</td>
</tr>
<tr>
<td></td>
<td>-6912</td>
</tr>
<tr>
<td></td>
<td>-27648</td>
</tr>
<tr>
<td>Measured value too low</td>
<td>-27649</td>
</tr>
<tr>
<td></td>
<td>-32512</td>
</tr>
<tr>
<td>Underflow</td>
<td>-32768</td>
</tr>
</tbody>
</table>

The represented resolution corresponds to 16 bits.

### Input ranges resistance temperature detector

<table>
<thead>
<tr>
<th>Range</th>
<th>Pt100 / Pt1000 -50...70 °C</th>
<th>Pt100 / Pt1000 -50...400 °C</th>
<th>Ni1000 -50...150 °C</th>
</tr>
</thead>
<tbody>
<tr>
<td>Overflow</td>
<td>&gt; 80.0 °C</td>
<td>&gt; 450.0 °C</td>
<td>&gt; 160.0 °C</td>
</tr>
<tr>
<td>Measured value too high</td>
<td></td>
<td>450.0 °C</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>400.1 °C</td>
<td>160.0 °C</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>150.1 °C</td>
</tr>
<tr>
<td>80.0 °C</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>70.1 °C</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Range

<table>
<thead>
<tr>
<th></th>
<th>Pt100 / Pt1000 -50...70 °C</th>
<th>Pt100 / Pt1000 -50...400 °C</th>
<th>Ni1000 -50...150 °C</th>
</tr>
</thead>
<tbody>
<tr>
<td>Normal range</td>
<td>:</td>
<td>400.0 °C</td>
<td>150.0 °C</td>
</tr>
<tr>
<td></td>
<td>:</td>
<td>:</td>
<td>:</td>
</tr>
<tr>
<td></td>
<td>70.0 °C</td>
<td>:</td>
<td>:</td>
</tr>
<tr>
<td></td>
<td>:</td>
<td>:</td>
<td>0.1 °C</td>
</tr>
<tr>
<td></td>
<td>0.1 °C</td>
<td>0.1 °C</td>
<td>:</td>
</tr>
<tr>
<td>0.0 °C</td>
<td>0.0 °C</td>
<td>0.0 °C</td>
<td>0.0 °C</td>
</tr>
<tr>
<td>-0.1 °C</td>
<td>-0.1 °C</td>
<td>:</td>
<td>:</td>
</tr>
<tr>
<td></td>
<td>:</td>
<td>:</td>
<td>-50.0 °C</td>
</tr>
<tr>
<td>Measured value too low</td>
<td>-50.1 °C</td>
<td>-50.1 °C</td>
<td>-50.1 °C</td>
</tr>
<tr>
<td></td>
<td>:</td>
<td>:</td>
<td>:</td>
</tr>
<tr>
<td></td>
<td>-60.0 °C</td>
<td>-60.0 °C</td>
<td>-60.0 °C</td>
</tr>
<tr>
<td>Underflow</td>
<td>&lt; -60.0 °C</td>
<td>&lt; -60.0 °C</td>
<td>&lt; -60.0 °C</td>
</tr>
</tbody>
</table>

### Digital value

<table>
<thead>
<tr>
<th>Range</th>
<th>Digital value</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Decimal</td>
<td>Hex.</td>
<td></td>
</tr>
<tr>
<td>Overflow</td>
<td>32767</td>
<td>7FFF</td>
<td></td>
</tr>
<tr>
<td>Measured value too high</td>
<td>4500</td>
<td>1194</td>
<td></td>
</tr>
<tr>
<td></td>
<td>:</td>
<td>:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>4001</td>
<td>0FA1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1600</td>
<td>0640</td>
<td></td>
</tr>
<tr>
<td></td>
<td>:</td>
<td>:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1501</td>
<td>05DD</td>
<td></td>
</tr>
<tr>
<td></td>
<td>800</td>
<td>0320</td>
<td></td>
</tr>
<tr>
<td></td>
<td>:</td>
<td>:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>701</td>
<td>02BD</td>
<td></td>
</tr>
<tr>
<td>Normal range</td>
<td>4000</td>
<td>0FA0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1500</td>
<td>05DC</td>
<td></td>
</tr>
<tr>
<td></td>
<td>700</td>
<td>02BC</td>
<td></td>
</tr>
<tr>
<td></td>
<td>:</td>
<td>:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>0001</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>0000</td>
<td></td>
</tr>
<tr>
<td></td>
<td>-1</td>
<td>FFFF</td>
<td></td>
</tr>
<tr>
<td></td>
<td>:</td>
<td>:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>-500</td>
<td>FE0C</td>
<td></td>
</tr>
<tr>
<td>Measured value too low</td>
<td>-501</td>
<td>FE0B</td>
<td></td>
</tr>
<tr>
<td></td>
<td>:</td>
<td>:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>-600</td>
<td>FDA8</td>
<td></td>
</tr>
<tr>
<td>Underflow</td>
<td>-32768</td>
<td>8000</td>
<td></td>
</tr>
</tbody>
</table>
## Output ranges voltage and current

<table>
<thead>
<tr>
<th>Range</th>
<th>-10...+10 V</th>
<th>0...20 mA</th>
<th>4...20 mA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Overflow</td>
<td>&gt;11.7589 V</td>
<td>&gt;23.5178 mA</td>
<td>&gt;22.8142 mA</td>
</tr>
<tr>
<td>Measured value too high</td>
<td>11.7589 V</td>
<td>23.5178 mA</td>
<td>22.8142 mA</td>
</tr>
<tr>
<td></td>
<td>10.0004 V</td>
<td>20.0007 mA</td>
<td>20.0006 mA</td>
</tr>
<tr>
<td>Normal range</td>
<td>10.0000 V</td>
<td>20.0000 mA</td>
<td>20.0000 mA</td>
</tr>
<tr>
<td></td>
<td>0.0004 V</td>
<td>0.0007 mA</td>
<td>4.0006 mA</td>
</tr>
<tr>
<td></td>
<td>0.0000 V</td>
<td>0.0000 mA</td>
<td>4.0000 mA</td>
</tr>
<tr>
<td></td>
<td>-0.0004 V</td>
<td>0 mA</td>
<td>3.9994 mA</td>
</tr>
<tr>
<td></td>
<td>-10.0000 V</td>
<td>0 mA</td>
<td>0 mA</td>
</tr>
<tr>
<td>Measured value too low</td>
<td>-10.0004 V</td>
<td>0 mA</td>
<td>0 mA</td>
</tr>
<tr>
<td></td>
<td>-11.7589 V</td>
<td>0 mA</td>
<td>0 mA</td>
</tr>
<tr>
<td>Underflow</td>
<td>0 V</td>
<td>0 mA</td>
<td>0 mA</td>
</tr>
</tbody>
</table>

### Range

<table>
<thead>
<tr>
<th>Range</th>
<th>Digital value</th>
<th>Decimal</th>
<th>Hex.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Overflow</td>
<td>&gt; 32511</td>
<td>&gt; 32511</td>
<td>&gt; 7EFF</td>
</tr>
<tr>
<td>Measured value too high</td>
<td>32511</td>
<td>27649</td>
<td>7EFF</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>6C01</td>
</tr>
<tr>
<td>Normal range</td>
<td>27648</td>
<td>1</td>
<td>6C00</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0</td>
<td>0000</td>
</tr>
<tr>
<td></td>
<td>-1</td>
<td>-6912</td>
<td>FFFF</td>
</tr>
<tr>
<td></td>
<td>-27648</td>
<td>-9400</td>
<td>E500</td>
</tr>
<tr>
<td>Measured value too low</td>
<td>-27649</td>
<td>-32512</td>
<td>93FF</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>8100</td>
</tr>
<tr>
<td>Underflow</td>
<td>&lt; -32512</td>
<td></td>
<td>&lt; 8100</td>
</tr>
</tbody>
</table>

The represented resolution corresponds to 16 bits.

## Technical data

### Technical data of the module

The system data of AC500 and S500 in Chapter 1.5.4.5.1 “System data AC500” on page 3058 are valid for standard version.
The system data of AC500-XC ∘ Chapter 1.5.4.6.1 “System data AC500-XC” on page 3099 are valid for the XC version.

Only additional details are therefore documented below.

The technical data are also valid for the XC version.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Process supply voltage</td>
<td></td>
</tr>
<tr>
<td>Connections</td>
<td>Terminals 1.8, 2.8, 3.8 and 4.8 for UP (+24 VDC) and 1.9, 2.9, 3.9 and 4.9 for ZP (0 VDC)</td>
</tr>
<tr>
<td>Protection against reverse voltage</td>
<td>yes</td>
</tr>
<tr>
<td>Rated protection fuse at UP</td>
<td>10 A fast</td>
</tr>
<tr>
<td>Rated value</td>
<td>24 VDC</td>
</tr>
<tr>
<td>Max. ripple</td>
<td>5 %</td>
</tr>
<tr>
<td>Current consumption</td>
<td></td>
</tr>
<tr>
<td>From UP</td>
<td>0.07 A + max. 0.5 A per output</td>
</tr>
<tr>
<td>From 24 VDC power supply at the terminals UP/L+ and ZP/M of the CPU/Bus Module</td>
<td>ca. 2 mA</td>
</tr>
<tr>
<td>Inrush current from UP (at power-up)</td>
<td>0.04 A²s</td>
</tr>
<tr>
<td>Galvanic isolation</td>
<td>Yes, per module</td>
</tr>
<tr>
<td>Max. power dissipation within the module</td>
<td>6 W (outputs unloaded)</td>
</tr>
<tr>
<td>Weight (without terminal unit)</td>
<td>ca. 125 g</td>
</tr>
<tr>
<td>Mounting position</td>
<td>Horizontal mounting or vertical with derating (output load reduced to 50 % at 40 °C)</td>
</tr>
<tr>
<td>Cooling</td>
<td>The natural convection cooling must not be hindered by cable ducts or other parts in the switch-gear cabinet.</td>
</tr>
</tbody>
</table>

**NOTICE!**

**Attention:**

All I/O channels (digital and analog) are protected against reverse polarity, reverse supply, short circuit and continuous overvoltage up to 30 VDC.

**Multiple overloads**

No effects of multiple overloads on isolated multi-channel modules occur, as every channel is protected individually by an internal smart high-side switch.

**Technical data of the digital inputs**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of channels per module</td>
<td>16</td>
</tr>
<tr>
<td>Distribution of the channels into groups</td>
<td>2 groups of 8 channels</td>
</tr>
</tbody>
</table>
### Technical data of the configurable digital inputs/outputs

Each of the configurable digital I/O channels can be defined as input or output by the user program. This is done by interrogating or allocating the corresponding channel.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of channels per module</td>
<td>8 inputs/outputs (with transistors)</td>
</tr>
<tr>
<td>Distribution of the channels into groups</td>
<td>1 group for 8 channels</td>
</tr>
<tr>
<td>If the channels are used as inputs</td>
<td></td>
</tr>
<tr>
<td>Channels DC16...DC23</td>
<td>Terminals 4.0...4.7</td>
</tr>
<tr>
<td>If the channels are used as outputs</td>
<td></td>
</tr>
<tr>
<td>Channels DC16...DC23</td>
<td>Terminals 4.0...4.7</td>
</tr>
<tr>
<td>Indication of the input/output signals</td>
<td>1 yellow LED per channel, the LED is ON when the input/output signal is high (signal 1)</td>
</tr>
<tr>
<td>Monitoring point of input/output indicator</td>
<td>LED is part of the input circuitry</td>
</tr>
<tr>
<td>Galvanic isolation</td>
<td>Yes, per module</td>
</tr>
</tbody>
</table>
### Technical data of the digital inputs/outputs if used as inputs

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of channels per module</td>
<td>8</td>
</tr>
<tr>
<td>Distribution of the channels into groups</td>
<td>1 group of 8 channels</td>
</tr>
<tr>
<td>Terminals of the channels DC16 to DC23</td>
<td>Terminals 4.0 to 4.7</td>
</tr>
<tr>
<td>Reference potential for all inputs</td>
<td>Terminals 1.9...4.9 (negative pole of the supply voltage, signal name ZP)</td>
</tr>
<tr>
<td>Indication of the input signals</td>
<td>1 yellow LED per channel, the LED is ON when the input signal is high (signal 1)</td>
</tr>
<tr>
<td>Monitoring point of input/output indicator</td>
<td>LED is part of the input circuitry</td>
</tr>
<tr>
<td>Input type (according EN 61131-2)</td>
<td>Type 1</td>
</tr>
<tr>
<td>Input delay (0-&gt;1 or 1-&gt;0)</td>
<td>Typ. 0.1 ms, configurable from 0.1...32 ms</td>
</tr>
<tr>
<td>Input signal voltage</td>
<td>24 VDC</td>
</tr>
<tr>
<td>0-Signal</td>
<td>-3 V...+5 V</td>
</tr>
<tr>
<td>Undefined Signal</td>
<td>&gt; +5 V...&lt; +15 V</td>
</tr>
<tr>
<td>1-Signal</td>
<td>+15 V...+30 V</td>
</tr>
<tr>
<td>Ripple with signal 0</td>
<td>Within -3 V...+5 V</td>
</tr>
<tr>
<td>Ripple with signal 1</td>
<td>Within +15 V...+30 V</td>
</tr>
<tr>
<td>Input current per channel</td>
<td></td>
</tr>
<tr>
<td>Input voltage +24 V</td>
<td>Typ. 5 mA</td>
</tr>
<tr>
<td>Input voltage +5 V</td>
<td>&gt; 1 mA</td>
</tr>
<tr>
<td>Input voltage +15 V</td>
<td>&gt; 2 mA</td>
</tr>
<tr>
<td>Input voltage +30 V</td>
<td>&lt; 8 mA</td>
</tr>
<tr>
<td>Max. cable length</td>
<td></td>
</tr>
<tr>
<td>shielded</td>
<td>1000 m</td>
</tr>
<tr>
<td>unshielded</td>
<td>600 m</td>
</tr>
</tbody>
</table>

* Due to the direct connection to the output, the demagnetizing varistor is also effective at the input (see figure) above. This is why the difference between UPx and the input signal must not exceed the clamp voltage of the varistor. The varistor limits the clamp voltage to approx. 36 V. Consequently, the input voltage must range from -12 V to +30 V when UPx = 24 V and from -6 V to +30 V when UPx = 30 V.

### Technical data of the digital inputs/outputs if used as outputs

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of channels per module</td>
<td>8</td>
</tr>
<tr>
<td>Distribution of the channels into groups</td>
<td>1 group of 8 channels</td>
</tr>
<tr>
<td>Terminals of the channels DC16 to DC23</td>
<td>Terminals 4.0 to 4.7</td>
</tr>
<tr>
<td>Reference potential for all outputs</td>
<td>Terminals 1.9...4.9 (negative pole of the supply voltage, signal name ZP)</td>
</tr>
<tr>
<td>Common power supply voltage</td>
<td>For all outputs terminals 1.8, 2.8, 3.8 and 4.8 (positive pole of the supply voltage, signal name UP)</td>
</tr>
<tr>
<td>Output voltage for signal 1</td>
<td>UP (-0.8 V)</td>
</tr>
</tbody>
</table>
### Parameter

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output delay (0-&gt;1 or 1-&gt;0)</td>
<td>On request</td>
</tr>
<tr>
<td>Output current</td>
<td></td>
</tr>
<tr>
<td>rated value per channel</td>
<td>500 mA at UP = 24 V</td>
</tr>
<tr>
<td>max. value (all channels together)</td>
<td>4 A</td>
</tr>
<tr>
<td>Leakage current with signal 0</td>
<td>&lt; 0.5 mA</td>
</tr>
<tr>
<td>Fuse for UP</td>
<td>10 A fast</td>
</tr>
<tr>
<td>Demagnetization with inductive DC load</td>
<td>Via internal varistors (see figure below this table)</td>
</tr>
<tr>
<td>Output switching frequency</td>
<td></td>
</tr>
<tr>
<td>With resistive load</td>
<td>On request</td>
</tr>
<tr>
<td>With inductive loads</td>
<td>Max. 0.5 Hz</td>
</tr>
<tr>
<td>With lamp loads</td>
<td>11 Hz max. at 5 W max.</td>
</tr>
<tr>
<td>Short-circuit-proof / overload-proof</td>
<td>Yes</td>
</tr>
<tr>
<td>Overload message (I &gt; 0.7 A)</td>
<td>Yes, after ca. 100 ms</td>
</tr>
<tr>
<td>Output current limitation</td>
<td>Yes, automatic reactivation after short circuit/overload</td>
</tr>
<tr>
<td>Resistance to feedback against 24 V signals</td>
<td>Yes (software-controlled supervision)</td>
</tr>
<tr>
<td>Max. cable length</td>
<td></td>
</tr>
<tr>
<td>Shielded</td>
<td>1000 m</td>
</tr>
<tr>
<td>Unshielded</td>
<td>600 m</td>
</tr>
</tbody>
</table>

The following drawing shows the circuitry of a digital input/output with the varistors for demagnetization when inductive loads are switched off.

![Circuit Diagram](image)

**Fig. 177: Digital input/output (circuit diagram)**

1. Digital input/output
2. For demagnetization when inductive loads are turned off

---

**Technical data of the fast counter**

*The fast counter of the module does not work if the module is connected to an FBP interface module or CS31 bus module.*
### Technical data of the analog inputs

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of channels per module</td>
<td>4</td>
</tr>
<tr>
<td>Distribution of channels into groups</td>
<td>1 group with 4 channels</td>
</tr>
<tr>
<td>Connection if channels AI0+ to AI3+</td>
<td>Terminals 3.0 to 3.3</td>
</tr>
<tr>
<td>Reference potential for AI0+ to AI3+</td>
<td>Terminal 3.4 (AI-) for voltage and RTD measurement</td>
</tr>
<tr>
<td></td>
<td>Terminal 1.9, 2.9, 3.9 and 4.9 for current measurement</td>
</tr>
<tr>
<td>Input type</td>
<td></td>
</tr>
<tr>
<td>Unipolar</td>
<td>Voltage 0 V...10 V, current or Pt100/Pt1000/Ni1000</td>
</tr>
<tr>
<td>Bipolar</td>
<td>Voltage -10 V...+10 V</td>
</tr>
<tr>
<td>Configurability</td>
<td>0 V...10 V, -10 V...+10 V, 0 mA...20 mA, 4 mA...20 mA, Pt100/1000, Ni1000 (each input can be configured individually)</td>
</tr>
<tr>
<td>Channel input resistance</td>
<td>Voltage: &gt; 100 kΩ</td>
</tr>
<tr>
<td></td>
<td>Current: ca. 330 Ω</td>
</tr>
<tr>
<td>Time constant of the input filter</td>
<td>Voltage: 100 µs</td>
</tr>
<tr>
<td></td>
<td>Current: 100 µs</td>
</tr>
<tr>
<td>Indication of the input signals</td>
<td>1 LED per channel (brightness depends on the value of the analog signal)</td>
</tr>
<tr>
<td>Conversion cycle</td>
<td>1 ms (for 4 inputs + 2 outputs); with RTDs Pt/Ni... 1 s</td>
</tr>
<tr>
<td>Resolution</td>
<td>Range 0 V...10 V: 12 bits</td>
</tr>
<tr>
<td></td>
<td>Range -10 V...+10 V: 12 bits + sign</td>
</tr>
<tr>
<td></td>
<td>Range 0 mA...20 mA: 12 bits</td>
</tr>
<tr>
<td></td>
<td>Range 4 mA...20 mA: 12 bits</td>
</tr>
<tr>
<td></td>
<td>Range RTD (Pt100, Pt1000, Ni1000): 0.1 °C</td>
</tr>
<tr>
<td>Conversion error of the analog values caused</td>
<td>Typ. 0.5 %, max. 1 %</td>
</tr>
<tr>
<td>by non-linearity, adjustment error at factory and resolution within the normal range</td>
<td>For XC version below 0 °C and above 60 °C: on request</td>
</tr>
</tbody>
</table>
### Technical data of the analog inputs, if used as digital inputs

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of channels per module</td>
<td>Max. 4</td>
</tr>
<tr>
<td>Distribution of channels into groups</td>
<td>1 group of 4 channels</td>
</tr>
<tr>
<td>Connections of the channels AI0+ to AI3+</td>
<td>Terminals 3.0 to 3.3</td>
</tr>
<tr>
<td>Reference potential for the inputs</td>
<td>Terminals 1.9, 2.9, 3.9 and 4.9 (ZP)</td>
</tr>
<tr>
<td>Indication of the input signals</td>
<td>1 LED per channel</td>
</tr>
<tr>
<td>Input signal voltage</td>
<td>24 VDC</td>
</tr>
<tr>
<td>Signal 0</td>
<td>-30 V...+5 V</td>
</tr>
<tr>
<td>Undefined signal</td>
<td>+5 V...+13 V</td>
</tr>
<tr>
<td>Signal 1</td>
<td>+13 V...+30 V</td>
</tr>
<tr>
<td>Input current per channel</td>
<td></td>
</tr>
<tr>
<td>Input voltage +24 V</td>
<td>Typ. 7 mA</td>
</tr>
<tr>
<td>Input voltage +5 V</td>
<td>Typ. 1.4 mA</td>
</tr>
<tr>
<td>Input voltage +15 V</td>
<td>Typ. 3.7 mA</td>
</tr>
<tr>
<td>Input voltage +30 V</td>
<td>&lt; 9 mA</td>
</tr>
<tr>
<td>Input resistance</td>
<td>ca. 3.5 kΩ</td>
</tr>
</tbody>
</table>

### Technical data of the analog outputs

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of channels per module</td>
<td>2</td>
</tr>
<tr>
<td>Distribution of channels into groups</td>
<td>1 group for 2 channels</td>
</tr>
<tr>
<td>Connection of the channels AO0+...AO1+</td>
<td>Terminals 3.5 and 3.6</td>
</tr>
<tr>
<td>Reference potential for AO0+ to AO1+</td>
<td>Terminal 3.7 (AO-) for voltage output</td>
</tr>
<tr>
<td>Output type</td>
<td></td>
</tr>
<tr>
<td>Unipolar</td>
<td>Current</td>
</tr>
<tr>
<td>Bipolar</td>
<td>Voltage</td>
</tr>
<tr>
<td>Galvanic isolation</td>
<td>Against internal supply and other modules</td>
</tr>
</tbody>
</table>
### Parameter Values

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Configurability</td>
<td>-10 V...+10 V, 0 mA...20 mA, 4 mA...20 mA (each output can be configured individually)</td>
</tr>
<tr>
<td>Output resistance (load) as current output</td>
<td>0 Ω...500 Ω</td>
</tr>
<tr>
<td>Output loadability as voltage output</td>
<td>±10 mA max.</td>
</tr>
<tr>
<td>Indication of the output signals</td>
<td>1 LED per channel (brightness depends on the value of the analog signal)</td>
</tr>
<tr>
<td>Resolution</td>
<td>12 bits (+ sign)</td>
</tr>
<tr>
<td>Conversion error of the analog values caused by non-linearity, adjustment error at factory and resolution within the normal range</td>
<td>Typ. 0.5 %, max. 1 %</td>
</tr>
<tr>
<td>Relationship between input signal and hex code</td>
<td>% Chapter 1.5.3.6.3.1.1.9.3 “Output ranges voltage and current” on page 2745</td>
</tr>
<tr>
<td>Unused outputs</td>
<td>Are configured as &quot;unused&quot; (default value) and can be left open-circuited</td>
</tr>
</tbody>
</table>

### Internal data exchange

<table>
<thead>
<tr>
<th></th>
<th>Without the Fast Counter</th>
<th>With the Fast Counter (only with AC500)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Digital inputs (bytes)</td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>Digital outputs (bytes)</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>Analog inputs (words)</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>Analog outputs (words)</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>Counter input data (words)</td>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td>Counter output data (words)</td>
<td>0</td>
<td>8</td>
</tr>
</tbody>
</table>

### Ordering data

<table>
<thead>
<tr>
<th>Part no.</th>
<th>Description</th>
<th>Product life cycle phase *)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1SAP 250 700 R0001</td>
<td>DA501, digital/analog input/output module, 16 DI, 8 DC, 4 AI, 2 AO</td>
<td>Active</td>
</tr>
<tr>
<td>1SAP 450 700 R0001</td>
<td>DA501-XC, digital/analog input/output module, 16 DI, 8 DC, 4 AI, 2 AO, XC version</td>
<td>Active</td>
</tr>
</tbody>
</table>

*) For planning and commissioning of new installations use modules in Active status only.

### DA502 - Digital/Analog input/output module

- 16 digital outputs, 24 VDC, 0.5 A max.
- 8 configurable digital inputs/outputs 24 VDC, 0.5 A max.
- 4 analog inputs, voltage, current and RTD, resolution 12 bits plus sign
- 2 analog outputs, voltage and current, resolution 12 bits plus sign
- Fast counter
- Module-wise electrically isolated
- XC version for use in extreme ambient conditions available

**Intended purpose**

The device can be used as a decentralized I/O extension module for S500 Communication Interface Modules (e.g., CI592-CS31, CI501-PNIO, CI541-DP, CI581-CN) or as a centralized extension module for AC500 CPUs (PM5xx).
### Functionality

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fast counter</td>
<td>Integrated, many configurable operating modes</td>
</tr>
<tr>
<td>Power supply</td>
<td>From the process supply voltage UP</td>
</tr>
<tr>
<td>LED displays</td>
<td>For system displays, signal states, errors and power supply</td>
</tr>
<tr>
<td>Internal supply voltage</td>
<td>Via the expansion bus interface (I/O bus)</td>
</tr>
<tr>
<td>External supply voltage</td>
<td>Via terminals UP and ZP (process supply voltage 24 VDC)</td>
</tr>
<tr>
<td>Effect of incorrect input terminal connection</td>
<td>Wrong or no signal detected, no damage up to 35 V</td>
</tr>
<tr>
<td>Required terminal unit</td>
<td>TU515 or TU516 * Chapter 1.5.3.5.2 “TU515, TU516, TU541 and TU542 for I/O modules” on page 2311</td>
</tr>
</tbody>
</table>

### Electrical connection

For a detailed description of the mounting, disassembly and electrical connection of the module, please refer to the System Assembly, Construction and Connection chapter \* Chapter 1.5.4.5 “AC500 (Standard)” on page 3058.

The electrical connection is carried out by using the 40 terminals of the terminal unit TU515/ TU516 \* Chapter 1.5.3.5.2 “TU515, TU516, TU541 and TU542 for I/O modules” on page 2311.

The assignment of the terminals:

<table>
<thead>
<tr>
<th>Terminal</th>
<th>Signal</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0</td>
<td>DO0</td>
<td>Signal of the digital output DO0</td>
</tr>
<tr>
<td>1.1</td>
<td>DO1</td>
<td>Signal of the digital output DO1</td>
</tr>
<tr>
<td>1.2</td>
<td>DO2</td>
<td>Signal of the digital output DO2</td>
</tr>
<tr>
<td>1.3</td>
<td>DO3</td>
<td>Signal of the digital output DO3</td>
</tr>
<tr>
<td>1.4</td>
<td>DO4</td>
<td>Signal of the digital output DO4</td>
</tr>
<tr>
<td>1.5</td>
<td>DO5</td>
<td>Signal of the digital output DO5</td>
</tr>
<tr>
<td>1.6</td>
<td>DO6</td>
<td>Signal of the digital output DO6</td>
</tr>
<tr>
<td>1.7</td>
<td>DO7</td>
<td>Signal of the digital output DO7</td>
</tr>
<tr>
<td>1.8</td>
<td>UP</td>
<td>Process voltage UP (24 VDC)</td>
</tr>
<tr>
<td>1.9</td>
<td>ZP</td>
<td>Process voltage ZP (0 VDC)</td>
</tr>
<tr>
<td>2.0</td>
<td>DO8</td>
<td>Signal of the digital output DO8</td>
</tr>
<tr>
<td>2.1</td>
<td>DO9</td>
<td>Signal of the digital output DO9</td>
</tr>
<tr>
<td>2.2</td>
<td>DO10</td>
<td>Signal of the digital output DO10</td>
</tr>
<tr>
<td>2.3</td>
<td>DO11</td>
<td>Signal of the digital output DO11</td>
</tr>
<tr>
<td>2.4</td>
<td>DO12</td>
<td>Signal of the digital output DO12</td>
</tr>
<tr>
<td>2.5</td>
<td>DO13</td>
<td>Signal of the digital output DO13</td>
</tr>
<tr>
<td>2.6</td>
<td>DO14</td>
<td>Signal of the digital output DO14</td>
</tr>
<tr>
<td>Terminal</td>
<td>Signal</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td>2.7</td>
<td>DO15</td>
<td>Signal of the digital output DO15</td>
</tr>
<tr>
<td>2.8</td>
<td>UP</td>
<td>Process voltage UP (24 VDC)</td>
</tr>
<tr>
<td>2.9</td>
<td>ZP</td>
<td>Process voltage ZP (0 VDC)</td>
</tr>
<tr>
<td>3.0</td>
<td>AI0+</td>
<td>Positive pole of analog input signal 0</td>
</tr>
<tr>
<td>3.1</td>
<td>AI1+</td>
<td>Positive pole of analog input signal 1</td>
</tr>
<tr>
<td>3.2</td>
<td>AI2+</td>
<td>Positive pole of analog input signal 2</td>
</tr>
<tr>
<td>3.3</td>
<td>AI3+</td>
<td>Positive pole of analog input signal 3</td>
</tr>
<tr>
<td>3.4</td>
<td>AI-</td>
<td>Negative pole of analog input signals 0 to 3</td>
</tr>
<tr>
<td>3.5</td>
<td>AO0+</td>
<td>Positive pole of analog output signal 0</td>
</tr>
<tr>
<td>3.6</td>
<td>AO1+</td>
<td>Positive pole of analog output signal 1</td>
</tr>
<tr>
<td>3.7</td>
<td>AO-</td>
<td>Negative pole of analog output signals 0 and 1</td>
</tr>
<tr>
<td>3.8</td>
<td>UP</td>
<td>Process voltage UP (24 VDC)</td>
</tr>
<tr>
<td>3.9</td>
<td>ZP</td>
<td>Process voltage ZP (0 VDC)</td>
</tr>
<tr>
<td>4.0</td>
<td>DC16</td>
<td>Signal of the configurable digital input/output DC16</td>
</tr>
<tr>
<td>4.1</td>
<td>DC17</td>
<td>Signal of the configurable digital input/output DC17</td>
</tr>
<tr>
<td>4.2</td>
<td>DC18</td>
<td>Signal of the configurable digital input/output DC18</td>
</tr>
<tr>
<td>4.3</td>
<td>DC19</td>
<td>Signal of the configurable digital input/output DC19</td>
</tr>
<tr>
<td>4.4</td>
<td>DC20</td>
<td>Signal of the configurable digital input/output DC20</td>
</tr>
<tr>
<td>4.5</td>
<td>DC21</td>
<td>Signal of the configurable digital input/output DC21</td>
</tr>
<tr>
<td>4.6</td>
<td>DC22</td>
<td>Signal of the configurable digital input/output DC22</td>
</tr>
<tr>
<td>4.7</td>
<td>DC23</td>
<td>Signal of the configurable digital input/output DC23</td>
</tr>
<tr>
<td>4.8</td>
<td>UP</td>
<td>Process voltage UP (24 VDC)</td>
</tr>
<tr>
<td>4.9</td>
<td>ZP</td>
<td>Process voltage ZP (0 VDC)</td>
</tr>
</tbody>
</table>

The internal power supply voltage for the module's circuitry is carried out via the I/O bus (provided by a bus module or a CPU). Thus, the current consumption from 24 VDC power supply at the terminals L+/UP and M/ZP of the CPU/bus module increases by 2 mA per DA502.

The external power supply connection is carried out via the UP (+24 VDC) and the ZP (0 VDC) terminals.

**NOTICE!**

Risk of damaging the PLC modules!

- Overvoltages and short circuits might damage the PLC modules.
  - Make sure that all voltage sources (supply voltage and process supply voltage) are switched off before you begin with operations at the system.
  - Never connect any voltages or signals to reserved terminals (marked with ---). Reserved terminals may carry internal voltages.
**NOTICE!**

**Risk of damaging the PLC modules!**

The PLC modules must not be removed while the plant is connected to a power supply.

Make sure that all voltage sources (supply and process voltage) are switched off before you
– connect or disconnect any signal or terminal block
– remove or replace a module.

---

**CAUTION!**

**Risk of imprecise and faulty measurements!**

Analog signals may be distorted seriously by external electromagnetic influences.

Use shielded wires when wiring analog signal sources. The cable shield must be grounded at both ends of the cable. Provide a potential equalization of a low resistance to avoid high potential differences between different parts of the plant.

---

**Fig. 178: Terminal assignment of the module**

The module provides several diagnosis functions ☝️ Chapter 1.5.3.6.3.1.2.7 “Diagnosis” on page 2772.

---

**Connection of the digital outputs**

The following figure shows the electrical connection of the digital output DO0. Proceed with the digital outputs DO1 to DO15 in the same way.
For a description of the meaning of the LEDs, please refer to the Displays chapter "State LEDs" on page 2775.

Connection of the configurable digital inputs/outputs

The following figure shows the electrical connection of the configurable digital input/output DC16 and DC17. DC16 is connected as an input and DC17 is connected as an output. Proceed with the configurable digital inputs/outputs DC18 to DC23 in the same way.
NOTICE!
Risk of influences to the connected sensors!
Some sensors may be influenced by the deactivated module outputs of DA502.
Connect a 470 Ω / 1 W resistor in series to inputs DC16/DC17 if they are used as fast counter inputs to avoid any influences.

For a description of the meaning of the LEDs, please refer to the Displays Chapter 1.5.3.6.3.1.2.8 "State LEDs" on page 2775 chapter.

Connection of resistance thermometers in 2-wire configuration to the analog inputs

When resistance thermometers (Pt100, Pt1000, Ni1000) are used, a constant current must flow through them to build the necessary voltage drop for the evaluation. For this, the module DA502 provides a constant current source which is multiplexed over max. 4 analog input channels.

The following figure shows the connection of resistance thermometers in 2-wire configuration to the analog input AI0. Proceed with the analog inputs AI1 to AI3 in the same way.

![Connection of resistance thermometers in 2-wire configuration to the analog inputs](image)

The following measuring ranges can be configured Chapter 1.5.3.6.3.1.2.6 “Parameterization” on page 2768 Chapter 1.5.3.6.3.1.2.9 “Measuring ranges” on page 2775:

<table>
<thead>
<tr>
<th>Sensor</th>
<th>Measuring Range</th>
<th>Configuration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pt100</td>
<td>-50 °C...+400 °C</td>
<td>2-wire configuration, 1 channel used</td>
</tr>
<tr>
<td>Pt1000</td>
<td>-50 °C...+400 °C</td>
<td>2-wire configuration, 1 channel used</td>
</tr>
<tr>
<td>Ni1000</td>
<td>-50 °C...+150 °C</td>
<td>2-wire configuration, 1 channel used</td>
</tr>
</tbody>
</table>

For a description of the function of the LEDs, please refer to Diagnosis and displays / Displays Chapter 1.5.3.6.3.1.2.8 “State LEDs” on page 2775.

The module DA502 performs a linearization of the resistance characteristic.
To avoid error messages from unused analog input channels, configure them as "unused".
Connection of resistance thermometers in 3-wire configuration to the analog inputs

When resistance thermometers (Pt100, Pt1000, Ni1000) are used, a constant current must flow through them to build the necessary voltage drop for the evaluation. For this, the module DA502 provides a constant current source which is multiplexed over max. 4 analog input channels.

The following figure shows the connection of resistance thermometers in 3-wire configuration to the analog inputs AI0 and AI1. Proceed with the analog inputs AI2 and AI3 in the same way.

**Fig. 180: Connection of resistance thermometers in 3-wire configuration to the analog inputs**

With 3-wire configuration, 2 adjacent analog channels belong together (e.g. the channels 0 and 1). In this case, both channels are configured according to the desired operating mode. The lower address must be the even address (channel 0), the next higher address must be the odd address (channel 1).

The constant current of one channel flows through the resistance thermometer. The constant current of the other channel flows through one of the cores. The module calculates the measured value from the two voltage drops and stores it under the input with the higher channel number (e.g. I1).

In order to keep measuring errors as small as possible, it is necessary to have all the involved conductors in the same cable. All the conductors must have the same cross section.

The following measuring ranges can be configured:

- **Pt100**
  - -50 °C...+400 °C
  - 3-wire configuration, 2 channels used
- **Pt1000**
  - -50 °C...+400 °C
  - 3-wire configuration, 2 channels used
- **Ni1000**
  - -50 °C...+150 °C
  - 3-wire configuration, 2 channels used

For a description of the function of the LEDs, please refer to Diagnosis and displays / Displays “State LEDs” on page 2775.

The module DA502 performs a linearization of the resistance characteristic.

To avoid error messages from unused analog input channels, configure them as "unused".
Connection of active-type analog sensors (Voltage) with electrically isolated power supply to the analog inputs

The following figure shows the connection of active-type analog sensors (voltage) with electrically isolated power supply to the analog input AI0. Proceed with the analog inputs AI1 to AI3 in the same way.

Fig. 181: Connection of active-type analog sensors (voltage) with electrically isolated power supply to the analog inputs

The following measuring ranges can be configured:

<table>
<thead>
<tr>
<th>Voltage</th>
<th>0 V...10 V</th>
<th>1 channel used</th>
</tr>
</thead>
<tbody>
<tr>
<td>Voltage</td>
<td>-10 V...+10 V</td>
<td>1 channel used</td>
</tr>
</tbody>
</table>

For a description of the function of the LEDs, please refer to Diagnosis and displays / Displays.

To avoid error messages from unused analog input channels, configure them as "unused".

Connection of active-type analog sensors (Current) with electrically isolated power supply to the analog inputs

The following figure shows the connection of active-type analog sensors (current) with electrically isolated power supply to the analog input AI0. Proceed with the analog inputs AI1 to AI3 in the same way.
The following measuring ranges can be configured:

- **Current**: 0 mA...20 mA
  - 1 channel used

- **Current**: 4 mA...20 mA
  - 1 channel used

For a description of the function of the LEDs, please refer to Diagnosis and displays / Displays “State LEDs on page 2775.

Unused input channels can be left open-circuited, because they are of low resistance.

**Connection of active-type analog sensors (Voltage) with no electrically isolated power supply to the analog inputs**

The following figure shows the connection of active-type analog sensors (voltage) without electrically isolated power supply to the analog input AI0. Proceed with the analog inputs AI1 to AI3 in the same way.
Fig. 183: Connection of active-type sensors (voltage) with no electrically isolated power supply to the analog inputs

CAUTION!
Risk of faulty measurements!
The negative pole at the sensors must not have too large a potential difference with respect to ZP (max. ±1 V within the full signal range).
Make sure that the potential difference never exceeds ±1 V.

The following measuring ranges can be configured:  
Chapter 1.5.3.6.3.1.2.6 “Parameterization” on page 2768  
Chapter 1.5.3.6.3.1.2.9 “Measuring ranges” on page 2775:

<table>
<thead>
<tr>
<th>Voltage</th>
<th>0 V...10 V</th>
<th>1 channel used</th>
</tr>
</thead>
<tbody>
<tr>
<td>Voltage</td>
<td>-10 V...+10 V</td>
<td>1 channel used</td>
</tr>
</tbody>
</table>

For a description of the function of the LEDs, please refer to Diagnosis and displays / Displays  
Chapter 1.5.3.6.3.1.2.8 “State LEDs” on page 2775.
To avoid error messages from unused analog input channels, configure them as "unused".

Connection of passive-type analog sensors (Current) to the analog inputs
The following figure shows the connection of passive-type analog sensors (current) to the analog input AI0. Proceed with the analog inputs AI1 to AI3 in the same way.
The following measuring ranges can be configured:

**Current**

- 4 mA...20 mA
- 1 channel used

For a description of the function of the LEDs, please refer to Diagnosis and displays / Displays:

**NOTICE!**

Risk of overloading the analog input!

- If an analog current sensor supplies more than 25 mA for more than 1 second during initialization, this input is switched off by the module (input protection).
- Use only sensors with fast initialization or without current peaks higher than 25 mA. If not possible, connect a 10-volt Zener diode in parallel to I+ and I-.

Unused input channels can be left open-circuited, because they are of low resistance.

**Connection of active-type analog sensors (Voltage) to differential analog inputs**

Differential inputs are very useful if analog sensors which are remotely non-isolated (e.g. the negative terminal is remotely earthed) are used.

Using differential inputs helps to considerably increase the measuring accuracy and to avoid earthing loops.

With differential input configurations, two adjacent analog channels belong together (e.g. the channels 0 and 1). In this case, both channels are configured according to the desired operating mode. The lower address must be the even address (channel 0), the next higher address must be the odd address (channel 1). The converted analog value is available at the higher address (channel 1).

The analog value is calculated by subtraction of the input value with the higher address from the input value of the lower address.

The converted analog value is available at the odd channel (higher address).
CAUTION!
Risk of faulty measurements!
The negative pole at the sensors must not have too large a potential difference
with respect to ZP (max. ±1 V within the full signal range).
Make sure that the potential difference never exceeds ±1 V.

The following figure shows the connection of active-type analog sensors (voltage) to differential analog inputs AI0 and AI1. Proceed with AI2 and AI3 in the same way.

![Connection of active-type analog sensors (voltage) to differential analog inputs](image)

**Fig. 185: Connection of active-type analog sensors (voltage) to differential analog inputs**

The following measuring ranges can be configured:

- Voltage 0 V...10 V with differential inputs, 2 channels used
- Voltage -10 V...+10 V with differential inputs, 2 channels used

For a description of the function of the LEDs, please refer to Diagnosis and displays / Displays

Use of analog inputs as digital inputs

Several (or all) analog inputs can be configured as digital inputs. The inputs are not electrically isolated against the other analog channels.

The following figure shows the connection of digital sensors to the analog input AI0. Proceed with the analog inputs AI1 to AI3 in the same way.
Fig. 186: Use of analog inputs as digital inputs

The following measuring ranges can be configured “Parameterization” on page 2768 “Measuring ranges” on page 2775:

<table>
<thead>
<tr>
<th>Digital input</th>
<th>24 V</th>
<th>1 channel used</th>
</tr>
</thead>
</table>

For a description of the function of the LEDs, please refer to Diagnosis and displays “State LEDs” on page 2775.

Connection of analog output loads (Voltage)

The following figure shows the connection of output loads to the analog output AO0. Proceed with the analog output AO1 in the same way.
The following measuring ranges can be configured:

- Voltage: -10 V...+10 V
- Load: ±10 mA max.
- 1 channel used

For a description of the function of the LEDs, please refer to Diagnosis and displays / Displays “State LEDs” on page 2775.

Unused analog outputs can be left open-circuited.

Connection of analog output loads (Current)

The following figure shows the connection of output loads to the analog output AO0. Proceed with the analog output AO1 in the same way.
The following measuring ranges can be configured 

<table>
<thead>
<tr>
<th>Current</th>
<th>0 mA...20 mA</th>
<th>Load 0 Ω...500 Ω</th>
<th>1 channel used</th>
</tr>
</thead>
<tbody>
<tr>
<td>Current</td>
<td>4 mA...20 mA</td>
<td>Load 0 Ω...500 Ω</td>
<td>1 channel used</td>
</tr>
</tbody>
</table>

For a description of the function of the LEDs, please refer to Diagnosis and displays / Displays 

<table>
<thead>
<tr>
<th>Current</th>
<th>0 mA...20 mA</th>
<th>Load 0 Ω...500 Ω</th>
<th>1 channel used</th>
</tr>
</thead>
<tbody>
<tr>
<td>Current</td>
<td>4 mA...20 mA</td>
<td>Load 0 Ω...500 Ω</td>
<td>1 channel used</td>
</tr>
</tbody>
</table>

For a description of the function of the LEDs, please refer to Diagnosis and displays / Displays 

| Chapter 1.5.3.6.3.1.2.8 “State LEDs” on page 2775.

Unused analog outputs can be left open-circuited.

### Internal data exchange

<table>
<thead>
<tr>
<th></th>
<th>Without the Fast Counter</th>
<th>With the Fast Counter (only with AC500)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Digital inputs (bytes)</td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>Digital outputs (bytes)</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>Analog inputs (words)</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>Analog outputs (words)</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>Counter input data (words)</td>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td>Counter output data (words)</td>
<td>0</td>
<td>8</td>
</tr>
</tbody>
</table>

### I/O configuration

The module itself does not store configuration data. It draws its parameterization data from the master device of the I/O bus (CPU or bus module) during power-up of the system.

Hence, replacing I/O modules is possible without any re-parameterization via software.
If the external power supply voltage via UP/ZP terminals fails, the I/O module loses its configuration data. The whole station has to be switched off and on again to re-configure the module.

Parameterization

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Module ID 1)</td>
<td>Internal</td>
<td>Internal value 1815, type WORD, Default 1815, EDS Slot / Index 0x0Y01</td>
</tr>
<tr>
<td>Ignore module</td>
<td>Internal</td>
<td>Internal value Yes, type BYTE, Default No</td>
</tr>
<tr>
<td>Parameter length</td>
<td>Internal</td>
<td>Internal value 8, type BYTE, Default 8, EDS Slot / Index 0x0Y02</td>
</tr>
<tr>
<td>Check supply</td>
<td>off</td>
<td>Internal value 0, type BYTE, Default 1, EDS Slot / Index 0x0Y03</td>
</tr>
<tr>
<td>Fast counter 2)</td>
<td>0 : 10</td>
<td>Internal value 0, type BYTE, Default 0, EDS Slot / Index Not for FBP</td>
</tr>
<tr>
<td>Behavior outputs at comm. error 5)</td>
<td>Off Last value 5 s, Last value 10 s Substitute value 5 s, Substitute value 10 s</td>
<td>Off Last value Off 0x00, EDS Slot / Index 0x0Y07</td>
</tr>
</tbody>
</table>

2) Setting:
- **On**: Error LED lights up at errors of all error classes, Failsafe mode off
- **Off by E4**: Error LED lights up at errors of error classes E1, E2 and E3, Failsafe mode off
- **Off by E3**: Error LED lights up at errors of error classes E1 and E2, Failsafe mode off
<table>
<thead>
<tr>
<th>Setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>On +Failsafe</td>
<td>Error LED lights up at errors of all error classes, Failsafe mode on *)</td>
</tr>
<tr>
<td>Off by E4 + Failsafe</td>
<td>Error LED lights up at errors of error classes E1, E2 and E3, Failsafe mode on *)</td>
</tr>
<tr>
<td>Off by E3 + Failsafe</td>
<td>Error LED lights up at errors of error classes E1 and E2, Failsafe mode on *)</td>
</tr>
</tbody>
</table>

1) With a faulty ID, the module reports a "parameter error" and does not perform cyclic process data transmission

2) For a description of the counter operating modes, please refer to the Fast Counter section  
   Chapter 1.5.3.6.1.2.9 “Fast counter” on page 2543

3) With CS31 without the parameter "Fast Counter"

The fast counter of the module does not work if the module is connected to a CS31 bus module.

5) The parameter Behavior outputs at comm. error is only analyzed if the Failsafe mode is ON.

### Group parameters for the digital part

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>Internal value</th>
<th>Internal value, type</th>
<th>Default</th>
<th>EDS Slot / Index</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input delay</td>
<td>0.1 ms</td>
<td>0</td>
<td>BYTE</td>
<td>0.1 ms</td>
<td>0x0Y05</td>
</tr>
<tr>
<td></td>
<td>1 ms</td>
<td>1</td>
<td></td>
<td>0x00</td>
<td></td>
</tr>
<tr>
<td></td>
<td>8 ms</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>32 ms</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Detect short circuit at out-</td>
<td>Off</td>
<td>0</td>
<td>BYTE</td>
<td>On</td>
<td>0x0Y06</td>
</tr>
<tr>
<td>puts</td>
<td>On</td>
<td>1</td>
<td></td>
<td>0x01</td>
<td></td>
</tr>
<tr>
<td>Substitute value at output</td>
<td>0...255</td>
<td>00h...FFh</td>
<td>BYTE</td>
<td>0</td>
<td>0x0Y08</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0x0000</td>
<td></td>
</tr>
</tbody>
</table>

* The parameters Behavior DO at comm. error is only analyzed if the Failsafe mode is ON.

### Group parameters for the analog part

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>Internal value</th>
<th>Internal value, type</th>
<th>Default</th>
<th>EDS Slot / Index</th>
</tr>
</thead>
<tbody>
<tr>
<td>Analog data format</td>
<td>Standard Reserved</td>
<td>0 255</td>
<td>BYTE</td>
<td>0</td>
<td>0x0Y04</td>
</tr>
</tbody>
</table>

* The parameter Behaviour AO at comm. error is only analyzed if the Failsafe mode is ON.
Channel parameters for the analog inputs (4x)

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>Internal value</th>
<th>Internal value, type</th>
<th>Default</th>
<th>EDS Slot / Index</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input 0, Channel configuration</td>
<td>see Table 588 “Channel configuration” on page 2770</td>
<td>see Table 588 “Channel configuration” on page 2770</td>
<td>BYTE</td>
<td>0</td>
<td>0x0Y09</td>
</tr>
<tr>
<td>Input 0, Check channel</td>
<td>see Table 589 “Channel monitoring” on page 2771</td>
<td>see Table 589 “Channel monitoring” on page 2771</td>
<td>BYTE</td>
<td>0</td>
<td>0x0Y0A</td>
</tr>
<tr>
<td>Input 3, Channel configuration</td>
<td>see Table 588 “Channel configuration” on page 2770</td>
<td>see Table 588 “Channel configuration” on page 2770</td>
<td>BYTE</td>
<td>0</td>
<td>0x0Y0F</td>
</tr>
<tr>
<td>Input 3, Check channel</td>
<td>see Table 589 “Channel monitoring” on page 2771</td>
<td>see Table 589 “Channel monitoring” on page 2771</td>
<td>BYTE</td>
<td>0</td>
<td>0x0Y10</td>
</tr>
</tbody>
</table>

Table 588: Channel configuration

<table>
<thead>
<tr>
<th>Internal value</th>
<th>Operating modes of the analog inputs, individually configurable</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 (default)</td>
<td>Not used</td>
</tr>
<tr>
<td>1</td>
<td>0 V...10 V</td>
</tr>
<tr>
<td>2</td>
<td>Digital input</td>
</tr>
<tr>
<td>3</td>
<td>0 mA...20 mA</td>
</tr>
<tr>
<td>4</td>
<td>4 mA...20 mA</td>
</tr>
<tr>
<td>5</td>
<td>-10 V...+10 V</td>
</tr>
<tr>
<td>8</td>
<td>2-wire Pt100 -50 °C...+400 °C</td>
</tr>
<tr>
<td>9</td>
<td>3-wire Pt100 -50 °C...+400 °C</td>
</tr>
<tr>
<td>10</td>
<td>0 V...10 V (voltage diff.)</td>
</tr>
<tr>
<td>11</td>
<td>-10 V...+10 V (voltage diff.)</td>
</tr>
<tr>
<td>14</td>
<td>2-wire Pt100 -50 °C...+70 °C</td>
</tr>
<tr>
<td>15</td>
<td>3-wire Pt100 -50 °C...+70 °C</td>
</tr>
<tr>
<td>16</td>
<td>2-wire Pt1000 -50 °C...+400 °C</td>
</tr>
<tr>
<td>17</td>
<td>3-wire Pt1000 -50 °C...+400 °C</td>
</tr>
<tr>
<td>18</td>
<td>2-wire Ni1000 -50 °C...+150 °C</td>
</tr>
</tbody>
</table>
### Internal value

<table>
<thead>
<tr>
<th>Internal value</th>
<th>Operating modes of the analog inputs, individually configurable</th>
</tr>
</thead>
<tbody>
<tr>
<td>19</td>
<td>3-wire Ni1000 -50 °C...+150 °C *)</td>
</tr>
</tbody>
</table>

*) In the operating modes with 3-wire configuration or with differential inputs, two adjacent analog inputs belong together (e.g. the channels 0 and 1). In these cases, both channels are configured in the desired operating mode. The lower address must be the even address (channel 0). The next higher address must be the odd address (channel 1). The converted analog value is available at the higher address (channel 1).

### Table 589: Channel monitoring

<table>
<thead>
<tr>
<th>Internal Value</th>
<th>Check Channel</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 (default)</td>
<td>Plausibility, cut wire, short circuit</td>
</tr>
<tr>
<td>3</td>
<td>Not used</td>
</tr>
</tbody>
</table>

### Channel parameters for the analog outputs (2x)

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>Internal value</th>
<th>Internal value, type</th>
<th>Default</th>
<th>EDS Slot / Index</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output 0, Channel configuration</td>
<td>see Table 590 “Channel configuration” on page 2772</td>
<td>see Table 590 “Channel configuration” on page 2772</td>
<td>BYTE</td>
<td>0</td>
<td>0x0Y11</td>
</tr>
<tr>
<td>Output 0, Check channel</td>
<td>see Table 591 “Channel monitoring” on page 2772</td>
<td>see Table 591 “Channel monitoring” on page 2772</td>
<td>BYTE</td>
<td>0</td>
<td>0x0Y12</td>
</tr>
<tr>
<td>Output 0, Substitute value</td>
<td>see Table 592 “Substitute value” on page 2772</td>
<td>see Table 592 “Substitute value” on page 2772</td>
<td>WORD</td>
<td>0</td>
<td>0x0Y13</td>
</tr>
<tr>
<td>Output 1, Channel configuration</td>
<td>see Table 590 “Channel configuration” on page 2772</td>
<td>see Table 590 “Channel configuration” on page 2772</td>
<td>BYTE</td>
<td>0</td>
<td>0x0Y14</td>
</tr>
<tr>
<td>Output 1, Check channel</td>
<td>see Table 591 “Channel monitoring” on page 2772</td>
<td>see Table 591 “Channel monitoring” on page 2772</td>
<td>BYTE</td>
<td>0</td>
<td>0x0Y15</td>
</tr>
<tr>
<td>Output 1, Substitute value</td>
<td>see Table 592 “Substitute value” on page 2772</td>
<td>see Table 592 “Substitute value” on page 2772</td>
<td>WORD</td>
<td>0</td>
<td>0x0Y16</td>
</tr>
</tbody>
</table>
Table 590: Channel configuration

<table>
<thead>
<tr>
<th>Internal value</th>
<th>Operating modes of the analog outputs, individually configurable</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 (default)</td>
<td>Not used</td>
</tr>
<tr>
<td>128</td>
<td>-10 V...+10 V</td>
</tr>
<tr>
<td>129</td>
<td>0 mA...20 mA</td>
</tr>
<tr>
<td>130</td>
<td>4 mA...20 mA</td>
</tr>
</tbody>
</table>

Table 591: Channel monitoring

<table>
<thead>
<tr>
<th>Internal value</th>
<th>Check channel</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Plausibility, cut wire, short circuit</td>
</tr>
<tr>
<td>3</td>
<td>None</td>
</tr>
</tbody>
</table>

Table 592: Substitute value

<table>
<thead>
<tr>
<th>Intended behavior of output channel when the control system stops</th>
<th>Required setting of the module parameter &quot;Behavior of outputs in case of a communication error&quot;</th>
<th>Required setting of the channel parameter &quot;Substitute value&quot;</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output OFF</td>
<td>Off</td>
<td>0</td>
</tr>
<tr>
<td>Last value infinite</td>
<td>Last value</td>
<td>0</td>
</tr>
<tr>
<td>Last value for 5 s and then turn off</td>
<td>Last value 5 s</td>
<td>0</td>
</tr>
<tr>
<td>Last value for 10 s and then turn off</td>
<td>Last value 10 s</td>
<td>0</td>
</tr>
<tr>
<td>Substitute value infinite</td>
<td>Substitute value</td>
<td>Depending on configuration</td>
</tr>
<tr>
<td>Substitute value for 5 s and then turn off</td>
<td>Substitute value 5 s</td>
<td>Depending on configuration</td>
</tr>
<tr>
<td>Substitute value for 10 s and then turn off</td>
<td>Substitute value 10 s</td>
<td>Depending on configuration</td>
</tr>
</tbody>
</table>

Diagnosis

In cases of short circuit or overload, the digital outputs are turned off. The module performs reactivation automatically. Thus, an acknowledgement of the errors is not necessary. The error message is stored via the LED.
<table>
<thead>
<tr>
<th>Class</th>
<th>Interface</th>
<th>Device</th>
<th>Module</th>
<th>Channel</th>
<th>Error Identifier</th>
<th>Error message</th>
<th>Remedy</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1)</td>
<td>2)</td>
<td>3)</td>
<td>4)</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Module error**

| 3     | 14       | 1...10 | 31     | 31      | 19               | Checksum error in the I/O module | Replace I/O module |
| 3     | 14       | 1...10 | 31     | 31      | 3                | Timeout in the I/O module       |                    |
| 3     | 14       | 1...10 | 31     | 31      | 40               | Different hard-/firmware versions in the module |                    |
| 3     | 14       | 1...10 | 31     | 31      | 43               | Internal error in the module     |                    |
| 3     | 14       | 1...10 | 31     | 31      | 36               | Internal data exchange failure   |                    |
| 3     | 14       | 1...10 | 31     | 31      | 9                | Overflow diagnosis buffer        | New start           |
| 3     | 14       | 1...10 | 31     | 31      | 26               | Parameter error                 | Check master        |
| 3     | 14       | 1...10 | 31     | 31      | 11               | Process voltage too low          | Check process voltage |

**Channel error DA502**

| 4     | 14       | 1...10 | 2      | 0...15   | 47               | Short-circuit at a digital output | Check connection |
| 4     | 11 / 12  | ADR     | 1...10 | 0...15   | 47               | Short-circuit at a digital output | Check connection |

**Channel error DA502**

| 4     | 14       | 1...10 | 1      | 16...19   | 48               | Analog value overflow or broken wire at an analog input | Check input value or terminal |
| 4     | 11 / 12  | ADR     | 1...10 | 16...19   | 48               | Analog value overflow or broken wire at an analog input | Check input value or terminal |
| 4     | 11 / 12  | ADR     | 1...10 | 16...19   | 47               | Analog value underflow at an analog input              | Check input value |
| 4     | 11 / 12  | ADR     | 1...10 | 16...19   | 47               | Analog value underflow at an analog input              | Check input value |
| 4     | 11 / 12  | ADR     | 1...10 | 20...21   | 4                | Analog value overflow at an analog output              | Check output value |
| 4     | 11 / 12  | ADR     | 1...10 | 20...21   | 4                | Analog value overflow at an analog output              | Check output value |
### PLC Automation with V3 CPUs
#### PLC integration > Device specifications

<table>
<thead>
<tr>
<th>E1...E4</th>
<th>d1</th>
<th>d2</th>
<th>d3</th>
<th>d4</th>
<th>Identifier 000...063</th>
<th>AC500 display</th>
<th>&lt;-- Display in</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class</td>
<td>Comp</td>
<td>Dev</td>
<td>Mod</td>
<td>Ch</td>
<td>Err</td>
<td>PS501 PLC browser</td>
<td></td>
</tr>
<tr>
<td>Byte 6</td>
<td>-</td>
<td>Byte 3</td>
<td>Byte 4</td>
<td>Byte 5</td>
<td>Byte 6 Bit 0...5</td>
<td>FBP diagnosis block</td>
<td></td>
</tr>
<tr>
<td>Bit 6...7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Class</th>
<th>Interface</th>
<th>Device</th>
<th>Module</th>
<th>Channel</th>
<th>Error Identifier</th>
<th>Error message</th>
<th>Remedy</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>14</td>
<td>1...10</td>
<td>3</td>
<td>20...21</td>
<td>7</td>
<td>Analog value underflow at an analog output</td>
<td>Check output value</td>
</tr>
<tr>
<td></td>
<td>11 / 12</td>
<td>ADR</td>
<td>1...10</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Remarks:**

1) In AC500, the following interface identifier applies:
   14 = I/O bus, 11 = COM1 (e.g. CS31 bus), 12 = COM2.

2) With "Device" the following allocation applies:
   31 = module itself,
   1...10 = decentralized communication interface module 1...10,
   ADR = hardware address (e.g. of the DC551)

3) With "Module" the following allocation applies depending on the master:
   Module error: I/O bus: 31 = Module itself; COM1/COM2: 1...10 = expansion 1...10
   Channel error: I/O bus = module type (1 = AI, 3 = AO, 4 = DC); COM1/COM2: 1...10 = expansion 1...10

4) In case of module errors, with channel "31 = module itself" is output.

5) Ch = 22...29 indicate the digital inputs/outputs DC16...DC23

6) Ch = 16...19 indicates the analog inputs AI0...AI3

7) Ch = 20...21 indicates the analog outputs AO0...AO1
### State LEDs

<table>
<thead>
<tr>
<th>LED</th>
<th>State</th>
<th>Color</th>
<th>LED = OFF</th>
<th>LED = ON</th>
<th>LED flashes</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>Output is OFF</td>
<td>Output is ON</td>
<td>--</td>
</tr>
<tr>
<td>Do0 to Do15</td>
<td>Digital output</td>
<td>Yellow</td>
<td></td>
<td></td>
<td>--</td>
</tr>
<tr>
<td></td>
<td>DC16 to DC23</td>
<td>Digital input/output</td>
<td>Yellow</td>
<td>Input/output is OFF</td>
<td>Input/output is ON ¹)</td>
</tr>
<tr>
<td></td>
<td>Ai0 to Ai3</td>
<td>Analog input</td>
<td>Yellow</td>
<td>Input is OFF</td>
<td>Input is ON ²)</td>
</tr>
<tr>
<td></td>
<td>Ao0 to Ao1</td>
<td>Analog output</td>
<td>Yellow</td>
<td>Output is OFF</td>
<td>Output is ON ²)</td>
</tr>
<tr>
<td></td>
<td>UP</td>
<td>Process supply voltage 24 VDC via terminal</td>
<td>Green</td>
<td>Process supply voltage is missing</td>
<td>Process supply voltage OK</td>
</tr>
<tr>
<td></td>
<td>CH-ERR1</td>
<td>Channel error, error messages in groups (digital inputs/outputs combined into the groups 1, 2, 3, 4)</td>
<td>Red</td>
<td>No error or process supply voltage is missing</td>
<td>Severe error within the corresponding group</td>
</tr>
<tr>
<td></td>
<td>CH-ERR2</td>
<td></td>
<td>Red</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>CH-ERR3</td>
<td></td>
<td>Red</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>CH-ERR4</td>
<td></td>
<td>Red</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>CH-ERR ³)</td>
<td>Module error</td>
<td>Red</td>
<td></td>
<td>Internal error</td>
</tr>
</tbody>
</table>

¹) Indication LED is ON even if an input signal is applied to the channel and the supply voltage is off. In this case the module is not operating and does not generate an input signal.

²) Brightness depends on the value of the analog signal

³) All of the LEDs CH-ERR1 to CH-ERR4 light up together

### Measuring ranges

#### Input ranges voltage, current and digital input

<table>
<thead>
<tr>
<th>Range</th>
<th>0...10 V</th>
<th>-10...+10 V</th>
<th>0...20 mA</th>
<th>4...20 mA</th>
<th>Digital input</th>
<th>Digital value</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Decimal</td>
<td>Hex.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Overflow</td>
<td>&gt; 11.7589</td>
<td>32767</td>
<td>7FFF</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Measured value too high</td>
<td>11.7589</td>
<td>32511</td>
<td>7EFF</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Normal range</td>
<td>10.0000</td>
<td>27649</td>
<td>6C01</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Normal range or measured value too low</td>
<td>0.0000</td>
<td>27648</td>
<td>6C00</td>
<td>1</td>
<td>0001</td>
<td>0000</td>
</tr>
<tr>
<td>Range</td>
<td>0...10 V</td>
<td>-10...+10 V</td>
<td>0...20 mA</td>
<td>4...20 mA</td>
<td>Digital input</td>
<td>Digital value</td>
</tr>
<tr>
<td>------------</td>
<td>----------</td>
<td>-------------</td>
<td>-----------</td>
<td>-----------</td>
<td>---------------</td>
<td>---------------</td>
</tr>
<tr>
<td></td>
<td>Decimal</td>
<td>Hex.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>-0.0004</td>
<td>-0.0004</td>
<td>3.9994</td>
<td>-1</td>
<td>FFFF</td>
<td></td>
</tr>
<tr>
<td></td>
<td>-1.7593</td>
<td></td>
<td></td>
<td>-4864</td>
<td>ED00</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>-6912</td>
<td>E500</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>-27648</td>
<td>9400</td>
<td></td>
</tr>
<tr>
<td>Measured value too low</td>
<td>-10.0004</td>
<td>-11.7589</td>
<td>&lt; 0.0000</td>
<td>&lt; 0.0000</td>
<td>93FF</td>
<td>8100</td>
</tr>
<tr>
<td>Underflow</td>
<td>&lt; 0.0000</td>
<td>&lt; 11.7589</td>
<td>&lt; 0.0000</td>
<td>&lt; 0.0000</td>
<td>32768</td>
<td>8000</td>
</tr>
</tbody>
</table>

The represented resolution corresponds to 16 bits.

**Input ranges resistance temperature detector**

<table>
<thead>
<tr>
<th>Range</th>
<th>Pt100 / Pt1000 -50...70 °C</th>
<th>Pt100 / Pt1000 -50...400 °C</th>
<th>Ni1000 -50...150 °C</th>
<th>Digital value</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Decimal</td>
<td>Hex.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Overflow</td>
<td>&gt; 80.0 °C</td>
<td>&gt; 450.0 °C</td>
<td>&gt; 160.0 °C</td>
<td>32767</td>
</tr>
<tr>
<td>Measured value too high</td>
<td>450.0 °C</td>
<td>400.1 °C</td>
<td>160.0 °C</td>
<td>150.1 °C</td>
</tr>
<tr>
<td></td>
<td>:</td>
<td>: 4001</td>
<td>: 1600</td>
<td>0640</td>
</tr>
<tr>
<td></td>
<td>:</td>
<td>:</td>
<td>: 1501</td>
<td>05DD</td>
</tr>
<tr>
<td></td>
<td>:</td>
<td>:</td>
<td>:</td>
<td></td>
</tr>
<tr>
<td>80.0 °C</td>
<td>:</td>
<td>:</td>
<td>: 800</td>
<td>0320</td>
</tr>
<tr>
<td>70.1 °C</td>
<td>:</td>
<td>:</td>
<td>: 701</td>
<td>02BD</td>
</tr>
<tr>
<td>Normal range</td>
<td>: 400.0 °C</td>
<td>: 150.0 °C</td>
<td>: 4000</td>
<td>0FA0</td>
</tr>
<tr>
<td></td>
<td>:</td>
<td>:</td>
<td>: 1500</td>
<td>05DC</td>
</tr>
<tr>
<td></td>
<td>:</td>
<td>:</td>
<td>: 700</td>
<td>02BC</td>
</tr>
<tr>
<td></td>
<td>:</td>
<td>:</td>
<td>: 1</td>
<td>0001</td>
</tr>
<tr>
<td></td>
<td>:</td>
<td>:</td>
<td>:</td>
<td></td>
</tr>
<tr>
<td>0.0 °C</td>
<td>: 0.0 °C</td>
<td>: 0.0 °C</td>
<td>: 0</td>
<td>0000</td>
</tr>
<tr>
<td></td>
<td>:</td>
<td>:</td>
<td>:</td>
<td></td>
</tr>
<tr>
<td>-0.1 °C</td>
<td>: -0.1 °C</td>
<td>: -0.1 °C</td>
<td>: -1</td>
<td>FFFF</td>
</tr>
<tr>
<td></td>
<td>:</td>
<td>:</td>
<td>:</td>
<td></td>
</tr>
<tr>
<td>-50.0 °C</td>
<td>: -50.0 °C</td>
<td>: -50.0 °C</td>
<td>: -500</td>
<td>FE0C</td>
</tr>
</tbody>
</table>
### Range

<table>
<thead>
<tr>
<th>Range</th>
<th>Pt100 / Pt1000</th>
<th>Pt100 / Pt1000</th>
<th>Ni1000</th>
<th>Digital value</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>-50...70 °C</td>
<td>-50...400 °C</td>
<td>-50...150 °C</td>
<td></td>
</tr>
<tr>
<td>Measured value too low</td>
<td>-50.1 °C</td>
<td>-50.1 °C</td>
<td>-50.1 °C</td>
<td>-501 FE0B</td>
</tr>
<tr>
<td></td>
<td>-60.0 °C</td>
<td>-60.0 °C</td>
<td>-60.0 °C</td>
<td>-600 FDA8</td>
</tr>
<tr>
<td>Underflow</td>
<td>&lt; -60.0 °C</td>
<td>&lt; -60.0 °C</td>
<td>&lt; -60.0 °C</td>
<td>-32768 8000</td>
</tr>
</tbody>
</table>

### Output ranges voltage and current

<table>
<thead>
<tr>
<th>Range</th>
<th>-10...+10 V</th>
<th>0...20 mA</th>
<th>4...20 mA</th>
<th>Digital value</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Decimal</td>
<td>Hex.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Overflow</td>
<td>0 V</td>
<td>0 mA</td>
<td>0 mA</td>
<td>&gt; 32511</td>
</tr>
<tr>
<td>Measured value too high</td>
<td>11.7589 V</td>
<td>23.5178 mA</td>
<td>22.8142 mA</td>
<td>32511 7EFF</td>
</tr>
<tr>
<td></td>
<td>10.0004 V</td>
<td>20.0007 mA</td>
<td>20.0006 mA</td>
<td>27649 6C01</td>
</tr>
<tr>
<td>Normal range</td>
<td>10.0000 V</td>
<td>20.0000 mA</td>
<td>20.0000 mA</td>
<td>27648 6C00</td>
</tr>
<tr>
<td></td>
<td>0.0000 V</td>
<td>0.0000 mA</td>
<td>0.0000 mA</td>
<td>0 0000</td>
</tr>
<tr>
<td></td>
<td>-0.0004 V</td>
<td>0 mA</td>
<td>3.9994 mA</td>
<td>-1 FFFF</td>
</tr>
<tr>
<td></td>
<td>-10.0000 V</td>
<td>0 mA</td>
<td>0 mA</td>
<td>-6912 E500</td>
</tr>
<tr>
<td>Measured value too low</td>
<td>-10.0004 V</td>
<td>0 mA</td>
<td>0 mA</td>
<td>-27649 93FF</td>
</tr>
<tr>
<td></td>
<td>-11.7589 V</td>
<td>0 mA</td>
<td>0 mA</td>
<td>-32512 8100</td>
</tr>
<tr>
<td>Underflow</td>
<td>0 V</td>
<td>0 mA</td>
<td>0 mA</td>
<td>&lt; -32512 &lt; 8100</td>
</tr>
</tbody>
</table>

The represented resolution corresponds to 16 bits.

### Technical data

#### Technical data of the module

The system data of AC500 and S500 \(\text{Chapter 1.5.4.5.1 “System data AC500” on page 3058}\) are valid for standard version.

The system data of AC500-XC \(\text{Chapter 1.5.4.6.1 “System data AC500-XC” on page 3099}\) are valid for the XC version.

Only additional details are therefore documented below.

The technical data are also valid for the XC version.
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Process supply voltage</td>
<td></td>
</tr>
<tr>
<td>Connections</td>
<td>Terminals 1.8, 2.8, 3.8 and 4.8 for UP (+24 VDC) and 1.9, 2.9, 3.9 and 4.9 for ZP (0 V)</td>
</tr>
<tr>
<td>Protection against reverse voltage</td>
<td>yes</td>
</tr>
<tr>
<td>Rated protection fuse at UP</td>
<td>10 A fast</td>
</tr>
<tr>
<td>Rated value</td>
<td>24 VDC</td>
</tr>
<tr>
<td>Max. ripple</td>
<td>5 %</td>
</tr>
<tr>
<td>Current consumption</td>
<td></td>
</tr>
<tr>
<td>From UP</td>
<td>0.07 A + max. 0.5 A per output</td>
</tr>
<tr>
<td>From 24 VDC power supply at the terminals UP/L+ and ZP/M of the CPU/bus module</td>
<td>ca. 2 mA</td>
</tr>
<tr>
<td>Inrush current from UP (at power-up)</td>
<td>0.04 A²s</td>
</tr>
<tr>
<td>Galvanic isolation</td>
<td>Yes, per module</td>
</tr>
<tr>
<td>Max. power dissipation within the module</td>
<td>6 W (outputs unloaded)</td>
</tr>
<tr>
<td>Weight (without terminal unit)</td>
<td>ca. 125 g</td>
</tr>
<tr>
<td>Mounting position</td>
<td>Horizontal mounting or vertical with derating (output load reduced to 50% at 40 °C)</td>
</tr>
<tr>
<td>Cooling</td>
<td>The natural convection cooling must not be hindered by cable ducts or other parts in the switch-gear cabinet.</td>
</tr>
</tbody>
</table>

**NOTICE!**

**Attention:**

All I/O channels (digital and analog) are protected against reverse polarity, reverse supply, short circuit and continuous overvoltage up to 30 VDC.

**Multiple overloads**

No effects of multiple overloads on isolated multi-channel modules occur, as every channel is protected individually by an internal smart high-side switch.

### Technical data of the digital outputs

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of channels per module</td>
<td>16 outputs (with transistors)</td>
</tr>
<tr>
<td>Distribution of the channels into groups</td>
<td>1 group of 16 channels</td>
</tr>
<tr>
<td>Connection of the channels</td>
<td></td>
</tr>
<tr>
<td>DO0 to DO7</td>
<td>Terminals 1.0 to 1.7</td>
</tr>
<tr>
<td>DO8 to DO15</td>
<td>Terminals 2.0 to 2.7</td>
</tr>
<tr>
<td>Indication of the output signals</td>
<td>1 yellow LED per channel, the LED is ON if the output signal is high (signal 1)</td>
</tr>
<tr>
<td>Monitoring point of output indicator</td>
<td>LED is controlled by process CPU</td>
</tr>
</tbody>
</table>
### Technical data of the configurable digital inputs/outputs

Each of the configurable digital I/O channels can be defined as input or output by the user program. This is done by interrogating or allocating the corresponding channel.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of channels per module</td>
<td>8 inputs/outputs (with transistors)</td>
</tr>
<tr>
<td>Distribution of the channels into groups</td>
<td>1 group for 8 channels</td>
</tr>
<tr>
<td>If the channels are used as inputs</td>
<td></td>
</tr>
<tr>
<td>Channels DC16...DC23</td>
<td>Terminals 4.0...4.7</td>
</tr>
<tr>
<td>If the channels are used as outputs</td>
<td></td>
</tr>
<tr>
<td>Channels DC16...DC23</td>
<td>Terminals 4.0...4.7</td>
</tr>
<tr>
<td>Indication of the input/output signals</td>
<td>1 yellow LED per channel, the LED is ON when the input/output signal is high (signal 1)</td>
</tr>
<tr>
<td>Monitoring point of input/output indicator</td>
<td>LED is part of the input circuitry</td>
</tr>
<tr>
<td>Galvanic isolation</td>
<td>Yes, per module</td>
</tr>
</tbody>
</table>
Technical data of the digital inputs/outputs if used as inputs

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of channels per module</td>
<td>8</td>
</tr>
<tr>
<td>Distribution of the channels into groups</td>
<td>1 group of 8 channels</td>
</tr>
<tr>
<td>Terminals of the channels DC16 to DC23</td>
<td>Terminals 4.0 to 4.7</td>
</tr>
<tr>
<td>Reference potential for all inputs</td>
<td>Terminals 1.9...4.9 (Negative pole of the supply voltage, signal name ZP)</td>
</tr>
<tr>
<td>Indication of the input signals</td>
<td>1 yellow LED per channel, the LED is ON when the input signal is high (signal 1)</td>
</tr>
<tr>
<td>Monitoring point of input/output indicator</td>
<td>LED is part of the input circuitry</td>
</tr>
<tr>
<td>Input type (according EN 61131-2)</td>
<td>Type 1</td>
</tr>
<tr>
<td>Input delay (0-&gt;1 or 1-&gt;0)</td>
<td>Typ. 0.1 ms, configurable from 0.1...32 ms</td>
</tr>
<tr>
<td>Input signal voltage</td>
<td>24 VDC</td>
</tr>
<tr>
<td>0-Signal</td>
<td>-3 V...+5 V</td>
</tr>
<tr>
<td>Undefined Signal</td>
<td>&gt; +5 V...&lt; +15 V</td>
</tr>
<tr>
<td>1-Signal</td>
<td>+15 V...+30 V</td>
</tr>
<tr>
<td>Ripple with signal 0</td>
<td>Within -3 V...+5 V</td>
</tr>
<tr>
<td>Ripple with signal 1</td>
<td>Within +15 V...+30 V</td>
</tr>
<tr>
<td>Input current per channel</td>
<td></td>
</tr>
<tr>
<td>Input voltage +24 V</td>
<td>Typ. 5 mA</td>
</tr>
<tr>
<td>Input voltage +5 V</td>
<td>&gt; 1 mA</td>
</tr>
<tr>
<td>Input voltage +15 V</td>
<td>&gt; 2 mA</td>
</tr>
<tr>
<td>Input voltage +30 V</td>
<td>&lt; 8 mA</td>
</tr>
<tr>
<td>Max. cable length</td>
<td></td>
</tr>
<tr>
<td>Shielded</td>
<td>1000 m</td>
</tr>
<tr>
<td>Unshielded</td>
<td>600 m</td>
</tr>
</tbody>
</table>

* Due to the direct connection to the output, the demagnetizing varistor is also effective at the input (see figure) above. This is why the difference between UPx and the input signal must not exceed the clamp voltage of the varistor. The varistor limits the clamp voltage to approx. 36 V. Consequently, the input voltage must range from -12 V to +30 V when UPx = 24 V and from -6 V to +30 V when UPx = 30 V.

Technical data of the digital inputs/outputs if used as outputs

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of channels per module</td>
<td>8</td>
</tr>
<tr>
<td>Distribution of the channels into groups</td>
<td>1 group of 8 channels</td>
</tr>
<tr>
<td>Terminals of the channels DC16 to DC23</td>
<td>Terminals 4.0 to 4.7</td>
</tr>
<tr>
<td>Reference potential for all outputs</td>
<td>Terminals 1.9...4.9 (negative pole of the supply voltage, signal name ZP)</td>
</tr>
<tr>
<td>Common power supply voltage</td>
<td>For all outputs terminals 1.8, 2.8, 3.8 and 4.8 (positive pole of the supply voltage, signal name UP)</td>
</tr>
<tr>
<td>Output voltage for signal 1</td>
<td>UP (-0.8 V)</td>
</tr>
<tr>
<td>Parameter</td>
<td>Value</td>
</tr>
<tr>
<td>-----------------------------------------------</td>
<td>-------------------------------------------------</td>
</tr>
<tr>
<td>Output delay (0-&gt;1 or 1-&gt;0)</td>
<td>On request</td>
</tr>
<tr>
<td>Output current</td>
<td></td>
</tr>
<tr>
<td>rated value per channel</td>
<td>500 mA at UP = 24 V</td>
</tr>
<tr>
<td>max. value (all channels together)</td>
<td>4 A</td>
</tr>
<tr>
<td>Leakage current with signal 0</td>
<td>&lt; 0.5 mA</td>
</tr>
<tr>
<td>Fuse for UP</td>
<td>10 A fast</td>
</tr>
<tr>
<td>Demagnetization with inductive DC load</td>
<td>Via internal varistors (see figure below this table)</td>
</tr>
<tr>
<td>Output switching frequency</td>
<td></td>
</tr>
<tr>
<td>With resistive load</td>
<td>On request</td>
</tr>
<tr>
<td>With inductive loads</td>
<td>Max. 0.5 Hz</td>
</tr>
<tr>
<td>With lamp loads</td>
<td>11 Hz max. at 5 W max.</td>
</tr>
<tr>
<td>Short-circuit-proof / overload-proof</td>
<td>Yes</td>
</tr>
<tr>
<td>Overload message (I &gt; 0.7 A)</td>
<td>Yes, after ca. 100 ms</td>
</tr>
<tr>
<td>Output current limitation</td>
<td>Yes, automatic reactivation after short circuit/ overload</td>
</tr>
<tr>
<td>Resistance to feedback against 24 V signals</td>
<td>Yes (software-controlled supervision)</td>
</tr>
<tr>
<td>Max. cable length</td>
<td></td>
</tr>
<tr>
<td>Shielded</td>
<td>1000 m</td>
</tr>
<tr>
<td>Unshielded</td>
<td>600 m</td>
</tr>
</tbody>
</table>

The following drawing shows the circuitry of a digital input/output with the varistors for demagnetization when inductive loads are switched off.

![Circuit Diagram](image)

**Fig. 189: Digital input/output (circuit diagram)**
1. Digital input/output
2. For demagnetization when inductive loads are turned off

**Technical data of the fast counter**

> The fast counter of the module does not work if the module is connected to a CS31 bus module.
### Technical data of the analog inputs

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of channels per module</td>
<td>4</td>
</tr>
<tr>
<td>Distribution of channels into groups</td>
<td>1 group with 4 channels</td>
</tr>
<tr>
<td>Connection if channels AI0+ to AI3+</td>
<td>Terminals 3.0 to 3.3</td>
</tr>
<tr>
<td>Reference potential for AI0+ to AI3+</td>
<td>Terminal 3.4 (AI-) for voltage and RTD measurement</td>
</tr>
<tr>
<td></td>
<td>Terminal 1.9, 2.9, 3.9 and 4.9 for current measurement</td>
</tr>
<tr>
<td>Input type</td>
<td></td>
</tr>
<tr>
<td>Unipolar</td>
<td>Voltage 0 V...10 V, current or Pt100/Pt1000/ Ni1000</td>
</tr>
<tr>
<td>Bipolar</td>
<td>Voltage -10 V...+10 V</td>
</tr>
<tr>
<td>Configurability</td>
<td>0 V...10 V, -10 V...+10 V, 0 mA...20 mA, 4 mA...20 mA, Pt100/1000, Ni1000 (each input can be configured individually)</td>
</tr>
<tr>
<td>Channel input resistance</td>
<td>Voltage: &gt; 100 kΩ</td>
</tr>
<tr>
<td></td>
<td>Current: ca. 330 Ω</td>
</tr>
<tr>
<td>Time constant of the input filter</td>
<td>Voltage: 100 µs</td>
</tr>
<tr>
<td></td>
<td>Current: 100 µs</td>
</tr>
<tr>
<td>Indication of the input signals</td>
<td>1 LED per channel (brightness depends on the value of the analog signal)</td>
</tr>
<tr>
<td>Conversion cycle</td>
<td>1 ms (for 4 inputs + 2 outputs); with RTDs Pt/ Ni... 1 s</td>
</tr>
<tr>
<td>Resolution</td>
<td>Range 0 V...10 V: 12 bits</td>
</tr>
<tr>
<td></td>
<td>Range -10 V...+10 V: 12 bits + sign</td>
</tr>
<tr>
<td></td>
<td>Range 0 mA...20 mA: 12 bits</td>
</tr>
<tr>
<td></td>
<td>Range 4 mA...20 mA: 12 bits</td>
</tr>
<tr>
<td></td>
<td>Range RTD (Pt100, PT1000, Ni1000): 0.1 °C</td>
</tr>
<tr>
<td>Conversion error of the analog values caused by non-linearity, adjustment error at factory and resolution within the normal range</td>
<td>Typ. 0.5 %, max. 1 %</td>
</tr>
<tr>
<td></td>
<td>For XC version below 0 °C and above 60 °C: on request</td>
</tr>
</tbody>
</table>
### Technical data of the analog inputs, if used as digital inputs

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of channels per module</td>
<td>Max. 4</td>
</tr>
<tr>
<td>Distribution of channels into groups</td>
<td>1 group of 4 channels</td>
</tr>
<tr>
<td>Connections of the channels AI0+ to AI3+</td>
<td>Terminals 3.0 to 3.3</td>
</tr>
<tr>
<td>Reference potential for the inputs</td>
<td>Terminals 1.9, 2.9, 3.9 and 4.9 (ZP)</td>
</tr>
<tr>
<td>Indication of the input signals</td>
<td>1 LED per channel</td>
</tr>
<tr>
<td>Input signal voltage</td>
<td>24 VDC</td>
</tr>
<tr>
<td>Signal 0</td>
<td>-30 V...+5 V</td>
</tr>
<tr>
<td>Undefined signal</td>
<td>+5 V...+13 V</td>
</tr>
<tr>
<td>Signal 1</td>
<td>+13 V...+30 V</td>
</tr>
<tr>
<td>Input current per channel</td>
<td></td>
</tr>
<tr>
<td>Input voltage +24 V</td>
<td>Typ. 7 mA</td>
</tr>
<tr>
<td>Input voltage +5 V</td>
<td>Typ. 1.4 mA</td>
</tr>
<tr>
<td>Input voltage +15 V</td>
<td>Typ. 3.7 mA</td>
</tr>
<tr>
<td>Input voltage +30 V</td>
<td>&lt; 9 mA</td>
</tr>
<tr>
<td>Input resistance</td>
<td>ca. 3.5 kΩ</td>
</tr>
</tbody>
</table>

### Technical data of the analog outputs

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of channels per module</td>
<td>2</td>
</tr>
<tr>
<td>Distribution of channels into groups</td>
<td>1 group for 2 channels</td>
</tr>
<tr>
<td>Connection of the channels AO0+...AO1+</td>
<td>Terminals 3.5 and 3.6</td>
</tr>
<tr>
<td>Reference potential for AO0+ to AO1+</td>
<td>Terminal 3.7 (AO-) for voltage output</td>
</tr>
<tr>
<td></td>
<td>Terminals 1.9, 2.9, 3.9 and 4.9 for current output</td>
</tr>
<tr>
<td>Output type</td>
<td></td>
</tr>
<tr>
<td>Unipolar</td>
<td>Current</td>
</tr>
<tr>
<td>Bipolar</td>
<td>Voltage</td>
</tr>
<tr>
<td>Galvanic isolation</td>
<td>Against internal supply and other modules</td>
</tr>
</tbody>
</table>
### Parameter | Value
--- | ---
Configurability | -10 V...+10 V, 0 mA...20 mA, 4 mA...20 mA (each output can be configured individually)
Output resistance (load), as current output | 0 Ω...500 Ω
Output loadability, as voltage output | ±10 mA max.
Indication of the output signals | 1 LED per channel (brightness depends on the value of the analog signal)
Resolution | 12 bits (+ sign)
Conversion error of the analog values caused by non-linearity, adjustment error at factory and resolution within the normal range | Typ. 0.5 %, max. 1 %
Relationship between input signal and hex code | ☛ Chapter 1.5.3.6.3.1.2.9.3 “Output ranges voltage and current” on page 2777
Unused outputs | Are configured as "unused" (default value) and can be left open-circuited

### Ordering data

<table>
<thead>
<tr>
<th>Part no.</th>
<th>Description</th>
<th>Product life cycle phase *)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1SAP 250 800 R0001</td>
<td>DA502, digital/analog input/output module, 16 DO, 8 DC, 4 Al, 2 AO</td>
<td>Active</td>
</tr>
<tr>
<td>1SAP 450 800 R0001</td>
<td>DA502-XC, digital/analog input/output module, 16 DO, 8 DC, 4 Al, 2 AO, XC version</td>
<td>Active</td>
</tr>
</tbody>
</table>

*) For planning and commissioning of new installations use modules in Active status only.
1.5.3.7 Communication interface modules (S500)

Hot swap

System requirements for hot swapping of I/O modules:
- Types of terminal units that support hot swapping of I/O modules have the appendix TU5xx-H.
- I/O modules as of index F0.

The following I/O bus masters support hot swapping of attached I/O modules:
- Communication interface modules CI5xx as of index F0.
- Processor modules PM56xx-2ETH with firmware version as of V3.2.0.

NOTICE!
Risk of damage to I/O modules!
Hot swapping is only allowed for I/O modules.
Processor modules and communication interface modules must not be removed or inserted during operation.

Conditions for hot swapping
- Digital outputs are not under load.
- Input/output voltages above safety extra low voltage/protective extra low voltages (SELV/PELV) are switched off.
- Modules are completely plugged on the terminal unit with both snap fit engaged before switching on loads or input/output voltage.

Hot swap
Further information about hot swap: System technology § Chapter 1.5.5.1.6 “Hot swap” on page 3148.

1.5.3.7.1 Compatibility of communication modules and communication interface modules

Table 593: Modbus TCP

<table>
<thead>
<tr>
<th>Communication module</th>
<th>Communication interface module</th>
<th>I/O expansion module S500</th>
<th>I/O expansion module S500-eCo</th>
<th>I/O expansion module S500-S</th>
<th>Applications</th>
</tr>
</thead>
<tbody>
<tr>
<td>Onboard Ethernet inter-face</td>
<td>CI521-MODTCP</td>
<td>x</td>
<td>x</td>
<td>--</td>
<td>high availability, remote I/O</td>
</tr>
<tr>
<td>CI522-MODTCP</td>
<td>x</td>
<td>x</td>
<td>--</td>
<td>high availability, remote I/O</td>
<td></td>
</tr>
</tbody>
</table>
Table 594: PROFINET IO RT

<table>
<thead>
<tr>
<th>Communication module</th>
<th>Communication interface module</th>
<th>I/O expansion module S500</th>
<th>I/O expansion module S500-eCo</th>
<th>I/O expansion module S500-S</th>
<th>Applications</th>
</tr>
</thead>
<tbody>
<tr>
<td>CM579-PNIO controller</td>
<td>CI501-PNIO</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>remote I/O, safety I/O</td>
</tr>
<tr>
<td>CM579-PNIO controller</td>
<td>CI502-PNIO</td>
<td>x</td>
<td>--</td>
<td>--</td>
<td>hot swap I/O</td>
</tr>
</tbody>
</table>

Table 595: CANopen

<table>
<thead>
<tr>
<th>Communication module</th>
<th>Communication interface module</th>
<th>I/O expansion module S500</th>
<th>I/O expansion module S500-eCo</th>
<th>I/O expansion module S500-S</th>
<th>Applications</th>
</tr>
</thead>
<tbody>
<tr>
<td>Onboard CAN interface</td>
<td>CI581-CN</td>
<td>--</td>
<td>--</td>
<td>--</td>
<td>remote I/O</td>
</tr>
<tr>
<td></td>
<td>CI582-CN</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 596: EtherCAT

<table>
<thead>
<tr>
<th>Communication module</th>
<th>Communication interface module</th>
<th>I/O expansion module S500</th>
<th>I/O expansion module S500-eCo</th>
<th>I/O expansion module S500-S</th>
<th>Applications</th>
</tr>
</thead>
<tbody>
<tr>
<td>CM579-ETHCAT master</td>
<td>CI511-ETHCAT</td>
<td>x</td>
<td>x</td>
<td>--</td>
<td>remote I/O</td>
</tr>
<tr>
<td></td>
<td>CI512-ETHCAT</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

1.5.3.7.2 CANopen
Comparison CI581 and CI582
CI581/CI582:
Technical data

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Interface</td>
<td>CAN</td>
</tr>
<tr>
<td>Protocol</td>
<td>CANopen</td>
</tr>
<tr>
<td>Power supply</td>
<td>From the process supply voltage UP</td>
</tr>
<tr>
<td>Supply of the electronic circuitry of</td>
<td>Through the expansion bus interface (I/O bus)</td>
</tr>
<tr>
<td>the I/O modules attached</td>
<td></td>
</tr>
<tr>
<td>Rotary switches</td>
<td>For setting the CANopen Node ID for configuration purposes (00h to FFh)</td>
</tr>
<tr>
<td>LED displays</td>
<td>For system displays, signal states, errors and power supply</td>
</tr>
<tr>
<td>External supply voltage</td>
<td>Via terminals ZP, UP and UP3 (process supply voltage 24 VDC)</td>
</tr>
<tr>
<td>Transmission rates</td>
<td>10 / 20 / 50 / 125 / 250 / 500 / 800 kbit/s 1 Mbit/s Auto transmission rate detection is supported</td>
</tr>
<tr>
<td>Bus connection</td>
<td>Depending on used terminal unit TU510: 9-pin D-sub connector TU518: 10-pin terminal block</td>
</tr>
<tr>
<td>Parameter</td>
<td>Value</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Processor</td>
<td>Hilscher netX100</td>
</tr>
<tr>
<td>Expandability</td>
<td>CI58x can only be used on onboard CAN interface and without any I/O expansion module (\text{Table 526 &quot;CANopen&quot; on page 2288.})</td>
</tr>
<tr>
<td>State display</td>
<td>Module state: PWR/RUN, CN-RUN, CN-ERR, E-ERR, I/O bus</td>
</tr>
<tr>
<td>Adjusting elements</td>
<td>2 rotary switches for generation of the node address</td>
</tr>
<tr>
<td>Ambient temperature</td>
<td>System data AC500 (\text{Chapter 1.5.4.5.1 &quot;System data AC500&quot; on page 3058}) and System data AC500 XC (\text{Chapter 1.5.4.6.1 &quot;System data AC500-XC&quot; on page 3099})</td>
</tr>
<tr>
<td>Current consumption</td>
<td>UP: 0.2 A UP3: 0.06 A + 0.5 A max. per output</td>
</tr>
<tr>
<td>Weight (without terminal unit)</td>
<td>Ca. 125 g</td>
</tr>
<tr>
<td>Process supply voltages UP/UP3</td>
<td></td>
</tr>
<tr>
<td>Rated value</td>
<td>24 VDC (for inputs and outputs)</td>
</tr>
<tr>
<td>Max. load for the terminals</td>
<td>10 A</td>
</tr>
<tr>
<td>Protection against reversed voltage</td>
<td>Yes</td>
</tr>
<tr>
<td>Rated protection fuse on UP/UP3</td>
<td>10 A fast</td>
</tr>
<tr>
<td>Galvanic isolation</td>
<td>CANopen interface against the rest of the module</td>
</tr>
<tr>
<td>Inrush current from UP (at power up)</td>
<td>On request</td>
</tr>
<tr>
<td>Current consumption via UP (normal operation)</td>
<td>0.2 A</td>
</tr>
<tr>
<td>Current consumption via UP3</td>
<td>0.06 A + 0.5 A max. per output</td>
</tr>
<tr>
<td>Connections</td>
<td>Terminals 2.8 and 3.8 for +24 V (UP) Terminals 4.8 for +24 V (UP3) Terminals 2.9, 3.9 and 4.9 for 0 V (ZP)</td>
</tr>
<tr>
<td>Max. power dissipation within the module</td>
<td>6 W</td>
</tr>
<tr>
<td>Reference potential for all digital inputs and outputs</td>
<td>Minus pole of the supply voltage, signal name ZP</td>
</tr>
<tr>
<td>Setting of the CANopen Node ID identifier</td>
<td>With 2 rotary switches at the front side of the module</td>
</tr>
<tr>
<td>Mounting position</td>
<td>Horizontal Or vertical with derating (output load reduced to 50 % at 40 °C per group)</td>
</tr>
<tr>
<td>Cooling</td>
<td>The natural convection cooling must not be hindered by cable ducts or other parts in the switch-gear cabinet.</td>
</tr>
<tr>
<td>Effect of incorrect input terminal connection</td>
<td>Wrong or no signal detected, no damage up to 35 V</td>
</tr>
<tr>
<td>Required terminal unit</td>
<td>TU509, TU510, TU517 or TU518 (\text{Chapter 1.5.3.5.3 &quot;TU517 and TU518 for communication interface modules&quot; on page 2317})</td>
</tr>
</tbody>
</table>
All I/O channels (digital and analog) are protected against reverse polarity, reverse supply, short circuit and continuous overvoltage up to 30 VDC.

The difference of those devices can be found in their input and output characteristics.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inputs and outputs</td>
<td>8 digital inputs (24 VDC; delay time configurable via software) 8 digital transistor outputs (24 VDC, 0.5 A max.) 4 analog inputs, configurable as:  -10 V...+10 V  0 V...+10 V  -10 V...+10 V (differential voltage)  0 mA...20 mA  4 mA...20 mA  Pt100, Pt1000, Ni1000 (for each 2-wire and 3-wire)  24 V digital input function 2 analog outputs, configurable as:  -10 V...+10 V  0 mA...20 mA  4 mA...20 mA</td>
</tr>
<tr>
<td>Resolution of the analog channels</td>
<td>12 bits</td>
</tr>
<tr>
<td>Fast counter</td>
<td>Integrated, configurable operating modes</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inputs and outputs</td>
<td>8 digital inputs (24 VDC) 8 digital transistor outputs (24 VDC, 0.5 A max.) 8 configurable digital inputs/outputs (24 VDC, 0.5 A max.)</td>
</tr>
</tbody>
</table>

**CI581-CN**

- 4 analog inputs (resolution 12 bits plus sign)
- 2 analog outputs (resolution 12 bits plus sign)
- 8 digital inputs 24 VDC
- 8 digital outputs 24 VDC, 0.5 A max
- Module-wise electrically isolated
- Fast counter
- XC version for use in extreme ambient conditions available
Intended purpose

The CANopen bus module CI581-CN is used as decentralized I/O module in CANopen networks. Depending on the used terminal unit the network connection is performed either via 9-pin female D-sub connector or via 10 terminals (screw or spring terminals) which are integrated in the terminal unit. The bus module contains 22 I/O channels with the following properties:

- 4 analog inputs (2.0...2.3)
- 2 analog outputs (2.5...2.6)
- 8 digital inputs 24 VDC in 1 group (3.0...3.7)
- 8 digital outputs 24 VDC in 1 group (4.0...4.7)
The inputs/outputs are electrically isolated from the CANopen network. There is no potential separation between the channels. The configuration of the analog inputs/outputs is performed by software.

For use in extreme ambient conditions (e.g. wider temperature and humidity range), a special XC version of the device is available.

### Functionality

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Interface</td>
<td>CAN</td>
</tr>
<tr>
<td>Protocol</td>
<td>CANopen</td>
</tr>
<tr>
<td>Power supply</td>
<td>From the process supply voltage UP</td>
</tr>
<tr>
<td>Supply of the electronic circuitry of the I/O modules attached</td>
<td>Through the expansion bus interface (I/O bus)</td>
</tr>
<tr>
<td>Rotary switches</td>
<td>For setting the CANopen Node ID for configuration purposes (00h to FFh)</td>
</tr>
<tr>
<td>LED displays</td>
<td>For system displays, signal states, errors and power supply</td>
</tr>
<tr>
<td>External supply voltage</td>
<td>Via terminals ZP, UP and UP3 (process supply voltage 24 VDC)</td>
</tr>
<tr>
<td>Transmission rates</td>
<td>10 / 20 / 50 / 125 / 250 / 500 / 800 kbit/s 1 Mbit/s Auto transmission rate detection is supported</td>
</tr>
<tr>
<td>Bus connection</td>
<td>Depending on used terminal unit TU510: 9-pin D-sub connector TU518: 10-pin terminal block</td>
</tr>
<tr>
<td>Processor</td>
<td>Hilscher netX100</td>
</tr>
<tr>
<td>Expandability</td>
<td>CI58x can only be used on onboard CAN interface and without any I/O expansion module</td>
</tr>
<tr>
<td>State display</td>
<td>Module state: PWR/RUN, CN-RUN, CN-ERR, E-ERR, I/O bus</td>
</tr>
<tr>
<td>Adjusting elements</td>
<td>2 rotary switches for generation of the node address</td>
</tr>
<tr>
<td>Ambient temperature</td>
<td>System data AC500</td>
</tr>
<tr>
<td>Current consumption</td>
<td>UP: 0.2 A UP3: 0.06 A + 0.5 A max. per output</td>
</tr>
<tr>
<td>Weight (without terminal unit)</td>
<td>Ca. 125 g</td>
</tr>
<tr>
<td>Process supply voltages UP/UP3</td>
<td>24 VDC (for inputs and outputs)</td>
</tr>
<tr>
<td>Max. load for the terminals</td>
<td>10 A</td>
</tr>
<tr>
<td>Protection against reversed voltage</td>
<td>Yes</td>
</tr>
<tr>
<td>Rated protection fuse on UP/UP3</td>
<td>10 A fast</td>
</tr>
<tr>
<td>Galvanic isolation</td>
<td>CANopen interface against the rest of the module</td>
</tr>
<tr>
<td>Inrush current from UP (at power up)</td>
<td>On request</td>
</tr>
<tr>
<td>Parameter</td>
<td>Value</td>
</tr>
<tr>
<td>-----------------------------------------------------</td>
<td>----------------------------------------------------------------------</td>
</tr>
<tr>
<td>Current consumption via UP (normal operation)</td>
<td>0.2 A</td>
</tr>
<tr>
<td>Current consumption via UP3</td>
<td>0.06 A + 0.5 A max. per output</td>
</tr>
<tr>
<td>Connections</td>
<td>Terminals 2.8 and 3.8 for +24 V (UP)</td>
</tr>
<tr>
<td></td>
<td>Terminal 4.8 for +24 V (UP3)</td>
</tr>
<tr>
<td></td>
<td>Terminals 2.9, 3.9 and 4.9 for 0 V (ZP)</td>
</tr>
<tr>
<td>Max. power dissipation within the module</td>
<td>6 W</td>
</tr>
<tr>
<td>Reference potential for all digital inputs and</td>
<td>Minus pole of the supply voltage, signal name ZP</td>
</tr>
<tr>
<td>outputs</td>
<td></td>
</tr>
<tr>
<td>Setting of the CANopen Node ID identifier</td>
<td>With 2 rotary switches at the front side of the module</td>
</tr>
<tr>
<td>Mounting position</td>
<td>Horizontal</td>
</tr>
<tr>
<td></td>
<td>Or vertical with derating (output load reduced to 50 % at 40 °C per group)</td>
</tr>
<tr>
<td>Cooling</td>
<td>The natural convection cooling must not be hindered by cable ducts or other parts in the switch-gear cabinet.</td>
</tr>
<tr>
<td>Effect of incorrect input terminal connection</td>
<td>Wrong or no signal detected, no damage up to 35 V</td>
</tr>
<tr>
<td>Required terminal unit</td>
<td>TU509, TU510, TU517 or TU518 Chapter 1.5.3.5.3 “TU517 and TU518 for communication interface modules” on page 2317</td>
</tr>
</tbody>
</table>

All I/O channels (digital and analog) are protected against reverse polarity, reverse supply, short circuit and continuous overvoltage up to 30 VDC.
### CI581-CN: Input/Output characteristics

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inputs and outputs</td>
<td>8 digital inputs (24 VDC; delay time configurable via software)</td>
</tr>
<tr>
<td></td>
<td>8 digital transistor outputs (24 VDC, 0.5 A max.)</td>
</tr>
<tr>
<td></td>
<td>4 analog inputs, configurable as:</td>
</tr>
<tr>
<td></td>
<td>● -10 V...+10 V</td>
</tr>
<tr>
<td></td>
<td>● 0 V...+10 V</td>
</tr>
<tr>
<td></td>
<td>● -10 V...+10 V (differential voltage)</td>
</tr>
<tr>
<td></td>
<td>● 0 mA...20 mA</td>
</tr>
<tr>
<td></td>
<td>● 4 mA...20 mA</td>
</tr>
<tr>
<td></td>
<td>● Pt100, Pt1000, Ni1000 (for each 2-wire and 3-wire)</td>
</tr>
<tr>
<td></td>
<td>● 24 V digital input function</td>
</tr>
<tr>
<td></td>
<td>2 analog outputs, configurable as:</td>
</tr>
<tr>
<td></td>
<td>● -10 V...+10 V</td>
</tr>
<tr>
<td></td>
<td>● 0 mA...20 mA</td>
</tr>
<tr>
<td></td>
<td>● 4 mA...20 mA</td>
</tr>
<tr>
<td>Resolution of the analog channels</td>
<td>12 bits</td>
</tr>
<tr>
<td>Fast counter</td>
<td>Integrated, configurable operating modes</td>
</tr>
</tbody>
</table>

### Electrical connection

The CANopen bus module is plugged on the I/O terminal units TU517 [Chapter 1.5.3.5.3 “TU517 and TU518 for communication interface modules” on page 2317 or TU518 [Chapter 1.5.3.5.3 “TU517 and TU518 for communication interface modules” on page 2317 and accordingly TU509 or TU510. Properly position the module and press until it locks in place.

The electrical connection of the I/O channels is established using the 30 terminals of the I/O terminal unit. I/O modules can be replaced without re-wiring the terminal units.

The terminals 2.8, 3.8, 2.9, 3.9 and 4.9 are electrically interconnected within the terminal unit and always have the same assignment, irrespective of the inserted module:

- Terminals 2.8 and 3.8: process supply voltage UP = +24 VDC
- Terminal 4.8: process supply voltage UP3 = +24 VDC
- Terminals 2.9, 3.9 and 4.9: process supply voltage ZP = 0 V

For a detailed description of the mounting, disassembly and electrical connection of the module, please refer to the System Assembly, Construction and Connection chapter [Chapter 1.5.4.5 “AC500 (Standard)” on page 3058.

With a separate UP3 power supply, the digital outputs can be switched off externally. This way, an emergency-off functionality can be realized.
Do not connect any voltages externally to the digital outputs!
Reason: External voltages at an output or several outputs may cause other outputs to be supplied via that voltage instead of voltage UP3 (reverse voltage). This is not the intended use.

**CAUTION!**
Risk of malfunctions by unintended use!
If the function cut off of the digital outputs should be used by deactivation of the supply voltage UP3, be sure that no external voltage is connected at the outputs DO0..DO7 and DC0..DC7.

Possibilities of connection

**Mounting on terminal units**
TU509 or TU510

The assignment of the 9-pin female D-sub connector for the CANopen signals

<table>
<thead>
<tr>
<th>Pin</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>---</td>
</tr>
<tr>
<td>2</td>
<td>CAN-</td>
</tr>
<tr>
<td>3</td>
<td>CAN_GND</td>
</tr>
<tr>
<td>4</td>
<td>---</td>
</tr>
<tr>
<td>5</td>
<td>---</td>
</tr>
<tr>
<td>6</td>
<td>---</td>
</tr>
<tr>
<td>7</td>
<td>CAN+</td>
</tr>
<tr>
<td>8</td>
<td>---</td>
</tr>
<tr>
<td>9</td>
<td>---</td>
</tr>
</tbody>
</table>

Shield     | Cable shield  | Functional earth |

Bus terminating resistors

The ends of the data lines have to be terminated with a 120 Ω bus terminating resistor. The bus terminating resistor is usually installed directly at the bus connector.

Fig. 190: CANopen interface, bus terminating resistors connected to the line ends
### Table 1: DeviceNet Interface Details

<table>
<thead>
<tr>
<th>No.</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>CAN_GND</td>
</tr>
<tr>
<td>2</td>
<td>CAN_L</td>
</tr>
<tr>
<td>3</td>
<td>Shield</td>
</tr>
<tr>
<td>4</td>
<td>CAN_H</td>
</tr>
<tr>
<td>5</td>
<td>Data line, shielded twisted pair</td>
</tr>
<tr>
<td>6</td>
<td>COMBICON connection, CANopen interface</td>
</tr>
</tbody>
</table>

![Diagram](image.png)

**Fig. 191: DeviceNet interface, bus terminating resistors connected to the line ends**

<table>
<thead>
<tr>
<th>No.</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>DeviceNet power supply</td>
</tr>
<tr>
<td>7</td>
<td>COMBICON connection, DeviceNet interface</td>
</tr>
<tr>
<td>8</td>
<td>Data lines, twisted pair cables</td>
</tr>
<tr>
<td>9</td>
<td>red</td>
</tr>
<tr>
<td>10</td>
<td>black</td>
</tr>
<tr>
<td>11</td>
<td>white</td>
</tr>
<tr>
<td>12</td>
<td>blue</td>
</tr>
<tr>
<td>13</td>
<td>bare</td>
</tr>
</tbody>
</table>

*The earthing of the shield should take place at the switch-gear. Please refer to Chapter 1.5.4.5.1 “System data AC500” on page 3058.*
Table 597: Assignment of the terminals

<table>
<thead>
<tr>
<th>Terminal</th>
<th>Signal</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0</td>
<td>CAN+</td>
<td>Non-inverted signal of the CAN Bus</td>
</tr>
<tr>
<td>1.1</td>
<td>CAN+</td>
<td>Non-inverted signal of the CAN Bus</td>
</tr>
<tr>
<td>1.2</td>
<td>CAN-</td>
<td>Inverted signal of the CAN Bus</td>
</tr>
<tr>
<td>1.3</td>
<td>CAN-</td>
<td>Inverted signal of the CAN Bus</td>
</tr>
<tr>
<td>1.4</td>
<td>Term+</td>
<td>CAN bus termination for CAN+ (for bus termination, Term+ must be connected with CAN+)</td>
</tr>
<tr>
<td>1.5</td>
<td>Term+</td>
<td>CAN bus termination for CAN+ (connecting alternative for terminal 1.4)</td>
</tr>
<tr>
<td>1.6</td>
<td>Term-</td>
<td>CAN bus termination for CAN- (for bus termination, Term- must be connected with CAN-)</td>
</tr>
<tr>
<td>1.7</td>
<td>Term-</td>
<td>CAN bus termination for CAN- (connecting alternative for terminal 1.6)</td>
</tr>
<tr>
<td>1.8</td>
<td>CAN-GND</td>
<td>Ground potential of the CAN bus</td>
</tr>
<tr>
<td>1.9</td>
<td>CAN-GND</td>
<td>Ground potential of the CAN bus</td>
</tr>
</tbody>
</table>

At the line ends of a bus segment, termination resistors must be connected. If TU517 or TU518 is used, the bus termination resistors can be enabled by connecting the terminals Term+ and Term- to the data lines CAN+ and CAN- (no external termination resistors are required, see illustration below).

The following figures show the different connection options for the CANopen bus module:
In the case of TU517/TU518, the termination resistors are not located inside the TU but inside the bus module CI581-CN. Hence, when removing the device from the TU, the bus termination resistors are no longer connected to the bus. The bus itself will not be disconnected if a device is removed.

The earthing of the shield should take place at the switch-gear cabinet. Please refer to the AC500 System-Data Chapter 1.5.4.5.1 "System data AC500" on page 3058.

Table 598: Assignment of the other terminals

<table>
<thead>
<tr>
<th>Terminal</th>
<th>Signal</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.0</td>
<td>AI0+</td>
<td>Positive pole of analog input signal 0</td>
</tr>
<tr>
<td>2.1</td>
<td>AI1+</td>
<td>Positive pole of analog input signal 1</td>
</tr>
<tr>
<td>2.2</td>
<td>AI2+</td>
<td>Positive pole of analog input signal 2</td>
</tr>
<tr>
<td>2.3</td>
<td>AI3+</td>
<td>Positive pole of analog input signal 3</td>
</tr>
<tr>
<td>2.4</td>
<td>AI-</td>
<td>Negative pole of analog input signals 0 to 3</td>
</tr>
<tr>
<td>2.5</td>
<td>AO0+</td>
<td>Positive pole of analog output signal 0</td>
</tr>
<tr>
<td>2.6</td>
<td>AO1+</td>
<td>Positive pole of analog output signal 1</td>
</tr>
<tr>
<td>2.7</td>
<td>AI-</td>
<td>Negative pole of analog output signals 0 and 1</td>
</tr>
<tr>
<td>2.8</td>
<td>UP</td>
<td>Process voltage UP (24 V DC)</td>
</tr>
<tr>
<td>2.9</td>
<td>ZP</td>
<td>Process voltage ZP (0 V DC)</td>
</tr>
<tr>
<td>3.0</td>
<td>DI0</td>
<td>Signal of the digital input DI0</td>
</tr>
<tr>
<td>3.1</td>
<td>DI1</td>
<td>Signal of the digital input DI1</td>
</tr>
<tr>
<td>3.2</td>
<td>DI2</td>
<td>Signal of the digital input DI2</td>
</tr>
<tr>
<td>3.3</td>
<td>DI3</td>
<td>Signal of the digital input DI3</td>
</tr>
<tr>
<td>Terminal</td>
<td>Signal</td>
<td>Description</td>
</tr>
<tr>
<td>---------</td>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td>3.4</td>
<td>DI4</td>
<td>Signal of the digital input DI4</td>
</tr>
<tr>
<td>3.5</td>
<td>DI5</td>
<td>Signal of the digital input DI5</td>
</tr>
<tr>
<td>3.6</td>
<td>DI6</td>
<td>Signal of the digital input DI6</td>
</tr>
<tr>
<td>3.7</td>
<td>DI7</td>
<td>Signal of the digital input DI7</td>
</tr>
<tr>
<td>3.8</td>
<td>UP</td>
<td>Process voltage UP (24 V DC)</td>
</tr>
<tr>
<td>3.9</td>
<td>ZP</td>
<td>Process voltage ZP (0 V DC)</td>
</tr>
<tr>
<td>4.0</td>
<td>DO0</td>
<td>Signal of the digital output DO0</td>
</tr>
<tr>
<td>4.1</td>
<td>DO1</td>
<td>Signal of the digital output DO1</td>
</tr>
<tr>
<td>4.2</td>
<td>DO2</td>
<td>Signal of the digital output DO2</td>
</tr>
<tr>
<td>4.3</td>
<td>DO3</td>
<td>Signal of the digital output DO3</td>
</tr>
<tr>
<td>4.4</td>
<td>DO4</td>
<td>Signal of the digital output DO4</td>
</tr>
<tr>
<td>4.5</td>
<td>DO5</td>
<td>Signal of the digital output DO5</td>
</tr>
<tr>
<td>4.6</td>
<td>DO6</td>
<td>Signal of the digital output DO6</td>
</tr>
<tr>
<td>4.7</td>
<td>DO7</td>
<td>Signal of the digital output DO7</td>
</tr>
<tr>
<td>4.8</td>
<td>UP3</td>
<td>Process voltage UP3 (24 V DC)</td>
</tr>
<tr>
<td>4.9</td>
<td>ZP</td>
<td>Process voltage ZP (0 V DC)</td>
</tr>
</tbody>
</table>

**WARNING!**
**Removal/Insertion under power**

The devices are not designed for removal or insertion under power. Because of unforeseeable consequences, it is not allowed to plug or unplug devices with the power being ON.

Make sure that all voltage sources (supply and process voltage) are switched off before you

- connect or disconnect any signal or terminal block
- remove, mount or replace a module.

Disconnecting any powered devices while energized in a hazardous location could result in an electric arc, which could create a flammable ignition resulting in fire or explosion.

Make sure that power is removed and that the area has been thoroughly checked to ensure that flammable materials are not present prior to proceeding.

The devices must not be opened when in operation. The same applies to the network interfaces.

**NOTICE!**
**Risk of damaging the PLC modules!**

Overvoltages and short circuits might damage the PLC modules.

- Make sure that all voltage sources (supply voltage and process supply voltage) are switched off before you begin with operations at the system.
- Never connect any voltages or signals to reserved terminals (marked with ---). Reserved terminals may carry internal voltages.
For the open-circuit detection (cut wire), each analog input channel is pulled up to "plus" by a high-resistance resistor. If nothing is connected, the maximum voltage will be read in then.

Generally, analog signals must be laid in shielded cables. The cable shields must be earthed at both sides of the cables. In order to avoid unacceptable potential differences between different parts of the installation, low resistance equipotential bonding conductors must be laid.

Only for simple applications (low electromagnetic disturbances, no high requirement on precision), the shielding can also be omitted.

Electrical connection of CANopen bus module CI581-CN:

Fig. 192: Connection of the bus module CI581-CN

The module provides several diagnosis functions "Diagnosis" on page 2813.

For the measuring ranges that can be configured, please refer to the sections Measuring Ranges "Measuring ranges" on page 2819 and Parameterization "Parameterization" on page 2809.

The meaning of the LEDs is described in the section for the state LEDs "State LEDs" on page 2817.

Bus length

The maximum possible bus length of a CAN network depends on bit rate (transmission rate) and cable type. The sum of all bus segments must not exceed the maximum bus length.
<table>
<thead>
<tr>
<th>Bit Rate (speed)</th>
<th>Bus Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Mbit/s</td>
<td>40 m</td>
</tr>
<tr>
<td>800 kbit/s</td>
<td>50 m</td>
</tr>
<tr>
<td>500 kbit/s</td>
<td>100 m</td>
</tr>
<tr>
<td>250 kbit/s</td>
<td>250 m</td>
</tr>
<tr>
<td>125 kbit/s</td>
<td>500 m</td>
</tr>
<tr>
<td>50 kbit/s</td>
<td>1000 m</td>
</tr>
</tbody>
</table>

Connection of the digital inputs

The following figure shows the electrical connection of the digital input DI0. Proceed with the digital inputs DI1 to DI7 in the same way.

![Diagram of digital input connection]

Fig. 193: Connection of the digital inputs to the module CI581-CN

Connection of the digital outputs

The following figure shows the electrical connection of the digital output DO0. Proceed with the digital outputs DO1 - DO7 in the same way.
Connection of resistance thermometers in 2-wire configuration to the analog inputs

When resistance thermometers (Pt100, Pt1000, Ni1000) are used, a constant current must flow to build the necessary voltage drop for the evaluation. For this, the module CI581-CN provides a constant current source which is multiplexed over the max. 4 analog input channels.

The following figure shows the connection of resistance thermometers in 2-wire configuration to the analog input AI0. Proceed with the analog inputs AI1 to AI3 in the same way.

![Connection of resistance thermometers in 2-wire configuration to the analog inputs](image.png)

Fig. 195: Connection of resistance thermometers in 2-wire configuration to the analog inputs
Pt100  2-wire configuration, 1 channel used
Pt1000  2-wire configuration, 1 channel used
Ni1000  2-wire configuration, 1 channel used

For the measuring ranges that can be configured, please refer to sections Measuring Ranges © Chapter 1.5.3.7.2.2.10 “Measuring ranges” on page 2819 and Parameterization © Chapter 1.5.3.7.2.2.7 “Parameterization” on page 2809.

The module CI581-CN performs a linearization of the resistance characteristic.

To avoid error messages, configure unused analog input channels as "unused".

Connection of resistance thermometers in 3-wire configuration to the analog inputs

When resistance thermometers (Pt100, Pt1000, Ni1000) are used, a constant current must flow through them to build the necessary voltage drop for the evaluation. For this, the module CI581-CN provides a constant current source which is multiplexed over the max. 4 analog input channels.

The constant current of one channel flows through the resistance thermometer. The constant current of the other channel flows through one of the cores. The module calculates the measured value from the two voltage drops and stores it under the input with the higher channel number (e.g. I1).

In order to keep measuring errors as small as possible, it is necessary to have all the involved conductors in the same cable. All the conductors must have the same cross section.

With 3-wire configuration, 2 adjacent analog channels belong together (e.g. the channels 0 and 1). In this case, both channels are configured according to the desired operating mode. The lower address must be the even address (channel 0), the next higher address must be the odd address (channel 1).

The following figure shows the connection of resistance thermometers in 3-wire configuration to the analog inputs AI0 and AI1. Proceed with the analog inputs AI2 and AI3 in the same way.

![Diagram of connection of resistance thermometers in 3-wire configuration to the analog inputs](image)
Connection of active-type analog sensors (Voltage) with electrically isolated power supply to the analog inputs

The following figure shows the connection of active-type analog sensors (voltage) with electrically isolated power supply to the analog input AI0. Proceed with the analog inputs AI1 to AI3 in the same way.

![Connection Diagram](image)

**Fig. 197:** Connection of active-type analog sensors (voltage) with electrically isolated power supply to the analog inputs

<table>
<thead>
<tr>
<th>Voltage</th>
<th>0...10 V</th>
<th>1 channel used</th>
</tr>
</thead>
<tbody>
<tr>
<td>Voltage</td>
<td>-10 V...+10 V</td>
<td>1 channel used</td>
</tr>
</tbody>
</table>

For the measuring ranges that can be configured, please refer to the sections Measuring Ranges "Chapter 1.5.3.7.2.2.10 "Measuring ranges" on page 2819 and Parameterization "Chapter 1.5.3.7.2.2.7 "Parameterization" on page 2809.

To avoid error messages, configure unused analog input channels as "unused".

Connection of active-type analog sensors (Current) with electrically isolated power supply to the analog inputs

The following figure shows the connection of active-type analog sensors (current) with electrically isolated power supply to the analog input AI0. Proceed with the analog inputs AI1 to AI3 in the same way.
Fig. 198: Connection of active-type analog sensors (current) with electrically isolated power supply to the analog inputs

<table>
<thead>
<tr>
<th>Current</th>
<th>0...20 mA</th>
<th>1 channel used</th>
</tr>
</thead>
<tbody>
<tr>
<td>Current</td>
<td>4...20 mA</td>
<td>1 channel used</td>
</tr>
</tbody>
</table>

For the measuring ranges that can be configured, please refer to the sections Measuring Ranges \Chapter 1.5.3.7.2.2.10 “Measuring ranges” on page 2819 and Parameterization \Chapter 1.5.3.7.2.2.7 “Parameterization” on page 2809.

Unused input channels can be left open-circuited, because they are of low resistance.

Connection of active-type analog sensors (Voltage) with no electrically isolated power supply to the analog inputs

The following figure shows the connection of active-type analog sensors (voltage) with no electrically isolated power supply to the analog input AI0. Proceed with the analog inputs AI1 to AI3 in the same way.
**Fig. 199: Connection of active-type sensors (voltage) with no electrically isolated power supply to the analog inputs**

**NOTICE!**

Risk of faulty measurements!

The negative pole/earthing potential at the sensors must not have too large a potential difference with respect to ZP (max. $\pm 1$ V within the full signal range).

Make sure that the potential difference never exceeds $\pm 1$ V.

<table>
<thead>
<tr>
<th>Voltage</th>
<th>$0\ldots10$ V</th>
<th>1 channel used</th>
</tr>
</thead>
<tbody>
<tr>
<td>Voltage</td>
<td>$-10$ V$\ldots+10$ V</td>
<td>1 channel used</td>
</tr>
</tbody>
</table>

For the measuring ranges that can be configured, please refer to the sections Measuring Ranges \( Chapter \) 1.5.3.7.2.2.10 "Measuring ranges" on page 2819 and Parameterization \( Chapter \) 1.5.3.7.2.2.7 "Parameterization" on page 2809.

To avoid error messages, configure unused analog input channels as "unused".

**Connection of passive-type analog sensors (Current) to the analog inputs**

The following figure shows the connection of passive-type analog sensors (current) to the analog input AI0. Proceed with the analog inputs AI1 to AI3 in the same way.
2.0 AI0+
2.1 AI1+
2.2 AI2+
2.3 AI3+
2.4 AI-
2.5 AO0+
2.6 AO1+
2.7 AO-
2.8 UP
2.9 ZP

Fig. 200: Connection of passive-type analog sensors (current) to the analog inputs

<table>
<thead>
<tr>
<th>Current</th>
<th>4...20 mA</th>
<th>1 channel used</th>
</tr>
</thead>
</table>

**CAUTION!**

**Risk of overloading the analog input!**

If an analog current sensor supplies more than 25 mA for more than 1 second during initialization, this input is switched off by the module (input protection).

Only use sensors with fast initialization or without current peaks higher than 25 mA. If not possible, connect a 10-volt Zener diode in parallel to I+ and I-.

Unused input channels can be left open-circuited, because they are of low resistance.

**Connection of active-type analog sensors (Voltage) to differential analog inputs**

Differential inputs are very useful if analog sensors which are remotely non-isolated (e.g. the negative terminal is remotely earthed) are used.

Using differential inputs helps to considerably increase the measuring accuracy and to avoid earthing loops.

With differential input configurations, two adjacent analog channels belong together (e.g. the channels 0 and 1). In this case, both channels are configured according to the desired operating mode. The lower address must be the even address (channel 0), the next higher address must be the odd address (channel 1). The converted analog value is available at the higher address (channel 1).

The analog value is calculated by subtraction of the input value with the higher address from the input value of the lower address.

The converted analog value is available at the odd channel (higher address).
**NOTICE!**

Risk of faulty measurements!

The negative pole/earthing potential at the sensors must not have too large a potential difference with respect to ZP (max. ± 1 V within the full signal range). Make sure that the potential difference never exceeds ± 1 V.

The following figure shows the connection of active-type analog sensors (voltage) to differential analog inputs AI0 and AI1. Proceed with AI2 and AI3 in the same way.

![Connection diagram](image)

Fig. 201: Connection of active-type analog sensors (voltage) to differential analog inputs

<table>
<thead>
<tr>
<th>Voltage</th>
<th>0...10 V</th>
<th>with differential inputs, 2 channels used</th>
</tr>
</thead>
<tbody>
<tr>
<td>Voltage</td>
<td>-10 V...+10 V</td>
<td>with differential inputs, 2 channels used</td>
</tr>
</tbody>
</table>

For the measuring ranges that can be configured, please refer to the sections Measuring Ranges "Chapter 1.5.3.7.2.2.10 “Measuring ranges” on page 2819 and Parameterization "Chapter 1.5.3.7.2.2.7 “Parameterization” on page 2809.

To avoid error messages, configure unused analog input channels as "unused".

**Use of analog inputs as digital inputs**

Several (or all) analog inputs can be configured as digital inputs. The inputs are not electrically isolated against the other analog channels.

The following figure shows the connection of digital sensors to the analog input AI0. Proceed with the analog inputs AI1 to AI3 in the same way.
Fig. 202: Use of analog inputs as digital inputs

<table>
<thead>
<tr>
<th>Digital input</th>
<th>24 V</th>
<th>1 channel used</th>
</tr>
</thead>
</table>

For the measuring ranges that can be configured, please refer to the sections Measuring Ranges &gt; Chapter 1.5.3.7.2.2.10 “Measuring ranges” on page 2819 and Parameterization &gt; Chapter 1.5.3.7.2.2.7 “Parameterization” on page 2809.

Connection of analog output loads (Voltage)

The following figure shows the connection of output loads to the analog output AO0. Proceed with the analog output AO1 in the same way.

Fig. 203: Connection of analog output loads (voltage)
Voltage | -10 V...+10 V | Load ± 10 mA max. | 1 channel used

For the measuring ranges that can be configured, please refer to the sections Measuring Ranges & Chapter 1.5.3.7.2.2.10 “Measuring ranges” on page 2819 and Parameterization & Chapter 1.5.3.7.2.2.7 “Parameterization” on page 2809.

Unused analog outputs can be left open-circuited.

Connection of analog output loads (Current)

The following figure shows the connection of output loads to the analog output AO0. Proceed with the analog output AO1 in the same way.

![Connection diagram for analog output loads](image)

Fig. 204: Connection of analog output loads (current)

<table>
<thead>
<tr>
<th>Current</th>
<th>0...20 mA</th>
<th>Load 0...500 Ω</th>
<th>1 channel used</th>
</tr>
</thead>
<tbody>
<tr>
<td>Current</td>
<td>4...20 mA</td>
<td>Load 0...500 Ω</td>
<td>1 channel used</td>
</tr>
</tbody>
</table>

For the measuring ranges that can be configured, please refer to the sections Measuring Ranges & Chapter 1.5.3.7.2.2.10 “Measuring ranges” on page 2819 and Parameterization & Chapter 1.5.3.7.2.2.7 “Parameterization” on page 2809.

Unused analog outputs can be left open-circuited.

Internal data exchange

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Digital inputs (bytes)</td>
<td>3</td>
</tr>
<tr>
<td>Digital outputs (bytes)</td>
<td>3</td>
</tr>
<tr>
<td>Analog inputs (words)</td>
<td>4</td>
</tr>
<tr>
<td>Analog outputs (words)</td>
<td>2</td>
</tr>
<tr>
<td>Counter input data (words)</td>
<td>4</td>
</tr>
<tr>
<td>Counter output data (words)</td>
<td>8</td>
</tr>
</tbody>
</table>
Addressing

A detailed description concerning addressing can be found in the documentation of ABB Control Builder Plus Software.

The CANopen bus module reads the position of the rotary switches only during power-up, i.e. changes of the switch position during operation will have no effect until the next module initialization.

The range of permitted CANopen slave addresses is 1 to 127. Setting a higher address (> 128) does not lead to an error response, but results in a special mode (DS401). In this special mode, the device creates the node address by subtracting the value 128 from the address switch’s value.

I/O configuration

The CI582-CN CANopen bus configuration is handled by CANopen master with the exception of the slave node ID (via rotary switches) and the transmission rate (automatic detection).

The digital I/O channels and the fast counter are configured via software.

Parameterization

Parameters of the module

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>Internal value</th>
<th>Internal value, type</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>Module ID 1)</td>
<td>Internal</td>
<td>0x1C84</td>
<td>WORD</td>
<td>0x1C84</td>
</tr>
<tr>
<td>Parameter length</td>
<td>Internal</td>
<td>54</td>
<td>BYTE</td>
<td>54</td>
</tr>
<tr>
<td>Error LED / Fail-safe function (table error LED / Fail-safe function ² Further information on page 2809)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>On</td>
<td>0</td>
<td>BYTE</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>Off by E4</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Off by E3</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>On + failsafe</td>
<td>16</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Off by E4 + failsafe</td>
<td>17</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Off by E3 + failsafe</td>
<td>18</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Reserved</td>
<td>0</td>
<td>0</td>
<td>ARRAY of 24 BYTES</td>
<td></td>
</tr>
<tr>
<td>Check supply (UP and UP3)</td>
<td>On</td>
<td>0</td>
<td>BYTE</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>Off</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Fast counter</td>
<td>0</td>
<td>0</td>
<td>BYTE</td>
<td>0</td>
</tr>
</tbody>
</table>

1) With a faulty ID, the module reports a "parameter error" and does not perform cyclic process data transmission

2) For a description of the counter operating modes, please refer to the Fast Counter section ² Chapter 1.5.3.6.1.2.9 "Fast counter" on page 2543.
<table>
<thead>
<tr>
<th>Setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>On</td>
<td>Error LED (S-ERR) lights up at errors of all error classes, failsafe mode off</td>
</tr>
<tr>
<td>Off by E4</td>
<td>Error LED (S-ERR) lights up at errors of error classes E1, E2 and E3, failsafe mode off</td>
</tr>
<tr>
<td>Off by E3</td>
<td>Error LED (S-ERR) lights up at errors of error classes E1 and E2, failsafe mode off</td>
</tr>
<tr>
<td>On +Failsafe</td>
<td>Error LED (S-ERR) lights up at errors of all error classes, failsafe mode on *)</td>
</tr>
<tr>
<td>Off by E4 + Failsafe</td>
<td>Error LED (S-ERR) lights up at errors of error classes E1, E2 and E3, failsafe mode on *)</td>
</tr>
<tr>
<td>Off by E3 + Failsafe</td>
<td>Error LED (S-ERR) lights up at errors of error classes E1 and E2, failsafe mode on *)</td>
</tr>
</tbody>
</table>

*) The parameters Behavior analog outputs at communication error and Behavior digital outputs at communication error are only evaluated if the failsafe function is enabled.

### Group parameters for the analog part

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>Internal value</th>
<th>Internal value, type</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>Analog data format</td>
<td>Standard</td>
<td>0</td>
<td>BYTE</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>Reserved</td>
<td>255</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Behavior analog outputs at communication error *)</td>
<td>Off</td>
<td>0</td>
<td>BYTE</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>Last value</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Last value 5 s</td>
<td>6</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Last value 10 s</td>
<td>11</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Substitute value</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Substitute value 5 s</td>
<td>7</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Substitute value 10 s</td>
<td>12</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

*) The parameter behavior analog outputs at communication error is only analyzed if the failsafe mode is ON.

### Channel parameters for the analog inputs (4x)

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>Internal value</th>
<th>Internal value, type</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input 0, Channel configuration</td>
<td>Operation modes of analog inputs</td>
<td>Operation modes of analog inputs</td>
<td>BYTE</td>
<td>0</td>
</tr>
<tr>
<td>Input 0, Check channel</td>
<td>Settings channel monitoring</td>
<td>Settings channel monitoring</td>
<td>BYTE</td>
<td>0</td>
</tr>
</tbody>
</table>
### Table 600: Channel configuration - Operating modes of the analog inputs

<table>
<thead>
<tr>
<th>Internal Value</th>
<th>Operating Modes (individually configurable)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 (default)</td>
<td>Not used</td>
</tr>
<tr>
<td>1</td>
<td>0…10 V</td>
</tr>
<tr>
<td>2</td>
<td>Digital input</td>
</tr>
<tr>
<td>3</td>
<td>0…20 mA</td>
</tr>
<tr>
<td>4</td>
<td>4…20 mA</td>
</tr>
<tr>
<td>5</td>
<td>-10 V…+10 V</td>
</tr>
<tr>
<td>8</td>
<td>2-wire Pt100 -50…+400 °C</td>
</tr>
<tr>
<td>9</td>
<td>3-wire Pt100 -50…+400 °C *)</td>
</tr>
<tr>
<td>10</td>
<td>0…10 V (voltage diff.) *)</td>
</tr>
<tr>
<td>11</td>
<td>-10 V…+10 V (voltage diff.) *)</td>
</tr>
<tr>
<td>14</td>
<td>2-wire Pt100 -50…+70 °C</td>
</tr>
<tr>
<td>15</td>
<td>3-wire Pt100 -50…+70 °C *)</td>
</tr>
<tr>
<td>16</td>
<td>2-wire Pt1000 -50…+400 °C</td>
</tr>
<tr>
<td>17</td>
<td>3-wire Pt1000 -50…+400 °C *)</td>
</tr>
<tr>
<td>18</td>
<td>2-wire Ni1000 -50…+150 °C</td>
</tr>
<tr>
<td>19</td>
<td>3-wire Ni1000 -50…+150 °C *)</td>
</tr>
</tbody>
</table>

*) In the operating modes with 3-wire configuration or with differential inputs, two adjacent analog inputs belong together (e.g. the channels 0 and 1). In these cases, both channels are configured in the desired operating mode. The lower address must be the even address (channel 0). The next higher address must be the odd address (channel 1). The converted analog value is available at the higher address (channel 1).

### Table 601: Channel monitoring

<table>
<thead>
<tr>
<th>Internal Value</th>
<th>Check Channel</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 (default)</td>
<td>Plausibility, cut wire, short circuit</td>
</tr>
<tr>
<td>3</td>
<td>Not used</td>
</tr>
</tbody>
</table>
### Channel parameters for the analog outputs (2x)

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>Internal value</th>
<th>Internal value, type</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output 0, Channel configuration</td>
<td>Operation modes of analog outputs</td>
<td>Operation modes of analog outputs</td>
<td>BYTE</td>
<td>0</td>
</tr>
<tr>
<td>Output 0, Check channel</td>
<td>Channel monitoring</td>
<td>Channel monitoring</td>
<td>BYTE</td>
<td>0</td>
</tr>
<tr>
<td>Output 0, Substitute value</td>
<td>Substitute value</td>
<td>Substitute value</td>
<td>WORD</td>
<td>0</td>
</tr>
<tr>
<td>Output 1, Channel configuration</td>
<td>Operation modes of analog outputs</td>
<td>Operation modes of analog outputs</td>
<td>BYTE</td>
<td>0</td>
</tr>
<tr>
<td>Output 1, Check channel</td>
<td>Channel monitoring</td>
<td>Channel monitoring</td>
<td>BYTE</td>
<td>0</td>
</tr>
<tr>
<td>Output 1, Substitute value</td>
<td>Substitute value</td>
<td>Substitute value</td>
<td>WORD</td>
<td>0</td>
</tr>
</tbody>
</table>

**Table 602: Channel configuration - Operating modes of the analog outputs**

<table>
<thead>
<tr>
<th>Internal value</th>
<th>Operating Modes (individually configurable)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 (default)</td>
<td>Not used</td>
</tr>
<tr>
<td>128</td>
<td>-10 V...+10 V</td>
</tr>
<tr>
<td>129</td>
<td>0...20 mA</td>
</tr>
<tr>
<td>130</td>
<td>4...20 mA</td>
</tr>
</tbody>
</table>

**Table 603: Channel monitoring**

<table>
<thead>
<tr>
<th>Internal value</th>
<th>Check channel</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Plausibility, cut wire, short circuit</td>
</tr>
<tr>
<td>3</td>
<td>None</td>
</tr>
</tbody>
</table>

**Table 604: Substitute value**

<table>
<thead>
<tr>
<th>Intended Behavior of Output Channel when the Control System Stops</th>
<th>Required Setting of the Module Parameter &quot;Behavior of Outputs in Case of a Communication Error&quot;</th>
<th>Required Setting of the Channel Parameter &quot;Substitute value&quot;</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output OFF</td>
<td>Off</td>
<td>0</td>
</tr>
<tr>
<td>Last value infinite</td>
<td>Last value</td>
<td>0</td>
</tr>
<tr>
<td>Last value for 5 s and then turn off</td>
<td>Last value 5 sec</td>
<td>0</td>
</tr>
<tr>
<td>Last value for 10 s and then turn off</td>
<td>Last value 10 sec</td>
<td>0</td>
</tr>
<tr>
<td>Substitute value infinite</td>
<td>Substitute value</td>
<td>Depending on configuration</td>
</tr>
</tbody>
</table>
### Intended Behavior of Output Channel when the Control System Stops

<table>
<thead>
<tr>
<th>Required Setting of the Module Parameter “Behavior of Outputs in Case of a Communication Error”</th>
<th>Required Setting of the Channel Parameter ”Substitute value”</th>
</tr>
</thead>
<tbody>
<tr>
<td>Substitute value for 5 s and then turn off</td>
<td>Substitute value 5 sec</td>
</tr>
<tr>
<td>Substitute value for 10 s and then turn off</td>
<td>Substitute value 10 sec</td>
</tr>
</tbody>
</table>

### Group parameters for the digital part

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>Internal value</th>
<th>Internal value, type</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input delay</td>
<td>0.1 ms</td>
<td>0</td>
<td>BYTE</td>
<td>0.1 ms 0x00</td>
</tr>
<tr>
<td></td>
<td>1 ms</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>8 ms</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>32 ms</td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Detect short circuit at outputs</td>
<td>Off</td>
<td>0</td>
<td>BYTE</td>
<td>On 0x01</td>
</tr>
<tr>
<td></td>
<td>On</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Behavior digital outputs at communication error 1)</td>
<td>Off</td>
<td>0</td>
<td>BYTE</td>
<td>Off 0x00</td>
</tr>
<tr>
<td></td>
<td>Last value</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Last value 5 sec</td>
<td>6</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Last value 10 sec</td>
<td>11</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Substitute value</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Substitute value 5 sec</td>
<td>7</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Substitute value 10 sec</td>
<td>12</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Substitute value at output</td>
<td>0 ... 255</td>
<td>00h ... FFh</td>
<td>BYTE</td>
<td>0 0x00</td>
</tr>
<tr>
<td>Detect voltage overflow at outputs 2)</td>
<td>Off</td>
<td>0</td>
<td>BYTE</td>
<td>Off 0x00</td>
</tr>
<tr>
<td></td>
<td>On</td>
<td>1</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

1) The parameter Behavior digital outputs at communication error is only analyzed if the failsafe mode is ON.

2) The state "externally voltage detected" appears if the output of a channel DC0..DC7 is to be switched on while an external voltage is connected. Chapter 1.5.3.7.2.3 “Electrical connection” on page 2792. In this case, the start-up is disabled as long as the external voltage is connected. The monitoring of this state and the resulting diagnosis message can be disabled by setting the parameters to "OFF".

### Diagnosis

Structure of the Diagnosis Block via CANOM_NODE_DIAG
In cases of short circuit or overload, the digital outputs are turned off. The module performs reactivation automatically. Thus, an acknowledgement of the errors is not necessary. The error message is stored via the LED.

<table>
<thead>
<tr>
<th>Byte Number</th>
<th>Description</th>
<th>Possible Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Diagnosis byte, slot number</td>
<td>31 = CI581-CN (e. g. error at integrated 8 DI / 8 DO)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 = 1st connected S500 I/O module</td>
</tr>
<tr>
<td></td>
<td></td>
<td>...</td>
</tr>
<tr>
<td></td>
<td></td>
<td>10 = 10th connected S500 I/O module</td>
</tr>
<tr>
<td>2</td>
<td>Diagnosis byte, module number</td>
<td>According to the I/O bus specification passed on by modules to the fieldbus master</td>
</tr>
<tr>
<td>3</td>
<td>Diagnosis byte, channel</td>
<td>According to the I/O bus specification passed on by modules to the fieldbus master</td>
</tr>
<tr>
<td>4</td>
<td>Diagnosis byte, error code</td>
<td>According to the I/O bus specification</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Bit 7 and bit 6, coded error class</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 = E1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 = E2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2 = E3</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3 = E4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Bit 0 to bit 5, coded error description</td>
</tr>
<tr>
<td>5</td>
<td>Diagnosis byte, flags</td>
<td>According to the I/O bus specification</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Bit 7: 1 = coming error</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Bit 6: 1 = leaving error</td>
</tr>
</tbody>
</table>

In cases of short circuit or overload, the digital outputs are turned off. The module performs reactivation automatically. Thus, an acknowledgement of the errors is not necessary. The error message is stored via the LED.
<table>
<thead>
<tr>
<th>Class</th>
<th>Interface</th>
<th>Device</th>
<th>Module</th>
<th>Channel</th>
<th>Error-Identifier</th>
<th>Error message</th>
<th>Remedy</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>-</td>
<td>31</td>
<td>31</td>
<td>31</td>
<td>43</td>
<td>Internal error in the module</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>-</td>
<td>31</td>
<td>31</td>
<td>31</td>
<td>36</td>
<td>Internal data exchange failure</td>
<td>Restart</td>
</tr>
<tr>
<td>3</td>
<td>-</td>
<td>31</td>
<td>31</td>
<td>31</td>
<td>9</td>
<td>Overflow diagnosis buffer</td>
<td>Check Master</td>
</tr>
<tr>
<td>3</td>
<td>-</td>
<td>31</td>
<td>31</td>
<td>31</td>
<td>26</td>
<td>Parameter error</td>
<td>Check Master</td>
</tr>
<tr>
<td>3</td>
<td>-</td>
<td>31</td>
<td>31</td>
<td>31</td>
<td>11</td>
<td>Process voltage UP too low</td>
<td>Check process supply voltage</td>
</tr>
<tr>
<td>3</td>
<td>-</td>
<td>31</td>
<td>31</td>
<td>31</td>
<td>45</td>
<td>Process voltage UP gone</td>
<td>Check process supply voltage</td>
</tr>
<tr>
<td>3</td>
<td>-</td>
<td>31/1...10</td>
<td>31</td>
<td>31</td>
<td>17</td>
<td>No communication with I/O device</td>
<td>Replace I/O module</td>
</tr>
<tr>
<td>3</td>
<td>-</td>
<td>1...10</td>
<td>31</td>
<td>31</td>
<td>32</td>
<td>Wrong I/O device type on socket</td>
<td>Replace I/O module / check configuration</td>
</tr>
<tr>
<td>4</td>
<td>-</td>
<td>1...10</td>
<td>31</td>
<td>31</td>
<td>31</td>
<td>At least one module does not support failsafe function</td>
<td>Check modules and parameterization</td>
</tr>
<tr>
<td>4</td>
<td>-</td>
<td>31</td>
<td>31</td>
<td>31</td>
<td>46</td>
<td>Voltage feedback on activated digital outputs 4)</td>
<td>Check terminals</td>
</tr>
<tr>
<td>4</td>
<td>-</td>
<td>31/1...10</td>
<td>31</td>
<td>31</td>
<td>34</td>
<td>No response during initialization of the I/O module</td>
<td>Replace I/O module</td>
</tr>
<tr>
<td>4</td>
<td>-</td>
<td>31</td>
<td>31</td>
<td>31</td>
<td>11</td>
<td>Process voltage UP3 too low</td>
<td>Check process supply voltage</td>
</tr>
<tr>
<td>Class</td>
<td>Interface</td>
<td>Device</td>
<td>Module</td>
<td>Channel</td>
<td>Error-Identifer</td>
<td>Error message</td>
<td>Remedy</td>
</tr>
<tr>
<td>-------</td>
<td>-----------</td>
<td>--------</td>
<td>--------</td>
<td>---------</td>
<td>----------------</td>
<td>---------------</td>
<td>--------</td>
</tr>
<tr>
<td>Byte 4</td>
<td>Byte 1</td>
<td>Byte 2</td>
<td>Byte 3</td>
<td>Byte 4</td>
<td>CANope</td>
<td>Process voltage UP3 gone</td>
<td>Check process supply voltage</td>
</tr>
<tr>
<td>Bit 6..7</td>
<td>Byte 0..5</td>
<td>CANope</td>
<td>CANope</td>
<td>CANope</td>
<td>CANope</td>
<td>Voltage overflow on outputs (above UP3 level)</td>
<td>Check terminals/ check process supply voltage</td>
</tr>
</tbody>
</table>

Channel error digital

| 4 | - | 31 | 2 | 0...7 | 46 | Voltage feedback on deactivated digital output | Check terminals |
| 4 | - | 31 | 2 | 0...7 | 47 | Short circuit at digital output | Check terminals |

Channel error analog

| 4 | - | 31 | 1 | 0..3 | 48 | Analog value overflow or broken wire at an analog input | Check value or check terminals |
| 4 | - | 31 | 1 | 0..3 | 7 | Analog value underflow at an analog input | Check value |
| 4 | - | 31 | 1 | 0..3 | 47 | Short circuit at an analog input | Check terminals |
| 4 | - | 31 | 3 | 0..1 | 4 | Analog value overflow at an analog output | Check output value |
| 4 | - | 31 | 3 | 0..1 | 7 | Analog value underflow at an analog output | Check output value |

Remarks:
In AC500, the following interface identifier applies:

"-" = Diagnosis via bus-specific function blocks; 0 .. 4 or 10 = position of the communication module; 14 = I/O bus; 31 = module itself

The identifier is not contained in the CI541-DP diagnosis block.

With "Device" the following allocation applies: 31 = module itself; 1..10 = decentralized communication interface module

With "Module" the following allocation applies:

31 = module itself

Channel error: module type (1 = AI, 2 = DO, 3 = AO)

This message appears if external voltages at one or more terminals DO0..DO7 cause other digital outputs to be fed by that voltage (voltage feedback, description in Electrical Connection © Chapter 1.5.3.7.2.2.3 “Electrical connection” on page 2792). All outputs of the digital output groups will be turned off for 5 seconds. The diagnosis message appears for the whole output group.

The voltage on digital outputs DO0..DO7 has overrun the process supply voltage UP3 (description in Electrical Connection © Chapter 1.5.3.7.2.2.3 “Electrical connection” on page 2792). Diagnosis message appears for the whole module.

This message appears if the output of a channel DO0..DO7 is to be switched on while an external voltage is connected. In this case, start-up is disabled while the external voltage is connected. Otherwise, this could produce reverse voltage flowing from this output to other digital outputs. This diagnosis message appears for each channel.

Short circuit: After a short circuit has been detected, the output is deactivated for 100ms seconds. Subsequently, a new start-up will be executed. This diagnosis message appears for each channel.

State LEDs

The state LEDs are located at the front of module. There are 2 different groups:

- The 5 system LEDs (PWR, CN-RUN, CN-ERR, S-ERR and I/O bus) show the operation states of the module and display possible errors.
- The 27 process LEDs (UP, UP3, inputs, outputs, CH-ERR1 to CH-ERR3) show the process supply voltage and the states of the inputs and outputs and display possible errors.

<table>
<thead>
<tr>
<th>LED</th>
<th>Color</th>
<th>OFF</th>
<th>ON</th>
<th>Flashing</th>
</tr>
</thead>
<tbody>
<tr>
<td>PWR/RUN</td>
<td>Green</td>
<td>Process supply voltage missing</td>
<td>Internal supply voltage OK, module ready for communication with I/O controller</td>
<td>Start-up / preparing communi-</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>cation</td>
</tr>
<tr>
<td></td>
<td>Yellow</td>
<td>---</td>
<td>---</td>
<td>---</td>
</tr>
</tbody>
</table>

States of the 5 system LEDs

2020/12/10
<table>
<thead>
<tr>
<th>LED</th>
<th>Color</th>
<th>OFF</th>
<th>ON</th>
<th>Flashing</th>
</tr>
</thead>
<tbody>
<tr>
<td>CN-RUN</td>
<td>Green</td>
<td>---</td>
<td>Device configured, CANopen bus in OPERATIONAL state and cyclic data exchange running</td>
<td>Flashing: CANopen bus in PRE-OPERATIONAL state and slave is being configured</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Single flash: CANopen bus in STOPPED state.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Flickering: Auto-detect is active</td>
</tr>
<tr>
<td>CN-ERR</td>
<td>Red</td>
<td>No system error</td>
<td>CANopen Bus is OFF</td>
<td>Flashing: Configuration error</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Single flash: error counter overflow due to too many error frames</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Double flash: A node-guard or a heartbeat event occurred</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Flickering: Auto-detect is active</td>
</tr>
<tr>
<td>S-ERR</td>
<td>Red</td>
<td>No error</td>
<td>Internal error</td>
<td>--</td>
</tr>
<tr>
<td>I/O bus</td>
<td>Green</td>
<td>No decentralized I/O modules connected or communication error</td>
<td>Decentralized I/O modules connected and operational</td>
<td>--</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>States of the 27 process LEDs:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>AI0 to AI3</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Yellow</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Input is OFF</td>
<td>Input is ON (brightness depends on the value of the analog signal)</td>
<td>--</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>AO0 to AO1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Output is OFF</td>
<td>Output is ON (brightness depends on the value of the analog signal)</td>
<td>--</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>DI0 to DI7</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Input is OFF</td>
<td>Input is ON (the input voltage is even displayed if the supply voltage is OFF)</td>
<td>--</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>DO0 to DO7</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Output is OFF</td>
<td>Output is ON</td>
<td>--</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>UP</td>
</tr>
<tr>
<td></td>
<td>Green</td>
<td>Process supply voltage missing</td>
<td>Process supply voltage OK and initialization finished</td>
<td>--</td>
</tr>
</tbody>
</table>
### Measuring ranges

#### Input ranges voltage, current and digital input

<table>
<thead>
<tr>
<th>Range</th>
<th>0...10 V</th>
<th>-10...+10 V</th>
<th>0...20 mA</th>
<th>4...20 mA</th>
<th>Digital input</th>
<th>Digital value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Overflow</td>
<td>&gt;11.7589</td>
<td>&gt;11.7589</td>
<td>&gt;23.5178</td>
<td>&gt;22.8142</td>
<td></td>
<td>32767 7FFF</td>
</tr>
<tr>
<td>Measured value too high</td>
<td>11.7589 :</td>
<td>10.0004</td>
<td>23.5178 :</td>
<td>20.0007</td>
<td>20.0006</td>
<td>32511 : 27649</td>
</tr>
<tr>
<td>Normal range</td>
<td>10.0000  :</td>
<td>0.0004</td>
<td>20.0000   :</td>
<td>4.0006</td>
<td>On</td>
<td>27648 : 6C00</td>
</tr>
<tr>
<td>Normal range or measured value too low</td>
<td>0.0000   :</td>
<td>0.0000</td>
<td>0         :</td>
<td>4</td>
<td>Off</td>
<td>0 : 0000</td>
</tr>
<tr>
<td>Measured value too low</td>
<td>-10.0004 :</td>
<td>-0.0004</td>
<td>3.9994 : 0</td>
<td>-1</td>
<td>-4864 : -9400</td>
<td></td>
</tr>
<tr>
<td>Underflow</td>
<td>&lt;0.0000</td>
<td>&lt;11.7589</td>
<td>&lt;0.0000</td>
<td>&lt;0.0000</td>
<td>-32768</td>
<td>8000</td>
</tr>
</tbody>
</table>

The represented resolution corresponds to 16 bits.

#### Input ranges resistance temperature detector

<table>
<thead>
<tr>
<th>Range</th>
<th>Pt100 / Pt1000 -50...400 °C</th>
<th>Ni1000 -50...150 °C</th>
<th>Digital value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Overflow</td>
<td>&gt; 450.0 °C</td>
<td>&gt; 160.0 °C</td>
<td>32767 7FFF</td>
</tr>
<tr>
<td>Measured value too high</td>
<td>450.0 °C :</td>
<td>400.1 °C</td>
<td>4500 : 0FA1</td>
</tr>
</tbody>
</table>
### Output ranges voltage and current

<table>
<thead>
<tr>
<th>Range</th>
<th>-10...+10 V</th>
<th>0...20 mA</th>
<th>4...20 mA</th>
<th>Digital value</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Decimal</td>
<td>Hex.</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Overflow</strong></td>
<td>0 V</td>
<td>0 mA</td>
<td>0 mA</td>
<td>&gt; 32511</td>
</tr>
<tr>
<td><strong>Measured value too high</strong></td>
<td>11.7589 V</td>
<td>23.5178 mA</td>
<td>22.8142 mA</td>
<td>32511</td>
</tr>
<tr>
<td></td>
<td>10.0004 V</td>
<td>20.0007 mA</td>
<td>20.0006 mA</td>
<td>27649</td>
</tr>
<tr>
<td><strong>Normal range</strong></td>
<td>10.0000 V</td>
<td>20.0000 mA</td>
<td>20.0000 mA</td>
<td>27648</td>
</tr>
<tr>
<td></td>
<td>0.0004 V</td>
<td>0.0007 mA</td>
<td>4.0006 mA</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>0.0000 V</td>
<td>0.0000 mA</td>
<td>4.0000 mA</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>-0.0004 V</td>
<td>0 mA</td>
<td>3.9994 mA</td>
<td>-1</td>
</tr>
<tr>
<td></td>
<td>-10.0000 V</td>
<td>0 mA</td>
<td>0 mA</td>
<td>-6912</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>-27648</td>
</tr>
</tbody>
</table>
Technical data

The system data of AC500 and S500 \(\text{\textsuperscript{\textregistered}}\) Chapter 1.5.4.5.1 “System data AC500” on page 3058 are valid for standard version.

The system data of AC500-XC \(\text{\textsuperscript{\textregistered}}\) Chapter 1.5.4.6.1 “System data AC500-XC” on page 3099 are valid for the XC version.

Only additional details are therefore documented below.

The technical data are also valid for the XC version.

Multiple overloads

No effects of multiple overloads on isolated multi-channel modules occur, as every channel is protected individually by an internal smart high-side switch.

Technical data of the digital inputs

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of channels per module</td>
<td>8</td>
</tr>
<tr>
<td>Distribution of the channels into groups</td>
<td>1 group of 8 channels</td>
</tr>
<tr>
<td>Terminals of the channels DI0 to DI7</td>
<td>Terminals 3.0 to 3.7</td>
</tr>
<tr>
<td>Reference potential for all inputs</td>
<td>Terminals 2.9 ... 4.9 (negative pole of the supply voltage, signal name ZP)</td>
</tr>
<tr>
<td>Indication of the input signals</td>
<td>1 yellow LED per channel, the LED is ON when the input signal is high (signal 1)</td>
</tr>
<tr>
<td>Input type (according EN 61131-2)</td>
<td>Type 1</td>
</tr>
<tr>
<td>Input delay (0-&gt;1 or 1-&gt;0)</td>
<td>Typ. 0.1 ms, configurable from 0.1...32 ms</td>
</tr>
<tr>
<td>Input signal voltage</td>
<td>24 VDC</td>
</tr>
<tr>
<td>Signal 0</td>
<td>-3 V...+5 V</td>
</tr>
<tr>
<td>Undefined signal</td>
<td>&gt; +5 V...&lt; +15 V</td>
</tr>
<tr>
<td>Signal 1</td>
<td>+15 V...+30 V</td>
</tr>
<tr>
<td>Ripple with signal 0</td>
<td>Within -3 V...+5 V</td>
</tr>
<tr>
<td>Ripple with signal 1</td>
<td>Within +15 V...+30 V</td>
</tr>
<tr>
<td>Input current per channel</td>
<td></td>
</tr>
<tr>
<td>Input voltage +24 V</td>
<td>Typ. 5 mA</td>
</tr>
<tr>
<td>Input voltage +5 V</td>
<td>&gt; 1 mA</td>
</tr>
</tbody>
</table>
### Parameter Data

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input voltage +15 V</td>
<td>&gt; 2 mA</td>
</tr>
<tr>
<td>Input voltage +30 V</td>
<td>&lt; 8 mA</td>
</tr>
<tr>
<td>Max. cable length</td>
<td></td>
</tr>
<tr>
<td>Shielded</td>
<td>1000 m</td>
</tr>
<tr>
<td>Unshielded</td>
<td>600 m</td>
</tr>
</tbody>
</table>

### Technical Data of the Digital Outputs

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of channels per module</td>
<td>8</td>
</tr>
<tr>
<td>Distribution of the channels into groups</td>
<td>1 group of 8 channels</td>
</tr>
<tr>
<td>Terminals of the channels DO0 to DO7</td>
<td>Terminals 4.0 to 4.7</td>
</tr>
<tr>
<td>Reference potential for all outputs</td>
<td>Terminals 2.9 ... 4.9 (negative pole of the supply voltage, signal name ZP)</td>
</tr>
<tr>
<td>Common power supply voltage</td>
<td>For all outputs terminal 4.8 (positive pole of the supply voltage, signal name UP3)</td>
</tr>
<tr>
<td>Output voltage for signal 1</td>
<td>UP3 (-0.8 V)</td>
</tr>
<tr>
<td>Output delay (0-&gt;1 or 1-&gt;0)</td>
<td>On request</td>
</tr>
<tr>
<td>Output current</td>
<td></td>
</tr>
<tr>
<td>Rated value per channel</td>
<td>500 mA at UP3 = 24 V</td>
</tr>
<tr>
<td>Max. value (all channels together)</td>
<td>4 A</td>
</tr>
<tr>
<td>Leakage current with signal 0</td>
<td>&lt; 0.5 mA</td>
</tr>
<tr>
<td>Fuse for UP3</td>
<td>10 A fast</td>
</tr>
<tr>
<td>Demagnetization with inductive DC load</td>
<td>Via internal varistors (see figure below this table)</td>
</tr>
<tr>
<td>Output switching frequency</td>
<td></td>
</tr>
<tr>
<td>With resistive load</td>
<td>On request</td>
</tr>
<tr>
<td>With inductive loads</td>
<td>Max. 0.5 Hz</td>
</tr>
<tr>
<td>With lamp loads</td>
<td>11 Hz max. at 5 W max.</td>
</tr>
<tr>
<td>Short-circuit-proof / overload-proof</td>
<td>Yes</td>
</tr>
<tr>
<td>Overload message (I &gt; 0.7 A)</td>
<td>Yes, after ca. 100 ms</td>
</tr>
<tr>
<td>Output current limitation</td>
<td>Yes, automatic reactivation after short circuit/overload</td>
</tr>
<tr>
<td>Resistance to feedback against 24 V signals</td>
<td>Yes (software-controlled supervision)</td>
</tr>
<tr>
<td>Max. cable length</td>
<td></td>
</tr>
<tr>
<td>Shielded</td>
<td>1000 m</td>
</tr>
<tr>
<td>Unshielded</td>
<td>600 m</td>
</tr>
</tbody>
</table>

The following drawing shows the circuitry of a digital input/output with the varistors for demagnetization when inductive loads are switched off.
Fig. 205: Digital input/output (circuit diagram)

1. Digital output
2. Varistors for demagnetization when inductive loads are turned off

Technical data of the analog inputs

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of channels per module</td>
<td>4</td>
</tr>
<tr>
<td>Distribution of channels into groups</td>
<td>1 group with 4 channels</td>
</tr>
<tr>
<td>Connection if channels AI0+ to AI3+</td>
<td>Terminals 2.0 to 2.3</td>
</tr>
<tr>
<td>Reference potential for AI0+ to AI3+</td>
<td>Terminal 2.4 (AI-) for voltage and RTD measurement</td>
</tr>
<tr>
<td></td>
<td>Terminal 2.9, 3.9 and 4.9 for current measurement</td>
</tr>
<tr>
<td>Input type</td>
<td>Unipolar: Voltage 0...10 V, current or Pt100/Pt1000/Ni1000</td>
</tr>
<tr>
<td></td>
<td>Bipolar: Voltage -10...+10 V</td>
</tr>
<tr>
<td>Galvanic isolation</td>
<td>Against CANopen Bus</td>
</tr>
<tr>
<td>Configurability</td>
<td>0...10 V, -10...+10 V, 0/4...20 mA, Pt100/1000/Ni1000 (each input can be configured individually)</td>
</tr>
<tr>
<td>Channel input resistance</td>
<td>Voltage: &gt; 100 kΩ</td>
</tr>
<tr>
<td></td>
<td>Current: ca. 330 Ω</td>
</tr>
<tr>
<td>Time constant of the input filter</td>
<td>Voltage: 100 µs</td>
</tr>
<tr>
<td></td>
<td>Current: 100 µs</td>
</tr>
<tr>
<td>Indication of the input signals</td>
<td>1 LED per channel (brightness depends on the value of the analog signal)</td>
</tr>
<tr>
<td>Conversion cycle</td>
<td>1 ms (for 4 inputs + 2 outputs); with RTDs Pt/Ni... 1 s</td>
</tr>
<tr>
<td>Resolution</td>
<td>Range 0...10 V: 12 bits</td>
</tr>
<tr>
<td></td>
<td>Range -10...+10 V: 12 bits + sign</td>
</tr>
<tr>
<td></td>
<td>Range 0...20 mA: 12 bits</td>
</tr>
<tr>
<td></td>
<td>Range 4...20 mA: 12 bits</td>
</tr>
<tr>
<td></td>
<td>Range RTD (Pt100, Pt1000, Ni1000): 0.1 °C</td>
</tr>
</tbody>
</table>
### Technical data of the analog inputs if used as digital inputs

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of channels per module</td>
<td>Max. 4</td>
</tr>
<tr>
<td>Distribution of channels into groups</td>
<td>1 group of 4 channels</td>
</tr>
<tr>
<td>Connections of the channels AI0+ to AI3+</td>
<td>Terminals 2.0 to 2.3</td>
</tr>
<tr>
<td>Reference potential for the inputs</td>
<td>Terminals 2.9, 3.9 and 4.9 (ZP)</td>
</tr>
<tr>
<td>Indication of the input signals</td>
<td>1 LED per channel</td>
</tr>
<tr>
<td>Input signal voltage</td>
<td>24 VDC</td>
</tr>
<tr>
<td>Input current per channel</td>
<td></td>
</tr>
<tr>
<td>Input voltage +24 V</td>
<td>Typ. 7 mA</td>
</tr>
<tr>
<td>Input voltage +5 V</td>
<td>Typ. 1.4 mA</td>
</tr>
<tr>
<td>Input voltage +15 V</td>
<td>Typ. 3.7 mA</td>
</tr>
<tr>
<td>Input voltage +30 V</td>
<td>&lt; 9 mA</td>
</tr>
<tr>
<td>Input resistance</td>
<td>Ca. 3.5 kΩ</td>
</tr>
</tbody>
</table>

### Technical data of the analog outputs

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of channels per module</td>
<td>2</td>
</tr>
<tr>
<td>Distribution of channels into groups</td>
<td>1 group for 2 channels</td>
</tr>
<tr>
<td>Connection of the channels AO0+...AO1+</td>
<td>Terminals 1.5...1.6</td>
</tr>
<tr>
<td>Reference potential for AO0+ to AO1+</td>
<td>Terminal 2.7 (AO-) for voltage output</td>
</tr>
<tr>
<td>Output type</td>
<td></td>
</tr>
<tr>
<td>Unipolar</td>
<td>Current</td>
</tr>
</tbody>
</table>
### Technical data of the fast counter

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Used inputs</td>
<td>Terminal 3.0 (DI0), 3.1 (DI1)</td>
</tr>
<tr>
<td>Used outputs</td>
<td>Terminal 4.0 (DO0)</td>
</tr>
<tr>
<td>Counting frequency</td>
<td>Depending on operation mode:</td>
</tr>
<tr>
<td></td>
<td>Mode 1 - 6: max. 200 kHz</td>
</tr>
<tr>
<td></td>
<td>Mode 7: max. 50 kHz</td>
</tr>
<tr>
<td></td>
<td>Mode 9: max. 35 kHz</td>
</tr>
<tr>
<td></td>
<td>Mode 10: max. 20 kHz</td>
</tr>
<tr>
<td>Detailed description</td>
<td>Fast Counter [Chapter 1.5.3.6.1.2.9 “Fast counter” on page 2543]</td>
</tr>
<tr>
<td>Operating modes</td>
<td>Operating modes [Chapter 1.5.3.6.1.2.9 “Fast counter” on page 2543]</td>
</tr>
</tbody>
</table>

### Ordering data

<table>
<thead>
<tr>
<th>Part no.</th>
<th>Description</th>
<th>Product life cycle phase *)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1SAP 228 100 R0001</td>
<td>CI581-CN, CANopen bus module with 8 DI, 8 DO, 4 AI and 2 AO</td>
<td>Active</td>
</tr>
<tr>
<td>1SAP 428 100 R0001</td>
<td>CI581-CN-XC, CANopen bus module with 8 DI, 8 DO, 4 AI and 2 AO, XC version</td>
<td>Active</td>
</tr>
</tbody>
</table>
*) For planning and commissioning of new installations use modules in Active status only.

**CI582-CN**

- 8 digital inputs 24 VDC
- 8 digital outputs 24 VDC, 0.5 A max.
- 8 configurable digital inputs/outputs 24 VDC, 0.5 A max.
- Module-wise electrically isolated
- Fast counter
- XC version for use in extreme ambient conditions available

1. I/O bus
2. Allocation between terminal number and signal name
3. 8 yellow LEDs to display the signal states of the configurable digital inputs/outputs (DC0 - DC7)
4. 8 yellow LEDs to display the signal states of the digital inputs (DI8 - DI15)
5. 8 yellow LEDs to display the signal states of the digital outputs (DO8 - DO15)
6. 2 green LEDs to display the supply voltage UP and UP3
7. 3 red LEDs to display errors (CH-ERR1, CH-ERR2, CH-ERR3)
8. 5 System LEDs: PWR/RUN, CN-RUN, CN-ERR, S-ERR, I/O-Bus
9. Label
10. 2 rotary switches for setting the CANopen node ID
11 10 terminals to connect the CANopen bus signals
12 Terminal unit
13 DIN rail
\* Sign for XC version

**Intended purpose**

The CANopen bus module CI582-CN is used as decentralized I/O module in CANopen networks. Depending on the terminal unit used, the network connection is performed either via a female 9-pin D-sub connector or via 10 terminals (screw or spring terminals) which are integrated in the terminal unit. The bus module contains 24 I/O channels with the following properties:

- 8 digital configurable inputs/outputs in 1 group (1.0...1.7)
- 8 digital inputs 24 VDC in 1 group (2.0...2.7)
- 8 digital outputs 24 VDC in 1 group (3.0...3.7)

The inputs/outputs are electrically isolated from the CANopen network. There is no potential separation between the channels. The configuration of the analog inputs/outputs is performed by software.

For use in extreme ambient conditions (e.g. wider temperature and humidity range), a special XC version of the device is available.

**Functionality**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Interface</td>
<td>CAN</td>
</tr>
<tr>
<td>Protocol</td>
<td>CANopen</td>
</tr>
<tr>
<td>Power supply</td>
<td>From the process supply voltage UP</td>
</tr>
<tr>
<td>Supply of the electronic circuitry of the I/O modules attached</td>
<td>Through the expansion bus interface (I/O bus)</td>
</tr>
<tr>
<td>Rotary switches</td>
<td>For setting the CANopen Node ID for configuration purposes (00h to FFh)</td>
</tr>
<tr>
<td>LED displays</td>
<td>For system displays, signal states, errors and power supply</td>
</tr>
<tr>
<td>External supply voltage</td>
<td>Via terminals ZP, UP and UP3 (process supply voltage 24 VDC)</td>
</tr>
<tr>
<td>Transmission rates</td>
<td>10 / 20 / 50 / 125 / 250 / 500 / 800 kbit/s 1 Mbit/s Auto transmission rate detection is supported</td>
</tr>
<tr>
<td>Bus connection</td>
<td>Depending on used terminal unit TU510: 9-pin D-sub connector TU518: 10-pin terminal block</td>
</tr>
<tr>
<td>Processor</td>
<td>Hilscher netX100</td>
</tr>
<tr>
<td>Expandability</td>
<td>CI582 can only be used on onboard CAN interface and without any I/O expansion module * Table 526 “CANopen” on page 2288.</td>
</tr>
<tr>
<td>State display</td>
<td>Module state: PWR/RUN, CN-RUN, CN-ERR, E-ERR, I/O bus</td>
</tr>
<tr>
<td>Adjusting elements</td>
<td>2 rotary switches for generation of the node address</td>
</tr>
<tr>
<td>Parameter</td>
<td>Value</td>
</tr>
<tr>
<td>-----------------------------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Ambient temperature</td>
<td>System data AC500 “Chapter 1.5.4.5.1 “System data AC500” on page 3058 System data AC500 XC “Chapter 1.5.4.6.1 “System data AC500-XC” on page 3099</td>
</tr>
<tr>
<td>Current consumption</td>
<td>UP: 0.2 A UP3: 0.06 A + 0.5 A max. per output</td>
</tr>
<tr>
<td>Weight (without terminal unit)</td>
<td>Ca. 125 g</td>
</tr>
<tr>
<td>Process supply voltages UP/UP3</td>
<td></td>
</tr>
<tr>
<td>Rated value</td>
<td>24 VDC (for inputs and outputs)</td>
</tr>
<tr>
<td>Max. load for the terminals</td>
<td>10 A</td>
</tr>
<tr>
<td>Protection against reversed voltage</td>
<td>Yes</td>
</tr>
<tr>
<td>Rated protection fuse on UP/UP3</td>
<td>10 A fast</td>
</tr>
<tr>
<td>Galvanic isolation</td>
<td>CANopen interface against the rest of the module</td>
</tr>
<tr>
<td>Inrush current from UP (at power up)</td>
<td>On request</td>
</tr>
<tr>
<td>Current consumption via UP (normal operation)</td>
<td>0.2 A</td>
</tr>
<tr>
<td>Current consumption via UP3</td>
<td>0.06 A + 0.5 A max. per output</td>
</tr>
<tr>
<td>Connections</td>
<td>Terminals 2.8 and 3.8 for +24 V (UP) Terminal 4.8 for +24 V (UP3) Terminals 2.9, 3.9 and 4.9 for 0 V (ZP)</td>
</tr>
<tr>
<td>Max. power dissipation within the module</td>
<td>6 W</td>
</tr>
<tr>
<td>Reference potential for all digital inputs and outputs</td>
<td>Minus pole of the supply voltage, signal name ZP</td>
</tr>
<tr>
<td>Setting of the CANopen Node ID identifier</td>
<td>With 2 rotary switches at the front side of the module</td>
</tr>
<tr>
<td>Mounting position</td>
<td>Horizontal Or vertical with derating (output load reduced to 50 % at 40 °C per group)</td>
</tr>
<tr>
<td>Cooling</td>
<td>The natural convection cooling must not be hindered by cable ducts or other parts in the switch-gear cabinet.</td>
</tr>
<tr>
<td>Effect of incorrect input terminal connection</td>
<td>Wrong or no signal detected, no damage up to 35 V</td>
</tr>
<tr>
<td>Required terminal unit</td>
<td>TU509, TU510, TU517 or TU518 “Chapter 1.5.3.5.3 “TU517 and TU518 for communication interface modules” on page 2317</td>
</tr>
</tbody>
</table>

All I/O channels (digital and analog) are protected against reverse polarity, reverse supply, short circuit and continuous overvoltage up to 30 VDC.
## Electrical connection

The CANopen bus module is plugged on the I/O terminal units TU517 "TU517 and TU518 for communication interface modules" on page 2317 or TU518 "TU517 and TU518 for communication interface modules" on page 2317 and accordingly TU509 or TU510. Properly position the module and press until it locks in place.

The electrical connection of the I/O channels is established using the 30 terminals of the I/O terminal unit. I/O modules can be replaced without re-wiring the terminal units.

The terminals 2.8, 3.8, 2.9, 3.9 and 4.9 are electrically interconnected within the terminal unit and always have the same assignment, irrespective of the inserted module:

- Terminals 2.8 and 3.8: process supply voltage $UP = +24$ VDC
- Terminal 4.8: process supply voltage $UP3 = +24$ VDC
- Terminals 2.9, 3.9 and 4.9: process supply voltage $ZP = 0$ V

For a detailed description of the mounting, disassembly and electrical connection of the module, please refer to the System Assembly, Construction and Connection chapter "AC500 (Standard)" on page 3058.

With a separate $UP3$ power supply, the digital outputs can be switched off externally. This way, an emergency-off functionality can be realized.

## Possibilities of connection

### Mounting on terminal units TU509 or TU510

The assignment of the 9-pin female D-sub connector for the CANopen signals:

<table>
<thead>
<tr>
<th>Terminal</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>---</td>
</tr>
<tr>
<td>2</td>
<td>CAN-</td>
</tr>
<tr>
<td>3</td>
<td>CAN_GND</td>
</tr>
<tr>
<td>4</td>
<td>---</td>
</tr>
<tr>
<td>5</td>
<td>---</td>
</tr>
<tr>
<td>6</td>
<td>---</td>
</tr>
<tr>
<td>7</td>
<td>CAN+</td>
</tr>
<tr>
<td>8</td>
<td>---</td>
</tr>
</tbody>
</table>

For reference: 3ADR010583, 1, en_US
Bus terminating resistors

The ends of the data lines have to be terminated with a 120 Ω bus terminating resistor. The bus terminating resistor is usually installed directly at the bus connector.

Fig. 206: CANopen interface, bus terminating resistors connected to the line ends

<table>
<thead>
<tr>
<th></th>
<th>CAN_GND</th>
<th>CAN_L</th>
<th>Shield</th>
<th>CAN_H</th>
<th>Data line, shielded twisted pair</th>
<th>COMBICON connection, CANopen interface</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Fig. 207: DeviceNet interface, bus terminating resistors connected to the line ends
The earthing of the shield should take place at the switch-gear. Please refer to Chapter 1.5.4.5.1 “System data AC500” on page 3058.

<table>
<thead>
<tr>
<th>Terminal</th>
<th>Signal</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0</td>
<td>CAN+</td>
<td>Non-inverted signal of the CAN Bus</td>
</tr>
<tr>
<td>1.1</td>
<td>CAN+</td>
<td>Non-inverted signal of the CAN Bus</td>
</tr>
<tr>
<td>1.2</td>
<td>CAN-</td>
<td>Inverted signal of the CAN Bus</td>
</tr>
<tr>
<td>1.3</td>
<td>CAN-</td>
<td>Inverted signal of the CAN Bus</td>
</tr>
<tr>
<td>1.4</td>
<td>Term+</td>
<td>CAN bus termination for CAN+ (for bus termination, Term+ must be connected with CAN+)</td>
</tr>
<tr>
<td>1.5</td>
<td>Term+</td>
<td>CAN bus termination for CAN+ (connecting alternative for terminal 1.4)</td>
</tr>
<tr>
<td>1.6</td>
<td>Term-</td>
<td>CAN bus termination for CAN- (for bus termination, Term- must be connected with CAN-)</td>
</tr>
<tr>
<td>1.7</td>
<td>Term-</td>
<td>CAN bus termination for CAN- (connecting alternative for terminal 1.6)</td>
</tr>
<tr>
<td>1.8</td>
<td>CAN-GND</td>
<td>Ground potential of the CAN bus</td>
</tr>
<tr>
<td>1.9</td>
<td>CAN-GND</td>
<td>Ground potential of the CAN bus</td>
</tr>
</tbody>
</table>

At the line ends of a bus segment, termination resistors must be connected. If TU517 or TU518 is used, the bus termination resistors can be enabled by connecting the terminals Term+ and Term- to the data lines CAN+ and CAN- (no external termination resistors are required, see illustration below).

The following figures show the different connection options for the CANopen bus module:
In the case of TU517/TU518, the termination resistors are not located inside the TU but inside the bus module CI581-CN. Hence, when removing the device from the TU, the bus termination resistors are no longer connected to the bus. The bus itself will not be disconnected if a device is removed.

The earthing of the shield should take place at the switch-gear cabinet. Please refer to the AC500 System-Data Chapter 1.5.4.5.1 “System data AC500” on page 3058.
<table>
<thead>
<tr>
<th>Terminal</th>
<th>Signal</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.0</td>
<td>DC0</td>
<td>Signal of the configurable digital input/output DC0</td>
</tr>
<tr>
<td>2.1</td>
<td>DC1</td>
<td>Signal of the configurable digital input/output DC1</td>
</tr>
<tr>
<td>2.2</td>
<td>DC2</td>
<td>Signal of the configurable digital input/output DC2</td>
</tr>
<tr>
<td>2.3</td>
<td>DC3</td>
<td>Signal of the configurable digital input/output DC3</td>
</tr>
<tr>
<td>2.4</td>
<td>DC4</td>
<td>Signal of the configurable digital input/output DC4</td>
</tr>
<tr>
<td>2.5</td>
<td>DC5</td>
<td>Signal of the configurable digital input/output DC5</td>
</tr>
<tr>
<td>2.6</td>
<td>DC6</td>
<td>Signal of the configurable digital input/output DC6</td>
</tr>
<tr>
<td>2.7</td>
<td>DC7</td>
<td>Signal of the configurable digital input/output DC7</td>
</tr>
<tr>
<td>2.8</td>
<td>UP</td>
<td>Process voltage UP (24 VDC)</td>
</tr>
<tr>
<td>2.9</td>
<td>ZP</td>
<td>Process voltage ZP (0 VDC)</td>
</tr>
<tr>
<td>3.0</td>
<td>DI8</td>
<td>Signal of the digital input DI8</td>
</tr>
<tr>
<td>3.1</td>
<td>DI9</td>
<td>Signal of the digital input DI9</td>
</tr>
<tr>
<td>3.2</td>
<td>DI10</td>
<td>Signal of the digital input DI10</td>
</tr>
<tr>
<td>3.3</td>
<td>DI11</td>
<td>Signal of the digital input DI11</td>
</tr>
<tr>
<td>3.4</td>
<td>DI12</td>
<td>Signal of the digital input DI12</td>
</tr>
<tr>
<td>3.5</td>
<td>DI13</td>
<td>Signal of the digital input DI13</td>
</tr>
<tr>
<td>3.6</td>
<td>DI14</td>
<td>Signal of the digital input DI14</td>
</tr>
<tr>
<td>3.7</td>
<td>DI15</td>
<td>Signal of the digital input DI15</td>
</tr>
<tr>
<td>3.8</td>
<td>UP</td>
<td>Process voltage UP (24 VDC)</td>
</tr>
<tr>
<td>3.9</td>
<td>ZP</td>
<td>Process voltage ZP (0 VDC)</td>
</tr>
<tr>
<td>4.0</td>
<td>DO8</td>
<td>Signal of the digital output DO8</td>
</tr>
<tr>
<td>4.1</td>
<td>DO9</td>
<td>Signal of the digital output DO9</td>
</tr>
<tr>
<td>4.2</td>
<td>DO10</td>
<td>Signal of the digital output DO10</td>
</tr>
<tr>
<td>4.3</td>
<td>DO11</td>
<td>Signal of the digital output DO11</td>
</tr>
<tr>
<td>4.4</td>
<td>DO12</td>
<td>Signal of the digital output DO12</td>
</tr>
<tr>
<td>4.5</td>
<td>DO13</td>
<td>Signal of the digital output DO13</td>
</tr>
<tr>
<td>4.6</td>
<td>DO14</td>
<td>Signal of the digital output DO14</td>
</tr>
<tr>
<td>4.7</td>
<td>DO15</td>
<td>Signal of the digital output DO15</td>
</tr>
<tr>
<td>4.8</td>
<td>UP3</td>
<td>Process voltage UP3 (24 VDC)</td>
</tr>
<tr>
<td>4.9</td>
<td>ZP</td>
<td>Process voltage ZP (0 VDC)</td>
</tr>
</tbody>
</table>
WARNING!
Removal/Insertion under power

The devices are not designed for removal or insertion under power. Because of unforeseeable consequences, it is not allowed to plug or unplug devices with the power being ON.

Make sure that all voltage sources (supply and process voltage) are switched off before you
- connect or disconnect any signal or terminal block
- remove, mount or replace a module.

Disconnecting any powered devices while energized in a hazardous location could result in an electric arc, which could create a flammable ignition resulting in fire or explosion.

Make sure that power is removed and that the area has been thoroughly checked to ensure that flammable materials are not present prior to proceeding.

The devices must not be opened when in operation. The same applies to the network interfaces.

NOTICE!
Risk of damaging the PLC modules!

Overvoltages and short circuits might damage the PLC modules.
- Make sure that all voltage sources (supply voltage and process supply voltage) are switched off before you begin with operations at the system.
- Never connect any voltages or signals to reserved terminals (marked with ---). Reserved terminals may carry internal voltages.

Electrical connection of CANopen bus module CI582-CN:

Fig. 208: Connection of the bus module CI582-CN

For a description of the meaning of the LEDs, please refer to the section for the state LEDs Chapter 1.5.3.7.2.3.9 “State LEDs” on page 2843.
Bus length
The maximum possible bus length of a CAN network depends on bit rate (transmission rate) and cable type. The sum of all bus segments must not exceed the maximum bus length.

<table>
<thead>
<tr>
<th>Bit Rate (speed)</th>
<th>Bus Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Mbit/s</td>
<td>40 m</td>
</tr>
<tr>
<td>800 kbit/s</td>
<td>50 m</td>
</tr>
<tr>
<td>500 kbit/s</td>
<td>100 m</td>
</tr>
<tr>
<td>250 kbit/s</td>
<td>250 m</td>
</tr>
<tr>
<td>125 kbit/s</td>
<td>500 m</td>
</tr>
<tr>
<td>50 kbit/s</td>
<td>1000 m</td>
</tr>
</tbody>
</table>

Connection of the digital inputs
The following figure shows the electrical connection of the digital input DI8. Proceed with the digital inputs DI9 to DI15 in the same way.

![Connection of the digital inputs](image)

**Fig. 209: Connection of the digital inputs to the module CI582-CN**

Connection of the digital outputs
The following figure shows the electrical connection of the digital output DO8. Proceed with the digital outputs DO9 - DO15 in the same way.

![Connection of the digital outputs](image)
Connection of the configurable digital inputs/outputs

The following figure shows the electrical connection of the configurable digital input/output DC0 and DC1. DC0 is connected as an input and DC1 is connected as an output. Proceed with the configurable digital inputs/outputs DC2 to DC7 in the same way.
Internal data exchange

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Digital inputs (bytes)</td>
<td>5</td>
</tr>
<tr>
<td>Digital outputs (bytes)</td>
<td>5</td>
</tr>
<tr>
<td>Counter input data (words)</td>
<td>4</td>
</tr>
<tr>
<td>Counter output data (words)</td>
<td>8</td>
</tr>
</tbody>
</table>

Addressing

A detailed description concerning addressing can be found in the documentation of ABB Control Builder Plus Software.

> The CANopen bus module reads the position of the rotary switches only during power-up, i.e. changes of the switch position during operation will have no effect until the next module initialization.

> The range of permitted CANopen slave addresses is 1 to 127. Setting a higher address (> 128) does not lead to an error response, but results in a special mode (DS401). In this special mode, the device creates the node address by subtracting the value 128 from the address switch's value.

I/O configuration

The CI582-CN CANopen bus configuration is handled by CANopen master with the exception of the slave node ID (via rotary switches) and the transmission rate (automatic detection).

The digital I/O channels and the fast counter are configured via software.

Parameterization

Parameters of the module

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>Internal value</th>
<th>Internal value, type</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>Module ID ¹)</td>
<td>Internal</td>
<td>0x1C89</td>
<td>WORD</td>
<td>0x1C89</td>
</tr>
<tr>
<td>Parameter length</td>
<td>Internal</td>
<td>38</td>
<td>BYTE</td>
<td>38</td>
</tr>
<tr>
<td>Error LED / failsafe function</td>
<td>On</td>
<td>0</td>
<td>BYTE</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>Off by E4</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Off by E3</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>On + failsafe</td>
<td>16</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Off by E4 + failsafe</td>
<td>17</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Off by E3 + failsafe</td>
<td>18</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Reserved</td>
<td>0</td>
<td>0</td>
<td>ARRAY of 24 BYTES</td>
<td></td>
</tr>
<tr>
<td>Check supply</td>
<td>On</td>
<td>0</td>
<td>BYTE</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Off</td>
<td>1</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
With a faulty ID, the module reports a "parameter error" and does not perform cyclic process data transmission.

For a description of the counter operating modes, please refer to the Fast Counter section Chapter 1.5.3.6.1.2.9 “Fast counter” on page 2543.

<table>
<thead>
<tr>
<th>Setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>On</td>
<td>Error LED (S-ERR) lights up at errors of all error classes, failsafe mode off</td>
</tr>
<tr>
<td>Off by E4</td>
<td>Error LED (S-ERR) lights up at errors of error classes E1, E2 and E3, failsafe mode off</td>
</tr>
<tr>
<td>Off by E3</td>
<td>Error LED (S-ERR) lights up at errors of error classes E1 and E2, failsafe mode off</td>
</tr>
<tr>
<td>On + Failsafe</td>
<td>Error LED (S-ERR) lights up at errors of all error classes, failsafe mode on *)</td>
</tr>
<tr>
<td>Off by E4 + Failsafe</td>
<td>Error LED (S-ERR) lights up at errors of error classes E1, E2 and E3, failsafe mode on *)</td>
</tr>
<tr>
<td>Off by E3 + Failsafe</td>
<td>Error LED (S-ERR) lights up at errors of error classes E1 and E2, failsafe mode on *)</td>
</tr>
</tbody>
</table>

*) The parameter Behavior DO at comm. error is only analyzed if the failsafe mode is ON.

---

Group parameters for the digital part

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>Internal value</th>
<th>Internal value, type</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input delay</td>
<td>0.1 ms</td>
<td>0</td>
<td>BYTE</td>
<td>0.1 ms</td>
</tr>
<tr>
<td></td>
<td>1 ms</td>
<td>1</td>
<td></td>
<td>0x00</td>
</tr>
<tr>
<td></td>
<td>8 ms</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>32 ms</td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Detect short circuit at outputs</td>
<td>Off</td>
<td>0</td>
<td>BYTE</td>
<td>On</td>
</tr>
<tr>
<td></td>
<td>On</td>
<td>1</td>
<td></td>
<td>0x01</td>
</tr>
<tr>
<td>Behavior DO at comm. error</td>
<td>Off</td>
<td>0</td>
<td>BYTE</td>
<td>Off</td>
</tr>
<tr>
<td></td>
<td>Last value</td>
<td>6</td>
<td></td>
<td>0x00</td>
</tr>
<tr>
<td></td>
<td>Last value 5 sec</td>
<td>11</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Last value 10 sec</td>
<td>11</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Substitute value</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Substitute value 5 sec</td>
<td>7</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Substitute value 10 sec</td>
<td>12</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Name</td>
<td>Value</td>
<td>Internal value</td>
<td>Internal value, type</td>
<td>Default</td>
</tr>
<tr>
<td>-------------------------------------------</td>
<td>-------------</td>
<td>----------------</td>
<td>----------------------</td>
<td>---------------</td>
</tr>
<tr>
<td>Substitute value at output</td>
<td>0 ... 65535</td>
<td>0000h ... FFFFh</td>
<td>WORD</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0x0000</td>
</tr>
<tr>
<td>Preventive voltage feedback monitoring for DC0..DC7 2)</td>
<td>Off</td>
<td>0</td>
<td>BYTE</td>
<td>Off</td>
</tr>
<tr>
<td></td>
<td>On</td>
<td>1</td>
<td></td>
<td>0x00</td>
</tr>
<tr>
<td>Detect voltage overflow at outputs 3)</td>
<td>Off</td>
<td>0</td>
<td>BYTE</td>
<td>Off</td>
</tr>
<tr>
<td></td>
<td>On</td>
<td>1</td>
<td></td>
<td>0x00</td>
</tr>
</tbody>
</table>

Remarks:

1) The parameter Behavior DO at comm. error is applied to DC and DO channels and only analyzed if the failsafe mode is ON.

2) The state "externally voltage detected" appears if the output of a channel DC0..DC7 is to be switched on while an external voltage is connected. In this case, start-up is disabled while the externally voltage is connected. The monitoring of this state and the resulting diagnosis message can be disabled by setting the parameters to "OFF".

3) The error state "voltage overflow at outputs" appears if external voltage at digital outputs DC0..DC7 and DO0..DO7 has exceeded the process supply voltage UP3 (see Electrical Connection "Electrical connection" on page 2829). The according diagnosis message "Voltage overflow on outputs " can be disabled by setting the parameters to "OFF". This parameter should only be disabled in exceptional cases as voltage overflow may produce reverse voltage.

Diagnosis

Structure of the diagnosis block via CANOM NODE_DIAG

<table>
<thead>
<tr>
<th>Byte Number</th>
<th>Description</th>
<th>Possible Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Diagnosis byte, slot number</td>
<td>31 = CI582-CN (e.g. error at integrated 8 DI / 8 DO)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 = 1st connected S500 I/O module</td>
</tr>
<tr>
<td></td>
<td></td>
<td>...</td>
</tr>
<tr>
<td></td>
<td></td>
<td>10 = 10th connected S500 I/O module</td>
</tr>
<tr>
<td>2</td>
<td>Diagnosis byte, module number</td>
<td>According to the I/O bus specification passed on by modules to the fieldbus master</td>
</tr>
<tr>
<td>3</td>
<td>Diagnosis byte, channel</td>
<td>According to the I/O bus specification passed on by modules to the fieldbus master</td>
</tr>
</tbody>
</table>
### Byte Number

<table>
<thead>
<tr>
<th>Byte Number</th>
<th>Description</th>
<th>Possible Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>Diagnosis byte, error code</td>
<td>According to the I/O bus specification&lt;br&gt;Bit 7 and bit 6, coded error class&lt;br&gt;0 = E1&lt;br&gt;1 = E2&lt;br&gt;2 = E3&lt;br&gt;3 = E4&lt;br&gt;Bit 0 to Bit 5, coded error description</td>
</tr>
<tr>
<td>5</td>
<td>Diagnosis byte, flags</td>
<td>According to the I/O bus specification&lt;br&gt;Bit 7: 1 = coming error&lt;br&gt;Bit 6: 1 = leaving error</td>
</tr>
</tbody>
</table>

In cases of short circuit or overload, the digital outputs are turned off. The module performs reactivation automatically. Thus, an acknowledgement of the errors is not necessary. The error message is stored via the LED.

### Module errors

<table>
<thead>
<tr>
<th>Class</th>
<th>Interface</th>
<th>Device</th>
<th>Module</th>
<th>Channel</th>
<th>Error identifier</th>
<th>Error message</th>
<th>Remedy</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td></td>
<td>31</td>
<td>31</td>
<td>31</td>
<td>19</td>
<td>Checksum error in the I/O module</td>
<td>Replace I/O module</td>
</tr>
<tr>
<td>3</td>
<td></td>
<td>31</td>
<td>31</td>
<td>31</td>
<td>3</td>
<td>Timeout in the I/O module</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td>31</td>
<td>31</td>
<td>31</td>
<td>40</td>
<td>Different hard-/firmware versions in the module</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td>31</td>
<td>31</td>
<td>31</td>
<td>43</td>
<td>Internal error in the module</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td>31</td>
<td>31</td>
<td>31</td>
<td>36</td>
<td>Internal data exchange failure</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td>31</td>
<td>31</td>
<td>31</td>
<td>9</td>
<td>Overflow diagnosis buffer</td>
<td>Restart</td>
</tr>
<tr>
<td>3</td>
<td></td>
<td>31</td>
<td>31</td>
<td>31</td>
<td>26</td>
<td>Parameter error</td>
<td>Check Master</td>
</tr>
<tr>
<td>Class</td>
<td>Interface</td>
<td>Device</td>
<td>Module</td>
<td>Channel</td>
<td>Error identifier</td>
<td>Error message</td>
<td>Remedy</td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>1)</td>
<td>2)</td>
<td>3)</td>
<td>4)</td>
<td>10</td>
<td>11</td>
<td>Process voltage UP too low</td>
<td>Check process supply voltage</td>
</tr>
<tr>
<td>2)</td>
<td>3)</td>
<td>4)</td>
<td>10</td>
<td>11</td>
<td>45</td>
<td>Process voltage UP gone</td>
<td>Check process supply voltage</td>
</tr>
<tr>
<td>3)</td>
<td>4)</td>
<td>10</td>
<td>11</td>
<td>45</td>
<td>Voltage feedback on activated digital outputs</td>
<td>Check terminals</td>
<td>Check terminals</td>
</tr>
<tr>
<td>4)</td>
<td>10</td>
<td>11</td>
<td>45</td>
<td>Voltage feedback on activated digital outputs</td>
<td>Check terminals</td>
<td>Check terminals</td>
<td></td>
</tr>
<tr>
<td>5)</td>
<td>10</td>
<td>11</td>
<td>46</td>
<td>No response during initialization of the I/O module</td>
<td>Replace I/O module</td>
<td>Replace I/O module</td>
<td></td>
</tr>
<tr>
<td>6)</td>
<td>10</td>
<td>11</td>
<td>45</td>
<td>No response during initialization of the I/O module</td>
<td>Replace I/O module</td>
<td>Replace I/O module</td>
<td></td>
</tr>
</tbody>
</table>
### Remarks:

1) In AC500, the following interface identifier applies:
   
   
   - "-" = Diagnosis via bus-specific function blocks; 0 ... 4 or 10 = position of the communication module; 14 = I/O bus; 31 = module itself
   
   The identifier is not contained in the CI542-DP diagnosis block.

2) With "Device" the following allocation applies: 31 = module itself, 1..10 = decentralized communication interface module

3) With "Module" the following allocation applies depending on the master:
   
   - Module error: 31 = module itself
   - Channel error: module type (1 = AI, 2 = DO, 3 = AO)

4) This message appears if external voltages at one or more terminals DC0..DC7 or DO0..DO7 cause other digital outputs to be supplied by that voltage (voltage feedback, see Electrical Connection Chapter 1.5.3.7.2.3.3 "Electrical connection" on page 2829). All outputs of the digital output groups will be turned off for 5 seconds. The diagnosis message appears for the whole output group.

5) The voltage at digital outputs DC0..DC7 and DO0..DO7 has exceeded the process supply voltage UP3 (see Electrical Connection Chapter 1.5.3.7.2.3.3 "Electrical connection" on page 2829). A diagnosis message appears for the whole module.
This message appears if the output of a channel DC0..DC7 or DO0..DO7 should be switched on while an external voltage is connected. In this case the start-up is disabled while the external voltage is connected. Otherwise, this could produce reverse voltage flowing from this output to other digital outputs. This diagnosis message appears for each channel.

Short circuit: After a short circuit has been detected, the output is deactivated for 100ms. Subsequently, a new start-up will be executed. This diagnosis message appears for each channel.

State LEDs

The LEDs are located at the front of the module. There are 2 different groups:

- The 5 system LEDs (PWR, CN-RUN, CN-ERR, S-ERR and I/O bus) show the operation states of the module and display possible errors.
- The 29 process LEDs (UP, UP3, inputs, outputs, CH-ERR1 to CH-ERR3) show the process supply voltage and the states of the inputs and outputs and display possible errors.

### States of the 5 system LEDs

<table>
<thead>
<tr>
<th>LED</th>
<th>Color</th>
<th>OFF</th>
<th>ON</th>
<th>Flashing</th>
</tr>
</thead>
<tbody>
<tr>
<td>PWR/RUN</td>
<td>Green</td>
<td>Process supply voltage missing</td>
<td>Internal supply voltage OK, module ready for communication with I/O controller</td>
<td>Start-up / preparing communication</td>
</tr>
<tr>
<td></td>
<td>Yellow</td>
<td>---</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>CN-RUN</td>
<td>Green</td>
<td>---</td>
<td>Device configured, CANopen bus in OPERATIONAL state and cyclic data exchange running</td>
<td>Flashing: CANopen bus in PRE-OPERATIONAL state and slave is being configured Single flash: CANopen bus in STOPPED state Flickering: Autodetect is active</td>
</tr>
<tr>
<td>CN-ERR</td>
<td>Red</td>
<td>No system error</td>
<td>CANopen Bus is OFF</td>
<td>Flashing: Configuration error Single flash: error counter overflow due to too many error frames Double flash: A node-guard or a heartbeat event occurred Flickering: Autodetect is active</td>
</tr>
<tr>
<td>S-ERR</td>
<td>Red</td>
<td>No error</td>
<td>Internal error</td>
<td>--</td>
</tr>
<tr>
<td>I/O bus</td>
<td>Green</td>
<td>No decentralized I/O modules connected or communication error</td>
<td>Decentralized I/O modules connected and operational</td>
<td>---</td>
</tr>
</tbody>
</table>
### Technical data

The system data of AC500 and S500 \( \mathcal{O} \) Chapter 1.5.4.5.1 “System data AC500” on page 3058 are valid for standard version.

The system data of AC500-XC \( \mathcal{O} \) Chapter 1.5.4.6.1 “System data AC500-XC” on page 3099 are valid for the XC version.

Only additional details are therefore documented below. The technical data are also valid for the XC version.

#### Multiple overloads

No effects of multiple overloads on isolated multi-channel modules occur, as every channel is protected individually by an internal smart high-side switch.

#### Technical data of the digital inputs

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of channels per module</td>
<td>8</td>
</tr>
<tr>
<td>Distribution of the channels into groups</td>
<td>1 group of 8 channels</td>
</tr>
<tr>
<td>Terminals of the channels DI0 to DI7</td>
<td>Terminals 3.0 to 3.7</td>
</tr>
<tr>
<td>Reference potential for all inputs</td>
<td>Terminals 2.9 ... 4.9 (negative pole of the supply voltage, signal name ZP)</td>
</tr>
<tr>
<td>Indication of the input signals</td>
<td>1 yellow LED per channel, the LED is ON when the input signal is high (signal 1)</td>
</tr>
<tr>
<td>Input type (according EN 61131-2)</td>
<td>Type 1</td>
</tr>
<tr>
<td>Input delay (0-&gt;1 or 1-&gt;0)</td>
<td>Typ. 0.1 ms, configurable from 0.1...32 ms</td>
</tr>
<tr>
<td>Input signal voltage</td>
<td>24 VDC</td>
</tr>
<tr>
<td>Signal 0</td>
<td>-3 V...+5 V</td>
</tr>
<tr>
<td>Undefined signal</td>
<td>&gt; +5 V...&lt; +15 V</td>
</tr>
<tr>
<td>Signal 1</td>
<td>+15 V...+30 V</td>
</tr>
</tbody>
</table>
### Technical data of the digital outputs

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of channels per module</td>
<td>8</td>
</tr>
<tr>
<td>Distribution of the channels into groups</td>
<td>1 group of 8 channels</td>
</tr>
<tr>
<td>Terminals of the channels DO0 to DO7</td>
<td>Terminals 4.0 to 4.7</td>
</tr>
<tr>
<td>Reference potential for all outputs</td>
<td>Terminals 2.9 ... 4.9 (negative pole of the supply voltage, signal name ZP)</td>
</tr>
<tr>
<td>Common power supply voltage</td>
<td>For all outputs terminal 4.8 (positive pole of the supply voltage, signal name UP3)</td>
</tr>
<tr>
<td>Output voltage for signal 1</td>
<td>UP3 (-0.8 V)</td>
</tr>
<tr>
<td>Output delay (0-&gt;1 or 1-&gt;0)</td>
<td>On request</td>
</tr>
<tr>
<td>Output current</td>
<td></td>
</tr>
<tr>
<td>Rated value per channel</td>
<td>500 mA at UP3 = 24 V</td>
</tr>
<tr>
<td>Max. value (all channels together)</td>
<td>4 A</td>
</tr>
<tr>
<td>Leakage current with signal 0</td>
<td>&lt; 0.5 mA</td>
</tr>
<tr>
<td>Fuse for UP3</td>
<td>10 A fast</td>
</tr>
<tr>
<td>Demagnetization with inductive DC load</td>
<td>Via internal varistors (see figure below this table)</td>
</tr>
<tr>
<td>Output switching frequency</td>
<td></td>
</tr>
<tr>
<td>With resistive load</td>
<td>On request</td>
</tr>
<tr>
<td>With inductive loads</td>
<td>Max. 0.5 Hz</td>
</tr>
<tr>
<td>With lamp loads</td>
<td>11 Hz max. at 5 W max.</td>
</tr>
<tr>
<td>Short-circuit-proof / overload-proof</td>
<td>Yes</td>
</tr>
<tr>
<td>Overload message (I &gt; 0.7 A)</td>
<td>Yes, after ca. 100 ms</td>
</tr>
<tr>
<td>Output current limitation</td>
<td>Yes, automatic reactivation after short circuit/overload</td>
</tr>
<tr>
<td>Resistance to feedback against 24 V signals</td>
<td>Yes (software-controlled supervision)</td>
</tr>
<tr>
<td>Max. cable length</td>
<td></td>
</tr>
<tr>
<td>Parameter</td>
<td>Value</td>
</tr>
<tr>
<td>--------------------</td>
<td>----------</td>
</tr>
<tr>
<td>Shielded</td>
<td>1000 m</td>
</tr>
<tr>
<td>Unshielded</td>
<td>600 m</td>
</tr>
</tbody>
</table>

The following drawing shows the circuitry of a digital input/output with the varistors for demagnetization when inductive loads are switched off.

![Circuit Diagram](image)

Fig. 212: Digital input/output (circuit diagram)

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Digital output</td>
</tr>
<tr>
<td>2</td>
<td>Varistors for demagnetization when inductive loads are turned off</td>
</tr>
</tbody>
</table>

**Technical data of the configurable digital inputs/outputs**

Each of the configurable I/O channels is defined as input or output by the user program. This is done by interrogating or allocating the corresponding channel.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of channels per module</td>
<td>8 inputs/outputs (with transistors)</td>
</tr>
<tr>
<td>Distribution of the channels into groups</td>
<td>1 group for 8 channels</td>
</tr>
<tr>
<td>If the channels are used as inputs</td>
<td></td>
</tr>
<tr>
<td>Channels DC0...DC07</td>
<td>Terminals 2.0...2.7</td>
</tr>
<tr>
<td>If the channels are used as outputs</td>
<td></td>
</tr>
<tr>
<td>Channels DC0...DC07</td>
<td>Terminals 2.0...2.7</td>
</tr>
<tr>
<td>Indication of the input/output signals</td>
<td>1 yellow LED per channel, the LED is ON when the input/output signal is high (signal 1)</td>
</tr>
<tr>
<td>Galvanic isolation</td>
<td>From the CANopen network</td>
</tr>
</tbody>
</table>

**Technical data of the digital inputs/outputs if used as inputs**

Please refer to the Technical Data of the Digital Inputs. Deviation: Chapter 1.5.3.7.2.3.10 “Technical data” on page 2844. Deviation:

Terminals of the channels DC0 to DC7: Terminals 2.0 to 2.7

Due to the direct connection to the output, the demagnetizing varistor is also effective at the input. This is why the difference between UPx and the input signal must not exceed the clamp voltage of the varistor. The varistor limits the clamp voltage to approx. 36 V. Consequently, the input voltage must range from -12 V to +30 V when UPx = 24 V and from -6 V to +30 V when UPx = 30 V.
Please refer to the Technical Data of the Digital Outputs "Technical data" on page 2844. Deviation:

Terminals of the channels DC0 to DC7: Terminals 2.0 to 2.7

The following drawing shows the circuitry of a digital input/output with the varistors for demagnetization when inductive loads are switched off.

Fig. 213: Digital input/output (circuit diagram)

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Digital input/output</td>
</tr>
<tr>
<td>2</td>
<td>For demagnetization when inductive loads are turned off</td>
</tr>
</tbody>
</table>

### Technical data of the fast counter

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Used inputs</td>
<td>Terminal 3.0 (DI8), 3.1 (DI9)</td>
</tr>
<tr>
<td>Used outputs</td>
<td>Terminal 4.0 (DO8)</td>
</tr>
<tr>
<td>Counting frequency</td>
<td>Depending on operation mode:</td>
</tr>
<tr>
<td></td>
<td>Mode 1 - 6: max. 200 kHz</td>
</tr>
<tr>
<td></td>
<td>Mode 7: max. 50 kHz</td>
</tr>
<tr>
<td></td>
<td>Mode 9: max. 35 kHz</td>
</tr>
<tr>
<td></td>
<td>Mode 10: max. 20 kHz</td>
</tr>
<tr>
<td>Detailed description</td>
<td>Fast Counter &quot;Chapter 1.5.3.6.1.2.9 &quot;Fast counter&quot; on page 2543</td>
</tr>
<tr>
<td>Operating modes</td>
<td>Operating modes &quot;Chapter 1.5.3.6.1.2.9 &quot;Fast counter&quot; on page 2543</td>
</tr>
</tbody>
</table>

### Ordering data

<table>
<thead>
<tr>
<th>Part no.</th>
<th>Description</th>
<th>Product life cycle phase *)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1SAP 228 200 R0001</td>
<td>CI582-CN, CANopen bus module with 8 DI, 8 DO and 8 DC</td>
<td>Active</td>
</tr>
<tr>
<td>1SAP 428 200 R0001</td>
<td>CI582-CN-XC, CANopen bus module with 8 DI, 8 DO and 8 DC, XC version</td>
<td>Active</td>
</tr>
</tbody>
</table>

*) For planning and commissioning of new installations use modules in Active status only.
1.5.3.7.3 EtherCAT

CI511-ETHCAT

- 4 analog inputs (resolution 12 bits plus sign)
- 2 analog outputs (resolution 12 bits plus sign)
- 8 digital inputs 24 VDC
- 8 digital outputs 24 VDC, 0.5 A max.
- Cam switch functionality (see also Extended Cam Switch Library)
- Extended Cam switch functionality *) (see also Extended Cam Switch Library)
- Module-wise electrically isolated - Expandability with up to 10 S500 I/O Modules *)

*) Applicable for device index C0 and above.

---

1. I/O bus
2. Allocation between terminal number and signal name
3. 6 yellow LEDs to display the signal states of the analog inputs/outputs (AI0 - AI3, AO0 - AO1)
4. 8 yellow LEDs to display the signal states of the digital inputs (DI0 - DI7)
5. 8 yellow LEDs to display the signal states of the digital outputs (DO0 - DO7)
6. 2 green LEDs to display the supply voltage UP and UP3
7. 3 red LEDs to display errors (CH-ERR1, CH-ERR2, CH-ERR3)
8. 5 system LEDs: PWR/RUN, NET, DC, S-ERR, I/O-Bus
9. 2 rotary switches (reserved for future extensions)
10. Label
11. Ethernet interfaces (ETH1, ETH2) on the terminal unit
12. Terminal unit
13. DIN rail
Intended purpose

The EtherCAT bus module CI511-ETHCAT is used as decentralized I/O module in EtherCAT networks. The network connection is performed via 2 RJ45 connectors which are integrated in the terminal unit. The bus module contains 22 I/O channels with the following properties:

- 4 analog inputs (1.0...1.3)
- 2 analog outputs (1.5...1.6)
- 8 digital inputs 24 VDC in 1 group (2.0...2.7)
- 8 digital outputs 24 VDC in 1 group (3.0...3.7)
- Cam switch functionality

The inputs/outputs are electrically isolated from the Ethernet network. There is no potential separation between the channels. The configuration of the analog inputs/outputs is performed by software.

Functionality

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Interface</td>
<td>Ethernet</td>
</tr>
<tr>
<td>Protocol</td>
<td>EtherCAT</td>
</tr>
<tr>
<td>Power supply</td>
<td>From the process supply voltage UP</td>
</tr>
<tr>
<td>Supply of the electronic circuitry of</td>
<td>Through the expansion bus interface (I/O bus)</td>
</tr>
<tr>
<td>the I/O expansion modules attached</td>
<td></td>
</tr>
<tr>
<td>Rotary switches</td>
<td>Not used; reserved for future extensions</td>
</tr>
<tr>
<td>Analog inputs</td>
<td>4 (configurable via software)</td>
</tr>
<tr>
<td>Analog outputs</td>
<td>2 (configurable via software)</td>
</tr>
<tr>
<td>Digital inputs</td>
<td>8 (24 VDC; delay time configurable via software)</td>
</tr>
<tr>
<td>Digital outputs</td>
<td>8 (24 VDC, 0.5 A max.)</td>
</tr>
<tr>
<td>LED displays</td>
<td>For system displays, signal states, errors and power supply</td>
</tr>
<tr>
<td>External supply voltage</td>
<td>Via terminals ZP, UP and UP3 (process supply voltage 24 VDC)</td>
</tr>
<tr>
<td>Effect of incorrect input terminal connection</td>
<td>Wrong or no signal detected, no damage up to 35 V</td>
</tr>
<tr>
<td>Required terminal unit</td>
<td>TU507 or TU508 [Chapter 1.5.3.5.1 “TU507-ETH and TU508-ETH for Ethernet communication interface modules” on page 2307]</td>
</tr>
</tbody>
</table>

Electrical connection

The EtherCAT bus module CI511-ETHCAT is plugged on the I/O terminal unit TU507-ETH or TU508-ETH. Properly seat the module and press until it locks in place. The terminal unit is mounted on a DIN rail or with 2 screws plus the additional accessory for wall mounting (TA526).

For a detailed description of the mounting, disassembly and electrical connection of the module, please refer to the System Assembly, Construction and Connection chapter [Chapter 1.5.4.5 “AC500 (Standard)” on page 3058].

The electrical connection of the I/O channels is carried out using the 30 terminals of the I/O terminal unit. I/O modules can be replaced without re-wiring the terminal units.
The terminals 1.8 and 2.8 as well as 1.9, 2.9 and 3.9 are electrically interconnected within the terminal unit and have always the same assignment, independent of the inserted module:

Terminals 1.8 and 2.8: Process supply voltage \( \text{UP} = +24\ \text{VDC} \)

Terminal 3.8: Process supply voltage \( \text{UP3} = +24\ \text{VDC} \)

Terminals 1.9, 2.9 and 3.9: Process supply voltage \( \text{ZP} = 0\ \text{V} \)

With a separate \( \text{UP3} \) power supply, the digital outputs can be switched off externally. This way, an emergency-off functionality can be realized.

The assignment of the other terminals:

<table>
<thead>
<tr>
<th>Terminal</th>
<th>Signal</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0 to 1.3</td>
<td>AI0 to AI3</td>
<td>Plus poles of the 4 analog inputs</td>
</tr>
<tr>
<td>1.4</td>
<td>AI-</td>
<td>Minus pole of the analog inputs</td>
</tr>
<tr>
<td>1.5 to 1.6</td>
<td>AO0 to AO1</td>
<td>Plus poles of the 2 analog outputs</td>
</tr>
<tr>
<td>1.7</td>
<td>AO-</td>
<td>Minus pole of the analog outputs</td>
</tr>
<tr>
<td>2.0 to 2.7</td>
<td>DI0 to DI7</td>
<td>8 digital inputs</td>
</tr>
<tr>
<td>3.0 to 3.7</td>
<td>DO0 to DO7</td>
<td>8 digital outputs</td>
</tr>
</tbody>
</table>

**WARNING!**

Removal/Insertion under power

The devices are not designed for removal or insertion under power. Because of unforeseeable consequences, it is not allowed to plug or unplug devices with the power being ON.

Make sure that all voltage sources (supply and process voltage) are switched off before you

- connect or disconnect any signal or terminal block
- remove, mount or replace a module.

Disconnecting any powered devices while energized in a hazardous location could result in an electric arc, which could create a flammable ignition resulting in fire or explosion.

Make sure that power is removed and that the area has been thoroughly checked to ensure that flammable materials are not present prior to proceeding.

The devices must not be opened when in operation. The same applies to the network interfaces.

**CAUTION!**

There is no electrical isolation between the analog circuitry and ZP/UP. Therefore, the analog sensors must be electrically isolated in order to avoid loops via the earth potential or the supply voltage.
CAUTION!
Because of their common reference potential, analog current inputs cannot be
circuited in series, neither within the module nor with channels of other mod-
ules.

For the open-circuit detection (cut wire), each channel is pulled up to "plus" by a
high-resistance resistor. If nothing is connected, the maximum voltage will be
read in then.

Analog signals are always laid in shielded cables. The cable shields are earthed at both ends of
the cables. In order to avoid unacceptable potential differences between different parts of the
installation, low resistance equipotential bonding conductors must be laid.

For simple applications (low disturbances, no high requirement on precision), the shielding can
also be omitted.

The following figures show the electrical connection of the Ethernet bus module CI511-ETHCAT.

Fig. 214: Connection of the bus module CI511-ETHCAT
1 4 analog inputs, configurable for 0...10 V, -10...+10 V, 0/4...20 mA, Pt100/Pt1000, Ni1000
   and digital signals
2 2 analog outputs, configurable for -10...+10 V, 0/4...20 mA
3 8 digital inputs 24 VDC
4 8 digital outputs 24 VDC, 0.5 A max.
In case of voltage feedback, 2 cases are distinguished:

1. The outputs are already active
   The output group will be switched off. A diagnosis message will appear. After 5 seconds, the module tries automatic reactivation.

2. The outputs are not active
   Only the output with voltage feedback will not be set to active. A diagnosis message will appear.

NOTICE!
Risk of faulty measurements!
The negative pole/earthing potential at the sensors must not have too large a potential difference with respect to ZP (max. ±1 V within the full signal range).
Make sure that the potential difference never exceeds ±1 V.

CAUTION!
The process supply voltage must be included within the earthing concept of the plant (e.g. earthing of the minus pole).

The module provide several diagnosis functions “Diagnosis” on page 2869.
The measuring ranges are described in the section Measuring Ranges “Parameterization” on page 2863 “Measuring ranges” on page 2872.
The function of the LEDs is described in the section State LEDs “Diagnosis” on page 2869.

Connection of resistance thermometers in 2-wire configuration
When resistance thermometers (Pt100, Pt1000, Ni1000) are used, a constant current must flow through them to build the necessary voltage drop for the evaluation. For this, the module C1511-ETHCAT provides a constant current source which is multiplexed over the max. 4 analog input channels.
The following figure shows the connection of resistance thermometers in 2-wire configuration.
Fig. 215: Connection of resistance thermometers in 2-wire configuration

1. Pt100 (2-wire), Pt1000 (2-wire), Ni1000 (2-wire); 1 analog sensor requires 1 channel

<table>
<thead>
<tr>
<th>Resistor</th>
<th>Measuring Range</th>
<th>Configuration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pt100</td>
<td>-50 °C...+400 °C</td>
<td>2-wire, 1 channel used</td>
</tr>
<tr>
<td>Pt1000</td>
<td>-50 °C...+400 °C</td>
<td>2-wire, 1 channel used</td>
</tr>
<tr>
<td>Ni1000</td>
<td>-50 °C...+150 °C</td>
<td>2-wire, 1 channel used</td>
</tr>
</tbody>
</table>

The measuring ranges are described in the section Measuring Ranges > Chapter 1.5.3.7.3.1.7 "Parameterization" on page 2863 and Chapter 1.5.3.7.3.1.10 "Measuring ranges" on page 2872.

The module CI511-ETHCAT performs a linearization of the resistance characteristic.

In order to avoid error messages from unused analog input channels, it is useful to configure them as "unused".

Connection of resistance thermometers in 3-wire configuration

When resistance thermometers (Pt100, Pt1000, Ni1000) are used, a constant current must flow through them to build the necessary voltage drop for the evaluation. For this, the module CI511-ETHCAT provides a constant current source which is multiplexed over the max. 4 analog input channels.

The following figure shows the connection of resistance thermometers in 3-wire configuration.
Fig. 216: Connection of resistance thermometers in 3-wire configuration

1. Pt100 (3-wire), Pt1000 (3-wire), Ni1000 (3-wire); 1 analog sensor requires 2 channels
2. Twisted pair within the cable
3. Return line: The return line is only needed once if measuring points are adjacent to each other. This saves wiring costs.

With 3-wire configuration, two adjacent analog channels belong together (e.g. the channels 0 and 1). In this case, both channels are configured according to the desired operating mode. The lower address must be the even address (channel 0), the next higher address must be the odd address (channel 1).

The constant current of one channel flows through the resistance thermometer. The constant current of the other channel flows through one of the cores. The module calculates the measured value from the two voltage drops and stores it under the input with the higher channel number (e.g. I1).

In order to keep measuring errors as small as possible, it is necessary, to have all the involved conductors in the same cable. All the conductors must have the same cross section.

<table>
<thead>
<tr>
<th></th>
<th>-50 °C...+400 °C</th>
<th>3-wire configuration, 2 channels used</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pt100</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Pt1000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ni1000</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The measuring ranges are described in the section Measuring Ranges “Parameterization” on page 2863 and “Measuring ranges” on page 2872.

The module CI511-ETHCAT performs a linearization of the resistance characteristic.

In order to avoid error messages from unused analog input channels, it is useful to configure them as "unused".
Connection of active-type analog sensors (Voltage) with electrically isolated power supply

The following figure shows the connection of active-type analog sensors (voltage) with electrically isolated power supply:

Fig. 217: Connection of active-type analog sensors (voltage) with electrically isolated power supply

1. 1 analog sensor requires 1 channel
2. By connecting to AI-, the electrically isolated voltage source of the sensor is referred to ZP
3. Electrically isolated power supply for the analog sensor

<table>
<thead>
<tr>
<th>Voltage</th>
<th>0...10 V</th>
<th>1 channel used</th>
</tr>
</thead>
<tbody>
<tr>
<td>Voltage</td>
<td>-10 V...+10 V</td>
<td>1 channel used</td>
</tr>
</tbody>
</table>

The measuring ranges are described in the section Measuring Ranges "Parameterization" on page 2863 or Chapter 1.5.3.7.3.1.10 “Measuring ranges” on page 2872.

In order to avoid error messages or long processing times, it is useful to configure unused analog input channels as "unused".

Connection of active-type analog sensors (Current) with electrically isolated power supply

The following figure shows the connection of active-type analog sensors (current) with electrically isolated power supply.
Fig. 218: Connection of active-type analog sensors (current) with electrically isolated power supply

1. 1 analog sensor requires 1 channel
2. Electrically isolated power supply for the analog sensor

<table>
<thead>
<tr>
<th>Current</th>
<th>0...20 mA</th>
<th>1 channel used</th>
</tr>
</thead>
<tbody>
<tr>
<td>Current</td>
<td>4...20 mA</td>
<td>1 channel used</td>
</tr>
</tbody>
</table>

The measuring ranges are described in the section Measuring Ranges & Chapter 1.5.3.7.3.1.7 “Parameterization” on page 2863 & Chapter 1.5.3.7.3.1.10 “Measuring ranges” on page 2872.

Unused input channels can be left open-circuited, because they are of low resistance.

Connection of active-type analog sensors (Voltage) with no electrically isolated power supply

The following figure shows the connection of active-type sensors (voltage) with no electrically isolated power supply.
Fig. 219: Connection of active-type sensors (voltage) with no electrically isolated power supply

1. Analog sensor requires 1 channel
2. Power supply not electrically isolated
3. The connection between the minus pole of the sensor and ZP has to be performed
4. Long cable

**NOTICE!**
Risk of faulty measurements!

The negative pole/earthing potential at the sensors must not have too large a potential difference with respect to ZP (max. ±1 V within the full signal range).

Make sure that the potential difference never exceeds ±1 V.

<table>
<thead>
<tr>
<th>Voltage</th>
<th>0...10 V</th>
<th>1 channel used</th>
</tr>
</thead>
<tbody>
<tr>
<td>Voltage</td>
<td>-10 V...+10 V *)</td>
<td>1 channel used</td>
</tr>
</tbody>
</table>

*) if the sensor can provide this signal range

The measuring ranges are described in the section Measuring Ranges © Chapter 1.5.3.7.3.1.7 “Parameterization” on page 2863 © Chapter 1.5.3.7.3.1.10 “Measuring ranges” on page 2872.

In order to avoid error messages or long processing times, it is useful to configure unused analog input channels as “unused”.

**Connection of passive-type analog sensors (Current)**

The following figure shows the connection of passive-type analog sensors (current).
Fig. 220: Connection of passive-type analog sensors (current)

1 analog sensor requires 1 channel

<table>
<thead>
<tr>
<th>Current</th>
<th>4...20 mA</th>
<th>1 channel used</th>
</tr>
</thead>
</table>

The measuring ranges are described in the section Measuring Ranges \ Chapter 1.5.3.7.3.1.17 “Parameterization” on page 2863 \ Chapter 1.5.3.7.3.1.10 “Measuring ranges” on page 2872.

**CAUTION!**

If, during initialization, an analog current sensor supplies more than 25 mA for more than 1 second into an analog input, this input is switched off by the module (input protection). In such cases, it is recommended, to protect the analog input by a 10-volt zener diode (in parallel to I+ and I-). But, in general, it is a better solution to prefer sensors with fast initialization or without current peaks higher than 25 mA.

Unused input channels can be left open-circuited, because they are of low resistance.

**Connection of active-type analog sensors (Voltage) to differential inputs**

Differential inputs are very useful, if analog sensors are used which are remotely non-isolated (e.g. the minus terminal is remotely earthed).

The evaluation using differential inputs helps to considerably increase the measuring accuracy and to avoid earthing loops.

With differential input configurations, two adjacent analog channels belong together (e.g. the channels 0 and 1). In this case, both channels are configured according to the desired operating mode. The lower address must be the even address (channel 0), the next higher address must be the odd address (channel 1). The converted analog value is available at the higher address (channel 1).

The analog value is calculated by subtraction of the input value with the higher address from the input value of the lower address.

The converted analog value is available at the odd channel (higher address).
Important: The earthing potential at the sensors must not have a too big potential difference with respect to ZP (max. ±1 V within the full signal range). Otherwise problems can occur concerning the common-mode input voltages of the involved analog inputs.

The following figure shows the connection of active-type analog sensors (voltage) to differential inputs.

![Diagram of sensor connection]

**Fig. 221: Connection of active-type analog sensors (voltage) to differential inputs**
1. 1 analog sensor requires 2 channels
2. Electrically isolated power supply for the analog sensor
3. Earthing at the sensor
4. 0 V..10 V / -10 V..+10 V connected to differential inputs

<table>
<thead>
<tr>
<th>Voltage</th>
<th>0 V..10 V</th>
<th>with differential inputs, 2 channels used</th>
</tr>
</thead>
<tbody>
<tr>
<td>Voltage</td>
<td>-10 V..+10 V</td>
<td>with differential inputs, 2 channels used</td>
</tr>
</tbody>
</table>

The measuring ranges are described in the section Measuring Ranges "Parameterization" on page 2863 and Chapter 1.5.3.7.3.1.10 "Measuring ranges" on page 2872.

In order to avoid error messages or long processing times, it is useful to configure unused analog input channels as "unused".

**Use of analog inputs as digital inputs**

Several (or all) analog inputs can be configured as digital input. The inputs are not electrically isolated against the other analog channels.

The following figure shows the use of analog inputs as digital inputs.
**Fig. 222: Use of analog inputs as digital inputs**

1. 1 digital signal requires 1 channel

<table>
<thead>
<tr>
<th>Digital input</th>
<th>24 V</th>
<th>1 channel used</th>
</tr>
</thead>
</table>

The measuring ranges are described in the section Measuring Ranges Page 2863 and Chapter 1.5.3.7.3.10 “Measuring ranges” on page 2872.

**Connection of analog output loads (Voltage, current)**

The following figure shows the connection of analog output loads (voltage, current).
Fig. 223: Connection of analog output loads (voltage, current)

1 analog load requires 1 channel

<table>
<thead>
<tr>
<th>Voltage</th>
<th>-10 V...+10 V</th>
<th>Load ±10 mA max.</th>
<th>1 channel used</th>
</tr>
</thead>
<tbody>
<tr>
<td>Current</td>
<td>0...20 mA</td>
<td>Load 0...500 Ω</td>
<td>1 channel used</td>
</tr>
<tr>
<td>Current</td>
<td>4...20 mA</td>
<td>Load 0...500 Ω</td>
<td>1 channel used</td>
</tr>
</tbody>
</table>

The measuring ranges are described in the section Measuring Ranges "Parameterization" on page 2863. Unused analog outputs can be left open-circuited.

Assignment of the Ethernet ports

The terminal unit for the communication interface module provides two Ethernet interfaces with the following pin assignment. The pin assignment is used for the EtherCAT master (communication module CM5xy-ETHCAT) as well.
Table 608: Pin assignment RJ45 jack:

<table>
<thead>
<tr>
<th>Interface</th>
<th>Pin</th>
<th>Signal</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
<td>TxD+</td>
<td>Transmit data +</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>TxD-</td>
<td>Transmit data -</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>RxD+</td>
<td>Receive data +</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>NC</td>
<td>not used</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>NC</td>
<td>not used</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>RxD-</td>
<td>Receive data -</td>
</tr>
<tr>
<td></td>
<td>7</td>
<td>NC</td>
<td>not used</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>NC</td>
<td>not used</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Shield</td>
<td>Cable shield</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Functional earth</td>
</tr>
</tbody>
</table>

For further information regarding wiring and cable types see chapter Ethernet § Chapter 1.5.4.4.7 “Ethernet connection details” on page 3084.

The EtherCAT network differentiates between input-connectors (IN) and output-connectors (OUT):

At the EtherCAT slaves (communication interface modules), the ETH1-connector is IN and the ETH2-connector is OUT.

At the EtherCAT master (communication module), the ETHCAT1 connector has to be used. The ETHCAT2 connector is reserved for future extensions.

Internal data exchange

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Digital inputs (bytes)</td>
<td>1</td>
</tr>
<tr>
<td>Digital outputs (bytes)</td>
<td>1</td>
</tr>
<tr>
<td>Analog inputs (words)</td>
<td>4</td>
</tr>
<tr>
<td>Analog outputs (words)</td>
<td>2</td>
</tr>
</tbody>
</table>

Addressing

The Ethernet bus module CI511-ETHCAT does not consider the position of the rotary switches at the front side of the module. The function of the rotary switches is reserved for future expansions.
I/O configuration

In order to be able to use the CI51X-ETHCAT with device index C0 or above properly, please download the corresponding device description (.xml-)files from http://www.abb.com/plc and install them to the device repository of your Automation Builder. This will allow you to use up to 10 Expandable S500 I/O modules as well as the Extended Cam Switch Library with your CI51X-ETHCAT device.

The CI511-ETHCAT does not store configuration data itself.
The analog I/O channels are configured via software.

Parameterization

Module parameter

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>Internal value</th>
<th>Internal value, type</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>Module ID</td>
<td>Internal</td>
<td>48155</td>
<td>WORD</td>
<td>48155</td>
</tr>
<tr>
<td>Parameter length</td>
<td>Internal</td>
<td>28</td>
<td>BYTE</td>
<td>28</td>
</tr>
<tr>
<td>Error LED / Fail-safe function 1)</td>
<td>On</td>
<td>0</td>
<td>BYTE</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>Off by E4</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Off by E3 On + failsafe Off by E4</td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Off by E3 + failsafe</td>
<td>16</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Off by E3 + failsafe</td>
<td>17</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Off by E3 + failsafe</td>
<td>19</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Check Supply</td>
<td>Off</td>
<td>0</td>
<td>BYTE</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>On</td>
<td>1</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 609: Error LED / Failsafe function 1)

<table>
<thead>
<tr>
<th>Setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>On</td>
<td>Error LED lights up at errors of all error classes, Failsafe mode off</td>
</tr>
<tr>
<td>Off by E4</td>
<td>Error LED lights up at errors of error classes E1, E2 and E3, Failsafe mode off</td>
</tr>
<tr>
<td>Off by E3</td>
<td>Error LED lights up at errors of error classes E1 and E2 auf, Failsafe mode off</td>
</tr>
<tr>
<td>On + failsafe</td>
<td>Error LED lights up at errors of all error classes, Failsafe mode on *)</td>
</tr>
<tr>
<td>Off by E4 + failsafe</td>
<td>Error LED lights up at errors of error classes E1, E2 and E3, Failsafe mode on *)</td>
</tr>
<tr>
<td>Off by E3 + failsafe</td>
<td>Error LED lights up at errors of error classes E1 and E2, Failsafe mode on *)</td>
</tr>
</tbody>
</table>

*) The parameters behaviourAOatCommunicationFault and behaviourDOatCommunicationFault are only analyzed if the Failsafe-mode is ON.
Group parameters of the cam switch

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>Internal value</th>
<th>Internal value, type</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>numOfUsed-Cams 1)</td>
<td>0 ... 32</td>
<td>0 ... 32</td>
<td>WORD</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>128...160</td>
<td>218...160</td>
<td></td>
<td></td>
</tr>
<tr>
<td>resolution 2)</td>
<td>0 ... 2</td>
<td>0 ... 2</td>
<td>DWORD</td>
<td>36000</td>
</tr>
<tr>
<td></td>
<td>-1</td>
<td>-1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>zeroShift 3)</td>
<td>0 ... 2</td>
<td>0 ... 2</td>
<td>DWORD</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>-1</td>
<td>-1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>EncoderBitResolution 4)</td>
<td>8 ... 32</td>
<td>8 ... 32</td>
<td>WORD</td>
<td>18</td>
</tr>
<tr>
<td>Reserve</td>
<td>-</td>
<td>-</td>
<td>WORD</td>
<td>-</td>
</tr>
</tbody>
</table>

1) The parameter numOfUsedCams defines the interrupt cycle time (Therefore, it takes effect to the accuracy of the track) and the behaviour of the module if the DC information is lost.

<table>
<thead>
<tr>
<th>Parameter setting for numOfUsed-Cams</th>
<th>Number of cams used</th>
<th>Interrupt cycle time</th>
<th>Behaviour if DC information is lost</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>50 µs</td>
<td>Module changes to &quot;safe-operational&quot; state; the outputs are activated through the user program</td>
</tr>
<tr>
<td>1...8</td>
<td>1...8</td>
<td>80 µs</td>
<td></td>
</tr>
<tr>
<td>9...16</td>
<td>9...16</td>
<td>100 µs</td>
<td></td>
</tr>
<tr>
<td>17...32</td>
<td>17...32</td>
<td>200 µs</td>
<td></td>
</tr>
<tr>
<td>128</td>
<td>0</td>
<td>50 µs</td>
<td>Module keeps in &quot;operational&quot; state; the outputs are activated through the user program</td>
</tr>
<tr>
<td>129...136</td>
<td>1...8</td>
<td>80 µs</td>
<td>Module keeps in &quot;operational&quot; state; the cam switch outputs are activated according to an interpolated timing information</td>
</tr>
<tr>
<td>137...144</td>
<td>9...16</td>
<td>100 µs</td>
<td></td>
</tr>
<tr>
<td>145...170</td>
<td>17...32</td>
<td>200 µs</td>
<td></td>
</tr>
</tbody>
</table>

2) The parameter resolution defines the angle resolution of the track. The value gives the number of increments related to 360°; e. g. the value 36,000 corresponds to an angle resolution of 0.01°.

3) The parameter zeroShift defines the zero shift. With it the encoder can be adjusted to the mounting position. The value of zeroShift is set in encoder-increments. It is not assigned to the parameter resolution of the cam switch.

4) The parameter EncoderBitResolution defines the resolution of the used encoder (in bits), e. g. with the default setting 18 bits the encoder has 196,608 divisions.
Channel parameters for the cam switch (max. 32x)

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>Internal value</th>
<th>Internal value, type</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>camToTrack0 *)</td>
<td>Digital Output 0 ... 7, none</td>
<td>0 ... 7, FF</td>
<td>BYTE</td>
<td>FF</td>
</tr>
<tr>
<td></td>
<td>:</td>
<td>:</td>
<td>:</td>
<td>:</td>
</tr>
<tr>
<td>camToTrack31</td>
<td>Digital Output 0 ... 7, none</td>
<td>0 ... 7, FF</td>
<td>BYTE</td>
<td>FF</td>
</tr>
</tbody>
</table>

*) The value of the parameter camToTrack# defines which DO (digital output) is assigned to the track. camToTrack0 = 3 for example means that track 0 is assigned to the digital output 3. If the value FFh is set to a track, no digital output is assigned to it.

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>Referred FB from extended Cam Switch Library *)</th>
<th>Internal value</th>
<th>Internal value, type</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>camType[0] 1)</td>
<td>Common Pulsed Timed Comfort Cam shift Binary shift Multiturn cam Time timed Reference Multiturn timed</td>
<td>MCX_CamSwitchSimple_c MCX_CamSwitchSimple_dc MCX_PulseSwitch_dc MCX_CamSwitchTimed_dc MCX_CamSwitchComfort_dc MCX_CamShift_dc MCX_BinaryShift_dc MCX_CamSwitchMulti_dc MCX_SwitchTimeTimed_dc MCX_CamSwitchMultiTimed_dc</td>
<td>0</td>
<td>BYTE</td>
<td>0</td>
</tr>
</tbody>
</table>

1) camType additionally to camToTrack identifies the type of each cam switch and enables the use of a specific Function Block from the Extended Cam Switch Library.

2) camType parameters and the Extended Camswitch Library are only available for CI511-ETHCAT and CI512-ETHCAT with device index C0 and above.
Group parameters for the analog part

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>Internal value</th>
<th>Internal value, type</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>Analog data format</td>
<td>Standard</td>
<td>0</td>
<td>BYTE</td>
<td>0</td>
</tr>
<tr>
<td>Behaviour AO at comm. error *)</td>
<td>Off</td>
<td>0</td>
<td>BYTE</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>Last value</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Last value 5 s</td>
<td>6</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Last value 10 s</td>
<td>11</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Substitute value</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Substitute value 5 s</td>
<td>7</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Substitute value 10 s</td>
<td>12</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

*) The parameter Behaviour AO at comm. error is only analyzed if the Failsafe-mode is ON.

Channel parameters for the analog inputs (4x)

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>Internal value</th>
<th>Internal value, type</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input 0, channel configuration</td>
<td>see ¹)</td>
<td>see ¹)</td>
<td>BYTE</td>
<td>0</td>
</tr>
<tr>
<td>Input 0, check channel</td>
<td>see ²)</td>
<td>see ²)</td>
<td>BYTE</td>
<td>0</td>
</tr>
<tr>
<td>Input 3, channel configuration</td>
<td>see ¹)</td>
<td>see ¹)</td>
<td>BYTE</td>
<td>0</td>
</tr>
<tr>
<td>Input 3, channel configuration</td>
<td>see ²)</td>
<td>see ²)</td>
<td>BYTE</td>
<td>0</td>
</tr>
</tbody>
</table>

Channel configuration ²)

<table>
<thead>
<tr>
<th>Internal value</th>
<th>Operating modes of the analog inputs, individually configurable</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 (default)</td>
<td>Not used</td>
</tr>
<tr>
<td>1</td>
<td>0...10 V</td>
</tr>
<tr>
<td>2</td>
<td>Digital input</td>
</tr>
<tr>
<td>3</td>
<td>0...20 mA</td>
</tr>
<tr>
<td>4</td>
<td>4...20 mA</td>
</tr>
<tr>
<td>5</td>
<td>-10 V...+10 V</td>
</tr>
<tr>
<td>8</td>
<td>2-wire Pt100 -50...+400 °C</td>
</tr>
<tr>
<td>9</td>
<td>3-wire Pt100 -50...+400 °C *)</td>
</tr>
<tr>
<td>10</td>
<td>0 V...10 V (voltage diff.) *)</td>
</tr>
<tr>
<td>11</td>
<td>-10 V...+10 V (voltage diff.) *)</td>
</tr>
<tr>
<td>14</td>
<td>2-wire Pt100 -50...+70 °C</td>
</tr>
<tr>
<td>15</td>
<td>3-wire Pt100 -50...+70 °C *)</td>
</tr>
</tbody>
</table>
Internal value | Operating modes of the analog inputs, individually configurable
---|---
16 | 2-wire Pt1000 -50...+400 °C
17 | 3-wire Pt1000 -50...+400 °C *)
18 | 2-wire Ni1000 -50...+150 °C
19 | 3-wire Ni1000 -50...+150 °C *)

*) In the operating modes with 3-wire configuration or with differential inputs, two adjacent analog inputs belong together (e.g. the channels 0 and 1). In these cases, both channels are configured in the desired operating mode. The lower address must be the even address (channel 0). The next higher address must be the odd address (channel 1). The converted analog value is available at the higher address (channel 1).

Table 610: Channel monitoring 2)

<table>
<thead>
<tr>
<th>Internal Value</th>
<th>Check channel</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Plausibility, cut wire, short circuit</td>
</tr>
<tr>
<td>3</td>
<td>not used</td>
</tr>
</tbody>
</table>

Channel parameters for the analog outputs (2x)

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>Internal value</th>
<th>Internal value, type</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output 0, channel configuration</td>
<td>see ³)</td>
<td>see ³)</td>
<td>BYTE</td>
<td>0</td>
</tr>
<tr>
<td>Output 0, check channel</td>
<td>see ⁴)</td>
<td>see ⁴)</td>
<td>BYTE</td>
<td>0</td>
</tr>
<tr>
<td>Output 0, substitute value</td>
<td>see ⁵)</td>
<td>see ⁵)</td>
<td>WORD</td>
<td>0</td>
</tr>
<tr>
<td>Output 1, channel configuration</td>
<td>see ³)</td>
<td>see ³)</td>
<td>BYTE</td>
<td>0</td>
</tr>
<tr>
<td>Output 1, check channel</td>
<td>see ⁴)</td>
<td>see ⁴)</td>
<td>BYTE</td>
<td>0</td>
</tr>
<tr>
<td>Output 1, substitute value</td>
<td>see ⁵)</td>
<td>see ⁵)</td>
<td>WORD</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 611: Channel configuration ³)

<table>
<thead>
<tr>
<th>Internal value</th>
<th>Operating modes of the analog outputs, individually configurable</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Not used (default)</td>
</tr>
<tr>
<td>128</td>
<td>-10 V...+10 V</td>
</tr>
<tr>
<td>129</td>
<td>0...20 mA</td>
</tr>
<tr>
<td>130</td>
<td>4...20 mA</td>
</tr>
</tbody>
</table>
Table 612: Channel monitoring 4)

<table>
<thead>
<tr>
<th>Internal value</th>
<th>Check channel</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Plausibility, cut wire, short circuit</td>
</tr>
<tr>
<td>3</td>
<td>None</td>
</tr>
</tbody>
</table>

Table 613: Substitute value 5)

<table>
<thead>
<tr>
<th>Intended behaviour of output channel when the control system stops</th>
<th>Required setting of the module parameter &quot;Behaviour of outputs in case of a communication error&quot;</th>
<th>Required setting of the channel parameter &quot;Substitute value&quot;</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output OFF</td>
<td>Off</td>
<td>0</td>
</tr>
<tr>
<td>Last value infinite</td>
<td>Last value</td>
<td>0</td>
</tr>
<tr>
<td>Last value for 5 s</td>
<td>Last value 5 s</td>
<td>0</td>
</tr>
<tr>
<td>Last value for 10 s</td>
<td>Last value 10 s</td>
<td>0</td>
</tr>
<tr>
<td>Substitute value infinite</td>
<td>Substitute value</td>
<td>Depending on configuration</td>
</tr>
<tr>
<td>Substitute value for 5 s</td>
<td>Substitute value 5 s</td>
<td>Depending on configuration</td>
</tr>
<tr>
<td>Substitute value for 10 s</td>
<td>Substitute value 10 s</td>
<td>Depending on configuration</td>
</tr>
</tbody>
</table>

Group parameters for the digital part

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>Internal value</th>
<th>Internal value, type</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input delay</td>
<td>0.01 ms</td>
<td>0</td>
<td>BYTE</td>
<td>0.01 ms 0x00</td>
</tr>
<tr>
<td></td>
<td>1 ms</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>8 ms</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>32 ms</td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Detect short circuits at outputs</td>
<td>Off</td>
<td>0</td>
<td>BYTE</td>
<td>On 0x01</td>
</tr>
<tr>
<td></td>
<td>On</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Behaviour DO at comm. error *)</td>
<td>Off</td>
<td>0</td>
<td>BYTE</td>
<td>Off 0x00</td>
</tr>
<tr>
<td></td>
<td>Last value</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Last value 5 sec</td>
<td>6</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Last value 10 sec</td>
<td>11</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Substitute value</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Substitute 5 sec</td>
<td>7</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Substitute 10 sec</td>
<td>12</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Substitute value at output</td>
<td>0 ... 255</td>
<td>00h ... FFh</td>
<td>BYTE</td>
<td>0 0x0000</td>
</tr>
</tbody>
</table>

*) The parameter behaviourDOatCommunicationFault is only analyzed if the Failsafe-mode is ON.
## Diagnosis

In cases of short circuit or overload, the digital outputs are turned off. The modules perform reactivation automatically. Thus an acknowledgement of the errors is not necessary. The error message is stored via the LED.

<table>
<thead>
<tr>
<th>E1..E4</th>
<th>d1</th>
<th>d2</th>
<th>d3</th>
<th>d4</th>
<th>Identifier 000..063</th>
<th>AC500-Display</th>
<th>&lt;- Display in</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class</td>
<td>Comp</td>
<td>Dev</td>
<td>Mod</td>
<td>Ch</td>
<td>Err</td>
<td>PS501 PLC Browser</td>
<td></td>
</tr>
<tr>
<td>Byte 6</td>
<td>Bit 6..7</td>
<td>Byte 3</td>
<td>Byte 4</td>
<td>Byte 5</td>
<td>Byte 6 Bit 0..5</td>
<td>ETHCAT Diagnosis block</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Class</th>
<th>Interface</th>
<th>Device</th>
<th>Module</th>
<th>Channel</th>
<th>Error identifier</th>
<th>Error message</th>
<th>Remedy</th>
</tr>
</thead>
<tbody>
<tr>
<td>1)</td>
<td>2)</td>
<td>3)</td>
<td>4)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Module error

<p>| 3 | - | 31 | 31 | 31 | 19 | Checksum error in the I/O module | Replace I/O module |
| 3 | - | 31 | 31 | 31 | 3  | Timeout in the I/O module       |                   |
| 3 | - | 31 | 31 | 31 | 40 | Different hard-/firmware versions in the module |                   |
| 3 | - | 31 | 31 | 31 | 43 | Internal error in the module     |                   |
| 3 | - | 31 | 31 | 31 | 36 | Internal data exchange failure   |                   |
| 3 | - | 31 | 31 | 31 | 20 | Slave-to-Slave malfunction       | Check configuration |
| 3 | - | 31 | 31 | 31 | 41 | Distributed Clock malfunction    | Check configuration |
| 3 | - | 31 | 31 | 31 | 9  | Overflow diagnosis buffer        | Restart            |
| 3 | - | 31 | 31 | 31 | 26 | Parameter error                  | Check master       |
| 3 | - | 31 | 31 | 31 | 11 | Process voltage UP too low       | Check process supply voltage UP |
| 4 | - | 31 | 31 | 31 | 45 | Process voltage UP3 too low      | Check process voltage |
| 4 | - | 31 | 31 | 31 | 34 | No response during initialization of the I/O module | Replace I/O module |</p>
<table>
<thead>
<tr>
<th>E1..E4</th>
<th>d1</th>
<th>d2</th>
<th>d3</th>
<th>d4</th>
<th>Identifier</th>
<th>AC500-Display</th>
<th>&lt;= Display in</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>000..063</td>
<td>PS501 PLC Browser</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Class</th>
<th>Inter-</th>
<th>Device</th>
<th>Module</th>
<th>Channel</th>
<th>Error</th>
<th>Error message</th>
<th>Remedy</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>face</td>
<td></td>
<td></td>
<td></td>
<td>identifier</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>3)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>4)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Class</th>
<th>Interface</th>
<th>Device</th>
<th>Module</th>
<th>Channel</th>
<th>Error identifier</th>
<th>Error message</th>
<th>Remedy</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>-</td>
<td>31</td>
<td>31</td>
<td>31</td>
<td>46</td>
<td>Voltage feedback on activated digital outputs 4)</td>
<td>Check terminals</td>
</tr>
</tbody>
</table>

**Channel error digital**

<table>
<thead>
<tr>
<th>Class</th>
<th>Interface</th>
<th>Device</th>
<th>Module</th>
<th>Channel</th>
<th>Error identifier</th>
<th>Error message</th>
<th>Remedy</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>-</td>
<td>31</td>
<td>2</td>
<td>0..7</td>
<td>46</td>
<td>Voltage feedback on deactivated digital output 5)</td>
<td>Check terminals</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Class</th>
<th>Interface</th>
<th>Device</th>
<th>Module</th>
<th>Channel</th>
<th>Error identifier</th>
<th>Error message</th>
<th>Remedy</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>-</td>
<td>31</td>
<td>2</td>
<td>0..7</td>
<td>47</td>
<td>Short circuit at digital output</td>
<td>Check terminals</td>
</tr>
</tbody>
</table>

**Channel error analog**

<table>
<thead>
<tr>
<th>Class</th>
<th>Interface</th>
<th>Device</th>
<th>Module</th>
<th>Channel</th>
<th>Error identifier</th>
<th>Error message</th>
<th>Remedy</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>-</td>
<td>31</td>
<td>1</td>
<td>0..3</td>
<td>48</td>
<td>Analog value overflow or broken wire at an analog input</td>
<td>Check value or check terminals</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Class</th>
<th>Interface</th>
<th>Device</th>
<th>Module</th>
<th>Channel</th>
<th>Error identifier</th>
<th>Error message</th>
<th>Remedy</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>-</td>
<td>31</td>
<td>1</td>
<td>0..3</td>
<td>7</td>
<td>Analog value underflow at an analog input</td>
<td>Check value</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Class</th>
<th>Interface</th>
<th>Device</th>
<th>Module</th>
<th>Channel</th>
<th>Error identifier</th>
<th>Error message</th>
<th>Remedy</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>-</td>
<td>31</td>
<td>1</td>
<td>0..3</td>
<td>47</td>
<td>Short circuit at an analog input</td>
<td>Check terminals</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Class</th>
<th>Interface</th>
<th>Device</th>
<th>Module</th>
<th>Channel</th>
<th>Error identifier</th>
<th>Error message</th>
<th>Remedy</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>-</td>
<td>31</td>
<td>3</td>
<td>0..1</td>
<td>48</td>
<td>Analog value overflow at an analog output</td>
<td>Check output value</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Class</th>
<th>Interface</th>
<th>Device</th>
<th>Module</th>
<th>Channel</th>
<th>Error identifier</th>
<th>Error message</th>
<th>Remedy</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>-</td>
<td>31</td>
<td>3</td>
<td>0..1</td>
<td>7</td>
<td>Analog value underflow at an analog output</td>
<td>Check output value</td>
</tr>
</tbody>
</table>

**Remarks:**

1) In AC500 the following interface identifier applies:

"-" = Diagnosis via bus-specific function blocks; 0 ... 4 or 10 = Position of the Communication Module; 14 = I/O bus; 31 = Module itself

The identifier is not contained in the CI511-ETHCAT diagnosis block.

2) With "Device" the following allocation applies:

31 = Module itself or ADR = Hardware address (e. g. of the DC551)
With "Module" the following allocation applies dependent of the master:

31 = Module itself (Module error) or Module type (1=AI, 2=DO, 3=AO; channel error)

Diagnosis message appears for the whole output group and not per channel. The message occurs if the output channel is already active.

Diagnosis message appears per channel. The message occurs if the output channel is not active.

**State LEDs**

The LEDs are located at the front of module. There are 2 different groups:
- The 5 system LEDs (PWR, NET, DC, S-ERR and I/O-Bus) show the operation state of the module and display possible errors.
- The 27 process LEDs (UP, UP3, inputs, outputs, CH-ERR1 to CH-ERR3) show the process supply voltage and the states of the inputs and outputs and display possible errors.

<table>
<thead>
<tr>
<th>LED</th>
<th>Color</th>
<th>Off</th>
<th>On</th>
<th>Flashing</th>
<th>1x Flash</th>
<th>2x Flash</th>
</tr>
</thead>
<tbody>
<tr>
<td>PWR/RUN</td>
<td>Green</td>
<td>Error in the internal supply voltage or process voltage missing</td>
<td>Internal supply voltage OK</td>
<td>Module is not configured</td>
<td>--</td>
<td>--</td>
</tr>
<tr>
<td></td>
<td>Yellow</td>
<td>--</td>
<td>--</td>
<td>--</td>
<td>--</td>
<td>--</td>
</tr>
<tr>
<td>NET</td>
<td>Green</td>
<td>Init</td>
<td>Operational</td>
<td>Pre-operational</td>
<td>Safe-operational</td>
<td>--</td>
</tr>
<tr>
<td></td>
<td>Red</td>
<td>No error</td>
<td>PDI Watchdog Timeout</td>
<td>Invalid Configuration</td>
<td>Unsolicited State Change</td>
<td>Application time out</td>
</tr>
<tr>
<td>DC *)</td>
<td>Green</td>
<td>Distributed Clock not active</td>
<td>Distributed Clock active</td>
<td>--</td>
<td>--</td>
<td>--</td>
</tr>
<tr>
<td></td>
<td>Red</td>
<td>--</td>
<td>--</td>
<td>--</td>
<td>--</td>
<td>--</td>
</tr>
<tr>
<td>S-ERR</td>
<td>Red</td>
<td>No error</td>
<td>Internal error</td>
<td>--</td>
<td>--</td>
<td>--</td>
</tr>
<tr>
<td>I/O-Bus</td>
<td>Green</td>
<td>No expansion modules connected or communica-</td>
<td>---</td>
<td>---</td>
<td>--</td>
<td>--</td>
</tr>
<tr>
<td></td>
<td>ETH1</td>
<td>No EtherCAT connection</td>
<td>Link OK No data transfer</td>
<td>Link OK Data transfer OK</td>
<td>--</td>
<td>--</td>
</tr>
<tr>
<td></td>
<td>Yellow</td>
<td>--</td>
<td>--</td>
<td>--</td>
<td>--</td>
<td>--</td>
</tr>
</tbody>
</table>
### Table 615: States of the 27 process LEDs

<table>
<thead>
<tr>
<th>LED</th>
<th>Color</th>
<th>OFF</th>
<th>ON</th>
<th>Flashing</th>
<th>1x Flash</th>
<th>2x Flash</th>
</tr>
</thead>
<tbody>
<tr>
<td>ETH2</td>
<td>Green</td>
<td>No EtherCAT connection</td>
<td>Link OK</td>
<td>Link OK</td>
<td>--</td>
<td>--</td>
</tr>
<tr>
<td></td>
<td>Yellow</td>
<td></td>
<td>No data transfer</td>
<td>Data transfer OK</td>
<td>--</td>
<td>--</td>
</tr>
</tbody>
</table>

*) The state of this LED is only significant if the camswitch functionality is enabled

<table>
<thead>
<tr>
<th>LED</th>
<th>Color</th>
<th>OFF</th>
<th>ON</th>
<th>Flashing</th>
<th>1x Flash</th>
<th>2x Flash</th>
</tr>
</thead>
<tbody>
<tr>
<td>AI0 to AI3</td>
<td>Yellow</td>
<td>Input is OFF</td>
<td>Input is ON (brightness</td>
<td>--</td>
<td>--</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>depends on the value of the</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>analog signal)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AO0 to AO1</td>
<td>Yellow</td>
<td>Output is OFF</td>
<td>Output is ON (brightness</td>
<td>--</td>
<td>--</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>depends on the value of the</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>analog signal)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DO0 to DO7</td>
<td>Yellow</td>
<td>Output is OFF</td>
<td>Output is ON</td>
<td>--</td>
<td>--</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(the input voltage is even</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>displayed if the supply</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>voltage is OFF)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>UP</td>
<td>Green</td>
<td>Process supply voltage</td>
<td>Process supply voltage</td>
<td>--</td>
<td>--</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>missing</td>
<td>OK and initialization</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>finished</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>UP3</td>
<td>Green</td>
<td>Process supply voltage</td>
<td>Process supply voltage</td>
<td>--</td>
<td>--</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>missing</td>
<td>OK</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CH-ERR1 to CH-ERR3</td>
<td>Red</td>
<td>No error or process supply voltage missing</td>
<td>Internal error</td>
<td>Error on one channel of the corresponding group</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Measuring ranges**

**Input ranges voltage, current and digital input**

<table>
<thead>
<tr>
<th>Range</th>
<th>0...10 V</th>
<th>-10...+10 V</th>
<th>0...20 mA</th>
<th>4...20 mA</th>
<th>Digital input</th>
<th>Digital value</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Overflow</strong></td>
<td>&gt;11.7589</td>
<td>&gt;11.7589</td>
<td>&gt;23.5178</td>
<td>&gt;22.8142</td>
<td></td>
<td>32767</td>
</tr>
</tbody>
</table>
### Input ranges resistance temperature detector

<table>
<thead>
<tr>
<th>Range</th>
<th>Pt100 / Pt1000 -50...400 °C</th>
<th>Ni1000 -50...150 °C</th>
<th>Digital value</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Overload</strong></td>
<td>&gt; 450.0 °C</td>
<td>&gt; 160.0 °C</td>
<td>32767</td>
</tr>
<tr>
<td><strong>Measured value too high</strong></td>
<td>450.0 °C</td>
<td>4500</td>
<td>1194</td>
</tr>
<tr>
<td></td>
<td>400.1 °C</td>
<td>4001</td>
<td>0FA1</td>
</tr>
<tr>
<td></td>
<td>160.0 °C</td>
<td>1600</td>
<td>0640</td>
</tr>
<tr>
<td></td>
<td>150.1 °C</td>
<td>1501</td>
<td>05DD</td>
</tr>
<tr>
<td></td>
<td>800</td>
<td>701</td>
<td>0320</td>
</tr>
<tr>
<td><strong>Normal range</strong></td>
<td>400.0 °C</td>
<td>4000</td>
<td>0FA0</td>
</tr>
<tr>
<td></td>
<td>150.0 °C</td>
<td>1500</td>
<td>05DC</td>
</tr>
<tr>
<td></td>
<td>0.1 °C</td>
<td>700</td>
<td>02BC</td>
</tr>
<tr>
<td></td>
<td>0.0 °C</td>
<td>1</td>
<td>0001</td>
</tr>
<tr>
<td></td>
<td>-0.1 °C</td>
<td>-1</td>
<td>FFFF</td>
</tr>
<tr>
<td></td>
<td>-50.0 °C</td>
<td>-500</td>
<td>FE0C</td>
</tr>
</tbody>
</table>

The represented resolution corresponds to 16 bits.
### Range

<table>
<thead>
<tr>
<th>Range</th>
<th>Pt100 / Pt1000 -50...400 °C</th>
<th>Ni1000 -50...150 °C</th>
<th>Digital value</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Decimal</td>
<td>Hex.</td>
<td></td>
</tr>
<tr>
<td>Measured value too low</td>
<td>-50.1 °C</td>
<td>-50.1 °C</td>
<td>-501 FE0B</td>
</tr>
<tr>
<td></td>
<td>-60.0 °C</td>
<td>-60.0 °C</td>
<td>-600 FDA8</td>
</tr>
<tr>
<td>Underflow</td>
<td>&lt; -60.0 °C</td>
<td>&lt; -60.0 °C</td>
<td>-32768 8000</td>
</tr>
</tbody>
</table>

### Output ranges voltage and current

<table>
<thead>
<tr>
<th>Range</th>
<th>-10...+10 V</th>
<th>0...20 mA</th>
<th>4...20 mA</th>
<th>Digital value</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Decimal</td>
<td>Hex.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Overflow</td>
<td>0 V</td>
<td>0 mA</td>
<td>0 mA</td>
<td>&gt; 32511</td>
</tr>
<tr>
<td>Measured value too high</td>
<td>11.7589 V</td>
<td>23.5178 mA</td>
<td>22.8142 mA</td>
<td>32511 7EFF</td>
</tr>
<tr>
<td></td>
<td>10.0004 V</td>
<td>20.0007 mA</td>
<td>20.0006 mA</td>
<td>27649 6C01</td>
</tr>
<tr>
<td>Normal range</td>
<td>10.0000 V</td>
<td>20.0000 mA</td>
<td>20.0000 mA</td>
<td>27648 6C00</td>
</tr>
<tr>
<td></td>
<td>0.0004 V</td>
<td>0.0007 mA</td>
<td>4.0006 mA</td>
<td>1 0001</td>
</tr>
<tr>
<td></td>
<td>0.0000 V</td>
<td>0.0000 mA</td>
<td>4.0000 mA</td>
<td>0 0000</td>
</tr>
<tr>
<td></td>
<td>-0.0004 V</td>
<td>0 mA</td>
<td>3.9994 mA</td>
<td>-1 FFFF</td>
</tr>
<tr>
<td></td>
<td>-10.0000 V</td>
<td>0 mA</td>
<td>0 mA</td>
<td>-6912 E500</td>
</tr>
<tr>
<td>Measured value too low</td>
<td>-10.0000 V</td>
<td>0 mA</td>
<td>0 mA</td>
<td>-27648 9400</td>
</tr>
<tr>
<td></td>
<td>-11.7589 V</td>
<td>0 mA</td>
<td>0 mA</td>
<td>-32512 8100</td>
</tr>
<tr>
<td>Underflow</td>
<td>0 V</td>
<td>0 mA</td>
<td>0 mA</td>
<td>&lt; -32512</td>
</tr>
</tbody>
</table>

The represented resolution corresponds to 16 bits.

### Technical data

The system data of AC500 and S500 Chapter 1.5.4.5.1 “System data AC500” on page 3058 are valid for standard version.

The system data of AC500-XC Chapter 1.5.4.6.1 “System data AC500-XC” on page 3099 are valid for the XC version.

Only additional details are therefore documented below.

The technical data are also valid for the XC version.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bus connection</td>
<td>2 x RJ45</td>
</tr>
<tr>
<td>Technology</td>
<td>Hilscher netX100</td>
</tr>
<tr>
<td>Transfer rate</td>
<td>10/100 Mbit/s (full-duplex)</td>
</tr>
<tr>
<td>Parameter</td>
<td>Value</td>
</tr>
<tr>
<td>----------------------------------------------------</td>
<td>----------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Transfer method</td>
<td>According to Ethernet II, IEE802.3</td>
</tr>
<tr>
<td>Ethernet</td>
<td>100 base-TX, internal switch, 2x RJ45 socket</td>
</tr>
<tr>
<td>Expandability (S500 I/O modules)</td>
<td>Up to 10 S500 I/O modules (Index C0 and above), not available (Index below C0)</td>
</tr>
<tr>
<td>Indicators</td>
<td>5 LEDs for state indication</td>
</tr>
<tr>
<td>Adjusting elements</td>
<td>2 rotary switches (used for future topology extensions)</td>
</tr>
<tr>
<td>Quantity of input/output data</td>
<td>CI512-ETHCAT: 10 bytes input and 14 bytes output CI511-ETHCAT: 18 bytes input and 18 bytes output</td>
</tr>
<tr>
<td>Limit of data for input and output</td>
<td>144 byte</td>
</tr>
<tr>
<td>Acyclic services</td>
<td>SDO (1500 bytes max.) Emergency ECAT_SLV_DIAG</td>
</tr>
<tr>
<td>Protective functions (according to CODESYS)</td>
<td>Protected against:</td>
</tr>
<tr>
<td></td>
<td>● short circuit</td>
</tr>
<tr>
<td></td>
<td>● reverse supply</td>
</tr>
<tr>
<td></td>
<td>● overvoltage</td>
</tr>
<tr>
<td></td>
<td>● reverse polarity</td>
</tr>
<tr>
<td></td>
<td>Electrical isolation to network</td>
</tr>
</tbody>
</table>

**Technical data of the module**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Process supply voltage UP/UP3</td>
<td></td>
</tr>
<tr>
<td>Rated value</td>
<td>24 VDC (for inputs and outputs)</td>
</tr>
<tr>
<td>Max. load for the terminals</td>
<td>10 A</td>
</tr>
<tr>
<td>Protection against reversed voltage</td>
<td>Yes</td>
</tr>
<tr>
<td>Rated protection fuse on UP/UP3</td>
<td>10 A fast</td>
</tr>
<tr>
<td>Electrical isolation</td>
<td>Ethernet interface against the rest of the module</td>
</tr>
<tr>
<td>Inrush current from UP (at power up)</td>
<td>On request</td>
</tr>
<tr>
<td>Current consumption via UP (normal operation)</td>
<td>0.2 A</td>
</tr>
<tr>
<td>Current consumption via UP3</td>
<td>0.06 A + 0.5 A max. per output</td>
</tr>
<tr>
<td>Connections</td>
<td>Terminals 1.8 and 2.8 for +24 V (UP) Terminals 3.8 for +24 V (UP3) Terminals 1.9, 2.9 and 3.9 for 0 V (ZP)</td>
</tr>
<tr>
<td>Max. power dissipation within the module</td>
<td>6 W</td>
</tr>
<tr>
<td>Number of digital inputs</td>
<td>8</td>
</tr>
<tr>
<td>Number of digital outputs</td>
<td>8</td>
</tr>
<tr>
<td>Number of analog inputs</td>
<td>4</td>
</tr>
<tr>
<td>Number of analog outputs</td>
<td>2</td>
</tr>
</tbody>
</table>
### Technical data of the digital inputs

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of channels per module</td>
<td>8</td>
</tr>
<tr>
<td>Distribution of the channels into groups</td>
<td>1 group of 8 channels</td>
</tr>
<tr>
<td>Terminals of the channels DI0 to DI7</td>
<td>Terminals 2.0 to 2.7</td>
</tr>
<tr>
<td>Reference potential for all inputs</td>
<td>Terminals 1.9...3.9 (Minus pole of the supply voltage, signal name ZP)</td>
</tr>
<tr>
<td>Indication of the input signals</td>
<td>1 yellow LED per channel, the LED is ON when the input signal is high (signal 1)</td>
</tr>
<tr>
<td>Input type (according EN 61131-2)</td>
<td>Type 1</td>
</tr>
<tr>
<td>Input delay (0-&gt;1 or 1-&gt;0)</td>
<td>Typ. 0.1 ms, configurable from 0.1...32 ms</td>
</tr>
<tr>
<td>Input signal voltage</td>
<td>24 VDC</td>
</tr>
<tr>
<td>0-Signal</td>
<td>-3 V...+5 V</td>
</tr>
<tr>
<td>Undefined Signal</td>
<td>&gt; +5 V...&lt; +15 V</td>
</tr>
<tr>
<td>1-Signal</td>
<td>+15 V...+30 V</td>
</tr>
<tr>
<td>Ripple with signal 0</td>
<td>Within -3 V...+5 V</td>
</tr>
<tr>
<td>Ripple with signal 1</td>
<td>Within +15 V...+30 V</td>
</tr>
<tr>
<td>Input current per channel</td>
<td></td>
</tr>
<tr>
<td>Input voltage</td>
<td>+24 V</td>
</tr>
<tr>
<td>Typ. 5 mA</td>
<td></td>
</tr>
</tbody>
</table>
### Technical data of the digital inputs

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input voltage +5 V</td>
<td>&gt; 1 mA</td>
</tr>
<tr>
<td>Input voltage +15 V</td>
<td>&gt; 2 mA</td>
</tr>
<tr>
<td>Input voltage +30 V</td>
<td>&lt; 8 mA</td>
</tr>
<tr>
<td>Max. cable length</td>
<td></td>
</tr>
<tr>
<td>Shielded</td>
<td>1000 m</td>
</tr>
<tr>
<td>Unshielded</td>
<td>600 m</td>
</tr>
</tbody>
</table>

### Technical data of the digital outputs

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of channels per module</td>
<td>8</td>
</tr>
<tr>
<td>Distribution of the channels into groups</td>
<td>1 group of 8 channels</td>
</tr>
<tr>
<td>Terminals of the channels DO0 to DO7</td>
<td>Terminals 3.0 to 3.7</td>
</tr>
<tr>
<td>Reference potential for all outputs</td>
<td>Terminals 1.9...3.9 (minus pole of the supply voltage, signal name ZP)</td>
</tr>
<tr>
<td>Common power supply voltage</td>
<td>For all outputs terminal 3.8 (plus pole of the supply voltage, signal name UP3)</td>
</tr>
<tr>
<td>Output voltage for signal 1</td>
<td>UP3 (-0.8 V)</td>
</tr>
<tr>
<td>Output delay (0-&gt;1 or 1-&gt;0)</td>
<td>On request</td>
</tr>
<tr>
<td>Output current</td>
<td></td>
</tr>
<tr>
<td>Rated value per channel</td>
<td>500 mA at UP3 = 24 V</td>
</tr>
<tr>
<td>Max. value (all channels together)</td>
<td>4 A</td>
</tr>
<tr>
<td>Leakage current with signal 0</td>
<td>&lt; 0.5 mA</td>
</tr>
<tr>
<td>Fuse for UP3</td>
<td>10 A fast</td>
</tr>
<tr>
<td>Demagnetization with inductive DC load</td>
<td>Via internal varistors (see figure below this table)</td>
</tr>
<tr>
<td>Output switching frequency</td>
<td></td>
</tr>
<tr>
<td>With resistive load</td>
<td>On request</td>
</tr>
<tr>
<td>With inductive loads</td>
<td>Max. 0.5 Hz</td>
</tr>
<tr>
<td>With lamp loads</td>
<td>11 Hz max. at 5 W max.</td>
</tr>
<tr>
<td>Short-circuit-proof / overload-proof</td>
<td>Yes</td>
</tr>
<tr>
<td>Overload message (I &gt; 0.7 A)</td>
<td>Yes, after ca. 100 ms</td>
</tr>
<tr>
<td>Output current limitation</td>
<td>Yes, automatic reactivation after short circuit/overload</td>
</tr>
<tr>
<td>Resistance to feedback against 24 V signals</td>
<td>Yes (software-controlled supervision)</td>
</tr>
<tr>
<td>Max. cable length</td>
<td></td>
</tr>
<tr>
<td>Shielded</td>
<td>1000 m</td>
</tr>
<tr>
<td>Unshielded</td>
<td>600 m</td>
</tr>
</tbody>
</table>

The following drawing shows the circuitry of a digital input/output with the varistors for demagnetization when inductive loads are switched off.
**Fig. 224: Digital input/output (circuit diagram)**

1. Digital output
2. Varistors for demagnetization when inductive loads are turned off

### Technical data of the analog inputs

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of channels per module</td>
<td>4</td>
</tr>
<tr>
<td>Distribution of channels into groups</td>
<td>1 group with 4 channels</td>
</tr>
<tr>
<td>Connection if channels AI0+ to AI3+</td>
<td>Terminals 1.0 to 1.3</td>
</tr>
<tr>
<td>Reference potential for AI0+ to AI3+</td>
<td>Terminal 1.4 (AI-) for voltage and RTD measurement</td>
</tr>
<tr>
<td></td>
<td>Terminals 1.9, 2.9 and 3.9 for current measurement</td>
</tr>
<tr>
<td>Input type</td>
<td></td>
</tr>
<tr>
<td>Unipolar</td>
<td>Voltage 0 V...10 V, current or Pt100/Pt1000/ Ni1000</td>
</tr>
<tr>
<td>Bipolar</td>
<td>Voltage -10 V...+10 V</td>
</tr>
<tr>
<td>Electrical isolation</td>
<td>Against Ethernet network</td>
</tr>
<tr>
<td>Configurability</td>
<td>0 V...10 V, -10 V...+10 V, 0/4 mA...20 mA, Pt100/1000, Ni1000 (each input can be configured individually)</td>
</tr>
<tr>
<td>Channel input resistance</td>
<td>Voltage: &gt; 100 kΩ</td>
</tr>
<tr>
<td></td>
<td>Current: ca. 330 Ω</td>
</tr>
<tr>
<td>Time constant of the input filter</td>
<td>Voltage: 100 μs</td>
</tr>
<tr>
<td></td>
<td>Current: 100 μs</td>
</tr>
<tr>
<td>Indication of the input signals</td>
<td>1 LED per channel (brightness depends on the value of the analog signal)</td>
</tr>
<tr>
<td>Conversion cycle</td>
<td>1 ms (for 4 inputs + 2 outputs); with RTDs Pt/ Ni... 1 s</td>
</tr>
<tr>
<td>Resolution</td>
<td>Range 0...10 V: 12 bits</td>
</tr>
<tr>
<td></td>
<td>Range -10...+10 V: 12 bits + sign</td>
</tr>
<tr>
<td></td>
<td>Range 0...20 mA: 12 bits</td>
</tr>
<tr>
<td></td>
<td>Range 4...20 mA: 12 bits</td>
</tr>
<tr>
<td></td>
<td>Range RTD (Pt100, PT1000, Ni1000): 0.1 °C</td>
</tr>
<tr>
<td>Conversion error of the analog values caused by non-linearity, adjustment error at factory and resolution within the normal range</td>
<td>Typ. 0.5 %, max. 1 %</td>
</tr>
</tbody>
</table>
### Technical data of the analog inputs, if used as digital inputs

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of channels per module</td>
<td>Max. 4</td>
</tr>
<tr>
<td>Distribution of channels into groups</td>
<td>1 group of 4 channels</td>
</tr>
<tr>
<td>Connections of the channels AI0+ to AI3+</td>
<td>Terminals 1.0 to 1.3</td>
</tr>
<tr>
<td>Reference potential for the inputs</td>
<td>Terminals 1.9, 2.9 and 3.9 (ZP)</td>
</tr>
<tr>
<td>Indication of the input signals</td>
<td>1 LED per channel</td>
</tr>
<tr>
<td>Input signal voltage</td>
<td>24 VDC</td>
</tr>
<tr>
<td>Signal 0</td>
<td>-30 V...+5 V</td>
</tr>
<tr>
<td>Undefined signal</td>
<td>+5 V ... +13 V</td>
</tr>
<tr>
<td>Signal 1</td>
<td>+13 V...+30 V</td>
</tr>
<tr>
<td>Input current per channel</td>
<td></td>
</tr>
<tr>
<td>Input voltage +24 V</td>
<td>Typ. 7 mA</td>
</tr>
<tr>
<td>Input voltage +5 V</td>
<td>Typ. 1.4 mA</td>
</tr>
<tr>
<td>Input voltage +15 V</td>
<td>Typ. 3.7 mA</td>
</tr>
<tr>
<td>Input voltage +30 V</td>
<td>&lt; 9 mA</td>
</tr>
<tr>
<td>Input resistance</td>
<td>Ca. 3.5 kΩ</td>
</tr>
</tbody>
</table>

### Technical data of the analog outputs

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of channels per module</td>
<td>2</td>
</tr>
<tr>
<td>Distribution of channels into groups</td>
<td>1 group for 2 channels</td>
</tr>
<tr>
<td>Connection of the channels AO0+...AO1+</td>
<td>Terminals 1.5...1.6</td>
</tr>
<tr>
<td>Reference potential for AO0+ to AO1+</td>
<td>Terminal 1.7 (AO-) for voltage outputTerminals 1.9, 2.9 and 3.9 (ZP) for current output</td>
</tr>
<tr>
<td>Output type</td>
<td>Unipolar Current</td>
</tr>
<tr>
<td></td>
<td>Bipolar Voltage</td>
</tr>
<tr>
<td>Electrical isolation</td>
<td>Against Ethernet network</td>
</tr>
<tr>
<td>Configurability</td>
<td>-10 V...+10 V, 0 mA...20 mA, 4 mA...20 mA (each output can be configured individually)</td>
</tr>
<tr>
<td>Parameter</td>
<td>Value</td>
</tr>
<tr>
<td>------------------------------------------------</td>
<td>----------------------------------------------------------------------</td>
</tr>
<tr>
<td>Output resistance (load), as current output</td>
<td>0 ... 500 Ω</td>
</tr>
<tr>
<td>Output loadability, as voltage output</td>
<td>± 10 mA max.</td>
</tr>
<tr>
<td>Indication of the output signals</td>
<td>1 LED per channel (brightness depends on the value of the analog signal)</td>
</tr>
<tr>
<td>Resolution</td>
<td>12 bits (+ sign)</td>
</tr>
<tr>
<td>Conversion error of the analog values caused by non-linearity, adjustment error at factory and resolution within the normal range</td>
<td>Typ. 0.5 %, max. 1 %</td>
</tr>
<tr>
<td>Relationship between input signal and hex code</td>
<td>Table Output Ranges Voltage and Current ☰ Chapter 1.5.3.7.3.1.10.3 “Output ranges voltage and current” on page 2874</td>
</tr>
<tr>
<td>Unused outputs</td>
<td>Are configured as unused (default value) and can be left open-circuited</td>
</tr>
</tbody>
</table>

### Ordering data

<table>
<thead>
<tr>
<th>Part no.</th>
<th>Description</th>
<th>Product life cycle phase *)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1SAP 220 900 R0001</td>
<td>CI511-ETHCAT, EtherCAT bus module, 8 DI, 8 DO, 4 AI and 2 AO</td>
<td>Active</td>
</tr>
</tbody>
</table>

*) For planning and commissioning of new installations use modules in Active status only.

### CI512-ETHCAT

- 8 digital inputs 24 VDC
- 8 digital outputs 24 VDC, 0.5 A max.
- 8 configurable digital inputs/outputs 24 VDC, 0.5 A max.
- Cam switch functionality (see also Extended Cam Switch Library)
- Extended Cam switch functionality *)
  (see also Extended Cam Switch Library)
- Module-wise electrically isolated
- Expandability with up to 10 S500 I/O modules *)

*) Applicable for device index C0 and above.
I/O bus
Allocation between terminal number and signal name
8 yellow LEDs to display the signal states of the digital configurable inputs/outputs (DC0 - DC7)
8 yellow LEDs to display the signal states of the digital inputs (DI0 - DI7)
8 yellow LEDs to display the signal states of the digital outputs (DO0 - DO7)
2 green LEDs to display the supply voltage UP and UP3
3 red LEDs to display errors (CH-ERR1, CH-ERR2, CH-ERR3)
5 System LEDs: PWR/RUN, NET, DC, S-ERR, I/O-Bus
2 rotary switches (reserved for future extensions)
Label
Ethernet interfaces (ETH1, ETH2) on the terminal unit
Terminal unit
DIN rail

Intended purpose

The EtherCAT bus module CI512-ETHCAT is used as decentralized I/O module in EtherCAT networks. The network connection is performed via 2 RJ45 connectors which are integrated in the terminal unit. The bus module contains 24 I/O channels with the following properties:

- 8 digital configurable inputs/outputs in 1 group (1.0...1.7)
- 8 digital inputs 24 VDC in 1 group (2.0...2.7)
- 8 digital outputs 24 VDC in 1 group (3.0...3.7)
- Cam switch functionality
The inputs/outputs are electrically isolated from the Ethernet network. There is no potential separation between the channels. The configuration of the configurable digital inputs/outputs is performed by software.

**Functionality**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Interface</td>
<td>Ethernet</td>
</tr>
<tr>
<td>Protocol</td>
<td>EtherCAT</td>
</tr>
<tr>
<td>Power supply</td>
<td>From the process supply voltage UP</td>
</tr>
<tr>
<td>Supply of the electronic circuitry of the I/O expansion modules attached</td>
<td>Through the expansion bus interface (I/O bus)</td>
</tr>
<tr>
<td>Rotary switches</td>
<td>Not used; reserved for future extensions</td>
</tr>
<tr>
<td>Configurable digital inputs/outputs</td>
<td>8 (configurable via software)</td>
</tr>
<tr>
<td>Digital inputs</td>
<td>8 (24 VDC; delay time configurable via software)</td>
</tr>
<tr>
<td>Digital outputs</td>
<td>8 (24 VDC, 0.5 A max.)</td>
</tr>
<tr>
<td>LED displays</td>
<td>For system displays, signal states, errors and power supply</td>
</tr>
<tr>
<td>External supply voltage</td>
<td>Via terminals ZP, UP and UP3 (process supply voltage 24 VDC)</td>
</tr>
<tr>
<td>Effect of incorrect input terminal connection</td>
<td>Wrong or no signal detected, no damage up to 35 V</td>
</tr>
<tr>
<td>Required terminal unit</td>
<td>TU507 or TU508  ( \copyright ) Chapter 1.5.3.5.1 &quot;TU507-ETH and TU508-ETH for Ethernet communication interface modules&quot; on page 2307</td>
</tr>
</tbody>
</table>

**Electrical connection**

The Ethernet bus module CI512-ETHCAT is plugged on the I/O terminal unit TU507-ETH or TU508-ETH. Properly seat the module and press until it locks in place. The terminal unit is mounted on a DIN rail or with 2 screws plus the additional accessory for wall mounting (TA526).

The electrical connection of the I/O channels is carried out using the 30 terminals of the I/O terminal unit. I/O modules can be replaced without re-wiring the terminal units.

For a detailed description of the mounting, disassembly and electrical connection of the module, please refer to the System Assembly chapter.

The terminals 1.8 and 2.8 as well as 1.9, 2.9 and 3.9 are electrically interconnected within the terminal unit and have always the same assignment, independent of the inserted module:

- Terminals 1.8 and 2.8: Process supply voltage UP = +24 VDC
- Terminal 3.8: Process supply voltage UP3 = +24 VDC
- Terminals 1.9, 2.9 and 3.9: Process supply voltage ZP = 0 V

With a separate UP3 power supply, the digital outputs can be switched off externally. This way, an emergency-off functionality can be realized.
The assignment of the other terminals:

<table>
<thead>
<tr>
<th>Terminals</th>
<th>Signal</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0 to 1.7</td>
<td>DC0 to DC7</td>
<td>8 digital inputs/outputs (configurable via software)</td>
</tr>
<tr>
<td>2.0 to 2.7</td>
<td>DI0 to DI7</td>
<td>8 digital inputs (delay time configurable via software)</td>
</tr>
<tr>
<td>3.0 to 3.7</td>
<td>DO0 to DO7</td>
<td>8 digital outputs</td>
</tr>
</tbody>
</table>

**WARNING!**

Removal/Insertion under power
The devices are not designed for removal or insertion under power. Because of unforeseeable consequences, it is not allowed to plug or unplug devices with the power being ON.

Make sure that all voltage sources (supply and process voltage) are switched off before you
- connect or disconnect any signal or terminal block
- remove, mount or replace a module.

Disconnecting any powered devices while energized in a hazardous location could result in an electric arc, which could create a flammable ignition resulting in fire or explosion.

Make sure that power is removed and that the area has been thoroughly checked to ensure that flammable materials are not present prior to proceeding.

The devices must not be opened when in operation. The same applies to the network interfaces.

**NOTICE!**

Risk of damaging the PLC modules!
- Overvoltages and short circuits might damage the PLC modules.
  - Make sure that all voltage sources (supply voltage and process supply voltage) are switched off before you begin with operations at the system.
  - Never connect any voltages or signals to reserved terminals (marked with ---). Reserved terminals may carry internal voltages.

The following figures show the electrical connection of the Ethernet bus module CI512-ETHCAT.
In case of voltage feedback, 2 cases are distinguished:

1. The outputs are already active

   The output group will be switched off. A diagnosis message will appear. After 5 seconds, the module tries automatic reactivation.

2. The outputs are not active

   Only the output with voltage feedback will not be set to active. A diagnosis message will appear.

CAUTION!

The process supply voltage must be included within the earthing concept of the plant (e.g. earthing of the minus pole).

The module provides several diagnosis functions \(\text{\textcopyright Chapter 1.5.3.7.3.2.9 "Diagnosis" on page 2889.}\)

Assignment of the Ethernet ports

The terminal unit for the communication interface module provides two Ethernet interfaces with the following pin assignment. The pin assignment is used for the EtherCAT master (communication module CM5xy-ETHCAT) as well.
Table 616: Pin assignment RJ45 jack:

<table>
<thead>
<tr>
<th>Interface</th>
<th>Pin</th>
<th>Signal</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
<td>TxD+</td>
<td>Transmit data +</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>TxD-</td>
<td>Transmit data -</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>RxD+</td>
<td>Receive data +</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>NC</td>
<td>not used</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>NC</td>
<td>not used</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>RxD-</td>
<td>Receive data -</td>
</tr>
<tr>
<td></td>
<td>7</td>
<td>NC</td>
<td>not used</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>NC</td>
<td>not used</td>
</tr>
<tr>
<td>Shield</td>
<td></td>
<td></td>
<td>Cable shield</td>
</tr>
</tbody>
</table>

For further information regarding wiring and cable types see chapter Ethernet ♫ Chapter 1.5.4.4.7 “Ethernet connection details” on page 3084.

The EtherCAT network differentiates between input-connectors (IN) and output-connectors (OUT):

- At the EtherCAT slaves (communication interface modules), the ETH1-connector is IN and the ETH2-connector is OUT.
- At the EtherCAT master (communication module), the ETHCAT1 connector has to be used. The ETHCAT2 connector is reserved for future extensions.

Internal data exchange

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Digital inputs (bytes)</td>
<td>1</td>
</tr>
<tr>
<td>Digital outputs (bytes)</td>
<td>1</td>
</tr>
<tr>
<td>Configurable digital inputs/outputs (bytes)</td>
<td>1 + 1</td>
</tr>
</tbody>
</table>

Addressing

The Ethernet bus module CI512-ETHCAT does not consider the position of the rotary switches at the front side of the module. The function of the rotary switches is reserved for future expansions.
I/O configuration

In order to be able to use the CI51X-ETHCAT with device index C0 or above properly, please download the corresponding device description (.xml-)files from http://www.abb.com/plc and install them to the device repository of your Automation Builder. This will allow you to use up to 10 Expandable S500 I/O modules as well as the Extended Cam Switch Library with your CI51X-ETHCAT device.

The CI512-ETHCAT does not store configuration data itself. The analog I/O channels are configured via software.

Parameterization

Module parameter

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>Internal value</th>
<th>Internal value, type</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>Module ID</td>
<td>Internal</td>
<td>49435</td>
<td>WORD</td>
<td>49435</td>
</tr>
<tr>
<td>Parameter length</td>
<td>Internal</td>
<td>10</td>
<td>BYTE</td>
<td>10</td>
</tr>
<tr>
<td>Error LED / Failsafe function 1)</td>
<td>On, Off by E4, Off by E3 On + failsafe Off by E4 + failsafe Off by E3 + failsafe</td>
<td>0, 1, 3, 16, 17, 19</td>
<td>BYTE</td>
<td>0</td>
</tr>
<tr>
<td>Check Supply</td>
<td>Off</td>
<td>0</td>
<td>BYTE</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 617: Error LED / Failsafe function 1)

<table>
<thead>
<tr>
<th>Setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>On</td>
<td>Error LED lights up at errors of all error classes, Failsafe mode off</td>
</tr>
<tr>
<td>Off by E4</td>
<td>Error LED lights up at errors of error classes E1, E2 and E3, Failsafe mode off</td>
</tr>
<tr>
<td>Off by E3</td>
<td>Error LED lights up at errors of error classes E1 and E2 auf, Failsafe mode off</td>
</tr>
<tr>
<td>Off + failsafe</td>
<td>Error LED lights up at errors of all error classes, Failsafe mode on *)</td>
</tr>
<tr>
<td>Off by E4 + failsafe</td>
<td>Error LED lights up at errors of error classes E1, E2 and E3, Failsafe mode on *)</td>
</tr>
<tr>
<td>Off by E3 + failsafe</td>
<td>Error LED lights up at errors of error classes E1 and E2, Failsafe mode on *)</td>
</tr>
</tbody>
</table>

*) The parameter behaviourDOatCommunicationFault is only analyzed if the Failsafe-mode is ON.
Group parameters of the cam switch

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>Internal value</th>
<th>Internal value, type</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>numOfUsed-Cams 1)</td>
<td>0 ... 32</td>
<td>0 ... 32</td>
<td>0 ... 32, 218...160</td>
<td>WORD</td>
</tr>
<tr>
<td></td>
<td>128...160</td>
<td></td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>resolution 2)</td>
<td>0 ... 2</td>
<td>0 ... 2</td>
<td>-1</td>
<td>DWORD</td>
</tr>
<tr>
<td></td>
<td>-1</td>
<td></td>
<td></td>
<td>36000</td>
</tr>
<tr>
<td>zeroShift 3)</td>
<td>0 ... 2</td>
<td>0 ... 2</td>
<td>-1</td>
<td>DWORD</td>
</tr>
<tr>
<td></td>
<td>-1</td>
<td></td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>EncoderBitResolution 4)</td>
<td>8 ... 32</td>
<td>8 ... 32</td>
<td>WORD</td>
<td>18</td>
</tr>
<tr>
<td>Reserve</td>
<td>-</td>
<td>-</td>
<td>WORD</td>
<td>-</td>
</tr>
</tbody>
</table>

Remarks:
1) The parameter numOfUsedCams defines the interrupt cycle time (Therefore, it takes effect to the accuracy of the track) and the behaviour of the module if the DC information is lost.

<table>
<thead>
<tr>
<th>Parameter setting for numOfUsed-Cams</th>
<th>Number of cams used</th>
<th>Interrupt cycle time</th>
<th>Behaviour if DC information is lost</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>50 µs</td>
<td>Module changes to &quot;safe-operational&quot; state; the outputs are activated through the user program</td>
</tr>
<tr>
<td>1...8</td>
<td>1...8</td>
<td>80 µs</td>
<td></td>
</tr>
<tr>
<td>9...16</td>
<td>9...16</td>
<td>100 µs</td>
<td>the outputs are activated through the user program</td>
</tr>
<tr>
<td>17...32</td>
<td>17...32</td>
<td>200 µs</td>
<td></td>
</tr>
<tr>
<td>128</td>
<td>0</td>
<td>50 µs</td>
<td>Module keeps in &quot;operational&quot; state; the outputs are activated through the user program</td>
</tr>
<tr>
<td>129...136</td>
<td>1...8</td>
<td>80 µs</td>
<td>Module keeps in &quot;operational&quot; state; the outputs are activated according to an interpolated timing information</td>
</tr>
<tr>
<td>137...144</td>
<td>9...16</td>
<td>100 µs</td>
<td></td>
</tr>
<tr>
<td>145...170</td>
<td>17...32</td>
<td>200 µs</td>
<td></td>
</tr>
</tbody>
</table>

2) The parameter resolution defines the angle resolution of the track. The value gives the number of increments related to 360°; e. g. the value 36,000 corresponds to an angle resolution of 0.01°.

3) The parameter zeroShift defines the zero shift. With it the encoder can be adjusted to the mounting position. The value of zeroShift is set in encoder-increments. It is not assigned to the parameter resolution of the cam switch.

4) The parameter EncoderBitResolution defines the resolution of the used encoder (in bits), e. g. with the default setting 18 bits the encoder has 196,608 divisions.
Channel parameters for the cam switch (max. 32x)

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>Internal value</th>
<th>Internal value, type</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>camToTrack0 1)</td>
<td>Digital Output 0 ... 15, none</td>
<td>0 ... 15, FF</td>
<td>BYTE</td>
<td>FF</td>
</tr>
<tr>
<td>:</td>
<td>:</td>
<td>:</td>
<td>:</td>
<td>:</td>
</tr>
<tr>
<td>camToTrack31</td>
<td>Digital Output 0 ... 15, none</td>
<td>0 ... 15, FF</td>
<td>BYTE</td>
<td>FF</td>
</tr>
</tbody>
</table>

1) The value of the parameter camToTrack# defines which DO (digital output) is assigned to the track. camToTrack0 = 3 for example means that track 0 is assigned to the digital output 3. If the value FFh is set to a track, no digital output is assigned to it.

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>Referred FB from extended Cam Switch Library 2)</th>
<th>Internal value</th>
<th>Internal value, type</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>cam-Type[0] 1)</td>
<td>Common, Pulsed, Timed, Comfort, Cam shift, Binary shift, Multiturn cam, Time timed, Reference, Multiturn timed</td>
<td>MCX_CamSwitchSimple_c, MCX_CamSwitchSimple_dc, MCX_PulseSwitch_dc, MCX_CamSwitchTimed_dc, MCX_CamSwitchComfort_dc, MCX_CamShift_dc, MCX_BinaryShift_dc, MCX_CamSwitchMulti_dc, MCX_SwitchTimeTimed_dc, MCX_BinaryReference_dc, MCX_CamSwitchMultiTimed_dc</td>
<td>0</td>
<td>BYTE</td>
<td>0</td>
</tr>
</tbody>
</table>

1) camType additionally to camToTrack identifies the type of each cam switch and enables the use of a specific Function Block from the Extended Cam Switch Library.

2) camType parameters and the Extended Camswitch Library Extended Camswitch Library are only available for CI511-ETHCAT and CI512-ETHCAT with device index C0 and above.

Group parameters for the digital part

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>Internal value</th>
<th>Internal value, type</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input delay</td>
<td>0.01 ms</td>
<td>0</td>
<td>BYTE</td>
<td>0.01 ms 0x00</td>
</tr>
<tr>
<td></td>
<td>1 ms</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>8 ms</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>32 ms</td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Detect short-circuit at outputs</td>
<td>Off</td>
<td>0</td>
<td>BYTE</td>
<td>On 0x01</td>
</tr>
<tr>
<td></td>
<td>On</td>
<td>1</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

2888

3ADR010583, 1, en_US

2020/12/10
<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>Internal value</th>
<th>Internal value, type</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>Behaviour DO at comm. error *)</td>
<td>Off</td>
<td>0</td>
<td>BYTE</td>
<td>Off 0x00</td>
</tr>
<tr>
<td>Last value</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Last value 5 sec</td>
<td>6</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Last value 10 sec</td>
<td>11</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Substitute value</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Substitute value 5 sec</td>
<td>7</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Substitute value 10 sec</td>
<td>12</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Substitute values DO</td>
<td>0...65535</td>
<td>0000h...FFFFh</td>
<td>WORD</td>
<td>0 0x0000</td>
</tr>
</tbody>
</table>

*) The parameter behaviourDOatCommunicationFault is only analyzed if the Failsafe-mode is ON.

**Diagnosis**

In cases of short circuit or overload, the digital outputs are turned off. The modules performs reactivation automatically. Thus an acknowledgement of the errors is not necessary. The error message is stored via the LED.

---

**E1..E4**

<table>
<thead>
<tr>
<th>d1</th>
<th>d2</th>
<th>d3</th>
<th>d4</th>
<th>Identifier 000..063</th>
<th>AC500-Display</th>
<th>&lt;-- Display in</th>
</tr>
</thead>
<tbody>
<tr>
<td>Comp</td>
<td>Dev</td>
<td>Mod</td>
<td>Ch</td>
<td>Err</td>
<td>PS501 PLC Browser</td>
<td>ETHCAT Diagnosis block</td>
</tr>
<tr>
<td>Byte 6 Bit 6..7</td>
<td>Byte 3</td>
<td>Byte 4</td>
<td>Byte 5</td>
<td>Byte 6 Bit 0..5</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Class</th>
<th>Interface</th>
<th>Device</th>
<th>Module</th>
<th>Channel</th>
<th>Error identifier</th>
<th>Error message</th>
<th>Remedy</th>
</tr>
</thead>
<tbody>
<tr>
<td>E1..E4</td>
<td>d1</td>
<td>d2</td>
<td>d3</td>
<td>d4</td>
<td>Identifier 000..063</td>
<td>AC500-Display</td>
<td>&lt;-- Display in</td>
</tr>
<tr>
<td>Class</td>
<td>Interface</td>
<td>Device</td>
<td>Module</td>
<td>Channel</td>
<td>Error identifier</td>
<td>Error message</td>
<td>Remedy</td>
</tr>
<tr>
<td>Mod</td>
<td>Ch</td>
<td>Err</td>
<td>PS501 PLC Browser</td>
<td>ETHCAT Diagnosis block</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Module error**

3 | - | 31 | 31 | 31 | 43 | Internal error in the module | Replace I/O module |

3 | - | 31 | 31 | 31 | 20 | Slave-to-Slave malfunction | Check configuration |

3 | - | 31 | 31 | 31 | 41 | Distributed Clock malfunction | Check configuration |

3 | - | 31 | 31 | 31 | 26 | Parameter error | Check master |

3 | - | 31 | 31 | 31 | 11 | Process voltage UP too low | Check process supply voltage |
<table>
<thead>
<tr>
<th>Class</th>
<th>Interface</th>
<th>Device</th>
<th>Module</th>
<th>Channel</th>
<th>Error identifier</th>
<th>Error message</th>
<th>Remedy</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>-</td>
<td>31</td>
<td>31</td>
<td>31</td>
<td>45</td>
<td>Process voltage UP3 too low</td>
<td>Check process voltage</td>
</tr>
<tr>
<td>4</td>
<td>-</td>
<td>31</td>
<td>31</td>
<td>31</td>
<td>34</td>
<td>No response during initialization of the I/O module</td>
<td>Replace I/O module</td>
</tr>
<tr>
<td>4</td>
<td>-</td>
<td>31</td>
<td>31</td>
<td>31</td>
<td>46</td>
<td>Voltage feedback on activated digital outputs</td>
<td>Check terminals</td>
</tr>
</tbody>
</table>

Channel error digital

<table>
<thead>
<tr>
<th>Class</th>
<th>Interface</th>
<th>Device</th>
<th>Module</th>
<th>Channel</th>
<th>Error identifier</th>
<th>Error message</th>
<th>Remedy</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>-</td>
<td>31</td>
<td>2</td>
<td>0..15</td>
<td>46</td>
<td>Voltage feedback on deactivated digital output</td>
<td>Check terminals</td>
</tr>
<tr>
<td>4</td>
<td>-</td>
<td>31</td>
<td>2</td>
<td>0..15</td>
<td>47</td>
<td>Short circuit at digital output</td>
<td>Check terminals</td>
</tr>
</tbody>
</table>

Remarks:

1) In AC500 the following interface identifier applies:
"." = Diagnosis via bus-specific function blocks; 0 or 4 or 10 = Position of the Communication Module; 14 = I/O-Bus; 31 = Module itself
The identifier is not contained in the CI512-ETHCAT diagnosis block.

2) With "Device" the following allocation applies:
31 = Module itself or ADR = Hardware address (e.g. of the DC551)

3) With "Module" the following allocation applies dependent of the master:
31 = Module itself (Module error) or Module type (1=AI, 2=DO, 3=AO; channel error)

4) Diagnosis message appears for the whole output group and not per channel. The message occurs if the output channel is already active.

5) Diagnosis message appears per channel. The message occurs if the output channel is not active.

State LEDs

The LEDs are located at the front of module. There are 2 different groups:
• The 5 system LEDs (PWR, NET, DC, S-ERR and I/O-Bus) show the operation state of the module and display possible errors.
• The 29 process LEDs (UP, UP3, inputs, outputs, CH-ERR1 to CH-ERR3) show the process supply voltage and the states of the inputs and outputs and display possible errors.

Table 618: States of the 5 system LEDs

<table>
<thead>
<tr>
<th>LED</th>
<th>Color</th>
<th>Off</th>
<th>On</th>
<th>Flashing</th>
<th>1x flash</th>
<th>2x flash</th>
</tr>
</thead>
<tbody>
<tr>
<td>PWR/RUN</td>
<td>Green</td>
<td>Error in the internal supply voltage or process voltage missing</td>
<td>Internal supply voltage OK</td>
<td>Module is not configured</td>
<td>--</td>
<td>--</td>
</tr>
<tr>
<td></td>
<td>Yellow</td>
<td>--</td>
<td>Operational</td>
<td>Pre-operational</td>
<td>--</td>
<td>--</td>
</tr>
<tr>
<td>NET</td>
<td>Green</td>
<td>Init</td>
<td>Operational</td>
<td>Pre-operational</td>
<td>Safe-operational</td>
<td>--</td>
</tr>
<tr>
<td></td>
<td>Red</td>
<td>No error</td>
<td>PDI Watchdog Timeout</td>
<td>Invalid Configuration</td>
<td>Unsolicited State Change</td>
<td>Application time out</td>
</tr>
<tr>
<td>DC *)</td>
<td>Green</td>
<td>Distributed Clock not active</td>
<td>Distributed Clock active</td>
<td>--</td>
<td>--</td>
<td>--</td>
</tr>
<tr>
<td></td>
<td>Red</td>
<td>--</td>
<td>--</td>
<td>--</td>
<td>--</td>
<td>--</td>
</tr>
<tr>
<td>S-ERR</td>
<td>Red</td>
<td>No error</td>
<td>Internal error</td>
<td>--</td>
<td>--</td>
<td>--</td>
</tr>
<tr>
<td>I/O-Bus</td>
<td>Green</td>
<td>No expansion modules connected or communicaion error</td>
<td>---</td>
<td>---</td>
<td>--</td>
<td>--</td>
</tr>
<tr>
<td>ETH1</td>
<td>Green</td>
<td>No EtherCAT connection</td>
<td>Link OK</td>
<td>Link OK Data transfer OK</td>
<td>--</td>
<td>--</td>
</tr>
<tr>
<td></td>
<td>Yellow</td>
<td>--</td>
<td>--</td>
<td>--</td>
<td>--</td>
<td>--</td>
</tr>
<tr>
<td>ETH2</td>
<td>Green</td>
<td>No EtherCAT connection</td>
<td>Link OK</td>
<td>Link OK Data transfer OK</td>
<td>--</td>
<td>--</td>
</tr>
<tr>
<td></td>
<td>Yellow</td>
<td>--</td>
<td>--</td>
<td>--</td>
<td>--</td>
<td>--</td>
</tr>
</tbody>
</table>

*) The state of this LED is only significant if the camswitch functionality is enabled

Table 619: States of the 29 process LEDs

<table>
<thead>
<tr>
<th>LED</th>
<th>Color</th>
<th>OFF</th>
<th>ON</th>
<th>Flashing</th>
</tr>
</thead>
<tbody>
<tr>
<td>DC0 to DC7</td>
<td>Yellow</td>
<td>Input/Output is OFF</td>
<td>Input/Output is ON</td>
<td>--</td>
</tr>
<tr>
<td>DI8 to DI15</td>
<td>Yellow</td>
<td>Input is OFF</td>
<td>Input is ON (the input voltage is even displayed if the supply voltage is OFF)</td>
<td>--</td>
</tr>
</tbody>
</table>
### Technical data

The system data of AC500 and S500 refer to Chapter 1.5.4.5.1 “System data AC500” on page 3058 and are valid for standard version.

The system data of AC500-XC refer to Chapter 1.5.4.6.1 “System data AC500-XC” on page 3099 and are valid for the XC version.

Only additional details are therefore documented below.

The technical data are also valid for the XC version.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bus connection</td>
<td>2 x RJ45</td>
</tr>
<tr>
<td>Technology</td>
<td>Hilscher netX100</td>
</tr>
<tr>
<td>Transfer rate</td>
<td>10/100 Mbit/s (full-duplex)</td>
</tr>
<tr>
<td>Transfer method</td>
<td>According to Ethernet II, IEE802.3</td>
</tr>
<tr>
<td>Ethernet</td>
<td>100 base-TX, internal switch, 2x RJ45 socket</td>
</tr>
<tr>
<td>Expandability (S500 I/O modules)</td>
<td>Up to 10 S500 I/O modules (Index C0 and above), not available (Index below C0)</td>
</tr>
<tr>
<td>Indicators</td>
<td>5 LEDs for state indication</td>
</tr>
<tr>
<td>Adjusting elements</td>
<td>2 rotary switches (used for future topology extensions)</td>
</tr>
<tr>
<td>Quantity of input/output data</td>
<td>CI512-ETHCAT: 10 bytes input and 14 bytes output</td>
</tr>
<tr>
<td></td>
<td>CI511-ETHCAT: 18 bytes input and 18 bytes output</td>
</tr>
<tr>
<td>Limit of data for input and output</td>
<td>144 byte</td>
</tr>
<tr>
<td>Acyclic services</td>
<td>SDO (1500 bytes max.)</td>
</tr>
<tr>
<td></td>
<td>Emergency ECAT_SLV_DIAG</td>
</tr>
<tr>
<td>Protective functions (according to CODESYS)</td>
<td>Protected against:</td>
</tr>
<tr>
<td></td>
<td>● short circuit</td>
</tr>
<tr>
<td></td>
<td>● reverse supply</td>
</tr>
<tr>
<td></td>
<td>● overvoltage</td>
</tr>
<tr>
<td></td>
<td>● reverse polarity</td>
</tr>
<tr>
<td></td>
<td>Electrical isolation to network</td>
</tr>
</tbody>
</table>
## Technical data of the module

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Process supply voltages UP/UP3</td>
<td></td>
</tr>
<tr>
<td>Rated value</td>
<td>24 VDC (for inputs and outputs)</td>
</tr>
<tr>
<td>Max. load for the terminals</td>
<td>10 A</td>
</tr>
<tr>
<td>Protection against reversed voltage</td>
<td>Yes</td>
</tr>
<tr>
<td>Rated protection fuse on UP/UP3</td>
<td>10 A fast</td>
</tr>
<tr>
<td>Electrical isolation</td>
<td>Ethernet interface against the rest of the module</td>
</tr>
<tr>
<td>Inrush current from UP (at power up)</td>
<td>On request</td>
</tr>
<tr>
<td>Current consumption via UP (normal operation)</td>
<td>0.15 A</td>
</tr>
<tr>
<td>Current consumption via UP3</td>
<td>0.06 A + 0.5 A max. per output</td>
</tr>
<tr>
<td>Connections</td>
<td>Terminals 1.8 and 2.8 for +24 V (UP)</td>
</tr>
<tr>
<td></td>
<td>Terminal 3.8 for +24 V (UP3)</td>
</tr>
<tr>
<td></td>
<td>Terminals 1.9, 2.9 and 3.9 for 0 V (ZP)</td>
</tr>
<tr>
<td>Max. power dissipation within the module</td>
<td>6 W</td>
</tr>
<tr>
<td>Number of digital inputs</td>
<td>8</td>
</tr>
<tr>
<td>Number of digital outputs</td>
<td>8</td>
</tr>
<tr>
<td>Number of configurable digital inputs/outputs</td>
<td>8</td>
</tr>
<tr>
<td>Reference potential for all digital inputs and outputs</td>
<td>Minus pole of the supply voltage, signal name ZP</td>
</tr>
<tr>
<td>Diagnosis</td>
<td>See Diagnosis and Displays Chapter 1.5.3.7.3.2.9 “Diagnosis” on page 2889</td>
</tr>
<tr>
<td>Operation and error displays</td>
<td>34 LEDs (totally)</td>
</tr>
<tr>
<td>Weight (without terminal unit)</td>
<td>Ca. 125 g</td>
</tr>
<tr>
<td>Mounting position</td>
<td>Horizontal</td>
</tr>
<tr>
<td></td>
<td>Or vertical with derating (output load reduced to 50 % at 40 °C per group)</td>
</tr>
<tr>
<td>Cooling</td>
<td>The natural convection cooling must not be hindered by cable ducts or other parts in the switch-gear cabinet.</td>
</tr>
</tbody>
</table>

**NOTICE!**

**Attention:**

All I/O channels (digital and analog) are protected against reverse polarity, reverse supply, short circuit and continuous overvoltage up to 30 VDC.

**Multiple overloads**

No effects of multiple overloads on isolated multi-channel modules occur, as every channel is protected individually by an internal smart high-side switch.
### Technical data of the digital inputs

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of channels per module</td>
<td>8</td>
</tr>
<tr>
<td>Distribution of the channels into groups</td>
<td>1 group of 8 channels</td>
</tr>
<tr>
<td>Terminals of the channels DI0 to DI7</td>
<td>Terminals 2.0 to 2.7</td>
</tr>
<tr>
<td>Reference potential for all inputs</td>
<td>Terminals 1.9...3.9 (Minus pole of the supply voltage, signal name ZP)</td>
</tr>
<tr>
<td>Indication of the input signals</td>
<td>1 yellow LED per channel, the LED is ON when the input signal is high (signal 1)</td>
</tr>
<tr>
<td>Input type (according EN 61131-2)</td>
<td>Type 1</td>
</tr>
<tr>
<td>Input delay (0-&gt;1 or 1-&gt;0)</td>
<td>Typ. 0.1 ms, configurable from 0.1...32 ms</td>
</tr>
<tr>
<td>Input signal voltage</td>
<td>24 VDC</td>
</tr>
<tr>
<td>0-Signal</td>
<td>-3 V...+5 V</td>
</tr>
<tr>
<td>undefined Signal</td>
<td>&gt; +5 V...&lt; +15 V</td>
</tr>
<tr>
<td>1-Signal</td>
<td>+15 V...+30 V</td>
</tr>
<tr>
<td>Ripple with signal 0</td>
<td>Within -3 V...+5 V</td>
</tr>
<tr>
<td>Ripple with signal 1</td>
<td>Within +15 V...+30 V</td>
</tr>
<tr>
<td>Input current per channel</td>
<td></td>
</tr>
<tr>
<td>Input voltage +24 V</td>
<td>Typ. 5 mA</td>
</tr>
<tr>
<td>Input voltage +5 V</td>
<td>&gt; 1 mA</td>
</tr>
<tr>
<td>Input voltage +15 V</td>
<td>&gt; 2 mA</td>
</tr>
<tr>
<td>Input voltage +30 V</td>
<td>&lt; 8 mA</td>
</tr>
<tr>
<td>Max. cable length</td>
<td></td>
</tr>
<tr>
<td>Shielded</td>
<td>1000 m</td>
</tr>
<tr>
<td>Unshielded</td>
<td>600 m</td>
</tr>
</tbody>
</table>

### Technical data of the digital outputs

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of channels per module</td>
<td>8</td>
</tr>
<tr>
<td>Distribution of the channels into groups</td>
<td>1 group of 8 channels</td>
</tr>
<tr>
<td>Terminals of the channels DO0 to DO7</td>
<td>Terminals 3.0 to 3.7</td>
</tr>
<tr>
<td>Reference potential for all outputs</td>
<td>Terminals 1.9...3.9 (minus pole of the supply voltage, signal name ZP)</td>
</tr>
<tr>
<td>Common power supply voltage</td>
<td>For all outputs terminal 3.8 (plus pole of the supply voltage, signal name UP3)</td>
</tr>
<tr>
<td>Output voltage for signal 1</td>
<td>UP3 (-0.8 V)</td>
</tr>
<tr>
<td>Output delay (0-&gt;1 or 1-&gt;0)</td>
<td>On request</td>
</tr>
<tr>
<td>Output current</td>
<td></td>
</tr>
<tr>
<td>Rated value per channel</td>
<td>500 mA at UP3 = 24 V</td>
</tr>
<tr>
<td>Max. value (all channels together)</td>
<td>4 A</td>
</tr>
<tr>
<td>Leakage current with signal 0</td>
<td>&lt; 0.5 mA</td>
</tr>
<tr>
<td>Parameter</td>
<td>Value</td>
</tr>
<tr>
<td>-----------------------------------------------</td>
<td>------------------------------------</td>
</tr>
<tr>
<td>Fuse for UP3</td>
<td>10 A fast</td>
</tr>
<tr>
<td>Demagnetization with inductive DC load</td>
<td>Via internal varistors (see figure below this table)</td>
</tr>
<tr>
<td>Output switching frequency</td>
<td></td>
</tr>
<tr>
<td>With resistive load</td>
<td>On request</td>
</tr>
<tr>
<td>With inductive loads</td>
<td>Max. 0.5 Hz</td>
</tr>
<tr>
<td>With lamp loads</td>
<td>11 Hz max. at 5 W max.</td>
</tr>
<tr>
<td>Short-circuit-proof / overload-proof</td>
<td>Yes</td>
</tr>
<tr>
<td>Overload message (I &gt; 0.7 A)</td>
<td>Yes, after ca. 100 ms</td>
</tr>
<tr>
<td>Output current limitation</td>
<td>Yes, automatic reactivation after short circuit/overload</td>
</tr>
<tr>
<td>Resistance to feedback against 24V signals</td>
<td>Yes (software-controlled supervision)</td>
</tr>
<tr>
<td>Max. cable length</td>
<td></td>
</tr>
<tr>
<td>Shielded</td>
<td>1000 m</td>
</tr>
<tr>
<td>Unshielded</td>
<td>600 m</td>
</tr>
</tbody>
</table>

The following drawing shows the circuitry of a digital input/output with the varistors for demagnetization when inductive loads are switched off.

![Circuit Diagram](image)

*Fig. 226: Digital input/output (circuit diagram)*

1  Digital Output
2  Varistors for demagnetization when inductive loads are turned off

Figure:

**Technical data of the configurable digital inputs/outputs**

Each of the configurable I/O channels is defined as input or output by the user program. This is done by interrogating or allocating the corresponding channel.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of channels per module</td>
<td>8 inputs/outputs (with transistors)</td>
</tr>
<tr>
<td>Distribution of the channels into groups</td>
<td>1 group for 8 channels</td>
</tr>
<tr>
<td>If the channels are used as inputs</td>
<td></td>
</tr>
<tr>
<td>Channels DC0...DC07</td>
<td>Terminals 1.0...1.7</td>
</tr>
<tr>
<td>If the channels are used as outputs</td>
<td></td>
</tr>
<tr>
<td>Channels DC0...DC07</td>
<td>Terminals 1.0...1.7</td>
</tr>
</tbody>
</table>
### Technical data of the digital inputs/outputs if used as inputs

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Indication of the input/output signals</td>
<td>1 yellow LED per channel, the LED is ON when the input/output signal is high (signal 1)</td>
</tr>
<tr>
<td>Electrical isolation</td>
<td>From the Ethernet network</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of channels per module</td>
<td>8</td>
</tr>
<tr>
<td>Distribution of the channels into groups</td>
<td>1 group of 8 channels</td>
</tr>
<tr>
<td>Terminals of the channels DC0 to DC7</td>
<td>Terminals 1.0 to 1.7</td>
</tr>
<tr>
<td>Reference potential for all inputs</td>
<td>Terminals 1.9...3.9 (Minus pole of the supply voltage, signal name ZP)</td>
</tr>
<tr>
<td>Indication of the input signals</td>
<td>1 yellow LED per channel, the LED is ON when the input signal is high (signal 1)</td>
</tr>
<tr>
<td>Input type (according EN 61131-2)</td>
<td>Type 1</td>
</tr>
<tr>
<td>Input delay (0-&gt;1 or 1-&gt;0)</td>
<td>Typ. 0.1 ms, configurable from 0.1...32 ms</td>
</tr>
<tr>
<td>Input signal voltage</td>
<td>24 VDC</td>
</tr>
<tr>
<td>0-Signal</td>
<td>-3 V...+5 V *)</td>
</tr>
<tr>
<td>Undefined Signal</td>
<td>&gt; +5 V...&lt; +15 V</td>
</tr>
<tr>
<td>1-Signal</td>
<td>+15 V...+30 V</td>
</tr>
<tr>
<td>Ripple with signal 0</td>
<td>Within -3 V...+5 V *)</td>
</tr>
<tr>
<td>Ripple with signal 1</td>
<td>Within +15 V...+30 V</td>
</tr>
<tr>
<td>Input current per channel</td>
<td></td>
</tr>
<tr>
<td>Input voltage +24 V</td>
<td>Typ. 5 mA</td>
</tr>
<tr>
<td>Input voltage +5 V</td>
<td>&gt; 1 mA</td>
</tr>
<tr>
<td>Input voltage +15 V</td>
<td>&gt; 2 mA</td>
</tr>
<tr>
<td>Input voltage +30 V</td>
<td>&lt; 8 mA</td>
</tr>
<tr>
<td>Max. cable length</td>
<td></td>
</tr>
<tr>
<td>Shielded</td>
<td>1000 m</td>
</tr>
<tr>
<td>Unshielded</td>
<td>600 m</td>
</tr>
</tbody>
</table>

*) Due to the direct connection to the output, the demagnetizing varistor is also effective at the input (see figure) above. This is why the difference between UPx and the input signal may not exceed the clamp voltage of the varistor. The varistor limits the voltage to approx. 36 V. Following this, the input voltage must range from -12 V to +30 V when UPx = 24 V and from -6 V to +30 V when UPx = 30 V.

### Technical data of the digital inputs/outputs if used as outputs

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of channels per module</td>
<td>8</td>
</tr>
<tr>
<td>Distribution of the channels into groups</td>
<td>1 group of 8 channels</td>
</tr>
<tr>
<td>Terminals of the channels DC0 to DC7</td>
<td>Terminals 1.0 to 1.7</td>
</tr>
<tr>
<td>Parameter</td>
<td>Value</td>
</tr>
<tr>
<td>--------------------------------------------------------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Reference potential for all outputs</td>
<td>Terminals 1.9...3.9 (minus pole of the supply voltage, signal name ZP)</td>
</tr>
<tr>
<td>Common power supply voltage</td>
<td>For all outputs terminal 3.8 (plus pole of the supply voltage, signal name UP3)</td>
</tr>
<tr>
<td>Output voltage for signal 1</td>
<td>UP3 (-0.8 V)</td>
</tr>
<tr>
<td>Output delay (0-&gt;1 or 1-&gt;0)</td>
<td>On request</td>
</tr>
<tr>
<td>Output current</td>
<td></td>
</tr>
<tr>
<td>Rated value per channel</td>
<td>500 mA at UP3 = 24 V</td>
</tr>
<tr>
<td>Max. value (all channels together)</td>
<td>4 A</td>
</tr>
<tr>
<td>Leakage current with signal 0</td>
<td>&lt; 0.5 mA</td>
</tr>
<tr>
<td>Fuse for UP3</td>
<td>10 A fast</td>
</tr>
<tr>
<td>Demagnetization with inductive DC load</td>
<td>Via internal varistors (see figure below this table)</td>
</tr>
<tr>
<td>Output switching frequency</td>
<td></td>
</tr>
<tr>
<td>With resistive load</td>
<td>On request</td>
</tr>
<tr>
<td>With inductive loads</td>
<td>Max. 0.5 Hz</td>
</tr>
<tr>
<td>With lamp loads</td>
<td>11 Hz max. at 5 W max.</td>
</tr>
<tr>
<td>Short-circuit-proof / overload-proof</td>
<td>Yes</td>
</tr>
<tr>
<td>Overload message (I &gt; 0.7 A)</td>
<td>Yes, after ca. 100 ms</td>
</tr>
<tr>
<td>Output current limitation</td>
<td>Yes, automatic reactivation after short circuit/overload</td>
</tr>
<tr>
<td>Resistance to feedback against 24V signals</td>
<td>Yes (software-controlled supervision)</td>
</tr>
<tr>
<td>Max. cable length</td>
<td></td>
</tr>
<tr>
<td>Shielded</td>
<td>1000 m</td>
</tr>
<tr>
<td>Unshielded</td>
<td>600 m</td>
</tr>
</tbody>
</table>

The following drawing shows the circuitry of a digital input/output with the varistors for demagnetization when inductive loads are switched off.

![Circuit Diagram](image)

*Fig. 227: Digital input/output (circuit diagram)*

1 Digital input/output
2 For demagnetization when inductive loads are turned off

**Ordering data**

<table>
<thead>
<tr>
<th>Part no.</th>
<th>Description</th>
<th>Product life cycle phase *)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1SAP 221 000 R0001</td>
<td>CI512-ETHCAT, EtherCAT bus module, 8 DI, 8 DO and 8 DC</td>
<td>Active</td>
</tr>
</tbody>
</table>
1.5.3.7.4 Modbus

CI521-MODTCP

- 4 analog inputs (resolution 12 bits plus sign)
- 2 analog outputs (resolution 12 bits plus sign)
- 8 digital inputs 24 VDC
- 8 digital outputs 24 VDC, 0.5 A max.
- Module-wise electrically isolated
- Fast Counter
- XC version for usage in extreme ambient conditions available

*) For planning and commissioning of new installations use modules in Active status only.
1. I/O bus
2. Allocation between terminal number and signal name
3. 6 yellow LEDs to display the signal states of the analog inputs/outputs (AI0 - AI3, AO0 - AO1)
4. 8 yellow LEDs to display the signal states of the digital inputs (DI0 - DI7)
5. 8 yellow LEDs to display the signal states of the digital outputs (DO0 - DO7)
6. 2 green LEDs to display the process supply voltage UP and UP3
7. 3 red LEDs to display errors (CH-ERR1, CH-ERR2, CH-ERR3)
8. 5 system LEDs: PWR/RUN, STA1 ETH, STA2 ETH, S-ERR, I/O-Bus
9. Label
10. 2 rotary switches for setting the IP address
11. Ethernet interfaces (ETH1, ETH2) on the terminal unit
12. Terminal unit
13. DIN rail
*+*: Sign for XC version
Intended purpose

The Modbus TCP bus module CI521-MODTCP is used as decentralized I/O module in Modbus TCP networks. The network connection is performed via 2 RJ45 connectors which are integrated in the terminal unit. The bus module contains 22 I/O channels with the following properties:

- 4 analog inputs (1.0...1.3)
- 2 analog outputs (1.5...1.6)
- 8 digital inputs 24 V DC in 1 group (2.0...2.7)
- 8 digital outputs 24 V DC in 1 group (3.0...3.7)

The inputs/outputs are electrically isolated from the Ethernet network. There is no potential separation between the channels. The configuration of the analog inputs/outputs is performed by software.

For usage in enhanced ambient conditions (e.g. wider temperature and humidity range), a special XC version of the device is available.

Functionality

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Interface</td>
<td>Ethernet</td>
</tr>
<tr>
<td>Protocol</td>
<td>Modbus TCP</td>
</tr>
<tr>
<td>Power supply</td>
<td>From the process supply voltage UP</td>
</tr>
<tr>
<td>Supply of the electronic circuitry of the I/O expansion modules attached</td>
<td>Through the expansion bus interface (I/O bus)</td>
</tr>
<tr>
<td>Rotary switches</td>
<td>for setting the last BYTE of the IP (00h to FFh)</td>
</tr>
<tr>
<td>Analog inputs</td>
<td>4 (configurable via software)</td>
</tr>
<tr>
<td>Analog outputs</td>
<td>2 (configurable via software)</td>
</tr>
<tr>
<td>Digital inputs</td>
<td>8 (24 VDC; delay time configurable via software)</td>
</tr>
<tr>
<td>Digital outputs</td>
<td>8 (24 VDC, 0.5 A max.)</td>
</tr>
<tr>
<td>LED displays</td>
<td>For system displays, signal states, errors and power supply</td>
</tr>
<tr>
<td>External supply voltage</td>
<td>Via terminals ZP, UP and UP3 (process supply voltage 24 VDC)</td>
</tr>
</tbody>
</table>
| Required terminal unit                   | TU507 or TU508  
> Chapter 1.5.3.5.1 “TU507-ETH and TU508-ETH for Ethernet communication interface modules” on page 2307 |

Electrical connection

The Ethernet bus module CI521-MODTCP is plugged on the I/O terminal unit TU507-ETH or TU508-ETH  
> Chapter 1.5.3.5.1 “TU507-ETH and TU508-ETH for Ethernet communication interface modules” on page 2307. Properly seat the module and press until it locks in place. The terminal unit is mounted on a DIN rail or with 2 screws plus the additional accessory for wall mounting (TA526  
> Chapter 1.5.3.8.1.4 “TA526 - Wall mounting accessory” on page 3035).

The electrical connection of the I/O channels is carried out using the 30 terminals of the I/O terminal unit. I/O modules can be replaced without re-wiring the terminal units.
For a detailed description of the mounting, disassembly and electrical connection of the module, please refer to the System Assembly, Construction and Connection chapter & Chapter 1.5.4.5 “AC500 (Standard)” on page 3058.

The terminals 1.8 and 2.8 as well as 1.9, 2.9 and 3.9 are electrically interconnected within the terminal unit and have always the same assignment, independent of the inserted module:

Terminals 1.8 and 2.8: Process supply voltage UP = +24 VDC
Terminal 3.8: Process supply voltage UP3 = +24 VDC
Terminals 1.9, 2.9 and 3.9: Process supply voltage ZP = 0 V

With a separate UP3 power supply, the digital outputs can be switched off externally. This way, an emergency-off functionality can be realized.

Conditions for undisturbed operating with older I/O expansion modules

All I/O expansion modules that are attached to the CI52x-MODTCP must be powered up together with the CI52x-MODTCP if the firmware version of these I/O expansion modules is V1.9 or lower.

The firmware version is related to the index. The index is printed on the module type label on the right side.

Modules as of index listed in the following table can be powered up independently.

<table>
<thead>
<tr>
<th>S500 I/O module type</th>
<th>First index with firmware version above 1.9</th>
</tr>
</thead>
<tbody>
<tr>
<td>AI523</td>
<td>D0</td>
</tr>
<tr>
<td>AI523-XC</td>
<td>D0</td>
</tr>
<tr>
<td>AI531</td>
<td>A3</td>
</tr>
<tr>
<td>AI531-XC</td>
<td>A0</td>
</tr>
<tr>
<td>AO523</td>
<td>D0</td>
</tr>
<tr>
<td>AO523-XC</td>
<td>D0</td>
</tr>
<tr>
<td>AX521</td>
<td>D0</td>
</tr>
<tr>
<td>AX521-XC</td>
<td>D0</td>
</tr>
<tr>
<td>AX522</td>
<td>D0</td>
</tr>
<tr>
<td>AX522-XC</td>
<td>D0</td>
</tr>
<tr>
<td>CD522</td>
<td>A2</td>
</tr>
<tr>
<td>CD522-XC</td>
<td>A0</td>
</tr>
<tr>
<td>DA501</td>
<td>A2</td>
</tr>
<tr>
<td>DA501-XC</td>
<td>A0</td>
</tr>
<tr>
<td>DA502</td>
<td>A1</td>
</tr>
<tr>
<td>DA502-XC</td>
<td>A1</td>
</tr>
<tr>
<td>DC522</td>
<td>D0</td>
</tr>
<tr>
<td>DC522-XC</td>
<td>D0</td>
</tr>
<tr>
<td>DC523</td>
<td>D0</td>
</tr>
</tbody>
</table>
**S500 I/O module type** | **First index with firmware version above 1.9**
--- | ---
DC523-XC | D0
DC532 | D0
DC532-XC | D0
DI524 | D0
DI524-XC | D0
DO524 | A2
DO524-XC | A2
DX522 | D0
DX522-XC | D0
DX531 | D0
AC522 | D0
PD501 | D0

**Do not connect any voltages externally to digital outputs!**

*Reason:* Externally voltages at an output or several outputs may cause that other outputs are supplied through that voltage instead of voltage UP3 (reverse voltage). This is not intended usage.

---

**CAUTION!**

**Risk of malfunction by not intended usage!**

If the function cut off of the digital outputs should be used by deactivation of the supply voltage UP3, be sure that no external voltage is connected at the outputs DO0..DO7.

---

The assignment of the other terminals:

<table>
<thead>
<tr>
<th>Terminal</th>
<th>Signal</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0</td>
<td>AI0+</td>
<td>Plus pole of analog input signal 0</td>
</tr>
<tr>
<td>1.1</td>
<td>AI1+</td>
<td>Plus pole of analog input signal 1</td>
</tr>
<tr>
<td>1.2</td>
<td>AI2+</td>
<td>Plus pole of analog input signal 2</td>
</tr>
<tr>
<td>1.3</td>
<td>AI3+</td>
<td>Plus pole of analog input signal 3</td>
</tr>
<tr>
<td>1.4</td>
<td>AI-</td>
<td>Minus pole of analog input signals 0 to 3</td>
</tr>
<tr>
<td>1.5</td>
<td>AO0+</td>
<td>Plus pole of analog output signal 0</td>
</tr>
<tr>
<td>1.6</td>
<td>AO1+</td>
<td>Plus pole of analog output signal 1</td>
</tr>
<tr>
<td>1.7</td>
<td>AI-</td>
<td>Minus pole of analog output signals 0 and 1</td>
</tr>
<tr>
<td>1.8</td>
<td>UP</td>
<td>Process voltage UP (24 VDC)</td>
</tr>
<tr>
<td>1.9</td>
<td>ZP</td>
<td>Process voltage ZP (0 VDC)</td>
</tr>
<tr>
<td>2.0</td>
<td>DI0</td>
<td>Signal of the digital input DI0</td>
</tr>
<tr>
<td>2.1</td>
<td>DI1</td>
<td>Signal of the digital input DI1</td>
</tr>
<tr>
<td>2.2</td>
<td>DI2</td>
<td>Signal of the digital input DI2</td>
</tr>
<tr>
<td>2.3</td>
<td>DI3</td>
<td>Signal of the digital input DI3</td>
</tr>
<tr>
<td>2.4</td>
<td>DI4</td>
<td>Signal of the digital input DI4</td>
</tr>
<tr>
<td>Terminal</td>
<td>Signal</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td>2.5</td>
<td>DI5</td>
<td>Signal of the digital input DI5</td>
</tr>
<tr>
<td>2.6</td>
<td>DI6</td>
<td>Signal of the digital input DI6</td>
</tr>
<tr>
<td>2.7</td>
<td>DI7</td>
<td>Signal of the digital input DI7</td>
</tr>
<tr>
<td>2.8</td>
<td>UP</td>
<td>Process voltage UP (24 VDC)</td>
</tr>
<tr>
<td>2.9</td>
<td>ZP</td>
<td>Process voltage ZP (0 VDC)</td>
</tr>
<tr>
<td>3.0</td>
<td>DO0</td>
<td>Signal of the digital output DO0</td>
</tr>
<tr>
<td>3.1</td>
<td>DO1</td>
<td>Signal of the digital output DO1</td>
</tr>
<tr>
<td>3.2</td>
<td>DO2</td>
<td>Signal of the digital output DO2</td>
</tr>
<tr>
<td>3.3</td>
<td>DO3</td>
<td>Signal of the digital output DO3</td>
</tr>
<tr>
<td>3.4</td>
<td>DO4</td>
<td>Signal of the digital output DO4</td>
</tr>
<tr>
<td>3.5</td>
<td>DO5</td>
<td>Signal of the digital output DO5</td>
</tr>
<tr>
<td>3.6</td>
<td>DO6</td>
<td>Signal of the digital output DO6</td>
</tr>
<tr>
<td>3.7</td>
<td>DO7</td>
<td>Signal of the digital output DO7</td>
</tr>
<tr>
<td>3.8</td>
<td>UP3</td>
<td>Process voltage UP3 (24 VDC)</td>
</tr>
<tr>
<td>3.9</td>
<td>ZP</td>
<td>Process voltage ZP (0 VDC)</td>
</tr>
</tbody>
</table>

**WARNING!**
Removal/Insertion under power

The devices are not designed for removal or insertion under power. Because of unforeseeable consequences, it is not allowed to plug or unplug devices with the power being ON.

Make sure that all voltage sources (supply and process voltage) are switched off before you

- connect or disconnect any signal or terminal block
- remove, mount or replace a module.

Disconnecting any powered devices while energized in a hazardous location could result in an electric arc, which could create a flammable ignition resulting in fire or explosion.

Make sure that power is removed and that the area has been thoroughly checked to ensure that flammable materials are not present prior to proceeding.

The devices must not be opened when in operation. The same applies to the network interfaces.

**NOTICE!**
Risk of damaging the PLC modules!

- Overvoltages and short circuits might damage the PLC modules.
  - Make sure that all voltage sources (supply voltage and process supply voltage) are switched off before you begin with operations at the system.
  - Never connect any voltages or signals to reserved terminals (marked with ---). Reserved terminals may carry internal voltages.

For the open-circuit detection (cut wire), each analog input channel is pulled up to "plus" by a high-resistance resistor. If nothing is connected, the maximum voltage will be read in then.
Generally, analog signals must be laid in shielded cables. The cable shields must be earthed at both sides of the cables. In order to avoid unacceptable potential differences between different parts of the installation, low resistance equipotential bonding conductors must be laid.

Only for simple applications (low electromagnetic disturbances, no high requirement on precision), the shielding can also be omitted.

The following figures show the electrical connection of the Ethernet bus module CI521-MODTCP.

**Fig. 228: Connection of the bus module CI521-MODTCP**

Further information is provided in the System Technology chapter CI52x-MODTCP Chapter 1.5.5.3.1 “Modbus communication interface module” on page 3200.

**Connection of the digital inputs**

The following figure shows the electrical connection of the digital input DI0. Proceed with the digital inputs DI1 to DI7 in the same way.
Fig. 229: Connection of the digital inputs to the module CI521-MODTCP

The meaning of the LEDs is described in Displays Chapter 1.5.3.7.4.1.8.2 “State LEDs” on page 2928.

Connection of the digital outputs

The following figure shows the electrical connection of the digital output DO0. Proceed with the digital outputs DO1 - DO7 in the same way.
Fig. 230: Connection of configurable digital inputs/outputs to the module CI521-MODTCP

The meaning of the LEDs is described in Displays Chapter 1.5.3.7.4.1.8.2 “State LEDs” on page 2928.

Connection of resistance thermometers in 2-wire configuration to the analog inputs

When resistance thermometers (Pt100, Pt1000, Ni1000) are used, a constant current must flow through them to build the necessary voltage drop for the evaluation. For this, the module CI521-MODTCP provides a constant current source which is multiplexed over the max. 4 analog input channels.

The following figure shows the connection of resistance thermometers in 2-wire configuration to the analog input AI0. Proceed with the analog inputs AI1 to AI3 in the same way.
Fig. 231: Connection of resistance thermometers in 2-wire configuration to the analog inputs

The following measuring ranges can be configured “Parameterization” on page 2918 and “Measuring ranges” on page 2930:

<table>
<thead>
<tr>
<th>Thermometer</th>
<th>Measuring Range</th>
<th>Configuration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pt100</td>
<td>-50 °C...+70 °C</td>
<td>2-wire config, 1 channel used</td>
</tr>
<tr>
<td>Pt100</td>
<td>-50 °C...+400 °C</td>
<td>2-wire config, 1 channel used</td>
</tr>
<tr>
<td>Pt1000</td>
<td>-50 °C...+400 °C</td>
<td>2-wire config, 1 channel used</td>
</tr>
<tr>
<td>Ni1000</td>
<td>-50 °C...+150 °C</td>
<td>2-wire config, 1 channel used</td>
</tr>
</tbody>
</table>

The function of the LEDs is described under Diagnosis and displays “Diagnosis and state LEDs” on page 2924.

To avoid error messages from unused analog input channels, configure them as "unused".

Connection of resistance thermometers in 3-wire configuration to the analog inputs

When resistance thermometers (Pt100, Pt1000, Ni1000) are used, a constant current must flow through them to build the necessary voltage drop for the evaluation. For this, the module CI521-MODTCP provides a constant current source which is multiplexed over the max. 4 analog input channels.

The following figure shows the connection of resistance thermometers in 3-wire configuration to the analog inputs AI0 and AI1. Proceed with the analog inputs AI2 and AI3 in the same way.
Fig. 232: Connection of resistance thermometers in 3-wire configuration to the analog inputs

With 3-wire configuration, 2 adjacent analog channels belong together (e. g. the channels 0 and 1). In this case, both channels are configured according to the desired operating mode. The lower address must be the even address (channel 0), the next higher address must be the odd address (channel 1).

The constant current of one channel flows through the resistance thermometer. The constant current of the other channel flows through one of the cores. The module calculates the measured value from the two voltage drops and stores it under the input with the higher channel number (e. g. I1).

In order to keep measuring errors as small as possible, it is necessary to have all the involved conductors in the same cable. All the conductors must have the same cross section.

The following measuring ranges can be configured ° Chapter 1.5.3.7.4.1.7 “Parameterization” on page 2918 and ° Chapter 1.5.3.7.4.1.9 “Measuring ranges” on page 2930:

<table>
<thead>
<tr>
<th>Type</th>
<th>Range</th>
<th>Configuration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pt100</td>
<td>-50 °C...+70 °C</td>
<td>3-wire used</td>
</tr>
<tr>
<td>Pt100</td>
<td>-50 °C...+400 °C</td>
<td>3-wire used</td>
</tr>
<tr>
<td>Pt1000</td>
<td>-50 °C...+400 °C</td>
<td>3-wire used</td>
</tr>
<tr>
<td>Ni1000</td>
<td>-50 °C...+150 °C</td>
<td>3-wire used</td>
</tr>
</tbody>
</table>

The function of the LEDs is described under Diagnosis and displays ° Displays ° Chapter 1.5.3.7.4.1.8 “Diagnosis and state LEDs” on page 2924.

The module CI521-MODTCP performs a linearization of the resistance characteristic.

To avoid error messages from unused analog input channels, configure them as "unused".

2908  3ADR010583, 1, en_US  2020/12/10
Connection of active-type analog sensors (voltage) with electrically isolated power supply to the analog inputs

The following figure shows the connection of active-type analog sensors (voltage) with electrically isolated power supply to the analog input AI0. Proceed with the analog inputs AI1 to AI3 in the same way.

![Connection diagram](image)

**Fig. 233: Connection of active-type analog sensors (voltage) with electrically isolated power supply to the analog inputs**

The following measuring ranges can be configured:  
- Chapter 1.5.3.7.4.1.7 “Parameterization” on page 2918 and  
- Chapter 1.5.3.7.4.1.9 “Measuring ranges” on page 2930:

<table>
<thead>
<tr>
<th>Voltage</th>
<th>0...10 V</th>
<th>1 channel used</th>
</tr>
</thead>
<tbody>
<tr>
<td>Voltage</td>
<td>-10 V...+10 V</td>
<td>1 channel used</td>
</tr>
</tbody>
</table>

The function of the LEDs is described under Diagnosis and displays / Displays:  
- Chapter 1.5.3.7.4.1.8 “Diagnosis and state LEDs” on page 2924.

To avoid error messages from unused analog input channels, configure them as "unused".

Connection of active-type analog sensors (Current) with electrically isolated power supply to the analog inputs

The following figure shows the connection of active-type analog sensors (current) with electrically isolated power supply to the analog input AI0. Proceed with the analog inputs AI1 to AI3 in the same way.
Fig. 234: Connection of active-type analog sensors (current) with electrically isolated power supply to the analog inputs

The following measuring ranges can be configured ➔ Chapter 1.5.3.7.4.1.7 “Parameterization” on page 2918 and ➔ Chapter 1.5.3.7.4.1.9 “Measuring ranges” on page 2930:

<table>
<thead>
<tr>
<th>Current</th>
<th>0...20 mA</th>
<th>1 channel used</th>
</tr>
</thead>
<tbody>
<tr>
<td>Current</td>
<td>4...20 mA</td>
<td>1 channel used</td>
</tr>
</tbody>
</table>

The function of the LEDs is described under Diagnosis and displays / Displays ➔ Chapter 1.5.3.7.4.1.8 “Diagnosis and state LEDs” on page 2924.

Unused input channels can be left open-circuited, because they are of low resistance.

To avoid error messages through unused analog input channels in measuring range 4...20 mA, these channels should be configured as “Not used”.

Connection of active-type analog sensors (Voltage) with no electrically isolated power supply to the analog inputs

The following figure shows the connection of active-type analog sensors (voltage) with no electrically isolated power supply to the analog input AI0. Proceed with the analog inputs AI1 to AI3 in the same way.
CAUTION!
Risk of faulty measurements!
The minus pole at the sensors must not have a too big potential difference with respect to ZP (max. ± 1 V).
Make sure that the potential difference never exceeds ± 1 V (also not with long cable lengths).

The following measuring ranges can be configured "Parameterization" on page 2918 and "Measuring ranges" on page 2930.

<table>
<thead>
<tr>
<th>Voltage</th>
<th>0...10 V</th>
<th>1 channel used</th>
</tr>
</thead>
<tbody>
<tr>
<td>Voltage</td>
<td>-10 V...+10 V</td>
<td>1 channel used</td>
</tr>
</tbody>
</table>

The function of the LEDs is described under Diagnosis and displays "Diagnosis and state LEDs" on page 2924.

To avoid error messages from unused analog input channels, configure them as "unused".

Connection of passive-type analog sensors (Current) to the analog inputs
The following figure shows the connection of passive-type analog sensors (current) to the analog input AI0. Proceed with the analog inputs AI1 to AI3 in the same way.
Fig. 236: Connection of passive-type analog sensors (current) to the analog inputs

The following measuring ranges can be configured  Chapter 1.5.3.7.4.1.7 “Parameterization” on page 2918 and  Chapter 1.5.3.7.4.1.9 “Measuring ranges” on page 2930:

| Current | 4...20 mA | 1 channel used |

The function of the LEDs is described under Diagnosis and displays / Displays  Chapter 1.5.3.7.4.1.8 “Diagnosis and state LEDs” on page 2924.

CAUTION!
Risk of overloading the analog input!
If an analog current sensor supplies more than 25 mA for more than 1 second during initialization, this input is switched off by the module (input protection).

Use only sensors with fast initialization or without current peaks higher than 25 mA. If not possible, connect a 10-volt zener diode in parallel to AIx+ and ZP.

Unused input channels can be left open-circuited, because they are of low resistance.

To avoid error messages through unused analog input channels in measuring range 4...20 mA, these channels should be configured as "Not used".

Connection of active-type analog sensors (Voltage) to differential analog inputs

Differential inputs are very useful, if analog sensors are used which are remotely non-isolated (e.g. the minus terminal is remotely earthed).

The evaluation using differential inputs helps to considerably increase the measuring accuracy and to avoid earthing loops.
With differential input configurations, two adjacent analog channels belong together (e.g. the channels 0 and 1). In this case, both channels are configured according to the desired operating mode. The lower address must be the even address (channel 0), the next higher address must be the odd address (channel 1). The converted analog value is available at the higher address (channel 1).

The analog value is calculated by subtraction of the input value with the higher address from the input value of the lower address.

The converted analog value is available at the odd channel (higher address).

**CAUTION!**
Risk of faulty measurements!
The minus pole at the sensors must not have a too big potential difference with respect to ZP (max. ±1 V).

Make sure that the potential difference never exceeds ±1 V.

The following figure shows the connection of active-type analog sensors (voltage) to differential analog inputs AI0 and AI1. Proceed with AI2 and AI3 in the same way.

![Connection diagram](image)

*Fig. 237: Connection of active-type analog sensors (voltage) to differential analog inputs AI0 and AI1. Proceed with AI2 and AI3 in the same way.*

The following measuring ranges can be configured [Chapter 1.5.3.7.4.1.7 “Parameterization” on page 2918](#) and [Chapter 1.5.3.7.4.1.9 “Measuring ranges” on page 2930](#):

<table>
<thead>
<tr>
<th>Voltage</th>
<th>0...10 V</th>
<th>With differential inputs, 2 channels used</th>
</tr>
</thead>
<tbody>
<tr>
<td>Voltage</td>
<td>-10 V...+10 V</td>
<td>With differential inputs, 2 channels used</td>
</tr>
</tbody>
</table>

The function of the LEDs is described under Diagnosis and displays / Displays [Chapter 1.5.3.7.4.1.8 “Diagnosis and state LEDs” on page 2924](#).
To avoid error messages from unused analog input channels, configure them as "unused".

**Use of analog inputs as digital inputs**

Several (or all) analog inputs can be configured as digital inputs [Chapter 1.5.3.7.4.1.10.5 “Technical data of the analog inputs if used as digital inputs” on page 2936. The inputs are not electrically isolated against the other analog channels.

The following figure shows the connection of digital sensors to the analog input AI0. Proceed with the analog inputs AI1 to AI3 in the same way.

![Use of analog inputs as digital inputs](image)

**Fig. 238: Use of analog inputs as digital inputs**

The following measuring ranges can be configured [Chapter 1.5.3.7.4.1.7 “Parameterization” on page 2918 and Chapter 1.5.3.7.4.1.9 “Measuring ranges” on page 2930:

<table>
<thead>
<tr>
<th>Digital input</th>
<th>24 V</th>
<th>1 channel used</th>
</tr>
</thead>
</table>

The function of the LEDs is described under Diagnosis and displays / Displays [Chapter 1.5.3.7.4.1.8 “Diagnosis and state LEDs” on page 2924.

**Connection of analog output loads (Voltage)**

The following figure shows the connection of output loads to the analog output AO0. Proceed with the analog output AO1 in the same way.
The following measuring ranges can be configured "Parameterization" on page 2918 and "Measuring ranges" on page 2930.

<table>
<thead>
<tr>
<th>Voltage</th>
<th>-10 V...+10 V</th>
<th>Load ±10 mA max.</th>
<th>1 channel used</th>
</tr>
</thead>
</table>

The function of the LEDs is described under Diagnosis and displays / Displays "Diagnosis and state LEDs" on page 2924.

Unused analog outputs can be left open-circuited.

**Connection of analog output loads (Current)**

The following figure shows the connection of output loads to the analog output AO0. Proceed with the analog output AO1 in the same way.
Fig. 240: Connection of analog output loads (current)

The following measuring ranges can be configured \( \text{Chapter 1.5.3.7.4.1.7 "Parameterization" on page 2918 and Chapter 1.5.3.7.4.1.9 "Measuring ranges" on page 2930.} \\

<table>
<thead>
<tr>
<th>Current</th>
<th>Load 0...500 Ω</th>
<th>1 channel used</th>
</tr>
</thead>
<tbody>
<tr>
<td>0...20 mA</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4...20 mA</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The function of the LEDs is described under Diagnosis and displays / Displays \( \text{Chapter 1.5.3.7.4.1.8 "Diagnosis and state LEDs" on page 2924.} \\

Unused analog outputs can be left open-circuited.

Assignment of the Ethernet ports

The terminal unit for the communication interface module provides two Ethernet interfaces with the following pin assignment:

<table>
<thead>
<tr>
<th>Interface assignment RJ45 jack:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Interface Pin Signal Description</td>
</tr>
<tr>
<td>----------------------------------</td>
</tr>
<tr>
<td>Ethernet RJ45 1 TxD+ Transmit data +</td>
</tr>
<tr>
<td>2 TxD- Transmit data -</td>
</tr>
<tr>
<td>3 RxD+ Receive data +</td>
</tr>
<tr>
<td>4 NC not used</td>
</tr>
<tr>
<td>5 NC not used</td>
</tr>
<tr>
<td>6 RxD- Receive data -</td>
</tr>
<tr>
<td>7 NC not used</td>
</tr>
<tr>
<td>Interface</td>
</tr>
<tr>
<td>------------</td>
</tr>
<tr>
<td>8</td>
</tr>
<tr>
<td>Shield</td>
</tr>
</tbody>
</table>

For further information regarding wiring and cable types see chapter Ethernet Chapter 1.5.4.5.4.7 “Ethernet connection details” on page 3084.

### Internal data exchange

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Digital inputs (bytes)</td>
<td>3</td>
</tr>
<tr>
<td>Digital outputs (bytes)</td>
<td>3</td>
</tr>
<tr>
<td>Analog inputs (words)</td>
<td>4</td>
</tr>
<tr>
<td>Analog outputs (words)</td>
<td>2</td>
</tr>
<tr>
<td>Counter input data (words)</td>
<td>4</td>
</tr>
<tr>
<td>Counter output data (words)</td>
<td>8</td>
</tr>
</tbody>
</table>

### Addressing

The module reads the position of the rotary switches only during power-up, i.e. changes of the switch position during operation will have no effect until the next module initialization.

The IP address of the CI521-MODTCP Module can be set with the “ABB IP Configuration Tool” Chapter 1.5.6.2.2.4.2 “Configuration of the IP settings with IP configuration tool” on page 3267.

If the last byte of the IP is set to 0, the address switch will be used instead.

Address switch position 255 is mapped to fixed IP 192.168.0.254 independent of other stored settings. This is a backup so the module can always get a valid IP address and can be configured by the “ABB IP Configuration Tool”.

Address switch position 0 is mapped to last byte equal 1 and DHCP enabled.

The factory setting for the IP is 192.168.0.x (last byte is address switch).

### I/O configuration

The CI521-MODTCP stores configuration parameters (IP address configuration, module parameters).

The analog/digital I/O channels are configured via software.

Details about configuration are described in Parameterization Chapter 1.5.3.7.4.1.7 “Parameterization” on page 2918.
## Parameterization

### Parameters of the module

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>Internal value</th>
<th>Internal value, type</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Module ID</strong> 1)</td>
<td>Internal</td>
<td>7400</td>
<td>WORD</td>
<td>7000</td>
</tr>
<tr>
<td><strong>Ignore Module</strong></td>
<td>Internal</td>
<td>0</td>
<td>BYTE</td>
<td>0</td>
</tr>
<tr>
<td><strong>Parameter length</strong></td>
<td>Internal</td>
<td>63</td>
<td>BYTE</td>
<td>63</td>
</tr>
</tbody>
</table>
| **Error LED / Fail-safe function** see table Error LED / Failsafe function  
See Table 621 "Error LED / Failsafe function" on page 2919 | On     | 0              | BYTE                | 0       |
<p>|                                                | Off by E4 | 1              |                      |         |
|                                                | Off by E3  | 3              |                      |         |
|                                                | On + failsafe | 16            |                      |         |
|                                                | Off by E4 + failsafe | 17         |                      |         |
|                                                | Off by E3 + failsafe | 19       |                      |         |
| <strong>Master IP for Write restriction 4)</strong>         | No master IP | Master IP     | ARRAY[0..3] OF BYTE | 0,0,0,0 |
| <strong>Master IP for Write restriction 4)</strong>         | No master IP | Master IP     | ARRAY[0..3] OF BYTE | 0,0,0,0 |
| <strong>Master IP for Write restriction 4)</strong>         | No master IP | Master IP     | ARRAY[0..3] OF BYTE | 0,0,0,0 |
| <strong>Master IP for Write restriction 4)</strong>         | No master IP | Master IP     | ARRAY[0..3] OF BYTE | 0,0,0,0 |
| <strong>Master IP for Write restriction 4)</strong>         | No master IP | Master IP     | ARRAY[0..3] OF BYTE | 0,0,0,0 |
| <strong>Master IP for Write restriction 4)</strong>         | No master IP | Master IP     | ARRAY[0..3] OF BYTE | 0,0,0,0 |
| <strong>Master IP for Write restriction 4)</strong>         | No master IP | Master IP     | ARRAY[0..3] OF BYTE | 0,0,0,0 |
| <strong>Timeout for Bus supervision</strong>                | No supervision | 0              | BYTE                | No supervision |
|                                                | 10 ms timeout | 1             |                      |         |
|                                                | 20 ms timeout | 2             |                      |         |
| <strong>IO Mapping Structure 3)</strong>                    | Fixed Mapping | Dynamic Mapping |                   |         |
|                                                | 0              | 1              | BYTE                | 0       |
| <strong>Reserved</strong>                                   | Internal       | 0              | ARRAY[0..2] OF BYTE | 0,0,0   |</p>
<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>Internal value</th>
<th>Internal value, type</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>Check supply</td>
<td>off</td>
<td>0</td>
<td>BYTE</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>on</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Fast counter</td>
<td>0</td>
<td>0</td>
<td>BYTE</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>10 3)</td>
<td>10</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

1) With a faulty ID, the Modules reports a "parameter error" and does not perform cyclic process data transmission.

2) Counter operating modes, see description of the Fast Counter Chapter 1.5.5.1.10 “Fast counters” on page 3194.

3) Fixed Mapping means each module has its own Modbus registers for data transfer independent of the IO bus constellation. See Modbus TCP Registers description for details Chapter 1.5.5.3.1.2 “Modbus TCP registers” on page 3201.

Dynamic mapping means the structure of the IO Date is dependent on the I/O bus constellation. Each I/O bus expansion module starts directly after the module before on the next Word adress.

4) If none of the parameters is set all masters / clients in the network have read and write rights on the CI52x-MODTCP device and its connected expansion modules.

If at least one parameter is set only the configured masters / clients have write rights on the CI52x-MODTCP device, all other masters / clients still have read access to the CI52x-MODTCP device.

### Table 621: Error LED / Failsafe function

<table>
<thead>
<tr>
<th>Setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>On</td>
<td>Error LED (S-ERR) lights up at errors of all error classes, Failsafe-mode off</td>
</tr>
<tr>
<td>Off by E4</td>
<td>Error LED (S-ERR) lights up at errors of error classes E1, E2 and E3, Failsafe-mode off</td>
</tr>
<tr>
<td>Off by E3</td>
<td>Error LED (S-ERR) lights up at errors of error classes E1 and E2, Failsafe-mode off</td>
</tr>
<tr>
<td>On +Failsafe</td>
<td>Error LED (S-ERR) lights up at errors of all error classes, Failsafe-mode on *)</td>
</tr>
<tr>
<td>Off by E4 + Failsafe</td>
<td>Error LED (S-ERR) lights up at errors of error classes E1, E2 and E3, Failsafe-mode on *)</td>
</tr>
<tr>
<td>Off by E3 + Failsafe</td>
<td>Error LED (S-ERR) lights up at errors of error classes E1 and E2, Failsafe-mode on *)</td>
</tr>
</tbody>
</table>

*) The parameters Behaviour AO at comm. error and Behaviour DO at comm. error are only analyzed if the Failsafe-mode is ON.
Group parameters for the analog part

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>Internal value</th>
<th>Internal value, type</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>Analog data format</td>
<td>Standard Reserved</td>
<td>0 255</td>
<td>BYTE</td>
<td>0</td>
</tr>
<tr>
<td>Behaviour AO at comm. error *)</td>
<td>Off Last value Last value 5 s Last value 10 s Substitute value Substitute value 5 s Substitute value 10 s</td>
<td>0 1 6 11 2 7 12</td>
<td>BYTE</td>
<td>0</td>
</tr>
</tbody>
</table>

*) The parameter Behaviour AO at comm. error is only analyzed if the Failsafe-mode is ON.

Channel parameters for the analog inputs (4x)

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>Internal value</th>
<th>Internal value, type</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input 0, Channel configuration</td>
<td>Table Operating modes of the analog inputs Table 622 “Channel configuration” on page 2921</td>
<td>Table Operating modes of the analog inputs Table 622 “Channel configuration” on page 2921</td>
<td>BYTE</td>
<td>0</td>
</tr>
<tr>
<td>Input 0, Check channel</td>
<td>Table Channel monitoring Table 623 “Channel monitoring” on page 2921</td>
<td>Table Channel monitoring Table 623 “Channel monitoring” on page 2921</td>
<td>BYTE</td>
<td>0</td>
</tr>
<tr>
<td>Input 3, Channel configuration</td>
<td>Table Operating modes of the analog inputs Table 622 “Channel configuration” on page 2921</td>
<td>Table Operating modes of the analog inputs Table 622 “Channel configuration” on page 2921</td>
<td>BYTE</td>
<td>0</td>
</tr>
<tr>
<td>Input 3, Check channel</td>
<td>Table Channel monitoring Table 623 “Channel monitoring” on page 2921</td>
<td>Table Channel monitoring Table 623 “Channel monitoring” on page 2921</td>
<td>BYTE</td>
<td>0</td>
</tr>
</tbody>
</table>
### Table 622: Channel configuration

<table>
<thead>
<tr>
<th>Internal value</th>
<th>Operating modes of the analog inputs, individually configurable</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 (default)</td>
<td>Not used</td>
</tr>
<tr>
<td>1</td>
<td>0...10 V</td>
</tr>
<tr>
<td>2</td>
<td>Digital input</td>
</tr>
<tr>
<td>3</td>
<td>0...20 mA</td>
</tr>
<tr>
<td>4</td>
<td>4...20 mA</td>
</tr>
<tr>
<td>5</td>
<td>-10 V...+10 V</td>
</tr>
<tr>
<td>8</td>
<td>2-wire Pt100 -50...+400 °C</td>
</tr>
<tr>
<td>9</td>
<td>3-wire Pt100 -50...+400 °C *)</td>
</tr>
<tr>
<td>10</td>
<td>0...10 V (voltage diff.) *)</td>
</tr>
<tr>
<td>11</td>
<td>-10 V...+10 V (voltage diff.) *)</td>
</tr>
<tr>
<td>14</td>
<td>2-wire Pt100 -50...+70 °C</td>
</tr>
<tr>
<td>15</td>
<td>3-wire Pt100 -50...+70 °C *)</td>
</tr>
<tr>
<td>16</td>
<td>2-wire Pt1000 -50...+400 °C</td>
</tr>
<tr>
<td>17</td>
<td>3-wire Pt1000 -50...+400 °C *)</td>
</tr>
<tr>
<td>18</td>
<td>2-wire Ni1000 -50...+150 °C</td>
</tr>
<tr>
<td>19</td>
<td>3-wire Ni1000 -50...+150 °C *)</td>
</tr>
</tbody>
</table>

*) In the operating modes with 3-wire configuration or with differential inputs, two adjacent analog inputs belong together (e.g. the channels 0 and 1). In these cases, both channels are configured in the desired operating mode. The lower address must be the even address (channel 0). The next higher address must be the odd address (channel 1). The converted analog value is available at the higher address (channel 1).

### Table 623: Channel monitoring

<table>
<thead>
<tr>
<th>Internal Value</th>
<th>Check Channel</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 (default)</td>
<td>Plausibility, cut wire, short circuit</td>
</tr>
<tr>
<td>3</td>
<td>Not used</td>
</tr>
</tbody>
</table>

### Channel parameters for the analog outputs (2x)

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>Internal value</th>
<th>Internal value, type</th>
<th>Default</th>
</tr>
</thead>
</table>
| Output 0, Channel configuration | Table Operating modes of the analog outputs  
|                              | Ÿ Table 624 “Channel configuration” on page 2922 |                | BYTE                | 0       |
| Output 0, Check channel     | Table Channel monitoring  
<p>|                              | Ÿ Table 625 “Channel monitoring” on page 2922   |                | BYTE                | 0       |</p>
<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>Internal value, type</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output 0, Substitute value</td>
<td>Table Substitute value Table 626 “Substitute value” on page 2922</td>
<td>WORD</td>
<td>0</td>
</tr>
<tr>
<td>Output 1, Channel configuration</td>
<td>Table Operating modes of the analog outputs Table 624 “Channel configuration” on page 2922</td>
<td>BYTE</td>
<td>0</td>
</tr>
<tr>
<td>Output 1, Check channel</td>
<td>Table Channel monitoring Table 625 “Channel monitoring” on page 2922</td>
<td>BYTE</td>
<td>0</td>
</tr>
<tr>
<td>Output 1, Substitute value</td>
<td>Table Substitute value Table 626 “Substitute value” on page 2922</td>
<td>WORD</td>
<td>0</td>
</tr>
</tbody>
</table>

**Table 624: Channel configuration**

<table>
<thead>
<tr>
<th>Internal value</th>
<th>Operating modes of the analog outputs, individually configurable</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 (default)</td>
<td>Not used</td>
</tr>
<tr>
<td>128</td>
<td>-10 V...+10 V</td>
</tr>
<tr>
<td>129</td>
<td>0...20 mA</td>
</tr>
<tr>
<td>130</td>
<td>4...20 mA</td>
</tr>
</tbody>
</table>

**Table 625: Channel monitoring**

<table>
<thead>
<tr>
<th>Internal value</th>
<th>Check channel</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Plausibility, cut wire, short circuit</td>
</tr>
<tr>
<td>3</td>
<td>None</td>
</tr>
</tbody>
</table>

**Table 626: Substitute value**

<table>
<thead>
<tr>
<th>Intended behaviour of output channel when the control system stops</th>
<th>Required setting of the module parameter &quot;Behaviour of outputs in case of a communication error&quot;</th>
<th>Required setting of the channel parameter &quot;Substitute value&quot;</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output OFF</td>
<td>Off</td>
<td>0</td>
</tr>
<tr>
<td>Last value infinite</td>
<td>Last value</td>
<td>0</td>
</tr>
<tr>
<td>Last value for 5 s and then turn off</td>
<td>Last value 5 sec</td>
<td>0</td>
</tr>
<tr>
<td>Last value for 10 s and then turn off</td>
<td>Last value 10 sec</td>
<td>0</td>
</tr>
<tr>
<td>Substitute value infinite</td>
<td>Substitute value</td>
<td>Depending on configuration</td>
</tr>
</tbody>
</table>
Intended behaviour of output channel when the control system stops

| Substitute value for 5 s and then turn off | Substitute value 5 sec | Depending on configuration |
| Substitute value for 10 s and then turn off | Substitute value 10 sec | Depending on configuration |

Required setting of the module parameter "Behaviour of outputs in case of a communication error"

Required setting of the channel parameter "Substitute value"

### Group parameters for the digital part

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>Internal value</th>
<th>Internal value, type</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input delay</td>
<td>0.1 ms</td>
<td>0</td>
<td>BYTE</td>
<td>0.1 ms</td>
</tr>
<tr>
<td></td>
<td>1 ms</td>
<td>1</td>
<td></td>
<td>0x00</td>
</tr>
<tr>
<td></td>
<td>8 ms</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>32 ms</td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Detect short circuit at outputs</td>
<td>Off</td>
<td>0</td>
<td>BYTE</td>
<td>On</td>
</tr>
<tr>
<td></td>
<td>On</td>
<td>1</td>
<td></td>
<td>0x01</td>
</tr>
<tr>
<td>Behaviour DO at comm. error 1)</td>
<td>Off</td>
<td>0</td>
<td>BYTE</td>
<td>Off</td>
</tr>
<tr>
<td></td>
<td>Last value</td>
<td>1</td>
<td></td>
<td>0x00</td>
</tr>
<tr>
<td></td>
<td>Last value 5 sec</td>
<td>6</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Last value 10 sec</td>
<td>11</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Substitute value</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Substitute value 5 sec</td>
<td>7</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Substitute value 10 sec</td>
<td>12</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Substitute value at output</td>
<td>0 ... 255</td>
<td>00h ... FFh</td>
<td>BYTE</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0x00000</td>
</tr>
<tr>
<td>Detect voltage overflow at outputs 2)</td>
<td>Off</td>
<td>0</td>
<td>BYTE</td>
<td>On</td>
</tr>
<tr>
<td></td>
<td>On</td>
<td>1</td>
<td></td>
<td>0x01</td>
</tr>
</tbody>
</table>

1) The parameters Behaviour DO at comm. error is only analyzed if the Failsafe-mode is ON.

2) The state "externally voltage detected" appears, if the output of a channel DC0..DC7 should be switched on while an externally voltage is connected. The state "externally voltage detected" is only analyzed if the Failsafe-mode is ON. In this case the start up is disabled, as long as the externally voltage is connected. The monitoring of this state and the resulting diagnosis message can be disabled by setting the parameters to "OFF".
## Diagnosis and state LEDs

### Structure of the diagnosis block

<table>
<thead>
<tr>
<th>Byte Number</th>
<th>Description</th>
<th>Possible Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Diagnosis Byte, slot number</td>
<td>31 = CI521-MODTCP (e.g. error at integrated 8 DI / 8 DO)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 = 1st connected S500 I/O Module</td>
</tr>
<tr>
<td></td>
<td></td>
<td>...</td>
</tr>
<tr>
<td></td>
<td></td>
<td>10 = 10th connected S500 I/O Module</td>
</tr>
<tr>
<td>2</td>
<td>Diagnosis Byte, module number</td>
<td>According to the I/O bus specification passed on by modules to the fieldbus master</td>
</tr>
<tr>
<td>3</td>
<td>Diagnosis Byte, channel</td>
<td>According to the I/O bus specification passed on by modules to the fieldbus master</td>
</tr>
<tr>
<td>4</td>
<td>Diagnosis Byte, error code</td>
<td>According to the I/O bus specification</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Bit 7 and bit 6, coded error class</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 = E1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 = E2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2 = E3</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3 = E4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Bit 0 to bit 5, coded error description</td>
</tr>
<tr>
<td>5</td>
<td>Diagnosis Byte, flags</td>
<td>According to the I/O bus specification</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Bit 7: 1 = coming error</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Bit 6: 1 = leaving error</td>
</tr>
</tbody>
</table>

In cases of short circuit or overload, the digital outputs are turned off. The modules performs reactivation automatically. Thus an acknowledgement of the errors is not necessary. The error message is stored via the LED.

> *For diagnosis firmware version ≥ 3.2.6 is required.*
| E1..E4 | d1 | d2 | d3 | d4 | Identifier 000..063 | AC500-Display | <= Display in
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Class</td>
<td>Comp</td>
<td>Dev</td>
<td>Mod</td>
<td>Ch</td>
<td>Err</td>
<td>PS501 PLC Browser</td>
<td></td>
</tr>
<tr>
<td>Byte 4</td>
<td>Byte 1</td>
<td>Byte 2</td>
<td>Byte 3</td>
<td>Byte 4 Bit 0..5</td>
<td>PNIO diagnosis block</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Class</th>
<th>Interface</th>
<th>Device</th>
<th>Module</th>
<th>Channel</th>
<th>Error-Identifier</th>
<th>Error message</th>
<th>Remedy</th>
</tr>
</thead>
<tbody>
<tr>
<td>1)</td>
<td>2)</td>
<td>3)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Module errors**

<p>| 3 | - | 31 | 31 | 31 | 19 | Checksum error in the I/O module | Replace I/O module |
| 3 | - | 31 | 31 | 31 | 3  | Timeout in the I/O module      |                   |
| 3 | - | 31 | 31 | 31 | 40 | Different hard-/firmware versions in the module |                   |
| 3 | - | 31 | 31 | 31 | 43 | Internal error in the module    |                   |
| 3 | - | 31 | 31 | 31 | 36 | Internal data exchange failure  |                   |
| 3 | - | 31 | 31 | 31 | 9  | Overflow diagnosis buffer       | Restart           |
| 3 | - | 31 | 31 | 31 | 26 | Parameter error                 | Check Master      |
| 3 | - | 31 | 31 | 31 | 11 | Process voltage UP too low      | Check process supply voltage |
| 3 | - | 31 | 31 | 31 | 45 | No process voltage UP           | Check process supply voltage |
| 3 | - | 31 | 31 | 31 | 17 | No communication with I/O module| Replace I/O module |
| 3 | - | 31/1...10 | 31 | 31 | 32 | Wrong I/O module type on socket | Replace I/O module / Check configuration |
| 4 | - | 1...10 | 31 | 31 | 31 | At least one module does not support failsafe function | Check modules and parameterization |</p>
<table>
<thead>
<tr>
<th>Class</th>
<th>Interface</th>
<th>Device</th>
<th>Module</th>
<th>Channel</th>
<th>Error Identifier</th>
<th>Error message</th>
<th>Remedy</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>-</td>
<td>1...10</td>
<td>31</td>
<td>5</td>
<td>8</td>
<td>I/O module removed from hot-swap terminal unit or defective module on hot-swap terminal unit</td>
<td>Plug I/O module, replace I/O module</td>
</tr>
<tr>
<td>4</td>
<td>-</td>
<td>1...10</td>
<td>31</td>
<td>5</td>
<td>28</td>
<td>Wrong I/O module plugged on hot-swap terminal unit</td>
<td>Remove wrong I/O module and plug projected I/O module</td>
</tr>
<tr>
<td>4</td>
<td>-</td>
<td>1...10</td>
<td>31</td>
<td>5</td>
<td>42</td>
<td>No communication with I/O module on hot-swap terminal unit</td>
<td>Replace I/O module</td>
</tr>
<tr>
<td>4</td>
<td>-</td>
<td>1...10</td>
<td>31</td>
<td>5</td>
<td>54</td>
<td>I/O module does not support hot swap</td>
<td>Power off system and replace I/O module</td>
</tr>
<tr>
<td>4</td>
<td>-</td>
<td>1...10</td>
<td>31</td>
<td>6</td>
<td>42</td>
<td>No communication with hot-swap terminal unit</td>
<td>Restart, if error persists replace terminal unit</td>
</tr>
<tr>
<td>4</td>
<td>-</td>
<td>31</td>
<td>31</td>
<td>31</td>
<td>46</td>
<td>Voltage feedback on activated digital outputs DO0...DO7 on UP3</td>
<td>Check terminals</td>
</tr>
<tr>
<td>4</td>
<td>-</td>
<td>31/1...10</td>
<td>31</td>
<td>31</td>
<td>34</td>
<td>No response during initialization of the I/O module</td>
<td>Replace I/O module</td>
</tr>
<tr>
<td>Class</td>
<td>Interface</td>
<td>Device</td>
<td>Module</td>
<td>Channel</td>
<td>Error-Identifier</td>
<td>Error message</td>
<td>Remedy</td>
</tr>
<tr>
<td>-------</td>
<td>-----------</td>
<td>--------</td>
<td>--------</td>
<td>---------</td>
<td>-----------------</td>
<td>---------------</td>
<td>--------</td>
</tr>
<tr>
<td>4</td>
<td>-</td>
<td>31</td>
<td>31</td>
<td>31</td>
<td>11</td>
<td>Process voltage UP3 too low</td>
<td>Check process supply voltage</td>
</tr>
<tr>
<td>4</td>
<td>-</td>
<td>31</td>
<td>31</td>
<td>31</td>
<td>45</td>
<td>No process voltage UP3</td>
<td>Check process supply voltage</td>
</tr>
<tr>
<td>4</td>
<td>-</td>
<td>31</td>
<td>31</td>
<td>31</td>
<td>10</td>
<td>Voltage overflow on outputs (above UP3 level)</td>
<td>Check terminals/ check process supply voltage</td>
</tr>
</tbody>
</table>

Channel error digital

| 4     | -         | 31     | 2      | 0...7   | 46              | Externally voltage detected at digital output DO0...DO7 | Check terminals |
| 4     | -         | 31     | 2      | 0...7   | 47              | Short circuit at digital output | Check terminals |

Channel error analog

| 4     | -         | 31     | 1      | 0..3    | 48              | Analog value overflow or broken wire at an analog input | Check value or check terminals |
| 4     | -         | 31     | 1      | 0..3    | 7               | Analog value underflow at an analog input | Check value |
| 4     | -         | 31     | 1      | 0..3    | 47              | Short circuit at an analog input | Check terminals |
| 4     | -         | 31     | 3      | 0..1    | 4               | Analog value overflow at an analog output | Check output value |
| 4     | -         | 31     | 3      | 0..1    | 7               | Analog value underflow at an analog output | Check output value |

Remarks:

Remarks:
In AC500 the following interface identifier applies:
"-" = Diagnosis via bus-specific function blocks; 0 ... 4 or 10 = Position of the Communication Module; 14 = I/O bus; 31 = Module itself
The identifier is not contained in the CI521-MODTCP diagnosis block.

With "Device" the following allocation applies: 31 = Module itself; 1..10 = Expansion module

With "Module" the following allocation applies:
31 = Module itself
Module type (1 = AI, 2 = DO, 3 = AO)

This message appears, if externally voltages at one or more terminals DO0...DO7 cause that other digital outputs are supplied through that voltage. Chapter 1.5.3.7.4.1.3 "Electrical connection" on page 2900. All outputs of the apply digital output groups will be turned off for 5 seconds. The diagnosis message appears for the whole output group.

The voltage on digital outputs DO0...DO7 has overrun the process supply voltage. Chapter 1.5.3.7.4.1.3 "Electrical connection" on page 2900. Diagnosis message appears for the whole module.

This message appears, if the output of a channel DO0...DO7 should be switched on while an externally voltage is connected. In this case the start up is disabled, as long as the externally voltage is connected. Otherwise this could produce reverse voltage from this output to other digital outputs. This diagnosis message appears per channel.

Short circuit: After a detected short circuit, the output is deactivated for 100ms. Then a new start up will be executed. This diagnosis message appears per channel.

In case of an I/O module doesn't support hot swapping, do not perform any hot-swap operations (also not on any other terminal units (slots)) as modules may be damaged or I/O bus communication may be disturbed.

Diagnosis for hot swap available as of version index F0.

---

State LEDs

The LEDs are located at the front of module. There are 2 different groups:
- The 5 system LEDs (PWR, STA1 ETH, STA2 ETH, S-ERR and I/O-Bus) show the operation state of the module and display possible errors.
- The 27 process LEDs (UP, UP3, inputs, outputs, CH-ERR1 to CH-ERR3) show the process supply voltage and the states of the inputs and outputs and display possible errors.

Table 627: States of the 5 system LEDs

<table>
<thead>
<tr>
<th>LED</th>
<th>Color</th>
<th>OFF</th>
<th>ON</th>
<th>Flashing</th>
</tr>
</thead>
<tbody>
<tr>
<td>PWR/RUN</td>
<td>Green</td>
<td>Process supply voltage missing</td>
<td>Internal supply voltage OK, module ready for communication with IO Controller</td>
<td>Start-up / preparing communication</td>
</tr>
<tr>
<td></td>
<td>Yellow</td>
<td>---</td>
<td>----</td>
<td>---</td>
</tr>
<tr>
<td>STA1 ETH (System LED &quot;BF&quot;)</td>
<td>Green</td>
<td>---</td>
<td>Device configured, cyclic data exchange running</td>
<td>Device configured, acyclic data exchange running</td>
</tr>
<tr>
<td>LED</td>
<td>Color</td>
<td>OFF</td>
<td>ON</td>
<td>Flashing</td>
</tr>
<tr>
<td>-------------------</td>
<td>-------</td>
<td>--------------------------</td>
<td>-----------------------------------------</td>
<td>-------------------------------</td>
</tr>
<tr>
<td>STA2 ETH</td>
<td>Green</td>
<td>Device has valid parameters</td>
<td>Device is running parameterization sequenze</td>
<td>Device has no parameters</td>
</tr>
<tr>
<td>(System LED &quot;SF&quot;)</td>
<td>Red</td>
<td>---</td>
<td>---</td>
<td>Device has invalid parameters</td>
</tr>
<tr>
<td>S-ERR</td>
<td>Red</td>
<td>No error</td>
<td>Internal error</td>
<td>--</td>
</tr>
<tr>
<td>I/O-Bus</td>
<td>Green</td>
<td>No expansion modules connected or communication error</td>
<td>Expansion modules connected and operational</td>
<td>---</td>
</tr>
<tr>
<td>ETH1</td>
<td>Green</td>
<td>No connection at Ethernet interface</td>
<td>Connected to Ethernet interface</td>
<td>---</td>
</tr>
<tr>
<td>Yellow</td>
<td>---</td>
<td>Device is transmitting telegrams</td>
<td>Device is transmitting telegrams</td>
<td>---</td>
</tr>
<tr>
<td>ETH2</td>
<td>Green</td>
<td>No connection at Ethernet interface</td>
<td>Connected to Ethernet interface</td>
<td>---</td>
</tr>
<tr>
<td>Yellow</td>
<td>---</td>
<td>Device is transmitting telegrams</td>
<td>Device is transmitting telegrams</td>
<td>---</td>
</tr>
</tbody>
</table>

*Table 628: States of the 27 process LEDs*

<table>
<thead>
<tr>
<th>LED</th>
<th>Color</th>
<th>OFF</th>
<th>ON</th>
<th>Flashing</th>
</tr>
</thead>
<tbody>
<tr>
<td>AI0 to AI3</td>
<td>Yellow</td>
<td>Input is OFF</td>
<td>Input is ON (brightness depends on the value of the analog signal)</td>
<td>--</td>
</tr>
<tr>
<td>AO0 to AO1</td>
<td>Yellow</td>
<td>Output is OFF</td>
<td>Output is ON (brightness depends on the value of the analog signal)</td>
<td>--</td>
</tr>
<tr>
<td>DI0 to DI7</td>
<td>Yellow</td>
<td>Input is OFF</td>
<td>Input is ON (the input voltage is even displayed if the supply voltage is OFF)</td>
<td>--</td>
</tr>
<tr>
<td>DO0 to DO7</td>
<td>Yellow</td>
<td>Output is OFF</td>
<td>Output is ON</td>
<td>--</td>
</tr>
<tr>
<td>UP</td>
<td>Green</td>
<td>Process supply voltage missing</td>
<td>Process supply voltage OK and initialization finished</td>
<td>--</td>
</tr>
</tbody>
</table>
### Measuring ranges

#### Input ranges voltage, current and digital input

<table>
<thead>
<tr>
<th>Range</th>
<th>0...10 V</th>
<th>-10...+10 V</th>
<th>0...20 mA</th>
<th>4...20 mA</th>
<th>Digital input</th>
<th>Digital value</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Overflow</strong></td>
<td>&gt;11.7589</td>
<td>&gt;11.7589</td>
<td>&gt;23.5178</td>
<td>&gt;22.8142</td>
<td></td>
<td>32767</td>
</tr>
<tr>
<td><strong>Measured value too high</strong></td>
<td>11.7589</td>
<td>11.7589</td>
<td>23.5178</td>
<td>22.8142</td>
<td></td>
<td>32511</td>
</tr>
<tr>
<td></td>
<td>10.0004</td>
<td>10.0004</td>
<td>20.0007</td>
<td>20.0006</td>
<td></td>
<td>27649</td>
</tr>
<tr>
<td><strong>Normal range</strong></td>
<td>10.0000</td>
<td>10.0000</td>
<td>20.0000</td>
<td>20.0000</td>
<td>On</td>
<td>27648</td>
</tr>
<tr>
<td></td>
<td>0.0004</td>
<td>0.0004</td>
<td>0.0007</td>
<td>4.0006</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td><strong>Underflow</strong></td>
<td>&lt;0.0000</td>
<td>&lt;11.7589</td>
<td>&lt;0.0000</td>
<td>&lt;0.0000</td>
<td></td>
<td>-32768</td>
</tr>
</tbody>
</table>

The represented resolution corresponds to 16 bits.

#### Input ranges resistance temperature detector

<table>
<thead>
<tr>
<th>Range</th>
<th>Pt100 / Pt1000 -50...70 °C</th>
<th>Pt100 / Pt1000 -50...400 °C</th>
<th>Ni1000 -50...150 °C</th>
<th>Digital value</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Overflow</strong></td>
<td>&gt; 80.0 °C</td>
<td>&gt; 450.0 °C</td>
<td>&gt; 160.0 °C</td>
<td>32767</td>
</tr>
<tr>
<td><strong>Measured value too high</strong></td>
<td>450.0 °C</td>
<td>400.1 °C</td>
<td>4500</td>
<td>1194</td>
</tr>
</tbody>
</table>

The represented resolution corresponds to 16 bits.
<table>
<thead>
<tr>
<th>Range</th>
<th>Pt100 / Pt1000 -50...70 °C</th>
<th>Pt100 / Pt1000 -50...400 °C</th>
<th>Ni1000 -50...150 °C</th>
<th>Digital value</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>160.0 °C</td>
<td>150.1 °C</td>
<td>1600</td>
</tr>
<tr>
<td>80.0 °C</td>
<td></td>
<td></td>
<td></td>
<td>800</td>
</tr>
<tr>
<td>70.1 °C</td>
<td></td>
<td></td>
<td></td>
<td>701</td>
</tr>
<tr>
<td>Normal range</td>
<td>70.0 °C</td>
<td>400.0 °C</td>
<td>150.0 °C</td>
<td>4000</td>
</tr>
<tr>
<td></td>
<td>:</td>
<td>:</td>
<td>:</td>
<td>1500</td>
</tr>
<tr>
<td></td>
<td>:</td>
<td>:</td>
<td>:</td>
<td>700</td>
</tr>
<tr>
<td>Normal range</td>
<td>-0.1 °C</td>
<td>-0.1 °C</td>
<td>-0.1 °C</td>
<td>-1</td>
</tr>
<tr>
<td></td>
<td>:</td>
<td>:</td>
<td>:</td>
<td>-500</td>
</tr>
<tr>
<td>Measured value too low</td>
<td>-50.1 °C</td>
<td>-50.1 °C</td>
<td>-50.1 °C</td>
<td>-501</td>
</tr>
<tr>
<td></td>
<td>:</td>
<td>:</td>
<td>:</td>
<td>-600</td>
</tr>
<tr>
<td>Underflow</td>
<td>&lt; -60.0 °C</td>
<td>&lt; -60.0 °C</td>
<td>&lt; -60.0 °C</td>
<td>-32768</td>
</tr>
</tbody>
</table>

**Output ranges voltage and current**

<table>
<thead>
<tr>
<th>Range</th>
<th>-10...+10 V</th>
<th>0...20 mA</th>
<th>4...20 mA</th>
<th>Digital value</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Decimal</td>
<td>Hex.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Overflow</td>
<td>0 V</td>
<td>0 mA</td>
<td>0 mA</td>
<td>&gt; 32511</td>
</tr>
<tr>
<td>Measured value too high</td>
<td>11.7589 V</td>
<td>23.5178 mA</td>
<td>22.8142 mA</td>
<td>32511</td>
</tr>
<tr>
<td></td>
<td>:</td>
<td>:</td>
<td>:</td>
<td>20.00006 mA</td>
</tr>
<tr>
<td>Normal range</td>
<td>10.0000 V</td>
<td>20.0000 mA</td>
<td>20.0000 mA</td>
<td>27648</td>
</tr>
<tr>
<td></td>
<td>:</td>
<td>:</td>
<td>:</td>
<td>4.0006 mA</td>
</tr>
<tr>
<td></td>
<td>0.0004 V</td>
<td>0.0007 mA</td>
<td>0 mA</td>
<td>-1</td>
</tr>
<tr>
<td></td>
<td>0.0000 V</td>
<td>0.0000 mA</td>
<td>4.0000 mA</td>
<td>-6912</td>
</tr>
<tr>
<td></td>
<td>-0.0004 V</td>
<td>0 mA</td>
<td>3.9994 mA</td>
<td>27648</td>
</tr>
<tr>
<td></td>
<td>-10.0000 V</td>
<td>0 mA</td>
<td>0 mA</td>
<td>27648</td>
</tr>
</tbody>
</table>

2020/12/10
### Technical data

The system data of AC500 and S500 \( \Rightarrow \) Chapter 1.5.4.5.1 “System data AC500” on page 3058 are valid for standard version.

The system data of AC500-XC \( \Rightarrow \) Chapter 1.5.4.6.1 “System data AC500-XC” on page 3099 are valid for the XC version.

Only additional details are therefore documented below.

The technical data are also valid for the XC version.

### Technical data of the module

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Process supply voltages UP/UP3</td>
<td></td>
</tr>
<tr>
<td>Rated value</td>
<td>24 VDC (for inputs and outputs)</td>
</tr>
<tr>
<td>Max. load for the terminals</td>
<td>10 A</td>
</tr>
<tr>
<td>Protection against reversed voltage</td>
<td>Yes</td>
</tr>
<tr>
<td>Rated protection fuse on UP/UP3</td>
<td>10 A fast</td>
</tr>
<tr>
<td>Electrical isolation</td>
<td>Ethernet interface against the rest of the module</td>
</tr>
<tr>
<td>Inrush current from UP (at power up)</td>
<td>On request</td>
</tr>
<tr>
<td>Current consumption via UP (normal operation)</td>
<td>0.2 A</td>
</tr>
<tr>
<td>Current consumption via UP3</td>
<td>0.06 A + 0.5 A max. per output</td>
</tr>
</tbody>
</table>
| Connections | Terminals 1.8 and 2.8 for +24 V (UP)  
Terminals 3.8 for +24 V (UP3)  
Terminals 1.9, 2.9 and 3.9 for 0 V (ZP) |
<p>| Max. power dissipation within the module | 6 W |
| Number of digital inputs | 8 |
| Number of digital outputs | 8 |
| Number of analog inputs | 4 |
| Number of analog outputs | 2 |
| Reference potential for all digital inputs and outputs | Minus pole of the supply voltage, signal name ZP |
| Ethernet | 10/100 base-TX, internal switch, 2 x RJ45 socket |</p>
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Setting of the IP address</td>
<td>With ABB IP config tool and 2 rotary switches at the front side of the module</td>
</tr>
<tr>
<td>Diagnose</td>
<td>See Diagnosis and Displays % Chapter 1.5.3.7.4.1.8 “Diagnosis and state LEDs” on page 2924</td>
</tr>
<tr>
<td>Operation and error displays</td>
<td>32 LEDs (totally)</td>
</tr>
<tr>
<td>Weight (without terminal unit)</td>
<td>Ca. 125 g</td>
</tr>
<tr>
<td>Mounting position</td>
<td>Horizontal or vertical with derating (output load reduced to 50 % at 40 °C per group)</td>
</tr>
<tr>
<td>Extended ambient temperature (XC version)</td>
<td>&gt; 60 °C on request</td>
</tr>
<tr>
<td>Cooling</td>
<td>The natural convection cooling must not be hindered by cable ducts or other parts in the switch-gear cabinet.</td>
</tr>
</tbody>
</table>

**NOTICE!**

**Attention:**

All I/O channels (digital and analog) are protected against reverse polarity, reverse supply, short circuit and continuous overvoltage up to 30 VDC.

### Technical data of the digital inputs

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of channels per module</td>
<td>8</td>
</tr>
<tr>
<td>Distribution of the channels into groups</td>
<td>1 group of 8 channels</td>
</tr>
<tr>
<td>Terminals of the channels DI0 to DI7</td>
<td>Terminals 2.0 to 2.7</td>
</tr>
<tr>
<td>Reference potential for all inputs</td>
<td>Terminals 1.9...3.9 (Minus pole of the supply voltage, signal name ZP)</td>
</tr>
<tr>
<td>Indication of the input signals</td>
<td>1 yellow LED per channel, the LED is ON when the input signal is high (signal 1)</td>
</tr>
<tr>
<td>Input type (according EN 61131-2)</td>
<td>Type 1</td>
</tr>
<tr>
<td>Input delay (0-&gt;1 or 1-&gt;0)</td>
<td>Typ. 0.1 ms, configurable from 0.1...32 ms</td>
</tr>
<tr>
<td>Input signal voltage</td>
<td>24 VDC</td>
</tr>
<tr>
<td>0-Signal</td>
<td>-3 V...+5 V</td>
</tr>
<tr>
<td>Undefined Signal</td>
<td>&gt; +5 V...&lt; +15 V</td>
</tr>
<tr>
<td>1-Signal</td>
<td>+15 V...+30 V</td>
</tr>
<tr>
<td>Ripple with signal 0</td>
<td>Within -3 V...+5 V</td>
</tr>
<tr>
<td>Ripple with signal 1</td>
<td>Within +15 V...+30 V</td>
</tr>
<tr>
<td>Input current per channel</td>
<td></td>
</tr>
<tr>
<td>Input voltage +24 V</td>
<td>Typ. 5 mA</td>
</tr>
<tr>
<td>Input voltage +5 V</td>
<td>&gt; 1 mA</td>
</tr>
<tr>
<td>Input voltage +15 V</td>
<td>&gt; 2 mA</td>
</tr>
<tr>
<td>Input voltage +30 V</td>
<td>&lt; 8 mA</td>
</tr>
<tr>
<td>Max. cable length</td>
<td></td>
</tr>
</tbody>
</table>
### Technical data of the digital outputs

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of channels per module</td>
<td>8</td>
</tr>
<tr>
<td>Distribution of the channels into groups</td>
<td>1 group of 8 channels</td>
</tr>
<tr>
<td>Terminals of the channels DO0 to DO7</td>
<td>Terminals 3.0 to 3.7</td>
</tr>
<tr>
<td>Reference potential for all outputs</td>
<td>Terminals 1.9...3.9 (minus pole of the supply voltage, signal name ZP)</td>
</tr>
<tr>
<td>Common power supply voltage</td>
<td>For all outputs terminal 3.8 (plus pole of the supply voltage, signal name UP3)</td>
</tr>
<tr>
<td>Output voltage for signal 1</td>
<td>UP3 (-0.8 V)</td>
</tr>
<tr>
<td>Output delay (0-&gt;1 or 1-&gt;0)</td>
<td>On request</td>
</tr>
<tr>
<td>Output current</td>
<td></td>
</tr>
<tr>
<td>Rated value per channel</td>
<td>500 mA at UP3 = 24 V</td>
</tr>
<tr>
<td>Max. value (all channels together)</td>
<td>4 A</td>
</tr>
<tr>
<td>Leakage current with signal 0</td>
<td>&lt; 0.5 mA</td>
</tr>
<tr>
<td>Fuse for UP3</td>
<td>10 A fast</td>
</tr>
<tr>
<td>Demagnetization with inductive DC load</td>
<td>Via internal varistors (see figure below this table)</td>
</tr>
<tr>
<td>Output switching frequency</td>
<td></td>
</tr>
<tr>
<td>With resistive load</td>
<td>On request</td>
</tr>
<tr>
<td>With inductive loads</td>
<td>Max. 0.5 Hz</td>
</tr>
<tr>
<td>With lamp loads</td>
<td>11 Hz max. at 5 W max.</td>
</tr>
<tr>
<td>Short-circuit-proof / overload-proof</td>
<td>Yes</td>
</tr>
<tr>
<td>Overload message (I &gt; 0.7 A)</td>
<td>Yes, after ca. 100 ms</td>
</tr>
<tr>
<td>Output current limitation</td>
<td>Yes, automatic reactivation after short circuit/ overload</td>
</tr>
<tr>
<td>Resistance to feedback against 24 V signals</td>
<td>Yes (software-controlled supervision)</td>
</tr>
<tr>
<td>Max. cable length</td>
<td></td>
</tr>
<tr>
<td>Shielded</td>
<td>1000 m</td>
</tr>
<tr>
<td>Unshielded</td>
<td>600 m</td>
</tr>
</tbody>
</table>

The following drawing shows the circuitry of a digital input/output with the varistors for demagnetization when inductive loads are switched off.
Fig. 241: Digital input/output (circuit diagram)
1 Digital Output
2 Varistors for demagnetization when inductive loads are turned off

Technical data of the analog inputs

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of channels per module</td>
<td>4</td>
</tr>
<tr>
<td>Distribution of channels into groups</td>
<td>1 group with 4 channels</td>
</tr>
<tr>
<td>Connection if channels AI0+ to AI3+</td>
<td>Terminals 1.0 to 1.3</td>
</tr>
<tr>
<td>Reference potential for AI0+ to AI3+</td>
<td>Terminal 1.4 (AI-) for voltage and RTD measurement</td>
</tr>
<tr>
<td></td>
<td>Terminal 1.9, 2.9 and 3.9 for current measurement</td>
</tr>
<tr>
<td>Input type</td>
<td>Unipolar</td>
</tr>
<tr>
<td></td>
<td>Voltage 0 ... 10 V, current or Pt100/Pt1000/Ni1000</td>
</tr>
<tr>
<td></td>
<td>Bipolar</td>
</tr>
<tr>
<td></td>
<td>Voltage -10 ... +10 V</td>
</tr>
<tr>
<td>Electrical isolation</td>
<td>Against Ethernet network</td>
</tr>
<tr>
<td>Configurability</td>
<td>0...10 V, -10...+10 V, 0/4...20 mA, Pt100/1000, Ni1000 (each input can be configured individually)</td>
</tr>
<tr>
<td>Channel input resistance</td>
<td>Voltage: &gt; 100 kΩ</td>
</tr>
<tr>
<td></td>
<td>Current: ca. 330 Ω</td>
</tr>
<tr>
<td>Time constant of the input filter</td>
<td>Voltage: 100 μs</td>
</tr>
<tr>
<td></td>
<td>Current: 100 μs</td>
</tr>
<tr>
<td>Indication of the input signals</td>
<td>1 LED per channel (brightness depends on the value of the analog signal)</td>
</tr>
<tr>
<td>Conversion cycle</td>
<td>1 ms (for 4 inputs + 2 outputs); with RTDs Pt/Ni... 1 s</td>
</tr>
<tr>
<td>Resolution</td>
<td>Range 0...10 V: 12 bits</td>
</tr>
<tr>
<td></td>
<td>Range -10...+10 V: 12 bits + sign</td>
</tr>
<tr>
<td></td>
<td>Range 0...20 mA: 12 bits</td>
</tr>
<tr>
<td></td>
<td>Range 4...20 mA: 12 bits</td>
</tr>
<tr>
<td></td>
<td>Range RTD (Pt100, PT1000, Ni1000): 0.1 °C</td>
</tr>
<tr>
<td>Conversion error of the analog values caused by non-linearity, adjustment error at factory and resolution within the normal range</td>
<td>Typ. 0.5 %, max. 1 %</td>
</tr>
</tbody>
</table>
### Relationship between input signal and hex code

The relationship between input signal and hex code is specified in Chapters 1.5.3.7.4.1.9.1 "Input ranges voltage, current and digital input" on page 2930 and 1.5.3.7.4.1.9.2 "Input ranges resistance temperature detector" on page 2930.

### Unused inputs

Unused inputs are configured as "unused" (default value).

### Overvoltage protection

Yes.

---

### Technical data of the analog inputs if used as digital inputs

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of channels per module</td>
<td>Max. 4</td>
</tr>
<tr>
<td>Distribution of channels into groups</td>
<td>1 group of 4 channels</td>
</tr>
<tr>
<td>Connections of the channels AI0+ to AI3+</td>
<td>Terminals 1.0 to 1.3</td>
</tr>
<tr>
<td>Reference potential for the inputs</td>
<td>Terminals 1.9, 2.9 and 3.9 (ZP)</td>
</tr>
<tr>
<td>Indication of the input signals</td>
<td>1 LED per channel</td>
</tr>
<tr>
<td>Input signal voltage</td>
<td>24 VDC</td>
</tr>
<tr>
<td>Signal 0</td>
<td>-30 V...+5 V</td>
</tr>
<tr>
<td>Undefined signal</td>
<td>+5 V ... +13 V</td>
</tr>
<tr>
<td>Signal 1</td>
<td>+13 V...+30 V</td>
</tr>
<tr>
<td>Input current per channel</td>
<td></td>
</tr>
<tr>
<td>Input voltage +24 V</td>
<td>Typ. 7 mA</td>
</tr>
<tr>
<td>Input voltage +5 V</td>
<td>Typ. 1.4 mA</td>
</tr>
<tr>
<td>Input voltage +15 V</td>
<td>Typ. 3.7 mA</td>
</tr>
<tr>
<td>Input voltage +30 V</td>
<td>&lt; 9 mA</td>
</tr>
<tr>
<td>Input resistance</td>
<td>Ca. 3.5 kΩ</td>
</tr>
</tbody>
</table>

---

### Technical data of the analog outputs

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of channels per module</td>
<td>2</td>
</tr>
<tr>
<td>Distribution of channels into groups</td>
<td>1 group for 2 channels</td>
</tr>
<tr>
<td>Connection of the channels AO0+...AO1+</td>
<td>Terminals 1.5...1.6</td>
</tr>
<tr>
<td>Reference potential for AO0+ to AO1+</td>
<td>Terminal 1.7 (AO-) for voltage outputTerminal 1.9, 2.9 and 3.9 for current output</td>
</tr>
<tr>
<td>Output type</td>
<td></td>
</tr>
<tr>
<td>Unipolar</td>
<td>Current</td>
</tr>
<tr>
<td>Bipolar</td>
<td>Voltage</td>
</tr>
<tr>
<td>Electrical isolation</td>
<td>Against internal supply and other modules</td>
</tr>
<tr>
<td>Configurability</td>
<td>-10...+10 V, 0...20 mA, 4...20 mA (each output can be configured individually)</td>
</tr>
</tbody>
</table>
Technical data of the fast counter

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Used inputs</td>
<td>Terminal 2.0 (DI0), 2.1 (DI1)</td>
</tr>
<tr>
<td>Used outputs</td>
<td>Terminal 3.0 (DO0)</td>
</tr>
<tr>
<td>Counting frequency</td>
<td>Depending on operation mode:</td>
</tr>
<tr>
<td></td>
<td>Mode 1 - 6: max. 200 kHz</td>
</tr>
<tr>
<td></td>
<td>Mode 7: max. 50 kHz</td>
</tr>
<tr>
<td></td>
<td>Mode 9: max. 35 kHz</td>
</tr>
<tr>
<td></td>
<td>Mode 10: max. 20 kHz</td>
</tr>
<tr>
<td>Detailed description</td>
<td>See Fast Counter, Chapter 1.5.5.1.10 “Fast counters” on page 3194</td>
</tr>
<tr>
<td>Operating modes</td>
<td>See Operating modes</td>
</tr>
</tbody>
</table>

Ordering data

<table>
<thead>
<tr>
<th>Part no.</th>
<th>Description</th>
<th>Product life cycle phase *)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1SAP 222 100 R0001</td>
<td>CI521-MODTCP, Modbus TCP bus module, 4 AI, 2 AO, 8 DI and 8 DO</td>
<td>Active</td>
</tr>
<tr>
<td>1SAP 422 100 R0001</td>
<td>CI521-MODTCP-XC, Modbus TCP bus module, 4 AI, 2 AO, 8 DI and 8 DO, XC version</td>
<td>Active</td>
</tr>
</tbody>
</table>
*) For planning and commissioning of new installations use modules in Active status only.

CI522-MODTCP

- 8 digital inputs 24 VDC
- 8 digital outputs 24 VDC, 0.5 A max.
- 8 configurable digital inputs/outputs 24 VDC, 0.5 A max.
- Module-wise electrically isolated
- Fast Counter
- XC version for usage in extreme ambient conditions available

1 I/O bus
2 Allocation between terminal number and signal name
3 8 yellow LEDs to display the signal states of the digital configurable inputs/outputs (DC0 - DC7)
4 8 yellow LEDs to display the signal states of the digital inputs (DI8 - DI15)
5 8 yellow LEDs to display the signal states of the digital outputs (DO8 - DO15)
6 2 green LEDs to display the process supply voltage UP and UP3
7 3 red LEDs to display errors (CH-ERR1, CH-ERR2, CH-ERR3)
8 5 system LEDs: PWR/RUN, STA1 ETH, STA2 ETH, S-ERR, I/O-Bus
9 Label
10 2 rotary switches for setting the IP address
11 Ethernet interfaces (ETH1, ETH2) on the terminal unit
12 Terminal unit
13 DIN rail
\(\text{Sign for XC version}\)

### Intended purpose

Modbus TCP bus module CI522-MODTCP is used as decentralized I/O module in Modbus TCP networks. The network connection is performed via 2 RJ45 connectors which are integrated in the terminal unit. The bus module contains 24 I/O channels with the following properties:

- 8 digital configurable inputs/outputs in 1 group (1.0...1.7)
- 8 digital inputs 24 VDC in 1 group (2.0...2.7)
- 8 digital outputs 24 VDC in 1 group (3.0...3.7)

The inputs/outputs are electrically isolated from the Ethernet network. There is no potential separation between the channels. The configuration of the configurable digital inputs/outputs is performed by software.

For usage in extreme ambient conditions (e.g. wider temperature and humidity range), a special XC version of the device is available.

### Functionality

<table>
<thead>
<tr>
<th>Interface</th>
<th>Ethernet</th>
</tr>
</thead>
<tbody>
<tr>
<td>Protocol</td>
<td>Modbus TCP</td>
</tr>
<tr>
<td>Power supply</td>
<td>From the process supply voltage UP</td>
</tr>
<tr>
<td>Supply of the electronic circuitry of the I/O expansion modules attached</td>
<td>Through the expansion bus interface (I/O bus)</td>
</tr>
<tr>
<td>Rotary switches</td>
<td>for setting the last BYTE of the IP ADDRESS (00h to FFh)</td>
</tr>
<tr>
<td>Configurable digital inputs/outputs</td>
<td>8 (configurable via software)</td>
</tr>
<tr>
<td>Digital inputs</td>
<td>8 (24 VDC; delay time configurable via software)</td>
</tr>
<tr>
<td>Digital outputs</td>
<td>8 (24 VDC, 0.5 A max.)</td>
</tr>
<tr>
<td>LED displays</td>
<td>For system displays, signal states, errors and power supply</td>
</tr>
<tr>
<td>External supply voltage</td>
<td>Via terminals ZP, UP and UP3 (process supply voltage 24 VDC)</td>
</tr>
<tr>
<td>Required terminal unit</td>
<td>TU507 or TU508 (^{%}) Chapter 1.5.3.5.1 “TU507-ETH and TU508-ETH for Ethernet communication interface modules” on page 2307</td>
</tr>
</tbody>
</table>
Electrical connection

The Ethernet bus module CI522-MODTCP is plugged on the I/O terminal unit TU507-ETH. Properly seat the module and press until it locks in place. The terminal unit is mounted on a DIN rail or with 2 screws plus the additional accessory for wall mounting (TA526). The electrical connection of the I/O channels is carried out using the 30 terminals of the I/O terminal unit. I/O modules can be replaced without re-wiring the terminal units.

For a detailed description of the mounting, disassembly and electrical connection of the module, please refer to the System Assembly, Construction and Connection chapter “Chapter 1.5.4.5 “AC500 (Standard)” on page 3058.

The terminals 1.8 and 2.8 as well as 1.9, 2.9 and 3.9 are electrically interconnected within the terminal unit and have always the same assignment, independent of the inserted module:

Terminals 1.8 and 2.8: Process supply voltage UP = +24 VDC

Terminal 3.8: Process supply voltage UP3 = +24 VDC

Terminals 1.9, 2.9 and 3.9: Process supply voltage ZP = 0 V

With a separate UP3 power supply, the digital outputs can be switched off externally. This way, an emergency-off functionality can be realized.

Conditions for undisturbed operating with older I/O expansion modules

All I/O expansion modules that are attached to the CI52x-MODTCP must be powered up together with the CI52x-MODTCP if the firmware version of these I/O expansion modules is V1.9 or lower.

The firmware version is related to the index. The index is printed on the module type label on the right side.

Modules as of index listed in the following table can be powered up independently.

<table>
<thead>
<tr>
<th>S500 I/O module type</th>
<th>First index with firmware version above 1.9</th>
</tr>
</thead>
<tbody>
<tr>
<td>AI523</td>
<td>D0</td>
</tr>
<tr>
<td>AI523-XC</td>
<td>D0</td>
</tr>
<tr>
<td>AI531</td>
<td>A3</td>
</tr>
<tr>
<td>AI531-XC</td>
<td>A0</td>
</tr>
<tr>
<td>AO523</td>
<td>D0</td>
</tr>
<tr>
<td>AO523-XC</td>
<td>D0</td>
</tr>
<tr>
<td>AX521</td>
<td>D0</td>
</tr>
<tr>
<td>AX521-XC</td>
<td>D0</td>
</tr>
<tr>
<td>AX522</td>
<td>D0</td>
</tr>
<tr>
<td>AX522-XC</td>
<td>D0</td>
</tr>
<tr>
<td>CD522</td>
<td>A2</td>
</tr>
<tr>
<td>CD522-XC</td>
<td>A0</td>
</tr>
</tbody>
</table>
Do not connect any voltages externally to digital outputs!

This ist not intended usage.

Reason: Externally voltages at one or more terminals DC0...DC7 or DO8...DO15 may cause that other digital outputs are supplied through that voltage instead of voltage UP3 (reverse voltage).

This is also possible, if DC channels are used as inputs. For this, the source for the input signals should be the impressed UP3 of the device.

This limitation does not apply for the input channels DI0..DI7.

CAUTION!
Risk of malfunction by not intended usage!
If the function cut off of the digital outputs should be used by deactivation of the supply voltage UP3, be sure that no external voltage is connected at the outputs DO8...DO15 and DC0...DC7.

The assignment of the other terminals:

<table>
<thead>
<tr>
<th>Terminal</th>
<th>Signal</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0</td>
<td>DC0</td>
<td>Signal of the configurable digital input/output DC0</td>
</tr>
<tr>
<td>1.1</td>
<td>DC1</td>
<td>Signal of the configurable digital input/output DC1</td>
</tr>
<tr>
<td>Terminal</td>
<td>Signal</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>-------</td>
<td>-------------</td>
</tr>
<tr>
<td>1.2</td>
<td>DC2</td>
<td>Signal of the configurable digital input/output DC2</td>
</tr>
<tr>
<td>1.3</td>
<td>DC3</td>
<td>Signal of the configurable digital input/output DC3</td>
</tr>
<tr>
<td>1.4</td>
<td>DC4</td>
<td>Signal of the configurable digital input/output DC4</td>
</tr>
<tr>
<td>1.5</td>
<td>DC5</td>
<td>Signal of the configurable digital input/output DC5</td>
</tr>
<tr>
<td>1.6</td>
<td>DC6</td>
<td>Signal of the configurable digital input/output DC6</td>
</tr>
<tr>
<td>1.7</td>
<td>DC7</td>
<td>Signal of the configurable digital input/output DC7</td>
</tr>
<tr>
<td>1.8</td>
<td>UP</td>
<td>Process voltage UP (24 VDC)</td>
</tr>
<tr>
<td>1.9</td>
<td>ZP</td>
<td>Process voltage ZP (0 VDC)</td>
</tr>
<tr>
<td>2.0</td>
<td>DI8</td>
<td>Signal of the digital input DI8</td>
</tr>
<tr>
<td>2.1</td>
<td>DI9</td>
<td>Signal of the digital input DI9</td>
</tr>
<tr>
<td>2.2</td>
<td>DI10</td>
<td>Signal of the digital input DI10</td>
</tr>
<tr>
<td>2.3</td>
<td>DI11</td>
<td>Signal of the digital input DI11</td>
</tr>
<tr>
<td>2.4</td>
<td>DI12</td>
<td>Signal of the digital input DI12</td>
</tr>
<tr>
<td>2.5</td>
<td>DI13</td>
<td>Signal of the digital input DI13</td>
</tr>
<tr>
<td>2.6</td>
<td>DI14</td>
<td>Signal of the digital input DI14</td>
</tr>
<tr>
<td>2.7</td>
<td>DI15</td>
<td>Signal of the digital input DI15</td>
</tr>
<tr>
<td>2.8</td>
<td>UP</td>
<td>Process voltage UP (24 VDC)</td>
</tr>
<tr>
<td>2.9</td>
<td>ZP</td>
<td>Process voltage ZP (0 VDC)</td>
</tr>
<tr>
<td>3.0</td>
<td>DO8</td>
<td>Signal of the digital output DO8</td>
</tr>
<tr>
<td>3.1</td>
<td>DO9</td>
<td>Signal of the digital output DO9</td>
</tr>
<tr>
<td>3.2</td>
<td>DO10</td>
<td>Signal of the digital output DO10</td>
</tr>
<tr>
<td>3.3</td>
<td>DO11</td>
<td>Signal of the digital output DO11</td>
</tr>
<tr>
<td>3.4</td>
<td>DO12</td>
<td>Signal of the digital output DO12</td>
</tr>
<tr>
<td>3.5</td>
<td>DO13</td>
<td>Signal of the digital output DO13</td>
</tr>
<tr>
<td>3.6</td>
<td>DO14</td>
<td>Signal of the digital output DO14</td>
</tr>
<tr>
<td>3.7</td>
<td>DO15</td>
<td>Signal of the digital output DO15</td>
</tr>
<tr>
<td>3.8</td>
<td>UP3</td>
<td>Process voltage UP3 (24 VDC)</td>
</tr>
<tr>
<td>3.9</td>
<td>ZP</td>
<td>Process voltage ZP (0 VDC)</td>
</tr>
</tbody>
</table>
**WARNING!**
*Removal/Insertion under power*

The devices are not designed for removal or insertion under power. Because of unforeseeable consequences, it is not allowed to plug or unplug devices with the power being ON.

Make sure that all voltage sources (supply and process voltage) are switched off before you
- connect or disconnect any signal or terminal block
- remove, mount or replace a module.

Disconnecting any powered devices while energized in a hazardous location could result in an electric arc, which could create a flammable ignition resulting in fire or explosion.

Make sure that power is removed and that the area has been thoroughly checked to ensure that flammable materials are not present prior to proceeding.

The devices must not be opened when in operation. The same applies to the network interfaces.

**NOTICE!**
*Risk of damaging the PLC modules!*

Overvoltages and short circuits might damage the PLC modules.
- Make sure that all voltage sources (supply voltage and process supply voltage) are switched off before you begin with operations at the system.
- Never connect any voltages or signals to reserved terminals (marked with ---). Reserved terminals may carry internal voltages.

The following figure shows the electrical connection of the Ethernet bus module CI522-MODTCP.

*Fig. 242: Connection of the bus module CI522-MODTCP*
Further information is provided in the System Technology chapter **CI52x-MODTCP** Chapter 1.5.5.3.1 “Modbus communication interface module” on page 3200.

### Connection of the digital inputs

The following figure shows the electrical connection of the digital input DI8. Proceed with the digital inputs DI9 to DI15 in the same way.

![Connection of the digital inputs to the module CI522-MODTCP](image)

*Fig. 243: Connection of the digital inputs to the module CI522-MODTCP*

The meaning of the LEDs is described in Displays **Chapter 1.5.3.7.4.2.8.1 “State LEDs”** on page 2955.

### Connection of the digital outputs

The following figure shows the electrical connection of the digital output DO8. Proceed with the digital outputs DO9 - DO15 in the same way.
The meaning of the LEDs is described in Displays Chapter 1.5.3.7.4.2.8.1 “State LEDs” on page 2955.

Connection of the configurable digital inputs/outputs

The following figure shows the electrical connection of the configurable digital input/output DC0 and DC1. DC0 is connected as an input and DC1 is connected as an output. Proceed with the configurable digital inputs/outputs DC2 to DC7 in the same way.

CAUTION!
If a DC channel is used as input, the source for the input signals should be the impressed UP3 of the device Chapter 1.5.3.7.4.2.3 “Electrical connection” on page 2940.
The meaning of the LEDs is described in Displays "Chapter 1.5.3.7.4.2.8.1 "State LEDs" on page 2955.

Assignment of the Ethernet ports

The terminal unit for the Communication Interface Module provides two Ethernet interfaces with the following pin assignment:

Table 629: Pin assignment RJ45 jack:

<table>
<thead>
<tr>
<th>Interface</th>
<th>Pin</th>
<th>Signal</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ethernet R.J45</td>
<td>1</td>
<td>TxD+</td>
<td>Transmit data +</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>TxD-</td>
<td>Transmit data -</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>RxD+</td>
<td>Receive data +</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>NC</td>
<td>not used</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>NC</td>
<td>not used</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>RxD-</td>
<td>Receive data -</td>
</tr>
<tr>
<td></td>
<td>7</td>
<td>NC</td>
<td>not used</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>NC</td>
<td>not used</td>
</tr>
<tr>
<td>Shield</td>
<td></td>
<td>Cable shield</td>
<td>Functional earth</td>
</tr>
</tbody>
</table>

For further information regarding wiring and cable types see chapter Ethernet "Chapter 1.5.4.5.4.7 "Ethernet connection details" on page 3084.
Internal data exchange

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Digital inputs (bytes)</td>
<td>5</td>
</tr>
<tr>
<td>Digital outputs (bytes)</td>
<td>5</td>
</tr>
<tr>
<td>Counter input data (words)</td>
<td>4</td>
</tr>
<tr>
<td>Counter output data (words)</td>
<td>8</td>
</tr>
</tbody>
</table>

Addressing

The IP address of the CI5221-MODTCP Module can be set with the "ABB IP Configuration Tool" Chapter 1.5.6.2.4.2 “Configuration of the IP settings with IP configuration tool” on page 3267.

If the last byte of the IP is set to 0, the address switch will be used instead.

Address switch position 255 is mapped to fixed IP 192.168.0.254 independent of other stored settings. This is a backup so the module can always get a valid IP address and can be configured by the “ABB IP Configuration Tool”.

Address switch position 0 is mapped to last byte equal 1 and DHCP enabled.

The factory setting for the IP is 192.168.0.x (last byte is address switch).

The module reads the position of the rotary switches only during power-up, i.e. changes of the switch position during operation will have no effect until the next module initialization.

I/O configuration

The CI522-MODTCP stores configuration parameters (IP address configuration, module parameters).

The digital I/O channels are configured via software.

Details about configuration are described in Parameterization Chapter 1.5.3.7.4.2.7 “Parameterization” on page 2947.

Parameterization

Parameters of the module

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>Internal value</th>
<th>Internal value, type</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>Module ID 1)</td>
<td>Internal</td>
<td>7405</td>
<td>WORD</td>
<td>7405</td>
</tr>
<tr>
<td>Ignore Module</td>
<td>Internal</td>
<td>0</td>
<td>BYTE</td>
<td>0</td>
</tr>
<tr>
<td>Parameter length</td>
<td>Internal</td>
<td>47</td>
<td>BYTE</td>
<td>47</td>
</tr>
<tr>
<td>Error LED / Failsafe function</td>
<td>On</td>
<td>0</td>
<td>BYTE</td>
<td>0</td>
</tr>
<tr>
<td>(Table Error LED / Failsafe function)</td>
<td>Off by E4</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Off by E3</td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>On + failsafe</td>
<td>16</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Off by E4 + failsafe</td>
<td>17</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Name</td>
<td>Value</td>
<td>Internal value</td>
<td>Internal value, type</td>
<td>Default</td>
</tr>
<tr>
<td>-------------------------------------------</td>
<td>----------------</td>
<td>----------------</td>
<td>----------------------</td>
<td>----------</td>
</tr>
<tr>
<td>Master IP for Write restriction 4)</td>
<td>No master IP</td>
<td>0,0,0,0 W,X,y,z</td>
<td>ARRAY[0..3] OF BYTE</td>
<td>0,0,0,0</td>
</tr>
<tr>
<td>Master IP for Write restriction 4)</td>
<td>Master IP</td>
<td>0,0,0,0 W,X,y,z</td>
<td>ARRAY[0..3] OF BYTE</td>
<td>0,0,0,0</td>
</tr>
<tr>
<td>Master IP for Write restriction 4)</td>
<td>No master IP</td>
<td>0,0,0,0 W,X,y,z</td>
<td>ARRAY[0..3] OF BYTE</td>
<td>0,0,0,0</td>
</tr>
<tr>
<td>Master IP for Write restriction 4)</td>
<td>Master IP</td>
<td>0,0,0,0 W,X,y,z</td>
<td>ARRAY[0..3] OF BYTE</td>
<td>0,0,0,0</td>
</tr>
<tr>
<td>Master IP for Write restriction 4)</td>
<td>No master IP</td>
<td>0,0,0,0 W,X,y,z</td>
<td>ARRAY[0..3] OF BYTE</td>
<td>0,0,0,0</td>
</tr>
<tr>
<td>Master IP for Write restriction 4)</td>
<td>Master IP</td>
<td>0,0,0,0 W,X,y,z</td>
<td>ARRAY[0..3] OF BYTE</td>
<td>0,0,0,0</td>
</tr>
<tr>
<td>Master IP for Write restriction 4)</td>
<td>No master IP</td>
<td>0,0,0,0 W,X,y,z</td>
<td>ARRAY[0..3] OF BYTE</td>
<td>0,0,0,0</td>
</tr>
<tr>
<td>Master IP for Write restriction 4)</td>
<td>Master IP</td>
<td>0,0,0,0 W,X,y,z</td>
<td>ARRAY[0..3] OF BYTE</td>
<td>0,0,0,0</td>
</tr>
<tr>
<td>Master IP for Write restriction 4)</td>
<td>No master IP</td>
<td>0,0,0,0 W,X,y,z</td>
<td>ARRAY[0..3] OF BYTE</td>
<td>0,0,0,0</td>
</tr>
<tr>
<td>Master IP for Write restriction 4)</td>
<td>Master IP</td>
<td>0,0,0,0 W,X,y,z</td>
<td>ARRAY[0..3] OF BYTE</td>
<td>0,0,0,0</td>
</tr>
<tr>
<td>Timeout for Bus supervision</td>
<td>No supervision</td>
<td>0</td>
<td>BYTE</td>
<td>No supervi-</td>
</tr>
<tr>
<td></td>
<td>10 ms timeout</td>
<td>1</td>
<td></td>
<td>sion</td>
</tr>
<tr>
<td></td>
<td>20 ms timeout</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>IO Mapping Structure 3)</td>
<td>Fixed Mapping</td>
<td>0</td>
<td>BYTE</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>Dynamic Mapping</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Reserved</td>
<td>Internal</td>
<td>0</td>
<td>ARRAY[0..2] OF BYTE</td>
<td>0,0,0</td>
</tr>
<tr>
<td>Check supply</td>
<td>off</td>
<td>0</td>
<td>BYTE</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>on</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Fast counter</td>
<td>0</td>
<td>0</td>
<td>BYTE</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>10 2)</td>
<td>10</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Remarks:

1) With a faulty ID, the module reports a "parameter error" and does not perform cyclic process data transmission.

2) Counter operating modes · Chapter 1.5.3.6.1.2.9 “Fast counter” on page 2543
Fixed Mapping means each module has its own Modbus registers for data transfer independent of the I/O bus constellation. See \textit{Modbus TCP Registers} description for details \cite{chap1.5.3.1.2} "Modbus TCP registers" on page 3201.

Dynamic mapping means the structure of the I/O Date is dependent on the I/O bus constellation. Each I/O bus expansion module starts directly after the module before on the next Word adress.

If none of the parameters is set all masters / clients in the network have read and write rights on the CI52x-MODTCP device and its connected expansion modules.

If at least one parameter is set only the configured masters / clients have write rights on the CI52x-MODTCP device, all other masters / clients still have read access to the CI52x-MODTCP device.

<table>
<thead>
<tr>
<th>Setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>On</td>
<td>Error LED (S-ERR) lights up at errors of all error classes, Failsafe-mode off</td>
</tr>
<tr>
<td>Off by E4</td>
<td>Error LED (S-ERR) lights up at errors of error classes E1, E2 and E3, Failsafe-mode off</td>
</tr>
<tr>
<td>Off by E3</td>
<td>Error LED (S-ERR) lights up at errors of error classes E1 and E2, Failsafe-mode off</td>
</tr>
<tr>
<td>On + Failsafe</td>
<td>Error LED (S-ERR) lights up at errors of all error classes, Failsafe-mode on *)</td>
</tr>
<tr>
<td>Off by E4 + Failsafe</td>
<td>Error LED (S-ERR) lights up at errors of error classes E1, E2 and E3, Failsafe-mode on *)</td>
</tr>
<tr>
<td>Off by E3 + Failsafe</td>
<td>Error LED (S-ERR) lights up at errors of error classes E1 and E2, Failsafe-mode on *)</td>
</tr>
</tbody>
</table>

*) The parameter Behaviour DO at comm. error is only analyzed if the Failsafe-mode is ON.

Group parameters for the digital part

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>Internal value</th>
<th>Internal value, type</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input delay</td>
<td>0.1 ms</td>
<td>0</td>
<td>BYTE</td>
<td>0.1 ms</td>
</tr>
<tr>
<td></td>
<td>1 ms</td>
<td>1</td>
<td></td>
<td>0x00</td>
</tr>
<tr>
<td></td>
<td>8 ms</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>32 ms</td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Detect short circuit at outputs</td>
<td>Off</td>
<td>0</td>
<td>BYTE</td>
<td>On</td>
</tr>
<tr>
<td></td>
<td>On</td>
<td>1</td>
<td></td>
<td>0x01</td>
</tr>
<tr>
<td>Name</td>
<td>Value</td>
<td>Internal value</td>
<td>Internal value, type</td>
<td>Default</td>
</tr>
<tr>
<td>----------------------------------------------------</td>
<td>---------------</td>
<td>----------------</td>
<td>----------------------</td>
<td>---------</td>
</tr>
<tr>
<td>Behaviour DO at comm. error ¹)</td>
<td>Off</td>
<td>0</td>
<td>BYTE</td>
<td>Off</td>
</tr>
<tr>
<td>Last value</td>
<td></td>
<td>1</td>
<td></td>
<td>0x00</td>
</tr>
<tr>
<td>Last value 5 sec</td>
<td></td>
<td>6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Last value 10 sec</td>
<td></td>
<td>11</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Substitute value</td>
<td></td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Substitute value 5 sec</td>
<td></td>
<td>7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Substitute value 10 sec</td>
<td></td>
<td>12</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Substitute value at output</td>
<td>0 ... 65535</td>
<td>0000h ... FFFFh</td>
<td>WORD</td>
<td>0</td>
</tr>
<tr>
<td>Preventive voltage feedback monitoring for DC0..DC7 ²)</td>
<td>Off</td>
<td>0</td>
<td>BYTE</td>
<td>Off</td>
</tr>
<tr>
<td>On</td>
<td></td>
<td>1</td>
<td></td>
<td>0x00</td>
</tr>
<tr>
<td>Detect voltage overflow at outputs ³)</td>
<td>Off</td>
<td>0</td>
<td>BYTE</td>
<td>Off</td>
</tr>
<tr>
<td>On</td>
<td></td>
<td>1</td>
<td></td>
<td>0x00</td>
</tr>
</tbody>
</table>

Remarks:

¹) The parameter Behaviour DO at comm. error is apply to DC and DO channels and only analyzed if the Failsafe-mode is ON.

²) The state "externally voltage detected" appears, if the output of a channel DC0...DC7 should be switched on while an externally voltage is connected. In this case the start up is disabled, as long as the externally voltage is connected. The monitoring of this state and the resulting diagnosis message can be disabled by setting the parameters to "OFF".

³) The error state "voltage overflow at outputs" appears, if externally voltage at digital outputs DC0...DC7 and accordingly DO8...DO15 has exceeded the process supply voltage UP3 (see description in section). The according diagnosis message "Voltage overflow on outputs " can be disabled by setting the parameters on "OFF". This parameter should only be disabled in exceptional cases for voltage overflow may produce reverse voltage.

**Diagnosis**

Structure of the Diagnosis Block
<table>
<thead>
<tr>
<th>Byte Number</th>
<th>Description</th>
<th>Possible Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Diagnosis Byte, slot number</td>
<td>31 = CI502-PNIO (e.g. error at integrated 8 DI / 8 DO)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 = 1st connected S500 I/O Module</td>
</tr>
<tr>
<td></td>
<td></td>
<td>...</td>
</tr>
<tr>
<td></td>
<td></td>
<td>10 = 10th connected S500 I/O Module</td>
</tr>
<tr>
<td>2</td>
<td>Diagnosis Byte, module number</td>
<td>According to the I/O bus specification passed on by modules to the fieldbus master</td>
</tr>
<tr>
<td>3</td>
<td>Diagnosis Byte, channel</td>
<td>According to the I/O bus specification passed on by modules to the fieldbus master</td>
</tr>
<tr>
<td>4</td>
<td>Diagnosis Byte, error code</td>
<td>According to the I/O bus specification</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Bit 7 and bit 6, coded error class</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 = E1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 = E2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2 = E3</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3 = E4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Bit 0 to bit 5, coded error description</td>
</tr>
<tr>
<td>5</td>
<td>Diagnosis Byte, flags</td>
<td>According to the I/O bus specification</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Bit 7: 1 = coming error</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Bit 6: 1 = leaving error</td>
</tr>
<tr>
<td>6</td>
<td>Reserved</td>
<td>0</td>
</tr>
</tbody>
</table>

In cases of short circuit or overload, the digital outputs are turned off. The modules performs reactivation automatically. Thus an acknowledgement of the errors is not necessary. The error message is stored via the LED.

*For diagnosis firmware version ≥ 3.2.6 is required.*
### Module errors

<table>
<thead>
<tr>
<th>Class</th>
<th>Interface</th>
<th>Device</th>
<th>Module</th>
<th>Channel</th>
<th>Error-Identifer</th>
<th>Error message</th>
<th>Remedy</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>31</td>
<td>31</td>
<td>31</td>
<td>19</td>
<td>Checksum error in the I/O module</td>
<td>Replace I/O module</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>31</td>
<td>31</td>
<td>31</td>
<td>3</td>
<td>Timeout in the I/O module</td>
<td>Check process supply voltage</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>31</td>
<td>31</td>
<td>31</td>
<td>40</td>
<td>Different hard-/firmware versions in the module</td>
<td>Check process supply voltage</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>31</td>
<td>31</td>
<td>31</td>
<td>43</td>
<td>Internal error in the module</td>
<td>Check process supply voltage</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>31</td>
<td>31</td>
<td>31</td>
<td>36</td>
<td>Internal data exchange failure</td>
<td>Replace I/O module</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>31</td>
<td>31</td>
<td>31</td>
<td>9</td>
<td>Overflow diagnosis buffer</td>
<td>Replace I/O module</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>31</td>
<td>31</td>
<td>31</td>
<td>26</td>
<td>Parameter error</td>
<td>Check process supply voltage</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>31</td>
<td>31</td>
<td>31</td>
<td>11</td>
<td>Process voltage UP too low</td>
<td>Replace I/O module</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>31</td>
<td>31</td>
<td>31</td>
<td>45</td>
<td>Process voltage UP gone</td>
<td>Replace I/O module</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>31/1...10</td>
<td>31</td>
<td>31</td>
<td>17</td>
<td>No communication with I/O module</td>
<td>Replace I/O module</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>1...10</td>
<td>31</td>
<td>31</td>
<td>32</td>
<td>Wrong I/O module type on socket</td>
<td>Replace I/O module / Check configuration</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>1...10</td>
<td>31</td>
<td>31</td>
<td>31</td>
<td>At least one module does not support failsafe function</td>
<td>Check modules and parameterization</td>
<td></td>
</tr>
<tr>
<td>E1..E4</td>
<td>d1</td>
<td>d2</td>
<td>d3</td>
<td>d4</td>
<td>Identifer</td>
<td>AC500-Display</td>
<td>LC501 PLC Browser</td>
</tr>
<tr>
<td>-------</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>-----------</td>
<td>---------------</td>
<td>------------------</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>000..063</td>
<td>&lt; Display in</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Class</th>
<th>Comp</th>
<th>Dev</th>
<th>Mod</th>
<th>Ch</th>
<th>Err</th>
<th>PS501 PLC Browser</th>
<th>PNIO diagnosis block</th>
</tr>
</thead>
<tbody>
<tr>
<td>Byte 4 Bit 6..7</td>
<td>-</td>
<td>Byte 1</td>
<td>Byte 2</td>
<td>Byte 3</td>
<td>Byte 4 Bit 0..5</td>
<td>PNIO diagnosis block</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Class</th>
<th>Interface</th>
<th>Device</th>
<th>Module</th>
<th>Channel</th>
<th>Error-</th>
<th>Error message</th>
<th>Remedy</th>
</tr>
</thead>
<tbody>
<tr>
<td>1)</td>
<td>2)</td>
<td>3)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>-</td>
<td>1...10</td>
<td>31</td>
<td>5</td>
<td>8</td>
<td>I/O module removed from hot-swap terminal unit or defective module on hot-swap terminal unit 8)</td>
<td>Plug I/O module, replace I/O module</td>
</tr>
<tr>
<td>4</td>
<td>-</td>
<td>1...10</td>
<td>31</td>
<td>5</td>
<td>28</td>
<td>Wrong I/O module plugged on hot-swap terminal unit 9)</td>
<td>Remove wrong I/O module and plug projected I/O module</td>
</tr>
<tr>
<td>4</td>
<td>-</td>
<td>1...10</td>
<td>31</td>
<td>5</td>
<td>42</td>
<td>No communication with I/O module on hot-swap terminal unit 9)</td>
<td>Replace I/O module</td>
</tr>
<tr>
<td>4</td>
<td>-</td>
<td>1...10</td>
<td>31</td>
<td>5</td>
<td>54</td>
<td>I/O module does not support hot swap 8) 9)</td>
<td>Power off system and replace I/O module</td>
</tr>
<tr>
<td>4</td>
<td>-</td>
<td>1...10</td>
<td>31</td>
<td>6</td>
<td>42</td>
<td>No communication with hot-swap terminal unit 9)</td>
<td>Restart, if error persists replace terminal unit</td>
</tr>
<tr>
<td>4</td>
<td>1...6</td>
<td>255</td>
<td>2</td>
<td>0</td>
<td>45</td>
<td>The connected Communication Module has no connection to the network</td>
<td>Check cabling</td>
</tr>
<tr>
<td>4</td>
<td>-</td>
<td>31</td>
<td>31</td>
<td>31</td>
<td>45</td>
<td>Process voltage UP3 too low</td>
<td>Check process voltage</td>
</tr>
<tr>
<td>Class</td>
<td>Interface</td>
<td>Device</td>
<td>Module</td>
<td>Channel</td>
<td>Error-Identifier</td>
<td>Error message</td>
<td>Remedy</td>
</tr>
<tr>
<td>-------</td>
<td>-----------</td>
<td>--------</td>
<td>--------</td>
<td>---------</td>
<td>-----------------</td>
<td>---------------</td>
<td>--------</td>
</tr>
<tr>
<td>4</td>
<td>-</td>
<td>31</td>
<td>31</td>
<td>31</td>
<td>46</td>
<td>Reverse voltage from digital outputs DO8...DO15 to UP3 4)</td>
<td>Check terminals</td>
</tr>
<tr>
<td>4</td>
<td>-</td>
<td>31/1...10</td>
<td>31</td>
<td>31</td>
<td>34</td>
<td>No response during initialization of the I/O module</td>
<td>Replace I/O module</td>
</tr>
<tr>
<td>4</td>
<td>-</td>
<td>31</td>
<td>31</td>
<td>31</td>
<td>11</td>
<td>Process voltage UP3 too low</td>
<td>Check process supply voltage</td>
</tr>
<tr>
<td>4</td>
<td>-</td>
<td>31</td>
<td>31</td>
<td>31</td>
<td>45</td>
<td>Process voltage UP3 gone</td>
<td>Check process supply voltage</td>
</tr>
<tr>
<td>4</td>
<td>-</td>
<td>31</td>
<td>31</td>
<td>31</td>
<td>10</td>
<td>Voltage overflow at outputs (above UP3 level) 5)</td>
<td>Check terminals/ check process supply voltage</td>
</tr>
</tbody>
</table>

Channel error digital

<table>
<thead>
<tr>
<th>Class</th>
<th>Interface</th>
<th>Device</th>
<th>Module</th>
<th>Channel</th>
<th>Error-Identifier</th>
<th>Error message</th>
<th>Remedy</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>-</td>
<td>31</td>
<td>2</td>
<td>8..15</td>
<td>46</td>
<td>Externally voltage detected at digital output DO8...DO15 6)</td>
<td>Check terminals</td>
</tr>
<tr>
<td>4</td>
<td>-</td>
<td>31</td>
<td>4</td>
<td>0...7</td>
<td>46</td>
<td>Externally voltage detected at digital output DC0...DC7 6)</td>
<td>Check terminals</td>
</tr>
<tr>
<td>4</td>
<td>-</td>
<td>31</td>
<td>2</td>
<td>0...7</td>
<td>47</td>
<td>Short circuit at digital output 7)</td>
<td>Check terminals</td>
</tr>
</tbody>
</table>

Remarks:
In AC500 the following interface identifier applies:

"." = Diagnosis via bus-specific function blocks; 0 ... 4 or 10 = Position of the Communication Module; 14 = I/O bus; 31 = Module itself

The identifier is not contained in the CI502-PNIO diagnosis block.

With "Device" the following allocation applies: 31 = Module itself, 1..10 = Expansion module

With "Module" the following allocation applies dependent of the master:

Module error: 31 = Module itself
Channel error: Module type (1 = AI, 2 = DO, 3 = AO)

This message appears, if externally voltages at one or more terminals DC0...DC7 oder DO8...DO15 cause that other digital outputs are supplied through that voltage (voltage feedback, see description in Electrical Connection  Chapter 1.5.3.7.4.2.3 “Electrical connection” on page 2940. All outputs of the apply digital output groups will be turned off for 5 seconds. The diagnosis message appears for the whole output group.

The voltage at digital outputs DC0...DC7 and accordingly DO8...DO15 has exceeded the process supply voltage UP3  Chapter 1.5.3.7.4.2.3 “Electrical connection” on page 2940. Diagnosis message appears for the whole module.

This message appears, if the output of a channel DC0...DC7 oder DO8...DO15 should be switched on while an externally voltage is connected. In this case the start up is disabled, as long as the externally voltage is connected. Otherwise this could produce reverse voltage from this output to other digital outputs. This diagnosis message appears per channel.

Short circuit: After a detected short circuit, the output is deactivated for 2000ms. Then a new start up will be executed. This diagnosis message appears per channel.

In case of an I/O module doesn't support hot swapping, do not perform any hot-swap operations (also not on any other terminal units (slots)) as modules may be damaged or I/O bus communication may be disturbed.

Diagnosis for hot swap available as of version index F0.

State LEDs

The LEDs are located at the front of module. There are 2 different groups:

- The 5 system LEDs (PWR, STA1 ETH, STA2 ETH, S-ERR and I/O-Bus) show the operation state of the module and display possible errors.
- The 29 process LEDs (UP, UP3, inputs, outputs, CH-ERR1 to CH-ERR3) show the process supply voltage and the states of the inputs and outputs and display possible errors.

Table 631: States of the 5 system LEDs

<table>
<thead>
<tr>
<th>LED</th>
<th>Color</th>
<th>OFF</th>
<th>ON</th>
<th>Flashing</th>
</tr>
</thead>
<tbody>
<tr>
<td>PWR/RUN</td>
<td>Green</td>
<td>Process supply voltage missing</td>
<td>Internal supply voltage OK, module ready for communication with IO Controller</td>
<td>Start-up / preparing communication</td>
</tr>
<tr>
<td>STA1 ETH</td>
<td>Green</td>
<td>---</td>
<td>Device configured, cyclic data exchange running</td>
<td>Device configured, acyclic data exchange running</td>
</tr>
</tbody>
</table>
### Table 632: States of the 29 process LEDs

<table>
<thead>
<tr>
<th>LED</th>
<th>Color</th>
<th>OFF</th>
<th>ON</th>
<th>Flashing</th>
</tr>
</thead>
<tbody>
<tr>
<td>DC0 to DC7</td>
<td>Yellow</td>
<td>Input/Output is OFF</td>
<td>Input/Output is ON</td>
<td>--</td>
</tr>
<tr>
<td>DI8 to DI15</td>
<td>Yellow</td>
<td>Input is OFF</td>
<td>Input is ON (the input voltage is even displayed if the supply voltage is OFF)</td>
<td>--</td>
</tr>
<tr>
<td>DO8 to DO15</td>
<td>Yellow</td>
<td>Output is OFF</td>
<td>Output is ON</td>
<td>--</td>
</tr>
<tr>
<td>UP</td>
<td>Green</td>
<td>Process supply voltage missing</td>
<td>Process supply voltage OK and initialization finished</td>
<td>--</td>
</tr>
<tr>
<td>UP3</td>
<td>Green</td>
<td>Process supply voltage missing</td>
<td>Process supply voltage OK</td>
<td>--</td>
</tr>
<tr>
<td>CH-ERR1 to CH-ERR3</td>
<td>Red</td>
<td>No error or process supply voltage missing</td>
<td>Internal error</td>
<td>Error on one channel of the corresponding group</td>
</tr>
</tbody>
</table>

### Technical data

The system data of AC500 and S500 Chapter 1.5.4.5.1 “System data AC500” on page 3058 are valid for standard version.
The system data of AC500-XC 

Chapter 1.5.4.6.1 “System data AC500-XC” on page 3099 are valid for the XC version.

Only additional details are therefore documented below.
The technical data are also valid for the XC version.

Technical data of the module

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Process supply voltages UP/UP3</td>
<td></td>
</tr>
<tr>
<td>Rated value</td>
<td>24 VDC (for inputs and outputs)</td>
</tr>
<tr>
<td>Max. load for the terminals</td>
<td>10 A</td>
</tr>
<tr>
<td>Protection against reversed voltage</td>
<td>Yes</td>
</tr>
<tr>
<td>Rated protection fuse on UP/UP3</td>
<td>10 A fast</td>
</tr>
<tr>
<td>Electrical isolation</td>
<td>Ethernet interface against the rest of the module</td>
</tr>
<tr>
<td>Inrush current from UP (at power up)</td>
<td>On request</td>
</tr>
<tr>
<td>Current consumption via UP (normal operation)</td>
<td>0.15 A</td>
</tr>
<tr>
<td>Current consumption via UP3</td>
<td>0.06 A + 0.5 A max. per output</td>
</tr>
<tr>
<td>Connections</td>
<td></td>
</tr>
<tr>
<td>Terminals 1.8 and 2.8 for +24 V (UP)</td>
<td></td>
</tr>
<tr>
<td>Terminal 3.8 for +24 V (UP3)</td>
<td></td>
</tr>
<tr>
<td>Terminals 1.9, 2.9 and 3.9 for 0 V (ZP)</td>
<td></td>
</tr>
<tr>
<td>Max. power dissipation within the module</td>
<td>6 W</td>
</tr>
<tr>
<td>Number of digital inputs</td>
<td>8</td>
</tr>
<tr>
<td>Number of digital outputs</td>
<td>8</td>
</tr>
<tr>
<td>Number of configurable digital inputs/outputs</td>
<td>8</td>
</tr>
<tr>
<td>Reference potential for all digital inputs and outputs</td>
<td>Minus pole of the supply voltage, signal name ZP</td>
</tr>
<tr>
<td>Ethernet</td>
<td>10/100 base-TX, internal switch, 2 x RJ45 socket</td>
</tr>
<tr>
<td>Setting of the IO Device identifier</td>
<td>With 2 rotary switches at the front side of the module</td>
</tr>
</tbody>
</table>
| Diagnosis                                    | See Diagnosis and Displays 
Chapter 1.5.3.7.4.2.8 “Diagnosis” on page 2950                  |
| Operation and error displays                 | 34 LEDs (totally)                                                   |
| Weight (without Terminal Unit)               | Ca. 125 g                                                           |
| Mounting position                            | Horizontal or vertical with derating (output load reduced to 50 % at 40°C per group) |
| Extended ambient temperature (XC version)     | > 60 °C on request                                                  |
| Cooling                                       | The natural convection cooling must not be hindered by cable ducts or other parts in the switch-gear cabinet. |
**NOTICE!**  
**Attention:**  
All I/O channels (digital and analog) are protected against reverse polarity, reverse supply, short circuit and continuous overvoltage up to 30 VDC.

---

### Multiple overloads

No effects of multiple overloads on isolated multi-channel modules occur, as every channel is protected individually by an internal smart high-side switch.

---

**Technical data of the digital inputs**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of channels per module</td>
<td>8</td>
</tr>
<tr>
<td>Distribution of the channels into groups</td>
<td>1 group of 8 channels</td>
</tr>
<tr>
<td>Terminals of the channels DI8 to DI15</td>
<td>Terminals 2.0 to 2.7</td>
</tr>
<tr>
<td>Reference potential for all inputs</td>
<td>Terminals 1.9...3.9 (Minus pole of the supply voltage, signal name ZP)</td>
</tr>
<tr>
<td>Indication of the input signals</td>
<td>1 yellow LED per channel, the LED is ON when the input signal is high (signal 1)</td>
</tr>
<tr>
<td>Input type (according EN 61131-2)</td>
<td>Type 1</td>
</tr>
<tr>
<td>Input delay (0-&gt;1 or 1-&gt;0)</td>
<td>Typ. 0.1 ms, configurable from 0.1...32 ms</td>
</tr>
<tr>
<td>Input signal voltage</td>
<td>24 VDC</td>
</tr>
<tr>
<td>Signal 0</td>
<td>-3 V...+5 V</td>
</tr>
<tr>
<td>Undefined Signal</td>
<td>&gt; +5 V...&lt; +15 V</td>
</tr>
<tr>
<td>Signal 1</td>
<td>+15 V...+30 V</td>
</tr>
<tr>
<td>Ripple with signal 0</td>
<td>Within -3 V...+5 V</td>
</tr>
<tr>
<td>Ripple with signal 1</td>
<td>Within +15 V...+30 V</td>
</tr>
<tr>
<td>Input current per channel</td>
<td></td>
</tr>
<tr>
<td>Input voltage +24 V</td>
<td>Typ. 5 mA</td>
</tr>
<tr>
<td>Input voltage +5 V</td>
<td>&gt; 1 mA</td>
</tr>
<tr>
<td>Input voltage +15 V</td>
<td>&gt; 2 mA</td>
</tr>
<tr>
<td>Input voltage +30 V</td>
<td>&lt; 8 mA</td>
</tr>
<tr>
<td>Max. cable length</td>
<td></td>
</tr>
<tr>
<td>Shielded</td>
<td>1000 m</td>
</tr>
<tr>
<td>Unshielded</td>
<td>600 m</td>
</tr>
</tbody>
</table>
# Technical data of the digital outputs

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of channels per module</td>
<td>8</td>
</tr>
<tr>
<td>Distribution of the channels into groups</td>
<td>1 group of 8 channels</td>
</tr>
<tr>
<td>Terminals of the channels DO8 to DO15</td>
<td>Terminals 3.0 to 3.7</td>
</tr>
<tr>
<td>Reference potential for all outputs</td>
<td>Terminals 1.9...3.9 (minus pole of the supply voltage, signal name ZP)</td>
</tr>
<tr>
<td>Common power supply voltage</td>
<td>For all outputs terminal 3.8 (plus pole of the supply voltage, signal name UP3)</td>
</tr>
<tr>
<td>Output voltage for signal 1</td>
<td>UP3 (-0.8 V)</td>
</tr>
<tr>
<td>Output delay (0-&gt;1 or 1-&gt;0)</td>
<td>On request</td>
</tr>
<tr>
<td>Output current</td>
<td></td>
</tr>
<tr>
<td>Rated value per channel</td>
<td>500 mA at UP3 = 24 V</td>
</tr>
<tr>
<td>Max. value (all channels together)</td>
<td>4 A</td>
</tr>
<tr>
<td>Leakage current with signal 0</td>
<td>&lt; 0.5 mA</td>
</tr>
<tr>
<td>Fuse for UP3</td>
<td>10 A fast</td>
</tr>
<tr>
<td>Demagnetization with inductive DC load</td>
<td>Via internal varistors (see figure below this table)</td>
</tr>
<tr>
<td>Output switching frequency</td>
<td></td>
</tr>
<tr>
<td>With resistive load</td>
<td>On request</td>
</tr>
<tr>
<td>With inductive loads</td>
<td>Max. 0.5 Hz</td>
</tr>
<tr>
<td>With lamp loads</td>
<td>11 Hz max. at 5 W max.</td>
</tr>
<tr>
<td>Short-circuit-proof / overload-proof</td>
<td>Yes</td>
</tr>
<tr>
<td>Overload message (I &gt; 0.7 A)</td>
<td>Yes, after ca. 100 ms</td>
</tr>
<tr>
<td>Output current limitation</td>
<td>Yes, automatic reactivation after short circuit/overload</td>
</tr>
<tr>
<td>Resistance to feedback against 24V signals</td>
<td>Yes (software-controlled supervision)</td>
</tr>
<tr>
<td>Max. cable length</td>
<td></td>
</tr>
<tr>
<td>Shielded</td>
<td>1000 m</td>
</tr>
<tr>
<td>Unshielded</td>
<td>600 m</td>
</tr>
</tbody>
</table>

The following drawing shows the circuitry of a digital input/output with the varistors for demagnetization when inductive loads are switched off.

![Circuit diagram](image)

Fig. 244: Digital input/output (circuit diagram)
1. Digital Output
2. Varistors for demagnetization when inductive loads are turned off
Technical data of the configurable digital inputs/outputs

Each of the configurable I/O channels is defined as input or output by the user program. This is done by interrogating or allocating the corresponding channel.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of channels per module</td>
<td>8 inputs/outputs (with transistors)</td>
</tr>
<tr>
<td>Distribution of the channels into groups</td>
<td>1 group for 8 channels</td>
</tr>
<tr>
<td>If the channels are used as inputs</td>
<td></td>
</tr>
<tr>
<td>Channels DC0...DC7</td>
<td>Terminals 1.0...1.7</td>
</tr>
<tr>
<td>If the channels are used as outputs</td>
<td></td>
</tr>
<tr>
<td>Channels DC0...DC7</td>
<td>Terminals 1.0...1.7</td>
</tr>
<tr>
<td>Indication of the input/output signals</td>
<td>1 yellow LED per channel, the LED is ON when the input/output signal is high (signal 1)</td>
</tr>
<tr>
<td>Electrical isolation</td>
<td>From the Ethernet network</td>
</tr>
</tbody>
</table>

Technical data of the digital inputs/outputs if used as inputs

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of channels per module</td>
<td>8</td>
</tr>
<tr>
<td>Distribution of the channels into groups</td>
<td>1 group of 8 channels</td>
</tr>
<tr>
<td>Terminals of the channels DC0 to DC7</td>
<td>Terminals 1.0 to 1.7</td>
</tr>
<tr>
<td>Reference potential for all inputs</td>
<td>Terminals 1.9...3.9 (Minus pole of the supply voltage, signal name ZP)</td>
</tr>
<tr>
<td>Indication of the input signals</td>
<td>1 yellow LED per channel, the LED is ON when the input signal is high (signal 1)</td>
</tr>
<tr>
<td>Input type (according EN 61131-2)</td>
<td>Type 1</td>
</tr>
<tr>
<td>Input delay (0-&gt;1 or 1-&gt;0)</td>
<td>Typ. 0.1 ms, configurable from 0.1...32 ms</td>
</tr>
<tr>
<td>Input signal voltage</td>
<td>24 VDC</td>
</tr>
<tr>
<td>Signal 0</td>
<td>-3 V...+5 V *)</td>
</tr>
<tr>
<td>Undefined Signal</td>
<td>&gt; +5 V...&lt; +15 V</td>
</tr>
<tr>
<td>Signal 1</td>
<td>+15 V...+30 V</td>
</tr>
<tr>
<td>Ripple with signal 0</td>
<td>Within -3 V...+5 V *)</td>
</tr>
<tr>
<td>Ripple with signal 1</td>
<td>Within +15 V...+30 V</td>
</tr>
<tr>
<td>Input current per channel</td>
<td></td>
</tr>
<tr>
<td>Input voltage +24 V</td>
<td>Typ. 5 mA</td>
</tr>
<tr>
<td>Input voltage +5 V</td>
<td>&gt; 1 mA</td>
</tr>
<tr>
<td>Input voltage +15 V</td>
<td>&gt; 2 mA</td>
</tr>
<tr>
<td>Input voltage +30 V</td>
<td>&lt; 8 mA</td>
</tr>
<tr>
<td>Max. cable length</td>
<td></td>
</tr>
<tr>
<td>Shielded</td>
<td>1000 m</td>
</tr>
<tr>
<td>Unshielded</td>
<td>600 m</td>
</tr>
</tbody>
</table>
*) Due to the direct connection to the output, the demagnetizing varistor is also effective at the input (see figure) above. This is why the difference between UPx and the input signal may not exceed the clamp voltage of the varistor. The varistor limits the voltage to approx. 36 V. Following this, the input voltage must range from -12 V to +30 V when UPx = 24 V and from -6 V to +30 V when UPx = 30 V.

Technical data of the digital inputs/outputs if used as outputs

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of channels per module</td>
<td>8</td>
</tr>
<tr>
<td>Distribution of the channels into groups</td>
<td>1 group of 8 channels</td>
</tr>
<tr>
<td>Terminals of the channels DC0 to DC7</td>
<td>Terminals 1.0 to 1.7</td>
</tr>
<tr>
<td>Reference potential for all outputs</td>
<td>Terminals 1.9...3.9 (minus pole of the supply voltage, signal name ZP)</td>
</tr>
<tr>
<td>Common power supply voltage</td>
<td>For all outputs terminal 3.8 (plus pole of the supply voltage, signal name UP3)</td>
</tr>
<tr>
<td>Output voltage for signal 1</td>
<td>UP3 (-0.8 V)</td>
</tr>
<tr>
<td>Output delay (0-&gt;1 or 1-&gt;0)</td>
<td>On request</td>
</tr>
<tr>
<td>Output current</td>
<td></td>
</tr>
<tr>
<td>Rated value per channel</td>
<td>500 mA at UP3 = 24 V</td>
</tr>
<tr>
<td>Max. value (all channels together)</td>
<td>4 A</td>
</tr>
<tr>
<td>Leakage current with signal 0</td>
<td>&lt; 0.5 mA</td>
</tr>
<tr>
<td>Fuse for UP3</td>
<td>10 A fast</td>
</tr>
<tr>
<td>Demagnetization with inductive DC load</td>
<td>Via internal varistors (see figure below this table)</td>
</tr>
<tr>
<td>Output switching frequency</td>
<td></td>
</tr>
<tr>
<td>With resistive load</td>
<td>On request</td>
</tr>
<tr>
<td>With inductive loads</td>
<td>Max. 0.5 Hz</td>
</tr>
<tr>
<td>With lamp loads</td>
<td>11 Hz max. at 5 W max.</td>
</tr>
<tr>
<td>Short-circuit-proof / overload-proof</td>
<td>Yes</td>
</tr>
<tr>
<td>Overload message (I &gt; 0.7 A)</td>
<td>Yes, after ca. 100 ms</td>
</tr>
<tr>
<td>Output current limitation</td>
<td>Yes, automatic reactivation after short circuit/overload</td>
</tr>
<tr>
<td>Resistance to feedback against 24V signals</td>
<td>Yes (software-controlled supervision)</td>
</tr>
<tr>
<td>Max. cable length</td>
<td></td>
</tr>
<tr>
<td>Shielded</td>
<td>1000 m</td>
</tr>
<tr>
<td>Unshielded</td>
<td>600 m</td>
</tr>
</tbody>
</table>

The following drawing shows the circuitry of a digital input/output with the varistors for demagnetization when inductive loads are switched off.
Technical data of the fast counter

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Used inputs</td>
<td>Terminal 2.0 (DI8), Terminal 2.1 (DI9)</td>
</tr>
<tr>
<td>Used outputs</td>
<td>Terminal 3.0 (DO8)</td>
</tr>
<tr>
<td>Counting frequency</td>
<td>Depending on operation mode:</td>
</tr>
<tr>
<td></td>
<td>Mode 1-6: max. 200 kHz</td>
</tr>
<tr>
<td></td>
<td>Mode 7: max. 50 kHz</td>
</tr>
<tr>
<td></td>
<td>Mode 9: max. 35 kHz</td>
</tr>
<tr>
<td></td>
<td>Mode 10: max. 20 kHz</td>
</tr>
<tr>
<td>Detailed description</td>
<td>See Fast Counter % Chapter 1.5.5.1.10 “Fast counters” on page 3194</td>
</tr>
<tr>
<td>Operating modes</td>
<td>See Operating modes</td>
</tr>
</tbody>
</table>

Ordering data

<table>
<thead>
<tr>
<th>Ordering No.</th>
<th>Scope of delivery</th>
<th>Product life cycle phase *)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1SAP 222 200 R0001</td>
<td>CI522-MODTCP, Modbus TCP bus module, 8 DC, 8 DI and 8 DO</td>
<td>Active</td>
</tr>
<tr>
<td>1SAP 422 200 R0001</td>
<td>CI522-MODTCP-XC, Modbus TCP bus module, 8 DC, 8 DI and 8 DO, XC version</td>
<td>Active</td>
</tr>
</tbody>
</table>

*) For planning and commissioning of new installations use modules in Active status only.

1.5.3.7.5 PROFINET

Comparison of the CIxyz-PNIO modules

The PROFINET IO devices combine the advantages of decentralized I/O modules with the reaction time of AC500 mounted central I/O modules. The devices for PROFINET provide the extension -PNIO in the device name.
The Communication Module CM579-PNIO acts as IO Controller in a PROFINET network. It is connected to the Processor Module via an internal communication bus. Depending on the Terminal Base, several Communication Modules can be used for one Processor Module.

The Communication Interface Modules CIxyz-PNIO act as IO Devices in a PROFINET network. Additionally the communication module CM589-PNIO(-4) can be used to setup a AC500 PLC to act as IO Device in a PROFINET network.

The difference of the CIxyz-PNIO devices can be found in their input and output characteristics

The characteristics for CM589-PNIO(-4) can be found in the device description for CM589-PNIO.

PROFINET IO devices CI50x-PNIO

Characteristics of CI50x-PNIO

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bus connection</td>
<td>2 x RJ45</td>
</tr>
<tr>
<td>Switch</td>
<td>Integrated</td>
</tr>
<tr>
<td>Technology</td>
<td>Hilscher netX100</td>
</tr>
<tr>
<td>Transfer rate</td>
<td>10/100 Mbit/s (full-duplex)</td>
</tr>
<tr>
<td>Transfer method</td>
<td>According to Ethernet II, IEE802.3</td>
</tr>
<tr>
<td>Ethernet</td>
<td>100 base-TX, internal switch, 2x RJ45 socket</td>
</tr>
<tr>
<td>Expandability</td>
<td>Max. 10 S500 I/O modules</td>
</tr>
<tr>
<td>Adjusting elements</td>
<td>2 rotary switches for generation of an explicit name</td>
</tr>
<tr>
<td>Supported protocols</td>
<td>RTC - real time cyclic protocol, class 1 *)</td>
</tr>
<tr>
<td></td>
<td>RTA - real time acyclic protocol</td>
</tr>
<tr>
<td></td>
<td>DCP - discovery and configuration protocol</td>
</tr>
<tr>
<td></td>
<td>CL-RPC - connectionless remote procedure Call</td>
</tr>
<tr>
<td></td>
<td>LLDP - link layer discovery protocol</td>
</tr>
<tr>
<td></td>
<td>MRP - MRP Client</td>
</tr>
<tr>
<td>Acyclic services</td>
<td>PNIO read / write sequence (max. 1024 bytes per telegram)</td>
</tr>
<tr>
<td></td>
<td>Process-Alarm service</td>
</tr>
<tr>
<td>Supported alarm types</td>
<td>Process Alarm, Diagnostic Alarm, Return of SubModule, Plug Alarm, Pull Alarm</td>
</tr>
<tr>
<td>Min. bus cycle</td>
<td>1 ms</td>
</tr>
<tr>
<td>Conformance class</td>
<td>CC A</td>
</tr>
<tr>
<td>Protective functions (according to IEC 61131-3)</td>
<td>Protected against:</td>
</tr>
<tr>
<td></td>
<td>• short circuit</td>
</tr>
<tr>
<td></td>
<td>• reverse supply</td>
</tr>
<tr>
<td></td>
<td>• overvoltage</td>
</tr>
<tr>
<td></td>
<td>• reverse polarity</td>
</tr>
<tr>
<td></td>
<td>Electrical isolation from the rest of the module</td>
</tr>
</tbody>
</table>

*) Priorization with the aid of VLAN-ID including priority level
Input/Output characteristics of CI501-PNIO

The PROFINET Bus Module CI501-PNIO is used as decentralized I/O module in PROFINET networks. The network connection is performed via 2 RJ45 connectors which are integrated in the Terminal Unit. The Bus Module contains 22 I/O channels with the following properties:

- 4 analog inputs (1.0...1.3), configurable as:
  - -10 ... +10 V
  - 0 ... +10 V
  - -10 ... +10 V (differential voltage)
  - 0 ... 20 mA
  - 4 ... 20 mA
  - Pt100, Pt1000, Ni1000 (for each 2-wire and 3-wire)
  - 24 V digital input function
- 2 analog outputs (1.5...1.6), configurable as:
  - -10 ... +10 V
  - 0 ... 20 mA
  - 4 ... 20 mA
- 8 digital inputs 24 V DC in 1 group (2.0...2.7)
- 8 digital transistor outputs 24 V DC (0.5 A max.) in 1 group (3.0...3.7)
- Resolution of the analog channels: 12 bits

The inputs/outputs are electrically isolated from the Ethernet network. There is no potential separation between the channels. The configuration of the analog inputs/outputs is performed by software.

For usage in enhanced ambient conditions (e.g. wider temperature and humidity range), a special XC version of the device is available.

Input/Output characteristics of CI502-PNIO

- 8 digital inputs 24 V DC
- 8 digital transistor outputs 24 V DC, 0.5 A max.
- 8 configurable digital inputs/outputs 24 V DC, 0.5 A max.
- Module-wise electrically isolated
- XC version for usage in extreme ambient conditions available

Technical data of the serial interfaces of CI504-PNIO

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of serial interfaces</td>
<td>3</td>
</tr>
<tr>
<td>Connectors for serial interfaces</td>
<td>X11 for COM1</td>
</tr>
<tr>
<td></td>
<td>X12 for COM2</td>
</tr>
<tr>
<td></td>
<td>X13 for COM3</td>
</tr>
<tr>
<td>Supported physical layers</td>
<td>RS-232</td>
</tr>
<tr>
<td></td>
<td>RS-422</td>
</tr>
<tr>
<td></td>
<td>RS-485</td>
</tr>
<tr>
<td>Supported protocols</td>
<td>ASCII</td>
</tr>
<tr>
<td>Transmission rate</td>
<td>Configurable from 300 bit/s to 115.200 bit/s</td>
</tr>
</tbody>
</table>
CI501-PNIO

- 4 analog inputs, 2 analog outputs, 8 digital inputs, 8 digital outputs
- Resolution 12 bits plus sign
- Module-wise electrically isolated
- Fast counter
- XC version for usage in extreme ambient conditions available

1. I/O bus
2. Allocation between terminal number and signal name
3. 6 yellow LEDs to display the signal states of the analog inputs/outputs (AI0 - AI3, AO0 - AO1)
4. 8 yellow LEDs to display the signal states of the digital inputs (DI0 - DI7)
5. 8 yellow LEDs to display the signal states of the digital outputs (DO0 - DO7)
6. 2 green LEDs to display the process supply voltage UP and UP3
7. 3 red LEDs to display errors (CH-ERR1, CH-ERR2, CH-ERR3)
8. 5 system LEDs: PWR/RUN, STA1 ETH, STA2 ETH, S-ERR, I/O-Bus
9. Label
10. 2 rotary switches for setting the I/O device identifier
11. Ethernet interfaces (ETH1, ETH2) on the terminal unit
12 Terminal unit
13 DIN rail
\(\text{\textsuperscript{\(\textcircled{\textbullet}\)}}\) Sign for XC version

**Intended purpose**

The PROFINET bus modules CI501-PNIO and CI502-PNIO are used as communication interface modules in PROFINET networks. The network connection is performed by Ethernet cables which are inserted in the RJ45 connectors in the terminal unit. An Ethernet switch in the communication interface module allows daisy chaining of the network.

For usage in enhanced ambient conditions (e.g. wider temperature and humidity range), a special XC version of the device is available.

**Functionality**

The bus module contains 22 I/O channels with the following properties:

- 4 configurable analog inputs (2-wire / single-ended) or 2 configurable analog inputs (3-wire / differential) (1.0...1.3)
- 2 analog outputs (1.5...1.6)
- 8 digital inputs 24 VDC in 1 group (2.0...2.7)
- 8 digital outputs 24 VDC, 0.5 A max. in 1 group (3.0...3.7)

The inputs/outputs are electrically isolated from the PROFINET network. There is no potential separation between the channels. The configuration of the analog inputs/outputs is performed by software.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Interface</td>
<td>Ethernet</td>
</tr>
<tr>
<td>Protocol</td>
<td>PROFINET IO RT</td>
</tr>
<tr>
<td>Power supply</td>
<td>From the process supply voltage UP</td>
</tr>
<tr>
<td>Supply of the electronic circuitry of the I/O expansion modules attached</td>
<td>Through the expansion bus interface (I/O bus)</td>
</tr>
<tr>
<td>Rotary switches</td>
<td>For setting the I/O device identifier for configuration purposes (00h to FFh)</td>
</tr>
<tr>
<td>LED displays</td>
<td>For system displays, signal states, errors and power supply</td>
</tr>
<tr>
<td>External supply voltage</td>
<td>Via terminals ZP, UP and UP3 (process supply voltage 24 VDC)</td>
</tr>
<tr>
<td>Effect of incorrect input terminal connection</td>
<td>Wrong or no signal detected, no damage up to 35 V</td>
</tr>
<tr>
<td>Required terminal unit</td>
<td>TU507 or TU508 (\text{\textsuperscript{(\textcircled{\textbullet})}}) Chapter 1.5.3.5.1 “TU507-ETH and TU508-ETH for Ethernet communication interface modules” on page 2307</td>
</tr>
</tbody>
</table>

**Electrical connection**

The Ethernet bus module CI501-PNIO is plugged on the I/O terminal unit TU507-ETH or TU508-ETH \(\text{\textsuperscript{\(\textcircled{\textbullet}\)}}\) Chapter 1.5.3.5.1 “TU507-ETH and TU508-ETH for Ethernet communication interface modules” on page 2307. Properly seat the module and press until it locks in place. The terminal unit is mounted on a DIN rail or with 2 screws plus the additional accessory for wall mounting (TA526 \(\text{\textsuperscript{\(\textcircled{\textbullet}\)}}\) Chapter 1.5.3.8.1.4 “TA526 - Wall mounting accessory” on page 3035).
The electrical connection of the I/O channels is carried out using the 30 terminals of the I/O terminal unit. I/O modules can be replaced without re-wiring the terminal units.

For a detailed description of the mounting, disassembly and electrical connection of the module, please refer to the System Assembly, Construction and Connection chapter $\text{* Chapter 1.5.4.5 “AC500 (Standard)” on page 3058.}$

The terminals 1.8 and 2.8 as well as 1.9, 2.9 and 3.9 are electrically interconnected within the terminal unit and have always the same assignment, independent of the inserted module:

Terminals 1.8 and 2.8: Process supply voltage $\text{UP} = +24 \text{ VDC}$

Terminal 3.8: Process supply voltage $\text{UP3} = +24 \text{ VDC}$

Terminals 1.9, 2.9 and 3.9: Process supply voltage $\text{ZP} = 0 \text{ V}$

With a separate UP3 power supply, the digital outputs can be switched off externally. This way, an emergency-off functionality can be realized.

Do not connect any voltages externally to digital outputs!

Reason: Externally voltages at an output or several outputs may cause that other outputs are supplied through that voltage instead of voltage UP3 (reverse voltage). This is not intended usage.

CAUTION!
Risk of malfunction by not intended usage!
If the function cut off of the digital outputs should be used by deactivation of the supply voltage UP3, be sure that no external voltage is connected at the outputs DO0...DO7.

The assignment of the other terminals:

<table>
<thead>
<tr>
<th>Terminal</th>
<th>Signal</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0</td>
<td>AI0+</td>
<td>Plus pole of analog input signal 0</td>
</tr>
<tr>
<td>1.1</td>
<td>AI1+</td>
<td>Plus pole of analog input signal 1</td>
</tr>
<tr>
<td>1.2</td>
<td>AI2+</td>
<td>Plus pole of analog input signal 2</td>
</tr>
<tr>
<td>1.3</td>
<td>AI3+</td>
<td>Plus pole of analog input signal 3</td>
</tr>
<tr>
<td>1.4</td>
<td>AI-</td>
<td>Minus pole of analog input signals 0 to 3</td>
</tr>
<tr>
<td>1.5</td>
<td>AO0+</td>
<td>Plus pole of analog output signal 0</td>
</tr>
<tr>
<td>1.6</td>
<td>AO1+</td>
<td>Plus pole of analog output signal 1</td>
</tr>
<tr>
<td>1.7</td>
<td>AI-</td>
<td>Minus pole of analog output signals 0 and 1</td>
</tr>
<tr>
<td>1.8</td>
<td>UP</td>
<td>Process voltage UP (24 VDC)</td>
</tr>
<tr>
<td>1.9</td>
<td>ZP</td>
<td>Process voltage ZP (0 VDC)</td>
</tr>
<tr>
<td>2.0</td>
<td>DI0</td>
<td>Signal of the digital input DI0</td>
</tr>
<tr>
<td>2.1</td>
<td>DI1</td>
<td>Signal of the digital input DI1</td>
</tr>
<tr>
<td>2.2</td>
<td>DI2</td>
<td>Signal of the digital input DI2</td>
</tr>
<tr>
<td>2.3</td>
<td>DI3</td>
<td>Signal of the digital input DI3</td>
</tr>
<tr>
<td>Terminal</td>
<td>Signal</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>--------</td>
<td>------------------------------</td>
</tr>
<tr>
<td>2.4</td>
<td>DI4</td>
<td>Signal of the digital input DI4</td>
</tr>
<tr>
<td>2.5</td>
<td>DI5</td>
<td>Signal of the digital input DI5</td>
</tr>
<tr>
<td>2.6</td>
<td>DI6</td>
<td>Signal of the digital input DI6</td>
</tr>
<tr>
<td>2.7</td>
<td>DI7</td>
<td>Signal of the digital input DI7</td>
</tr>
<tr>
<td>2.8</td>
<td>UP</td>
<td>Process voltage UP (24 VDC)</td>
</tr>
<tr>
<td>2.9</td>
<td>ZP</td>
<td>Process voltage ZP (0 VDC)</td>
</tr>
<tr>
<td>3.0</td>
<td>DO0</td>
<td>Signal of the digital output DO0</td>
</tr>
<tr>
<td>3.1</td>
<td>DO1</td>
<td>Signal of the digital output DO1</td>
</tr>
<tr>
<td>3.2</td>
<td>DO2</td>
<td>Signal of the digital output DO2</td>
</tr>
<tr>
<td>3.3</td>
<td>DO3</td>
<td>Signal of the digital output DO3</td>
</tr>
<tr>
<td>3.4</td>
<td>DO4</td>
<td>Signal of the digital output DO4</td>
</tr>
<tr>
<td>3.5</td>
<td>DO5</td>
<td>Signal of the digital output DO5</td>
</tr>
<tr>
<td>3.6</td>
<td>DO6</td>
<td>Signal of the digital output DO6</td>
</tr>
<tr>
<td>3.7</td>
<td>DO7</td>
<td>Signal of the digital output DO7</td>
</tr>
<tr>
<td>3.8</td>
<td>UP3</td>
<td>Process voltage UP3 (24 VDC)</td>
</tr>
<tr>
<td>3.9</td>
<td>ZP</td>
<td>Process voltage ZP (0 VDC)</td>
</tr>
</tbody>
</table>

**WARNING!**

**Removal/Insertion under power**

The devices are not designed for removal or insertion under power. Because of unforeseeable consequences, it is not allowed to plug or unplug devices with the power being ON.

Make sure that all voltage sources (supply and process voltage) are switched off before you
- connect or disconnect any signal or terminal block
- remove, mount or replace a module.

Disconnecting any powered devices while energized in a hazardous location could result in an electric arc, which could create a flammable ignition resulting in fire or explosion.

Make sure that power is removed and that the area has been thoroughly checked to ensure that flammable materials are not present prior to proceeding.

The devices must not be opened when in operation. The same applies to the network interfaces.

**NOTICE!**

**Risk of damaging the PLC modules!**

Overvoltages and short circuits might damage the PLC modules.
- Make sure that all voltage sources (supply voltage and process supply voltage) are switched off before you begin with operations at the system.
- Never connect any voltages or signals to reserved terminals (marked with ---). Reserved terminals may carry internal voltages.
For the open-circuit detection (cut wire), each analog input channel is pulled up to "plus" by a high-resistance resistor. If nothing is connected, the maximum voltage will be read in then.

Generally, analog signals must be laid in shielded cables. The cable shields must be earthed at both sides of the cables. In order to avoid unacceptable potential differences between different parts of the installation, low resistance equipotential bonding conductors must be laid.

Only for simple applications (low electromagnetic disturbances, no high requirement on precision), the shielding can also be omitted.

The following figures show the electrical connection of the Ethernet bus module CI501-PNIO.

Further information is provided in the System Technology chapter PROFINET.

Connection of the digital inputs

The following figure shows the electrical connection of the digital input DI0. Proceed with the digital inputs DI1 to DI7 in the same way.
The meaning of the LEDs is described in Displays Chapter 1.5.3.5.2.8.2 “State LEDs” on page 2993.

**Connection of the digital outputs**

The following figure shows the electrical connection of the digital output DO0. Proceed with the digital outputs DO1 - DO7 in the same way.
The meaning of the LEDs is described in Displays \( \Leftrightarrow \) Chapter 1.5.3.7.5.2.8.2 “State LEDs” on page 2993.

**Connection of resistance thermometers in 2-wire configuration to the analog inputs**

When resistance thermometers (Pt100, Pt1000, Ni1000) are used, a constant current must flow through them to build the necessary voltage drop for the evaluation. For this, the module CI501-PNIO provides a constant current source which is multiplexed over the max. 4 analog input channels.

The following figure shows the connection of resistance thermometers in 2-wire configuration to the analog input AI0. Proceed with the analog inputs AI1 to AI3 in the same way.
The following measuring ranges can be configured “Parameterization” on page 2982 “Input ranges voltage, current and digital input” on page 2995:

<table>
<thead>
<tr>
<th>Sensor</th>
<th>Measuring Range</th>
<th>Configuration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pt100</td>
<td>-50 °C...+400 °C</td>
<td>2-wire configuration, 1 channel used</td>
</tr>
<tr>
<td>Pt1000</td>
<td>-50 °C...+400 °C</td>
<td>2-wire configuration, 1 channel used</td>
</tr>
<tr>
<td>Ni1000</td>
<td>-50 °C...+150 °C</td>
<td>2-wire configuration, 1 channel used</td>
</tr>
</tbody>
</table>

The function of the LEDs is described under Diagnosis and displays / Displays “Diagnosis and state LEDs” on page 2988.

The module CI501-PNIO performs a linearization of the resistance characteristic.

To avoid error messages from unused analog input channels, configure them as "unused".

**Connection of resistance thermometers in 3-wire configuration to the analog inputs**

When resistance thermometers (Pt100, Pt1000, Ni1000) are used, a constant current must flow through them to build the necessary voltage drop for the evaluation. For this, the module CI501-PNIO provides a constant current source which is multiplexed over the max. 4 analog input channels.

The following figure shows the connection of resistance thermometers in 3-wire configuration to the analog inputs AI0 and AI1. Proceed with the analog inputs AI2 and AI3 in the same way.
With 3-wire configuration, 2 adjacent analog channels belong together (e.g. the channels 0 and 1). In this case, both channels are configured according to the desired operating mode. The lower address must be the even address (channel 0), the next higher address must be the odd address (channel 1).

The constant current of one channel flows through the resistance thermometer. The constant current of the other channel flows through one of the cores. The module calculates the measured value from the two voltage drops and stores it under the input with the higher channel number (e.g. 1).

In order to keep measuring errors as small as possible, it is necessary to have all the involved conductors in the same cable. All the conductors must have the same cross section.

The following measuring ranges can be configured: Chapter 1.5.3.7.5.2.7 "Parameterization" on page 2982 Chapter 1.5.3.7.5.2.9.1 "Input ranges voltage, current and digital input" on page 2995:

<table>
<thead>
<tr>
<th>Sensor</th>
<th>Temperature Range</th>
<th>Configuration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pt100</td>
<td>-50 °C...+70 °C</td>
<td>3-wire config. 2 channels used</td>
</tr>
<tr>
<td>Pt100</td>
<td>-50 °C...+400 °C</td>
<td>3-wire config. 2 channels used</td>
</tr>
<tr>
<td>Pt1000</td>
<td>-50 °C...+400 °C</td>
<td>3-wire config. 2 channels used</td>
</tr>
<tr>
<td>Ni1000</td>
<td>-50 °C...+150 °C</td>
<td>3-wire config. 2 channels used</td>
</tr>
</tbody>
</table>

The function of the LEDs is described under Diagnosis and displays / Displays Chapter 1.5.3.7.5.2.8 "Diagnosis and state LEDs" on page 2988.

The module CI501-PNIO performs a linearization of the resistance characteristic.

To avoid error messages from unused analog input channels, configure them as "unused".
Connection of active-type analog sensors (Voltage) with electrically isolated power supply to the analog inputs

The following figure shows the connection of active-type analog sensors (voltage) with electrically isolated power supply to the analog input AI0. Proceed with the analog inputs AI1 to AI3 in the same way.

The following measuring ranges can be configured:

<table>
<thead>
<tr>
<th>Voltage</th>
<th>0 V...10 V</th>
<th>1 channel used</th>
</tr>
</thead>
<tbody>
<tr>
<td>Voltage</td>
<td>-10 V...+10 V</td>
<td>1 channel used</td>
</tr>
</tbody>
</table>

The function of the LEDs is described under Diagnosis and displays.

To avoid error messages from unused analog input channels, configure them as "unused".

Connection of active-type analog sensors (Current) with electrically isolated power supply to the analog inputs

The following figure shows the connection of active-type analog sensors (current) with electrically isolated power supply to the analog input AI0. Proceed with the analog inputs AI1 to AI3 in the same way.
The following measuring ranges can be configured in Chapter 1.5.3.7.2.7 “Parameterization” on page 2982 and Chapter 1.5.3.7.2.9.1 “Input ranges voltage, current and digital input” on page 2995:

<table>
<thead>
<tr>
<th>Current</th>
<th>0 mA...20 mA</th>
<th>1 channel used</th>
</tr>
</thead>
<tbody>
<tr>
<td>Current</td>
<td>4 mA...20 mA</td>
<td>1 channel used</td>
</tr>
</tbody>
</table>

The function of the LEDs is described under Diagnosis and displays / Displays in Chapter 1.5.3.7.2.8 “Diagnosis and state LEDs” on page 2988.

Unused input channels can be left open-circuited, because they are of low resistance.

To avoid error messages through unused analog input channels in measuring range 4 mA...20 mA, these channels should be configured as "Not used".

Connection of active-type analog sensors (Voltage) with no electrically isolated power supply to the analog inputs

The following figure shows the connection of active-type analog sensors (voltage) with no electrically isolated power supply to the analog input AI0. Proceed with the analog inputs AI1 to AI3 in the same way.
CAUTION!
Risk of faulty measurements!
The minus pole at the sensors must not have a too big potential difference with respect to ZP (max. ±1 V).

Make sure that the potential difference never exceeds ±1 V (also not with long cable lengths).

The following measuring ranges can be configured: Chapter 1.5.3.7.5.2.7 “Parameterization” on page 2982.

<table>
<thead>
<tr>
<th>Voltage</th>
<th>0 V...10 V</th>
<th>1 channel used</th>
</tr>
</thead>
<tbody>
<tr>
<td>Voltage</td>
<td>-10 V...+10 V</td>
<td>1 channel used</td>
</tr>
</tbody>
</table>

The function of the LEDs is described under Diagnosis and displays / Displays Chapter 1.5.3.7.5.2.8 “Diagnosis and state LEDs” on page 2988.

To avoid error messages from unused analog input channels, configure them as "unused".

Connection of passive-type analog sensors (Current) to the analog inputs

The following figure shows the connection of passive-type analog sensors (current) to the analog input AI0. Proceed with the analog inputs AI1 to AI3 in the same way.
The following measuring ranges can be configured on page 2982:

- Current
  - 4 mA...20 mA
  - 1 channel used

The function of the LEDs is described under Diagnosis and displays on page 2988.

**CAUTION!**
Risk of overloading the analog input!

If an analog current sensor supplies more than 25 mA for more than 1 second during initialization, this input is switched off by the module (input protection).

Use only sensors with fast initialization or without current peaks higher than 25 mA. If not possible, connect a 10-volt zener diode in parallel to AIx+ and ZP.

Unused input channels can be left open-circuited, because they are of low resistance.

To avoid error messages through unused analog input channels in measuring range 4 mA...20 mA, these channels should be configured as "Not used".

**Connection of active-type analog sensors (Voltage) to differential analog inputs**

Differential inputs are very useful, if analog sensors are used which are remotely non-isolated (e.g. the minus terminal is remotely earthed).

The evaluation using differential inputs helps to considerably increase the measuring accuracy and to avoid earthing loops.
With differential input configurations, two adjacent analog channels belong together (e.g. the channels 0 and 1). In this case, both channels are configured according to the desired operating mode. The lower address must be the even address (channel 0), the next higher address must be the odd address (channel 1). The converted analog value is available at the higher address (channel 1).

The analog value is calculated by subtraction of the input value with the higher address from the input value of the lower address.

The converted analog value is available at the odd channel (higher address).

**CAUTION!**
**Risk of faulty measurements!**
The minus pole at the sensors must not have a too big potential difference with respect to ZP (max. ±1 V).

Make sure that the potential difference never exceeds ±1 V.

The following figure shows the connection of active-type analog sensors (voltage) to differential analog inputs AI0 and AI1. Proceed with AI2 and AI3 in the same way.

The following measuring ranges can be configured:

<table>
<thead>
<tr>
<th>Voltage</th>
<th>0 V...10 V</th>
<th>With differential inputs, 2 channels used</th>
</tr>
</thead>
<tbody>
<tr>
<td>Voltage</td>
<td>-10 V...+10 V</td>
<td>With differential inputs, 2 channels used</td>
</tr>
</tbody>
</table>

The function of the LEDs is described under Diagnosis and displays / Displays. See Chapter 1.5.3.7.5.2.8 “Diagnosis and state LEDs” on page 2988.
To avoid error messages from unused analog input channels, configure them as “unused”.

**Use of analog inputs as digital inputs**

Several (or all) analog inputs can be configured as digital inputs. The inputs are not electrically isolated against the other analog channels.

The following figure shows the connection of digital sensors to the analog input AI0. Proceed with the analog inputs AI1 to AI3 in the same way.

![Fig. 246: Use of analog inputs as digital inputs](image)

The following measuring ranges can be configured "Parameterization" on page 2982 "Input ranges voltage, current and digital input" on page 2995:

<table>
<thead>
<tr>
<th>Digital input</th>
<th>24 V</th>
<th>1 channel used</th>
</tr>
</thead>
<tbody>
<tr>
<td>Effect of incorrect input terminal connection</td>
<td>Wrong or no signal detected, no damage up to 35 V</td>
<td></td>
</tr>
</tbody>
</table>

The function of the LEDs is described under Diagnosis and displays / Displays "Diagnosis and state LEDs" on page 2988.

**Connection of analog output loads (Voltage)**

The following figure shows the connection of output loads to the analog output AO0. Proceed with the analog output AO1 in the same way.
Fig. 247: Connection of analog output loads (voltage)

The following measuring ranges can be configured \(\text{\textcopyright Chapter 1.5.3.7.5.2.7 "Parameterization" on page 2982 \textcopyright Chapter 1.5.3.7.5.2.9.1 "Input ranges voltage, current and digital input" on page 2995.}

<table>
<thead>
<tr>
<th>Voltage</th>
<th>-10 V...+10 V</th>
<th>Load ±10 mA max.</th>
<th>1 channel used</th>
</tr>
</thead>
</table>

The function of the LEDs is described under Diagnosis and displays / Displays \(\text{\textcopyright Chapter 1.5.3.7.5.2.8 "Diagnosis and status LEDs" on page 2988.}

Unused analog outputs can be left open-circuited.

**Connection of analog output loads (Current)**

The following figure shows the connection of output loads to the analog output AO0. Proceed with the analog output AO1 in the same way.
The following measuring ranges can be configured “Parameterization” on page 2982 “Input ranges voltage, current and digital input” on page 2995:

<table>
<thead>
<tr>
<th>Current</th>
<th>Load</th>
<th>1 channel used</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 mA...20 mA</td>
<td>0 Ω...500 Ω</td>
<td></td>
</tr>
<tr>
<td>4 mA...20 mA</td>
<td>0 Ω...500 Ω</td>
<td></td>
</tr>
</tbody>
</table>

The function of the LEDs is described under Diagnosis and displays “Diagnosis and state LEDs” on page 2988.

Unused analog outputs can be left open-circuited.

Assignment of the Ethernet ports

The terminal unit for the communication interface module provides two Ethernet interfaces with the following pin assignment:

<table>
<thead>
<tr>
<th>Interface</th>
<th>Pin</th>
<th>Signal</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ethernet RJ45</td>
<td>1</td>
<td>TxD+</td>
<td>Transmit data +</td>
</tr>
<tr>
<td>2</td>
<td>TxD-</td>
<td>Transmit data -</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>RxD+</td>
<td>Receive data +</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>NC</td>
<td>not used</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>NC</td>
<td>not used</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>RxD-</td>
<td>Receive data -</td>
<td></td>
</tr>
<tr>
<td>Interface</td>
<td>Pin</td>
<td>Signal</td>
<td>Description</td>
</tr>
<tr>
<td>-----------</td>
<td>-----</td>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td>7</td>
<td>NC</td>
<td>not used</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>NC</td>
<td>not used</td>
<td></td>
</tr>
<tr>
<td>Shield</td>
<td></td>
<td>Cable shield</td>
<td>Functional earth</td>
</tr>
</tbody>
</table>

For further information regarding wiring and cable types see chapter Ethernet Chapter 1.5.4.5.4.7 “Ethernet connection details” on page 3084.

Internal data exchange

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Digital inputs (bytes)</td>
<td>3</td>
</tr>
<tr>
<td>Digital outputs (bytes)</td>
<td>3</td>
</tr>
<tr>
<td>Analog inputs (words)</td>
<td>4</td>
</tr>
<tr>
<td>Analog outputs (words)</td>
<td>2</td>
</tr>
<tr>
<td>Counter input data (words)</td>
<td>4</td>
</tr>
<tr>
<td>Counter output data (words)</td>
<td>8</td>
</tr>
</tbody>
</table>

Addressing

The module reads the position of the rotary switches only during power-up, i.e. changes of the switch position during operation will have no effect until the next module initialization.

I/O configuration

The CI501-PNIO stores some PROFINET configuration parameters (I/O device identifier, I/O device type and IP address configuration). No more configuration data is stored.

The analog/digital I/O channels are configured via software.

Details about configuration are described in Parameterization Chapter 1.5.3.7.5.2.7 “Parameterization” on page 2982.

Parameterization

Parameters of the module

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>Internal value</th>
<th>Internal value, type</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>Module ID 1)</td>
<td>Internal</td>
<td>7000</td>
<td>WORD</td>
<td>7000</td>
</tr>
<tr>
<td>Parameter length</td>
<td>Internal</td>
<td>25</td>
<td>BYTE</td>
<td>25</td>
</tr>
<tr>
<td>Name</td>
<td>Value</td>
<td>Internal value</td>
<td>Internal value, type</td>
<td>Default</td>
</tr>
<tr>
<td>------------------------------------------------</td>
<td>------------------------------</td>
<td>----------------</td>
<td>----------------------</td>
<td>---------</td>
</tr>
<tr>
<td>Error LED / Failsafe function see Table 634 “Error LED / Failsafe function” on page 2984</td>
<td>On</td>
<td>0</td>
<td>BYTE</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>Off by E4</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Off by E3</td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>On + failsafe</td>
<td>16</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Off by E4 + failsafe</td>
<td>17</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Off by E3 + failsafe</td>
<td>19</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Process cycle time ²)</td>
<td>1 ms process cycle time</td>
<td>1</td>
<td>BYTE</td>
<td>1 ms</td>
</tr>
<tr>
<td></td>
<td>2 ms process cycle time</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>3 ms process cycle time</td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>4 ms process cycle time</td>
<td>4</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>5 ms process cycle time</td>
<td>5</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>6 ms process cycle time</td>
<td>6</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>7 ms process cycle time</td>
<td>7</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>8 ms process cycle time</td>
<td>8</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>9 ms process cycle time</td>
<td>9</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>10 ms process cycle time</td>
<td>10</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>11 ms process cycle time</td>
<td>11</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>12 ms process cycle time</td>
<td>12</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>13 ms process cycle time</td>
<td>13</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>14 ms process cycle time</td>
<td>14</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>15 ms process cycle time</td>
<td>15</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>16 ms process cycle time</td>
<td>16</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Check supply</td>
<td>off</td>
<td>0</td>
<td>BYTE</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>on</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Fast counter</td>
<td>0</td>
<td>0</td>
<td>BYTE</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>: 10 ³)</td>
<td>10</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Remarks:

²) Table 634 “Error LED / Failsafe function” on page 2984
3) The value 10 is used for mistake.
With a faulty ID, the modules reports a "parameter error" and does not perform cyclic process data transmission.

As for device index C0 the parameter is no longer evaluated.

Counter operating modes, see description of the Fast counter § Chapter 1.5.3.6.1.2.9 “Fast counter” on page 2543.

<table>
<thead>
<tr>
<th>Setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>On</td>
<td>Error LED (S-ERR) lights up at errors of all error classes, Failsafe-mode off</td>
</tr>
<tr>
<td>Off by E4</td>
<td>Error LED (S-ERR) lights up at errors of error classes E1, E2 and E3, Failsafe-mode off</td>
</tr>
<tr>
<td>Off by E3</td>
<td>Error LED (S-ERR) lights up at errors of error classes E1 and E2, Failsafe-mode off</td>
</tr>
<tr>
<td>On +Failsafe</td>
<td>Error LED (S-ERR) lights up at errors of all error classes, Failsafe-mode on *)</td>
</tr>
<tr>
<td>Off by E4 + Failsafe</td>
<td>Error LED (S-ERR) lights up at errors of error classes E1, E2 and E3, Failsafe-mode on *)</td>
</tr>
<tr>
<td>Off by E3 + Failsafe</td>
<td>Error LED (S-ERR) lights up at errors of error classes E1 and E2, Failsafe-mode on *)</td>
</tr>
</tbody>
</table>

*) The parameters Behaviour AO at comm. error and Behaviour DO at comm. error are only analyzed if the Failsafe-mode is ON.

Group parameters for the analog part

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>Internal value</th>
<th>Internal value, type</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>Analog data format</td>
<td>Standard Reserved</td>
<td>0 255</td>
<td>BYTE</td>
<td>0</td>
</tr>
<tr>
<td>Behaviour AO at comm. error *)</td>
<td>Off</td>
<td>0 1 6 11 2 7 12</td>
<td>BYTE</td>
<td>0</td>
</tr>
</tbody>
</table>

*) The parameter Behaviour AO at comm. error is only analyzed if the Failsafe-mode is ON.
### Channel parameters for the analog inputs (4x)

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>Internal value</th>
<th>Internal value, type</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input 0, Channel configuration</td>
<td>Table Operating modes of the analog inputs ↪ Table 635 “Channel configuration” on page 2985</td>
<td>Table Operating modes of the analog inputs ↪ Table 635 “Channel configuration” on page 2985</td>
<td>BYTE</td>
<td>0</td>
</tr>
<tr>
<td>Input 0, Check channel</td>
<td>Table Channel monitoring ↪ Table 636 “Channel monitoring” on page 2986</td>
<td>Table Channel monitoring ↪ Table 636 “Channel monitoring” on page 2986</td>
<td>BYTE</td>
<td>0</td>
</tr>
<tr>
<td>Input 3, Channel configuration</td>
<td>Table Operating modes of the analog inputs ↪ Table 635 “Channel configuration” on page 2985</td>
<td>Table Operating modes of the analog inputs ↪ Table 635 “Channel configuration” on page 2985</td>
<td>BYTE</td>
<td>0</td>
</tr>
<tr>
<td>Input 3, Check channel</td>
<td>Table Channel monitoring ↪ Table 636 “Channel monitoring” on page 2986</td>
<td>Table Channel monitoring ↪ Table 636 “Channel monitoring” on page 2986</td>
<td>BYTE</td>
<td>0</td>
</tr>
</tbody>
</table>

**Table 635: Channel configuration**

<table>
<thead>
<tr>
<th>Internal value</th>
<th>Operating modes of the analog inputs, individually configurable</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 (default)</td>
<td>Not used</td>
</tr>
<tr>
<td>1</td>
<td>0 V...10 V</td>
</tr>
<tr>
<td>2</td>
<td>Digital input</td>
</tr>
<tr>
<td>3</td>
<td>0 mA...20 mA</td>
</tr>
<tr>
<td>4</td>
<td>4 mA...20 mA</td>
</tr>
<tr>
<td>5</td>
<td>-10 V...+10 V</td>
</tr>
<tr>
<td>8</td>
<td>2-wire Pt100 -50 °C...+400 °C</td>
</tr>
<tr>
<td>9</td>
<td>3-wire Pt100 -50 °C...+400 °C *)</td>
</tr>
<tr>
<td>10</td>
<td>0 V...10 V (voltage diff.) *)</td>
</tr>
<tr>
<td>11</td>
<td>-10 V...+10 V (voltage diff.) *)</td>
</tr>
<tr>
<td>14</td>
<td>2-wire Pt100 -50 °C...+70 °C</td>
</tr>
<tr>
<td>15</td>
<td>3-wire Pt100 -50 °C...+70 °C *)</td>
</tr>
<tr>
<td>16</td>
<td>2-wire Pt1000 -50 °C...+400 °C</td>
</tr>
<tr>
<td>17</td>
<td>3-wire Pt1000 -50 °C...+400 °C *)</td>
</tr>
<tr>
<td>18</td>
<td>2-wire Ni1000 -50 °C...+150 °C</td>
</tr>
</tbody>
</table>
*) In the operating modes with 3-wire configuration or with differential inputs, two adjacent analog inputs belong together (e.g. the channels 0 and 1). In these cases, both channels are configured in the desired operating mode. The lower address must be the even address (channel 0). The next higher address must be the odd address (channel 1). The converted analog value is available at the higher address (channel 1).

<table>
<thead>
<tr>
<th>Internal Value</th>
<th>Operating modes of the analog inputs, individually configurable</th>
</tr>
</thead>
<tbody>
<tr>
<td>19</td>
<td>3-wire Ni1000 -50 °C...+150 °C *)</td>
</tr>
</tbody>
</table>

Table 636: Channel monitoring

<table>
<thead>
<tr>
<th>Internal Value</th>
<th>Check Channel</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 (default)</td>
<td>Plausibility, cut wire, short circuit</td>
</tr>
<tr>
<td>3</td>
<td>Not used</td>
</tr>
</tbody>
</table>

Channel parameters for the analog outputs (2x)

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output 0, Channel configuration</td>
<td>Table Operating modes of the analog outputs \Further information on page 2987</td>
</tr>
<tr>
<td>Internal value</td>
<td>Table Operating modes of the analog outputs \Further information on page 2987</td>
</tr>
<tr>
<td>Default</td>
<td>BYTE</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

| Output 0, Check channel     | Table Channel monitoring \Table 638 “Channel monitoring” on page 2987 |
| Internal value              | Table Channel monitoring \Table 638 “Channel monitoring” on page 2987 |
| Default                     | BYTE                                                                  |
| 0                           | 0                                                                    |

| Output 0, Substitute value  | Table Substitute value \Table 639 “Substitute value” on page 2987 |
| Internal value              | Table Substitute value \Table 639 “Substitute value” on page 2987 |
| Default                     | WORD                                                                 |
| 0                           | 0                                                                    |

| Output 1, Channel configuration | Table Operating modes of the analog outputs \Further information on page 2987 |
| Internal value              | Table Operating modes of the analog outputs \Further information on page 2987 |
| Default                     | BYTE                                                                  |
| 0                           | 0                                                                    |

| Output 1, Check channel     | Table Channel monitoring \Table 638 “Channel monitoring” on page 2987 |
| Internal value              | Table Channel monitoring \Table 638 “Channel monitoring” on page 2987 |
| Default                     | BYTE                                                                  |
| 0                           | 0                                                                    |

| Output 1, Substitute value  | Table Substitute value \Table 639 “Substitute value” on page 2987 |
| Internal value              | Table Substitute value \Table 639 “Substitute value” on page 2987 |
| Default                     | WORD                                                                 |
| 0                           | 0                                                                    |
### Table 637: Channel configuration

<table>
<thead>
<tr>
<th>Internal value</th>
<th>Operating modes of the analog outputs, individually configurable</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 (default)</td>
<td>Not used</td>
</tr>
<tr>
<td>128</td>
<td>-10 V...+10 V</td>
</tr>
<tr>
<td>129</td>
<td>0 mA...20 mA</td>
</tr>
<tr>
<td>130</td>
<td>4 mA...20 mA</td>
</tr>
</tbody>
</table>

### Table 638: Channel monitoring

<table>
<thead>
<tr>
<th>Internal value</th>
<th>Check channel</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Plausibility, cut wire, short circuit</td>
</tr>
<tr>
<td>3</td>
<td>None</td>
</tr>
</tbody>
</table>

### Table 639: Substitute value

<table>
<thead>
<tr>
<th>Intended behaviour of output channel when the control system stops</th>
<th>Required setting of the module parameter &quot;Behaviour of outputs in case of a communication error&quot;</th>
<th>Required setting of the channel parameter &quot;Substitute value&quot;</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output OFF</td>
<td>Off</td>
<td>0</td>
</tr>
<tr>
<td>Last value infinite</td>
<td>Last value</td>
<td>0</td>
</tr>
<tr>
<td>Last value for 5 s and then turn off</td>
<td>Last value 5 sec</td>
<td>0</td>
</tr>
<tr>
<td>Last value for 10 s and then turn off</td>
<td>Last value 10 sec</td>
<td>0</td>
</tr>
<tr>
<td>Substitute value infinite</td>
<td>Substitute value</td>
<td>Depending on configuration</td>
</tr>
<tr>
<td>Substitute value for 5 s and then turn off</td>
<td>Substitute value 5 sec</td>
<td>Depending on configuration</td>
</tr>
<tr>
<td>Substitute value for 10 s and then turn off</td>
<td>Substitute value 10 sec</td>
<td>Depending on configuration</td>
</tr>
</tbody>
</table>

### Group parameters for the digital part

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>Internal value</th>
<th>Internal value, type</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input delay</td>
<td>0.1 ms</td>
<td>0</td>
<td>BYTE</td>
<td>0.1 ms 0x00</td>
</tr>
<tr>
<td></td>
<td>1 ms</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>8 ms</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>32 ms</td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Detect short circuit at outputs</td>
<td>Off</td>
<td>0</td>
<td>BYTE</td>
<td>On 0x01</td>
</tr>
<tr>
<td></td>
<td>On</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Name</td>
<td>Value</td>
<td>Internal value</td>
<td>Internal value, type</td>
<td>Default</td>
</tr>
<tr>
<td>-------------------------------------------</td>
<td>------------------------</td>
<td>----------------</td>
<td>----------------------</td>
<td>---------------</td>
</tr>
<tr>
<td>Behaviour DO at comm. error ¹)</td>
<td>Off</td>
<td>0</td>
<td>BYTE</td>
<td>Off 0x00</td>
</tr>
<tr>
<td></td>
<td>Last value</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Last value 5 sec</td>
<td>6</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Last value 10 sec</td>
<td>11</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Substitute value</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Substitute value 5 sec</td>
<td>7</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Substitute value 10 sec</td>
<td>12</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Substitute value at output</td>
<td>0...255</td>
<td>00h...FFh</td>
<td>BYTE</td>
<td>0 0x0000</td>
</tr>
<tr>
<td>Detect voltage overflow at outputs ²)</td>
<td>Off</td>
<td>0</td>
<td>BYTE</td>
<td>On 0x01</td>
</tr>
<tr>
<td></td>
<td>On</td>
<td>1</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

¹) The parameters Behaviour DO at comm. error is only analyzed if the Failsafe-mode is ON.
²) The state "externally voltage detected" appears, if the output of a channel DC0...DC7 should be switched on while an externally voltage is connected. In this case the start up is disabled, as long as the externally voltage is connected. The monitoring of this state and the resulting diagnosis message can be disabled by setting the parameters to "OFF".

Diagnosis and state LEDs

Structure of the diagnosis block via PNIO_DEV_ALARM function block

<table>
<thead>
<tr>
<th>Byte Number</th>
<th>Description</th>
<th>Possible Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Diagnosis Byte, slot number</td>
<td>31 = CI501-PNIO (e. g. error at integrated 8 DI / 8 DO)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 = 1st connected S500 I/O module</td>
</tr>
<tr>
<td></td>
<td></td>
<td>...</td>
</tr>
<tr>
<td></td>
<td></td>
<td>10 = 10th connected S500 I/O module</td>
</tr>
<tr>
<td>2</td>
<td>Diagnosis Byte, module number</td>
<td>According to the I/O bus specification passed on by modules to the fieldbus master</td>
</tr>
<tr>
<td>3</td>
<td>Diagnosis Byte, channel</td>
<td>According to the I/O bus specification passed on by modules to the fieldbus master</td>
</tr>
<tr>
<td>Byte Number</td>
<td>Description</td>
<td>Possible Values</td>
</tr>
<tr>
<td>-------------</td>
<td>----------------------------------</td>
<td>----------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>4</td>
<td>Diagnosis Byte, error code</td>
<td>According to the I/O bus specification</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Bit 7 and bit 6, coded error class</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 = E1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 = E2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2 = E3</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3 = E4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Bit 0 to bit 5, coded error description</td>
</tr>
<tr>
<td>5</td>
<td>Diagnosis Byte, flags</td>
<td>According to the I/O bus specification</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Bit 7: 1 = coming error</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Bit 6: 1 = leaving error</td>
</tr>
</tbody>
</table>

In cases of short circuit or overload, the digital outputs are turned off. The modules performs reactivation automatically. Thus an acknowledgement of the errors is not necessary. The error message is stored via the LED.

<table>
<thead>
<tr>
<th>E1...E4</th>
<th>d1</th>
<th>d2</th>
<th>d3</th>
<th>d4</th>
<th>Identifier</th>
<th>AC500-Display</th>
<th>← Display in</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class</td>
<td>Comp</td>
<td>Dev</td>
<td>Mod</td>
<td>Ch</td>
<td>Err</td>
<td>PS501 PLC Browser</td>
<td></td>
</tr>
<tr>
<td>Byte 4 Bit 6...7</td>
<td>Byte 1</td>
<td>Byte 2</td>
<td>Byte 3</td>
<td>Byte 4 Bit 0...5</td>
<td>PNIO diagnosis block</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Class</th>
<th>Interface</th>
<th>Device</th>
<th>Module</th>
<th>Channel</th>
<th>Error-Identifier</th>
<th>Error message</th>
<th>Remedy</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1)</td>
<td>2)</td>
<td>3)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Module errors

<table>
<thead>
<tr>
<th>Module</th>
<th>Error Identifier</th>
<th>Error message</th>
<th>Remedy</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>19</td>
<td>Checksum error in the I/O module</td>
<td>Replace I/O module</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>Timeout in the I/O module</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>40</td>
<td>Different hard-/firmware versions in the module</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>43</td>
<td>Internal error in the module</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>36</td>
<td>Internal data exchange failure</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>9</td>
<td>Overflow diagnosis buffer</td>
<td>Restart</td>
</tr>
<tr>
<td>3</td>
<td>26</td>
<td>Parameter error</td>
<td>Check master</td>
</tr>
<tr>
<td>Class</td>
<td>Interface</td>
<td>Device</td>
<td>Module</td>
</tr>
<tr>
<td>-------</td>
<td>-----------</td>
<td>--------</td>
<td>--------</td>
</tr>
<tr>
<td>3</td>
<td>-</td>
<td>31</td>
<td>31</td>
</tr>
<tr>
<td>3</td>
<td>-</td>
<td>31</td>
<td>31</td>
</tr>
<tr>
<td>3</td>
<td>-</td>
<td>31/1...10</td>
<td>31</td>
</tr>
<tr>
<td>3</td>
<td>-</td>
<td>1...10</td>
<td>31</td>
</tr>
<tr>
<td>4</td>
<td>-</td>
<td>1...10</td>
<td>31</td>
</tr>
<tr>
<td>4</td>
<td>-</td>
<td>1...10</td>
<td>31</td>
</tr>
<tr>
<td>4</td>
<td>-</td>
<td>1...10</td>
<td>31</td>
</tr>
<tr>
<td>4</td>
<td>-</td>
<td>1...10</td>
<td>31</td>
</tr>
<tr>
<td>E1...E4</td>
<td>d1</td>
<td>d2</td>
<td>d3</td>
</tr>
<tr>
<td>--------</td>
<td>----</td>
<td>----</td>
<td>----</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Class</td>
<td>Comp</td>
<td>Dev</td>
<td>Mod</td>
</tr>
<tr>
<td>Byte 4</td>
<td>-</td>
<td>Byte 1</td>
<td>Byte 2</td>
</tr>
<tr>
<td>Byte 6...7</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Class</th>
<th>Interface</th>
<th>Device</th>
<th>Module</th>
<th>Channel</th>
<th>Error-Identifier</th>
<th>Error message</th>
<th>Remedy</th>
</tr>
</thead>
<tbody>
<tr>
<td>1)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>-</td>
<td>1...10</td>
<td>31</td>
<td>5</td>
<td>54</td>
<td>I/O module does not support hot swap 5)</td>
<td>Power off system and replace I/O module</td>
</tr>
<tr>
<td>4</td>
<td>-</td>
<td>1...10</td>
<td>31</td>
<td>6</td>
<td>42</td>
<td>No communication with hot-swap terminal unit 5)</td>
<td>Restart, if error persists replace terminal unit</td>
</tr>
<tr>
<td>4</td>
<td>-</td>
<td>31</td>
<td>31</td>
<td>31</td>
<td>46</td>
<td>Voltage feedback on activated digital outputs DO0...DO7 on UP3 4)</td>
<td>Check terminals</td>
</tr>
<tr>
<td>4</td>
<td>-</td>
<td>31/1...10</td>
<td>31</td>
<td>31</td>
<td>34</td>
<td>No response during initialization of the I/O module</td>
<td>Replace I/O module</td>
</tr>
<tr>
<td>4</td>
<td>-</td>
<td>31</td>
<td>31</td>
<td>31</td>
<td>11</td>
<td>Process voltage UP3 too low</td>
<td>Check process supply voltage</td>
</tr>
<tr>
<td>4</td>
<td>1...6</td>
<td>255</td>
<td>2</td>
<td>0</td>
<td>45</td>
<td>The connected Communication Module has no connection to the network</td>
<td>Check cabling</td>
</tr>
<tr>
<td>4</td>
<td>-</td>
<td>31</td>
<td>31</td>
<td>31</td>
<td>45</td>
<td>No process voltage UP3</td>
<td>Check process supply voltage</td>
</tr>
<tr>
<td>4</td>
<td>-</td>
<td>31</td>
<td>31</td>
<td>31</td>
<td>10</td>
<td>Voltage overflow on outputs (above UP3 level) 5)</td>
<td>Check terminals/check process supply voltage</td>
</tr>
</tbody>
</table>

Channel error digital
<table>
<thead>
<tr>
<th>E1...E4</th>
<th>d1</th>
<th>d2</th>
<th>d3</th>
<th>d4</th>
<th>Identifier</th>
<th>AC500-Display</th>
<th>&lt;= Display in</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class</td>
<td>Comp</td>
<td>Dev</td>
<td>Mod</td>
<td>Ch</td>
<td>Err</td>
<td>PS501 PLC Browser</td>
<td></td>
</tr>
<tr>
<td>Byte 4 Bit 6...7</td>
<td>-</td>
<td>Byte 1</td>
<td>Byte 2</td>
<td>Byte 3</td>
<td>Byte 4 Bit 0...5</td>
<td>PNIO diagnosis block</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Class</th>
<th>Interface</th>
<th>Device</th>
<th>Module</th>
<th>Channel</th>
<th>Error-Identifier</th>
<th>Error message</th>
<th>Remedy</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>-</td>
<td>31</td>
<td>2</td>
<td>0...7</td>
<td>46</td>
<td>Externally voltage detected at digital output DO0...DO7 6)</td>
<td>Check terminals</td>
</tr>
<tr>
<td>4</td>
<td>-</td>
<td>31</td>
<td>2</td>
<td>0...7</td>
<td>47</td>
<td>Short circuit at digital output 7)</td>
<td>Check terminals</td>
</tr>
</tbody>
</table>

Channel error analog

<p>| | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>-</td>
<td>31</td>
<td>1</td>
<td>0...3</td>
<td>48</td>
<td>Analog value overflow or broken wire at an analog input</td>
<td>Check value or check terminals</td>
</tr>
<tr>
<td>4</td>
<td>-</td>
<td>31</td>
<td>1</td>
<td>0...3</td>
<td>7</td>
<td>Analog value underflow at an analog input</td>
<td>Check value</td>
</tr>
<tr>
<td>4</td>
<td>-</td>
<td>31</td>
<td>1</td>
<td>0...3</td>
<td>47</td>
<td>Short circuit at an analog input</td>
<td>Check terminals</td>
</tr>
<tr>
<td>4</td>
<td>-</td>
<td>31</td>
<td>3</td>
<td>0...1</td>
<td>4</td>
<td>Analog value overflow at an analog output</td>
<td>Check output value</td>
</tr>
<tr>
<td>4</td>
<td>-</td>
<td>31</td>
<td>3</td>
<td>0...1</td>
<td>7</td>
<td>Analog value underflow at an analog output</td>
<td>Check output value</td>
</tr>
</tbody>
</table>

Remarks:

1) In AC500 the following interface identifier applies:
   
   
   "." = Diagnosis via bus-specific function blocks; 0...4 or 10 = Position of the communication module; 14 = I/O bus; 31 = Module itself
   
   The identifier is not contained in the CI501-PNIO diagnosis block.

2) With "Device" the following allocation applies: 31 = Module itself; 1...10 = Expansion module

3) With "Module" the following allocation applies:
   
   31 = Module itself
   
   Module type (1 = AI, 2 = DO, 3 = AO)
This message appears, if externally voltages at one or more terminals DO0...DO7 cause that other digital outputs are supplied through that voltage. Chapter 1.5.3.7.5.2.3 “Electrical connection” on page 2966. All outputs of the apply digital output groups will be turned off for 5 seconds. The diagnosis message appears for the whole output group.

The voltage on digital outputs DO0...DO7 has overrun the process supply voltage UP3. Chapter 1.5.3.7.5.2.3 “Electrical connection” on page 2966. Diagnosis message appears for the whole module.

This message appears, if the output of a channel DO0...DO7 should be switched on while an externally voltage is connected. In this case the start up is disabled, as long as the externally voltage is connected. Otherwise this could produce reverse voltage from this output to other digital outputs. This diagnosis message appears per channel.

Short circuit: After a detected short circuit, the output is deactivated for 100 ms. Then a new start up will be executed. This diagnosis message appears per channel.

In case of an I/O module doesn’t support hot swapping, do not perform any hot-swap operations (also not on any other terminal units (slots)) as modules may be damaged or I/O bus communication may be disturbed.

Diagnosis for hot swap available as of version index F0.

State LEDs

The LEDs are located at the front of module. There are 2 different groups:

- The 5 system LEDs (PWR, STA1 ETH, STA2 ETH, S-ERR and I/O-Bus) show the operation state of the module and display possible errors.
- The 27 process LEDs (UP, UP3, inputs, outputs, CH-ERR1 to CH-ERR3) show the process supply voltage and the states of the inputs and outputs and display possible errors.

Table 640: States of the 5 system LEDs

<table>
<thead>
<tr>
<th>LED</th>
<th>Color</th>
<th>OFF</th>
<th>ON</th>
<th>Flashing</th>
</tr>
</thead>
<tbody>
<tr>
<td>PWR/RUN</td>
<td>Green</td>
<td>Process supply voltage missing</td>
<td>Internal supply voltage OK, module ready for communication with IO Controller</td>
<td>Start-up / preparing communication</td>
</tr>
<tr>
<td></td>
<td>Yellow</td>
<td>---</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>STA1 ETH</td>
<td>Green</td>
<td>---</td>
<td>Device configured, cyclic data exchange running</td>
<td>---</td>
</tr>
<tr>
<td>(System LED “BF”)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Red</td>
<td>---</td>
<td>Device is not configured</td>
<td></td>
</tr>
<tr>
<td>STA2 ETH</td>
<td>Green</td>
<td>---</td>
<td>---</td>
<td>Got identification request from I/O controller</td>
</tr>
<tr>
<td>(System LED “SF”)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Red</td>
<td>No system error</td>
<td>System error (collective error)</td>
<td>---</td>
</tr>
<tr>
<td>S-ERR</td>
<td>Red</td>
<td>No error</td>
<td>Internal error</td>
<td>--</td>
</tr>
<tr>
<td>LED</td>
<td>Color</td>
<td>OFF</td>
<td>ON</td>
<td>Flashing</td>
</tr>
<tr>
<td>----------</td>
<td>-------</td>
<td>------------------------------------------</td>
<td>-----------------------------------------</td>
<td>---------------------------------------</td>
</tr>
<tr>
<td>I/O-Bus</td>
<td>Green</td>
<td>No expansion modules connected or com-</td>
<td>Expansion modules connected and operatio-</td>
<td>---</td>
</tr>
<tr>
<td></td>
<td></td>
<td>munication error</td>
<td>nal</td>
<td></td>
</tr>
<tr>
<td>ETH1</td>
<td>Green</td>
<td>No connection at Ethernet interface</td>
<td>Connected to Ethernet interface</td>
<td>---</td>
</tr>
<tr>
<td></td>
<td>Yellow</td>
<td>---</td>
<td>Device is transmitting telegrams</td>
<td>Device is transmitting telegrams</td>
</tr>
<tr>
<td>ETH2</td>
<td>Green</td>
<td>No connection at Ethernet interface</td>
<td>Connected to Ethernet interface</td>
<td>---</td>
</tr>
<tr>
<td></td>
<td>Yellow</td>
<td>---</td>
<td>Device is transmitting telegrams</td>
<td>Device is transmitting telegrams</td>
</tr>
</tbody>
</table>

Table 641: States of the 27 process LEDs

<table>
<thead>
<tr>
<th>LED</th>
<th>Color</th>
<th>OFF</th>
<th>ON</th>
<th>Flashing</th>
</tr>
</thead>
<tbody>
<tr>
<td>AI0 to AI3</td>
<td>Yellow</td>
<td>Input is OFF</td>
<td>Input is ON (brightness depends on the value of the analog signal)</td>
<td>--</td>
</tr>
<tr>
<td>AO0 to AO1</td>
<td>Yellow</td>
<td>Output is OFF</td>
<td>Output is ON (brightness depends on the value of the analog signal)</td>
<td>--</td>
</tr>
<tr>
<td>DI0 to DI7</td>
<td>Yellow</td>
<td>Input is OFF</td>
<td>Input is ON (the input voltage is even displayed if the supply voltage is OFF)</td>
<td>--</td>
</tr>
<tr>
<td>DO0 to DO7</td>
<td>Yellow</td>
<td>Output is OFF</td>
<td>Output is ON</td>
<td>--</td>
</tr>
<tr>
<td>UP</td>
<td>Green</td>
<td>Process supply voltage missing</td>
<td>Process supply voltage OK and initializat-</td>
<td>--</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>ion finished</td>
<td></td>
</tr>
<tr>
<td>UP3</td>
<td>Green</td>
<td>Process supply voltage missing</td>
<td>Process supply voltage OK</td>
<td>--</td>
</tr>
<tr>
<td>CH-ERR1 to CH-ERR3</td>
<td>Red</td>
<td>No error or process supply voltage missing</td>
<td>Internal error</td>
<td>Error on one channel of the corresponding group</td>
</tr>
</tbody>
</table>

PLC Automation with V3 CPUs
PLC integration > Device specifications
### Measuring ranges

**Input ranges voltage, current and digital input**

<table>
<thead>
<tr>
<th>Range</th>
<th>0...10 V</th>
<th>-10...+10 V</th>
<th>0...20 mA</th>
<th>4...20 mA</th>
<th>Digital input</th>
</tr>
</thead>
<tbody>
<tr>
<td>Overflow</td>
<td>&gt;11.7589</td>
<td>&gt;11.7589</td>
<td>&gt;23.5178</td>
<td>&gt;22.8142</td>
<td></td>
</tr>
<tr>
<td>Measured value too high</td>
<td>11.7589</td>
<td>11.7589</td>
<td>23.5178</td>
<td>22.8142</td>
<td>On</td>
</tr>
<tr>
<td>:</td>
<td>10.0004</td>
<td>10.0004</td>
<td>20.0007</td>
<td>20.0006</td>
<td></td>
</tr>
<tr>
<td>Normal range</td>
<td>10.0000</td>
<td>10.0000</td>
<td>20.0000</td>
<td>20.0000</td>
<td>Off</td>
</tr>
<tr>
<td>Normal range or measured</td>
<td>0.0004</td>
<td>0.0004</td>
<td>0.0007</td>
<td>4.0006</td>
<td></td>
</tr>
<tr>
<td>value too low</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>Measured value too low</td>
<td>-10.0004</td>
<td>-10.0004</td>
<td>-10.0000</td>
<td>3.9994</td>
<td></td>
</tr>
<tr>
<td>-1.7593</td>
<td></td>
<td></td>
<td></td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>Underflow</td>
<td>&lt;0.0000</td>
<td>=-11.7589</td>
<td>&lt;0.0000</td>
<td>&lt;0.0000</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Range</th>
<th>Digital value</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Decimal</td>
</tr>
<tr>
<td>Overflow</td>
<td>32767</td>
</tr>
<tr>
<td>Measured value too high</td>
<td>32511</td>
</tr>
<tr>
<td>:</td>
<td>27649</td>
</tr>
<tr>
<td>Normal range</td>
<td>27648</td>
</tr>
<tr>
<td>Normal range or measured</td>
<td>0</td>
</tr>
<tr>
<td>value too low</td>
<td>-1</td>
</tr>
<tr>
<td>:</td>
<td>-4864</td>
</tr>
<tr>
<td>:</td>
<td>-6912</td>
</tr>
<tr>
<td>:</td>
<td>27648</td>
</tr>
<tr>
<td>Measured value too low</td>
<td>-27649</td>
</tr>
<tr>
<td>:</td>
<td>32512</td>
</tr>
<tr>
<td>Underflow</td>
<td>-32768</td>
</tr>
</tbody>
</table>

The represented resolution corresponds to 16 bits.
## Input ranges resistance temperature detector

<table>
<thead>
<tr>
<th>Range</th>
<th>Pt100 / Pt1000 -50...+70 °C</th>
<th>Pt100 / Pt1000 -50...400 °C</th>
<th>Ni1000 -50...150 °C</th>
</tr>
</thead>
<tbody>
<tr>
<td>Overflow</td>
<td>&gt; 80.0 °C</td>
<td>&gt; 450.0 °C</td>
<td>&gt; 160.0 °C</td>
</tr>
<tr>
<td>Measured value too high</td>
<td>80.0 °C</td>
<td>450.0 °C</td>
<td>160.0 °C</td>
</tr>
<tr>
<td></td>
<td></td>
<td>400.1 °C</td>
<td>150.1 °C</td>
</tr>
<tr>
<td>Normal range</td>
<td>400.0 °C</td>
<td>150.0 °C</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.1 °C</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.0 °C</td>
<td></td>
</tr>
<tr>
<td></td>
<td>-0.1 °C</td>
<td>-0.1 °C</td>
<td></td>
</tr>
<tr>
<td></td>
<td>-50.0 °C</td>
<td>-50.0 °C</td>
<td></td>
</tr>
<tr>
<td>Measured value too low</td>
<td>&lt; -60.0 °C</td>
<td>-50.1 °C</td>
<td>-50.1 °C</td>
</tr>
<tr>
<td></td>
<td></td>
<td>-60.0 °C</td>
<td></td>
</tr>
<tr>
<td>Underflow</td>
<td>&lt; -60.0 °C</td>
<td>&lt; -60.0 °C</td>
<td>&lt; -60.0 °C</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Range</th>
<th>Digital value</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Decimal</td>
</tr>
<tr>
<td>Overflow</td>
<td>32767</td>
</tr>
<tr>
<td>Measured value too high</td>
<td>4500</td>
</tr>
<tr>
<td></td>
<td>4001</td>
</tr>
<tr>
<td></td>
<td>1600</td>
</tr>
<tr>
<td></td>
<td>1501</td>
</tr>
<tr>
<td></td>
<td>800</td>
</tr>
<tr>
<td></td>
<td>701</td>
</tr>
<tr>
<td>Normal range</td>
<td>4000</td>
</tr>
<tr>
<td></td>
<td>1500</td>
</tr>
<tr>
<td></td>
<td>700</td>
</tr>
<tr>
<td></td>
<td>1</td>
</tr>
</tbody>
</table>
### Output ranges voltage and current

<table>
<thead>
<tr>
<th>Range</th>
<th>-10...+10 V</th>
<th>0...20 mA</th>
<th>4...20 mA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Overflow</td>
<td>&gt; 11.7589 V</td>
<td>&gt; 23.5178 mA</td>
<td>&gt; 22.8142 mA</td>
</tr>
<tr>
<td>Measured value too high</td>
<td>11.7589 V</td>
<td>23.5178 mA</td>
<td>22.8142 mA</td>
</tr>
<tr>
<td></td>
<td>:</td>
<td>:</td>
<td>:</td>
</tr>
<tr>
<td></td>
<td>10.0004 V</td>
<td>20.0007 mA</td>
<td>20.0006 mA</td>
</tr>
<tr>
<td>Normal range</td>
<td>10.0000 V</td>
<td>20.0000 mA</td>
<td>20.0000 mA</td>
</tr>
<tr>
<td></td>
<td>:</td>
<td>:</td>
<td>:</td>
</tr>
<tr>
<td></td>
<td>0.0004 V</td>
<td>4.0000 mA</td>
<td>4.0000 mA</td>
</tr>
<tr>
<td>Measured value too low</td>
<td>-10.0004 V</td>
<td>0 mA</td>
<td>0 mA</td>
</tr>
<tr>
<td></td>
<td>:</td>
<td>:</td>
<td>:</td>
</tr>
<tr>
<td></td>
<td>-11.7589 V</td>
<td>0 mA</td>
<td>0 mA</td>
</tr>
<tr>
<td>Underflow</td>
<td>&lt; -11.7589 V</td>
<td>0 mA</td>
<td>0 mA</td>
</tr>
</tbody>
</table>

### Measured value too low

<table>
<thead>
<tr>
<th>Range</th>
<th>Digital value</th>
<th>Decimal</th>
<th>Hex.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Measured value too low</td>
<td>-501</td>
<td>FEOB</td>
<td>FE0B</td>
</tr>
<tr>
<td></td>
<td>: -600</td>
<td>: FEO0C</td>
<td>FDA8</td>
</tr>
</tbody>
</table>

### Underflow

<table>
<thead>
<tr>
<th>Range</th>
<th>Digital value</th>
<th>Decimal</th>
<th>Hex.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Underflow</td>
<td>-32768</td>
<td>8000</td>
<td></td>
</tr>
<tr>
<td>Range</td>
<td>Digital value</td>
<td></td>
<td></td>
</tr>
<tr>
<td>-----------------------------------</td>
<td>------------------------</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Decimal</td>
<td>Hex.</td>
<td></td>
</tr>
<tr>
<td>-</td>
<td>-1</td>
<td>FFFF</td>
<td></td>
</tr>
<tr>
<td></td>
<td>-6912</td>
<td>E500</td>
<td></td>
</tr>
<tr>
<td></td>
<td>-27648</td>
<td>9400</td>
<td></td>
</tr>
<tr>
<td>Measured value too low</td>
<td>-27649</td>
<td>93FF</td>
<td></td>
</tr>
<tr>
<td></td>
<td>:</td>
<td>:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>-32512</td>
<td>8100</td>
<td></td>
</tr>
<tr>
<td>Underflow</td>
<td>&lt; -32512</td>
<td>&lt; 8100</td>
<td></td>
</tr>
</tbody>
</table>

The represented resolution corresponds to 16 bits.

**Technical data**

The system data of AC500 and S500 © *Chapter 1.5.4.5.1 “System data AC500” on page 3058* are valid for standard version.

The system data of AC500-XC © *Chapter 1.5.4.6.1 “System data AC500-XC” on page 3099* are valid for the XC version.

Only additional details are therefore documented below.

The technical data are also valid for the XC version.

**Technical data of the module**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Process supply voltages UP/UP3</td>
<td></td>
</tr>
<tr>
<td>Rated value</td>
<td>24 VDC (for inputs and outputs)</td>
</tr>
<tr>
<td>Max. load for the terminals</td>
<td>10 A</td>
</tr>
<tr>
<td>Protection against reversed voltage</td>
<td>Yes</td>
</tr>
<tr>
<td>Rated protection fuse on UP/UP3</td>
<td>10 A fast</td>
</tr>
<tr>
<td>Electrical isolation</td>
<td>Ethernet interface against the rest of the module</td>
</tr>
<tr>
<td>Inrush current from UP (at power up)</td>
<td>On request</td>
</tr>
<tr>
<td>Current consumption via UP (normal operation)</td>
<td>0.2 A</td>
</tr>
<tr>
<td>Current consumption via UP3</td>
<td>0.06 A + 0.5 A max. per output</td>
</tr>
<tr>
<td>Connections</td>
<td>Terminals 1.8 and 2.8 for +24 V (UP)</td>
</tr>
<tr>
<td></td>
<td>Terminal 3.8 for +24 V (UP3)</td>
</tr>
<tr>
<td></td>
<td>Terminals 1.9, 2.9 and 3.9 for 0 V (ZP)</td>
</tr>
<tr>
<td>Max. power dissipation within the module</td>
<td>6 W</td>
</tr>
<tr>
<td>Number of digital inputs</td>
<td>8</td>
</tr>
<tr>
<td>Number of digital outputs</td>
<td>8</td>
</tr>
<tr>
<td>Number of analog inputs</td>
<td>4</td>
</tr>
<tr>
<td>Number of analog outputs</td>
<td>2</td>
</tr>
<tr>
<td>Input data length</td>
<td>2 bytes</td>
</tr>
</tbody>
</table>
### Parameter | Value
---|---
Output data length | 2 bytes
Reference potential for all digital inputs and outputs | Minus pole of the supply voltage, signal name ZP
Setting of the IO device identifier | With 2 rotary switches at the front side of the module
Diagnose | See Diagnosis and Displays "Chapter 1.5.3.7.5.2.8 "Diagnosis and state LEDs" on page 2988"
Operation and error displays | 32 LEDs (totally)
Weight (without terminal unit) | Ca. 125 g
Mounting position | Horizontal or vertical with derating (output load reduced to 50 % at 40 °C per group)
Extended ambient temperature (XC version) | >60 °C on request
Cooling | The natural convection cooling must not be hindered by cable ducts or other parts in the switch-gear cabinet.

---

**NOTICE!**  
**Attention:**  
All I/O channels (digital and analog) are protected against reverse polarity, reverse supply, short circuit and continuous overvoltage up to 30 VDC.

---

**Multiple overloads**  
*No effects of multiple overloads on isolated multi-channel modules occur, as every channel is protected individually by an internal smart high-side switch.*

---

### Parameter | Value
---|---
Bus connection | 2 x RJ45
Switch | Integrated
Technology | Hilscher netX100
Transfer rate | 10/100 Mbit/s (full-duplex)
Transfer method | According to Ethernet II, IEE802.3
Ethernet | 100 base-TX, internal switch, 2x RJ45 socket
Expandability | Max. 10 S500 I/O modules
Adjusting elements | 2 rotary switches for generation of an explicit name
**Parameter** | **Value**
---|---
Supported protocols | RTC - real time cyclic protocol, class 1 *)
RTA - real time acyclic protocol
DCP - discovery and configuration protocol
CL-RPC - connectionless remote procedure Call
LLDP - link layer discovery protocol
MRP - MRP Client

Acyclic services | PNIO read / write sequence (max. 1024 bytes per telegram)
Process-Alarm service

Supported alarm types | Process Alarm, Diagnostic Alarm, Return of SubModule, Plug Alarm, Pull Alarm

Min. bus cycle | 1 ms

Conformance class | CC A

Protective functions (according to IEC 61131-3) | Protected against:
- short circuit
- reverse supply
- overvoltage
- reverse polarity
Electrical isolation from the rest of the module

*) Priorization with the aid of VLAN-ID including priority level

---

**Technical data of the digital inputs**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of channels per module</td>
<td>8</td>
</tr>
<tr>
<td>Distribution of the channels into groups</td>
<td>1 group of 8 channels</td>
</tr>
<tr>
<td>Terminals of the channels DI0 to DI7</td>
<td>Terminals 2.0 to 2.7</td>
</tr>
<tr>
<td>Reference potential for all inputs</td>
<td>Terminals 1.9...3.9 (Minus pole of the supply voltage, signal name ZP)</td>
</tr>
<tr>
<td>Indication of the input signals</td>
<td>1 yellow LED per channel, the LED is ON when the input signal is high (signal 1)</td>
</tr>
<tr>
<td>Input type (according EN 61131-2)</td>
<td>Type 1</td>
</tr>
<tr>
<td>Input delay (0-&gt;1 or 1-&gt;0)</td>
<td>Typ. 0.1 ms, configurable from 0.1...32 ms</td>
</tr>
<tr>
<td>Input signal voltage</td>
<td>24 VDC</td>
</tr>
<tr>
<td>0-Signal</td>
<td>-3 V...+5 V</td>
</tr>
<tr>
<td>Undefined Signal</td>
<td>&gt; +5 V...&lt; +15 V</td>
</tr>
<tr>
<td>1-Signal</td>
<td>+15 V...+30 V</td>
</tr>
<tr>
<td>Ripple with signal 0</td>
<td>Within -3 V...+5 V</td>
</tr>
<tr>
<td>Ripple with signal 1</td>
<td>Within +15 V...+30 V</td>
</tr>
<tr>
<td>Input current per channel</td>
<td></td>
</tr>
<tr>
<td>Input voltage +24 V</td>
<td>Typ. 5 mA</td>
</tr>
<tr>
<td>Input voltage +5 V</td>
<td>&gt; 1 mA</td>
</tr>
</tbody>
</table>
### Technical data of the digital outputs

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of channels per module</td>
<td>8</td>
</tr>
<tr>
<td>Distribution of the channels into groups</td>
<td>1 group of 8 channels</td>
</tr>
<tr>
<td>Terminals of the channels DO0 to DO7</td>
<td>Terminals 3.0 to 3.7</td>
</tr>
<tr>
<td>Reference potential for all outputs</td>
<td>Terminals 1.9...3.9 (minus pole of the supply voltage, signal name ZP)</td>
</tr>
<tr>
<td>Common power supply voltage</td>
<td>For all outputs terminal 3.8 (plus pole of the supply voltage, signal name UP3)</td>
</tr>
<tr>
<td>Output voltage for signal 1</td>
<td>UP3 (-0.8 V)</td>
</tr>
<tr>
<td>Output delay (0-&gt;1 or 1-&gt;0)</td>
<td>On request</td>
</tr>
<tr>
<td>Output current</td>
<td></td>
</tr>
<tr>
<td>Rated value per channel</td>
<td>500 mA at UP3 = 24 V</td>
</tr>
<tr>
<td>Max. value (all channels together)</td>
<td>4 A</td>
</tr>
<tr>
<td>Leakage current with signal 0</td>
<td>&lt; 0.5 mA</td>
</tr>
<tr>
<td>Fuse for UP3</td>
<td>10 A fast</td>
</tr>
<tr>
<td>Demagnetization with inductive DC load</td>
<td>Via internal varistors (see figure below this table)</td>
</tr>
<tr>
<td>Output switching frequency</td>
<td></td>
</tr>
<tr>
<td>With resistive load</td>
<td>On request</td>
</tr>
<tr>
<td>With inductive loads</td>
<td>Max. 0.5 Hz</td>
</tr>
<tr>
<td>With lamp loads</td>
<td>11 Hz max. at 5 W max.</td>
</tr>
<tr>
<td>Short-circuit-proof / overload-proof</td>
<td>Yes</td>
</tr>
<tr>
<td>Overload message (I &gt; 0.7 A)</td>
<td>Yes, after ca. 100 ms</td>
</tr>
<tr>
<td>Output current limitation</td>
<td>Yes, automatic reactivation after short circuit/overload</td>
</tr>
<tr>
<td>Resistance to feedback against 24 V signals</td>
<td>Yes (software-controlled supervision)</td>
</tr>
<tr>
<td>Max. cable length</td>
<td></td>
</tr>
<tr>
<td>Shielded</td>
<td>1000 m</td>
</tr>
<tr>
<td>Unshielded</td>
<td>600 m</td>
</tr>
</tbody>
</table>

The following drawing shows the circuitry of a digital input/output with the varistors for demagnetization when inductive loads are switched off.
Technical data of the analog inputs

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of channels per module</td>
<td>4</td>
</tr>
<tr>
<td>Distribution of channels into groups</td>
<td>1 group with 4 channels</td>
</tr>
<tr>
<td>Connection if channels AI0+ to AI3+</td>
<td>Terminals 1.0 to 1.3</td>
</tr>
<tr>
<td>Reference potential for AI0+ to AI3+</td>
<td>Terminal 1.4 (AI-) for voltage and RTD measurement</td>
</tr>
<tr>
<td></td>
<td>Terminal 1.9, 2.9 and 3.9 for current measurement</td>
</tr>
<tr>
<td>Input type</td>
<td>Unipolar: Voltage 0 V...10 V, current or Pt100/Pt1000/ Ni1000</td>
</tr>
<tr>
<td></td>
<td>Bipolar: Voltage -10 V...+10 V</td>
</tr>
<tr>
<td>Electrical isolation</td>
<td>Against Ethernet network</td>
</tr>
<tr>
<td>Configurability</td>
<td>0 V...10 V, -10 V...+10 V, 0 mA...20 mA, 4 mA...20 mA Pt100/1000, Ni1000 (each input can be configured individually)</td>
</tr>
<tr>
<td>Channel input resistance</td>
<td>Voltage: &gt; 100 kΩ</td>
</tr>
<tr>
<td></td>
<td>Current: ca. 330 Ω</td>
</tr>
<tr>
<td>Time constant of the input filter</td>
<td>Voltage: 100 µs</td>
</tr>
<tr>
<td></td>
<td>Current: 100 µs</td>
</tr>
<tr>
<td>Indication of the input signals</td>
<td>1 LED per channel (brightness depends on the value of the analog signal)</td>
</tr>
<tr>
<td>Conversion cycle</td>
<td>1 ms (for 4 inputs + 2 outputs); with RTDs Pt/ Ni... 1 s</td>
</tr>
<tr>
<td>Resolution</td>
<td>Range 0 V...10 V: 12 bits</td>
</tr>
<tr>
<td></td>
<td>Range -10 V...+10 V: 12 bits + sign</td>
</tr>
<tr>
<td></td>
<td>Range 0 mA...20 mA: 12 bits</td>
</tr>
<tr>
<td></td>
<td>Range 4 mA...20 mA: 12 bits</td>
</tr>
<tr>
<td></td>
<td>Range RTD (Pt100, PT1000, Ni1000): 0.1 °C</td>
</tr>
<tr>
<td>Conversion error of the analog values</td>
<td>Typ. 0.5 %, max. 1 %</td>
</tr>
<tr>
<td>caused by non-linearity, adjustment</td>
<td>error at factory and resolution within the normal range</td>
</tr>
</tbody>
</table>

1. Digital output
2. Varistors for demagnetization when inductive loads are turned off
### Parameter | Value
---|---
Relationship between input signal and hex code | Tables Input ranges voltage, current and digital input and Input range resistance temperature detector Chapter 1.5.3.7.5.2.9.1 “Input ranges voltage, current and digital input” on page 2995
Unused inputs | Are configured as "unused" (default value)
Overvoltage protection | Yes

**Technical data of the analog inputs, if used as digital inputs**

| Parameter | Value |
---|---|
Number of channels per module | Max. 4 |
Distribution of channels into groups | 1 group of 4 channels |
Connections of the channels AI0+ to AI3+ | Terminals 1.0 to 1.3 |
Reference potential for the inputs | Terminals 1.9, 2.9 and 3.9 (ZP) |
Indication of the input signals | 1 LED per channel |
Input signal voltage | 24 VDC |
| Signal 0 | -30 V...+5 V |
| Undefined signal | +5 V ... +13 V |
| Signal 1 | +13 V...+30 V |
Input current per channel | |
| Input voltage +24 V | Typ. 7 mA |
| Input voltage +5 V | Typ. 1.4 mA |
| Input voltage +15 V | Typ. 3.7 mA |
| Input voltage +30 V | < 9 mA |
Input resistance | Ca. 3.5 kΩ |

**Technical data of the analog outputs**

| Parameter | Value |
---|---|
Number of channels per module | 2 |
Distribution of channels into groups | 1 group for 2 channels |
Connection of the channels AO0+...AO1+ | Terminals 1.5...1.6 |
Reference potential for AO0+ to AO1+ | Terminal 1.7 (AO-) for voltage output terminal 1.9, 2.9 and 3.9 for current output |
Output type | |
| Unipolar | Current |
| Bipolar | Voltage |
Electrical isolation | Against internal supply and other modules |
Configurability | -10 V...+10 V, 0 mA...20 mA, 4 mA...20 mA (each output can be configured individually) |
Output resistance (load), as current output | 0 Ω...500 Ω |
### Technical data of the fast counter

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Used inputs</td>
<td>Terminal 2.0 (DI0), 2.1 (DI1)</td>
</tr>
<tr>
<td>Used outputs</td>
<td>Terminal 3.0 (DO0)</td>
</tr>
<tr>
<td>Counting frequency</td>
<td>Depending on operation mode:</td>
</tr>
<tr>
<td></td>
<td>Mode 1 - 6: max. 200 kHz</td>
</tr>
<tr>
<td></td>
<td>Mode 7: max. 50 kHz</td>
</tr>
<tr>
<td></td>
<td>Mode 9: max. 35 kHz</td>
</tr>
<tr>
<td></td>
<td>Mode 10: max. 20 kHz</td>
</tr>
<tr>
<td>Detailed description</td>
<td>See Fast Counter (^\circ) Chapter 1.5.5.1.10 “Fast counters” on page 3194</td>
</tr>
<tr>
<td>Operating modes</td>
<td>See Operating Modes</td>
</tr>
</tbody>
</table>

### Ordering data

<table>
<thead>
<tr>
<th>Part no.</th>
<th>Description</th>
<th>Product life cycle phase *)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1SAP 220 600 R0001</td>
<td>CI501-PNIO (V3), PROFINET bus module, 8 DI, 8 DO, 4 AI and 2 AO</td>
<td>Active</td>
</tr>
<tr>
<td>1SAP 420 600 R0001</td>
<td>CI501-PNIO-XC (V3), PROFINET bus module, 8 DI, 8 DO, 4 AI and 2 AO, XC version</td>
<td>Active</td>
</tr>
</tbody>
</table>

*) For planning and commissioning of new installations use modules in Active status only.
**CI502-PNIO (-XC)**

- 8 digital inputs 24 VDC
- 8 digital outputs 24 VDC, 0.5 A max.
- 8 configurable digital inputs/outputs 24 VDC, 0.5 A max.
- Module-wise electrically isolated
- Fast counter
- XC version for usage in extreme ambient conditions available

---

1. I/O bus
2. Allocation between terminal number and signal name
3. 8 yellow LEDs to display the signal states of the digital configurable inputs/outputs (DC0 - DC7)
4. 8 yellow LEDs to display the signal states of the digital inputs (DI8 - DI15)
5. 8 yellow LEDs to display the signal states of the digital outputs (DO8 - DO15)
6. 2 green LEDs to display the process supply voltage UP and UP3
7. 3 red LEDs to display errors (CH-ERR1, CH-ERR2, CH-ERR3)
8. 5 system LEDs: PWR/RUN, STA1 ETH, STA2 ETH, S-ERR, I/O-Bus
9. Label
10. 2 rotary switches for setting the IO device identifier
11. Ethernet interfaces (ETH1, ETH2) on the terminal unit
12. Terminal unit
13. DIN rail

\*\*\* Sign for XC version
Intended purpose

The PROFINET bus module CI502-PNIO is used as communication interface module in PROFINET networks. The network connection is performed via 2 RJ45 connectors which are integrated in the terminal unit.

For usage in extreme ambient conditions (e.g. wider temperature and humidity range), a special XC version of the device is available.

Functionality

The CI502 Bus Module contains 24 I/O channels with the following properties:

- 8 digital configurable inputs/outputs
- 8 digital inputs: 24 VDC
- 8 digital outputs: 24 VDC, 0.5 A max.

The inputs/outputs are electrically isolated from the Ethernet network. There is no potential separation between the channels. The configuration of the analog inputs/outputs is performed by software.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Interface</td>
<td>Ethernet</td>
</tr>
<tr>
<td>Protocol</td>
<td>PROFINET IO RT</td>
</tr>
<tr>
<td>Power supply</td>
<td>From the process supply voltage UP</td>
</tr>
<tr>
<td>Supply of the electronic circuitry of the I/O</td>
<td>Through the expansion bus interface (I/O bus)</td>
</tr>
<tr>
<td>expansion modules attached</td>
<td></td>
</tr>
<tr>
<td>Rotary switches</td>
<td>For setting the IO device identifier for configuration purposes (00h to FFh)</td>
</tr>
<tr>
<td>Configurable digital inputs/outputs</td>
<td>8 (configurable via software)</td>
</tr>
<tr>
<td>Digital inputs</td>
<td>8 (24 VDC; delay time configurable via software)</td>
</tr>
<tr>
<td>Digital outputs</td>
<td>8 (24 VDC, 0.5 A max.)</td>
</tr>
<tr>
<td>LED displays</td>
<td>For system displays, signal states, errors and power supply</td>
</tr>
<tr>
<td>External supply voltage</td>
<td>Via terminals ZP, UP and UP3 (process supply voltage 24 VDC)</td>
</tr>
<tr>
<td>Effect of incorrect input terminal connection</td>
<td>Wrong or no signal detected, no damage up to 35 V</td>
</tr>
<tr>
<td>Required terminal unit</td>
<td>TU507-ETH or TU508-ETH § Chapter 1.5.3.5.1 “TU507-ETH and TU508-ETH for Ethernet communication interface modules” on page 2307</td>
</tr>
</tbody>
</table>

Electrical connection

The Ethernet bus module CI502-PNIO is plugged on the I/O terminal unit TU507-ETH § Chapter 1.5.3.5.1 “TU507-ETH and TU508-ETH for Ethernet communication interface modules” on page 2307 or TU508-ETH § Chapter 1.5.3.5.1 “TU507-ETH and TU508-ETH for Ethernet communication interface modules” on page 2307. Properly seat the module and press until it locks in place. The terminal unit is mounted on a DIN rail or with 2 screws plus the additional accessory for wall mounting (TA526 § Chapter 1.5.3.8.1.4 “TA526 - Wall mounting accessory” on page 3035).
The electrical connection of the I/O channels is carried out using the 30 terminals of the I/O terminal unit. I/O modules can be replaced without re-wiring the terminal units.

For a detailed description of the mounting, disassembly and electrical connection of the module, please refer to the System Assembly, Construction and Connection chapter & Chapter 1.5.4.5 “AC500 (Standard)” on page 3058.

The terminals 1.8 and 2.8 as well as 1.9, 2.9 and 3.9 are electrically interconnected within the terminal unit and have always the same assignment, independent of the inserted module:
Terminals 1.8 and 2.8: Process supply voltage UP = +24 VDC
Terminal 3.8: Process supply voltage UP3 = +24 VDC
Terminals 1.9, 2.9 and 3.9: Process supply voltage ZP = 0 V.

The assignment of the other terminals:

<table>
<thead>
<tr>
<th>Terminal</th>
<th>Signal</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0</td>
<td>DC0</td>
<td>Signal of the configurable digital input/output DC0</td>
</tr>
<tr>
<td>1.1</td>
<td>DC1</td>
<td>Signal of the configurable digital input/output DC1</td>
</tr>
<tr>
<td>1.2</td>
<td>DC2</td>
<td>Signal of the configurable digital input/output DC2</td>
</tr>
<tr>
<td>1.3</td>
<td>DC3</td>
<td>Signal of the configurable digital input/output DC3</td>
</tr>
<tr>
<td>1.4</td>
<td>DC4</td>
<td>Signal of the configurable digital input/output DC4</td>
</tr>
</tbody>
</table>

With a separate UP3 power supply, the digital outputs can be switched off externally. This way, an emergency-off functionality can be realized.

Do not connect any voltages externally to digital outputs!
This is not intended usage.
Reason: Externally voltages at one or more terminals DC0..DC7 or DO0..DO7 may cause that other digital outputs are supplied through that voltage instead of voltage UP3 (reverse voltage).
This is also possible, if DC channels are used as inputs. For this, the source for the input signals should be the impressed UP3 of the device.
This limitation does not apply for the input channels DI0..DI7.

CAUTION!
Risk of malfunction by not intended usage!
If the function cut off of the digital outputs should be used by deactivation of the supply voltage UP3, be sure that no external voltage is conncted at the outputs DO0...DO7 and DC0...DC7.

The assignment of the other terminals:
<table>
<thead>
<tr>
<th>Terminal</th>
<th>Signal</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.5</td>
<td>DC5</td>
<td>Signal of the configurable digital input/output DC5</td>
</tr>
<tr>
<td>1.6</td>
<td>DC6</td>
<td>Signal of the configurable digital input/output DC6</td>
</tr>
<tr>
<td>1.7</td>
<td>DC7</td>
<td>Signal of the configurable digital input/output DC7</td>
</tr>
<tr>
<td>1.8</td>
<td>UP</td>
<td>Process voltage UP (24 VDC)</td>
</tr>
<tr>
<td>1.9</td>
<td>ZP</td>
<td>Process voltage ZP (0 VDC)</td>
</tr>
<tr>
<td>2.0</td>
<td>DI8</td>
<td>Signal of the digital input DI8</td>
</tr>
<tr>
<td>2.1</td>
<td>DI9</td>
<td>Signal of the digital input DI9</td>
</tr>
<tr>
<td>2.2</td>
<td>DI10</td>
<td>Signal of the digital input DI10</td>
</tr>
<tr>
<td>2.3</td>
<td>DI11</td>
<td>Signal of the digital input DI11</td>
</tr>
<tr>
<td>2.4</td>
<td>DI12</td>
<td>Signal of the digital input DI12</td>
</tr>
<tr>
<td>2.5</td>
<td>DI13</td>
<td>Signal of the digital input DI13</td>
</tr>
<tr>
<td>2.6</td>
<td>DI14</td>
<td>Signal of the digital input DI14</td>
</tr>
<tr>
<td>2.7</td>
<td>DI15</td>
<td>Signal of the digital input DI15</td>
</tr>
<tr>
<td>2.8</td>
<td>UP</td>
<td>Process voltage UP (24 VDC)</td>
</tr>
<tr>
<td>2.9</td>
<td>ZP</td>
<td>Process voltage ZP (0 VDC)</td>
</tr>
<tr>
<td>3.0</td>
<td>DO8</td>
<td>Signal of the digital output DO8</td>
</tr>
<tr>
<td>3.1</td>
<td>DO9</td>
<td>Signal of the digital output DO9</td>
</tr>
<tr>
<td>3.2</td>
<td>DO10</td>
<td>Signal of the digital output DO10</td>
</tr>
<tr>
<td>3.3</td>
<td>DO11</td>
<td>Signal of the digital output DO11</td>
</tr>
<tr>
<td>3.4</td>
<td>DO12</td>
<td>Signal of the digital output DO12</td>
</tr>
<tr>
<td>3.5</td>
<td>DO13</td>
<td>Signal of the digital output DO13</td>
</tr>
<tr>
<td>3.6</td>
<td>DO14</td>
<td>Signal of the digital output DO14</td>
</tr>
<tr>
<td>3.7</td>
<td>DO15</td>
<td>Signal of the digital output DO15</td>
</tr>
<tr>
<td>3.8</td>
<td>UP3</td>
<td>Process voltage UP3 (24 VDC)</td>
</tr>
<tr>
<td>3.9</td>
<td>ZP</td>
<td>Process voltage ZP (0 VDC)</td>
</tr>
</tbody>
</table>
WARNING!

Removal/Insertion under power

The devices are not designed for removal or insertion under power. Because of unforeseeable consequences, it is not allowed to plug or unplug devices with the power being ON.

Make sure that all voltage sources (supply and process voltage) are switched off before you

– connect or disconnect any signal or terminal block
– remove, mount or replace a module.

Disconnecting any powered devices while energized in a hazardous location could result in an electric arc, which could create a flammable ignition resulting in fire or explosion.

Make sure that power is removed and that the area has been thoroughly checked to ensure that flammable materials are not present prior to proceeding.

The devices must not be opened when in operation. The same applies to the network interfaces.

NOTICE!

Risk of damaging the PLC modules!

Overvoltages and short circuits might damage the PLC modules.

– Make sure that all voltage sources (supply voltage and process supply voltage) are switched off before you begin with operations at the system.
– Never connect any voltages or signals to reserved terminals (marked with ---). Reserved terminals may carry internal voltages.

The following figure shows the electrical connection of the Ethernet bus module CI502-PNIO. Further information is provided in the System Technology chapter PROFINET.

[Diagram of electrical connection]

Further information is provided in the System Technology chapter PROFINET.
Connection of the Digital inputs

The following figure shows the electrical connection of the digital input DI8. Proceed with the digital inputs DI9 to DI15 in the same way.

The meaning of the LEDs is described in Displays Chapter 1.5.3.7.5.3.8.1 “State LEDs” on page 3020.
Connection of the Digital outputs

The following figure shows the electrical connection of the digital output DO8. Proceed with the digital outputs DO9 - DO15 in the same way.

The meaning of the LEDs is described in Displays  Chapter 1.5.3.7.5.3.8.1 “State LEDs” on page 3020.
Connection of the configurable digital inputs/outputs

The following figure shows the electrical connection of the configurable digital input/output DC0 and DC1. DC0 is connected as an input and DC1 is connected as an output. Proceed with the configurable digital inputs/outputs DC2 to DC7 in the same way.

CAUTION!

If a DC channel is used as input, the source for the input signals should be the impressed UP3 of the device % Chapter 1.5.3.7.5.3.3 “Electrical connection” on page 3006.

The meaning of the LEDs is described in Displays % Chapter 1.5.3.7.5.3.8.1 “State LEDs” on page 3020.

Assignment of the Ethernet ports

The terminal unit for the communication interface module provides two Ethernet interfaces with the following pin assignment:

Table 642: Pin assignment RJ45 jack:

<table>
<thead>
<tr>
<th>Interface</th>
<th>Pin</th>
<th>Signal</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>TxD+</td>
<td>Transmit data +</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>TxD-</td>
<td>Transmit data -</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>RxD+</td>
<td>Receive data +</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>NC</td>
<td>not used</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>NC</td>
<td>not used</td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td>RxD-</td>
<td>Receive data -</td>
</tr>
<tr>
<td>7</td>
<td>7</td>
<td>NC</td>
<td>not used</td>
</tr>
<tr>
<td>8</td>
<td>8</td>
<td>NC</td>
<td>not used</td>
</tr>
<tr>
<td>Shield</td>
<td></td>
<td>Cable shield</td>
<td>Functional earth</td>
</tr>
</tbody>
</table>
For further information regarding wiring and cable types see chapter Ethernet
uplicates Chapter 1.5.4.5.4.7 “Ethernet connection details” on page 3084.

Internal data exchange

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Digital inputs (bytes)</td>
<td>5</td>
</tr>
<tr>
<td>Digital outputs (bytes)</td>
<td>5</td>
</tr>
<tr>
<td>Counter input data (words)</td>
<td>4</td>
</tr>
<tr>
<td>Counter output data (words)</td>
<td>8</td>
</tr>
</tbody>
</table>

Addressing

The module reads the position of the rotary switches only during power-up, i.e.
changes of the switch position during operation will have no effect until the next
module initialization.

I/O configuration

The CI502-PNIO stores some PROFINET configuration parameters (I/O device identifier, I/O
device type and IP address configuration). No more configuration data is stored.

The digital I/O channels are configured via software.

Details about configuration are described in Parameterization Chapter 1.5.3.7.5.3.7 “Para-
terization” on page 3013.

Parameterization

Parameters of the module

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>Internal value</th>
<th>Internal value, type</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>Module ID 1)</td>
<td>Internal</td>
<td>7005</td>
<td>WORD</td>
<td>7005</td>
</tr>
<tr>
<td>Parameter length</td>
<td>Internal</td>
<td>8</td>
<td>BYTE</td>
<td>8</td>
</tr>
<tr>
<td>Error LED / Failsafe function (Table Error LED / Failsafe function Further information on page 3013)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>On</td>
<td>0</td>
<td>BYTE</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>Off by E4</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Off by E3</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>On + failsafe</td>
<td>16</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Off by E4 + failsafe</td>
<td>17</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Off by E3 + failsafe</td>
<td>19</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Name</td>
<td>Value</td>
<td>Internal value</td>
<td>Internal value, type</td>
<td>Default</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>------------------------------</td>
<td>----------------</td>
<td>----------------------</td>
<td>---------</td>
</tr>
<tr>
<td>Process cycle time</td>
<td>1 ms process cycle time</td>
<td>1</td>
<td>BYTE</td>
<td>1 ms</td>
</tr>
<tr>
<td></td>
<td>2 ms process cycle time</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>3 ms process cycle time</td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>4 ms process cycle time</td>
<td>4</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>5 ms process cycle time</td>
<td>5</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>6 ms process cycle time</td>
<td>6</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>7 ms process cycle time</td>
<td>7</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>8 ms process cycle time</td>
<td>8</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>9 ms process cycle time</td>
<td>9</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>10 ms process cycle time</td>
<td>10</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>11 ms process cycle time</td>
<td>11</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>12 ms process cycle time</td>
<td>12</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>13 ms process cycle time</td>
<td>13</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>14 ms process cycle time</td>
<td>14</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>15 ms process cycle time</td>
<td>15</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>16 ms process cycle time</td>
<td>16</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Check supply</td>
<td>off</td>
<td>0</td>
<td>BYTE</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>on</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Fast counter</td>
<td>0</td>
<td>0</td>
<td>BYTE</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>: 10 2)</td>
<td>10</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

1) With a faulty ID, the module reports a “parameter error” and does not perform cyclic process data transmission.

2) Counter operating modes  

*Chapter 1.5.3.6.1.2.9 “Fast counter” on page 2543*
Table 643: Table Error LED / Failsafe function

<table>
<thead>
<tr>
<th>Setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>On</td>
<td>Error LED (S-ERR) lights up at errors of all error classes, Failsafe-mode off</td>
</tr>
<tr>
<td>Off by E4</td>
<td>Error LED (S-ERR) lights up at errors of error classes E1, E2 and E3, Failsafe-mode off</td>
</tr>
<tr>
<td>Off by E3</td>
<td>Error LED (S-ERR) lights up at errors of error classes E1 and E2, Failsafe-mode off</td>
</tr>
<tr>
<td>On + Failsafe</td>
<td>Error LED (S-ERR) lights up at errors of all error classes, Failsafe-mode on *)</td>
</tr>
<tr>
<td>Off by E4 + Failsafe</td>
<td>Error LED (S-ERR) lights up at errors of error classes E1, E2 and E3, Failsafe-mode on *)</td>
</tr>
<tr>
<td>Off by E3 + Failsafe</td>
<td>Error LED (S-ERR) lights up at errors of error classes E1 and E2, Failsafe-mode on *)</td>
</tr>
</tbody>
</table>

*) The parameter Behaviour DO at comm. error is only analyzed if the Failsafe-mode is ON.

Group parameters for the digital part

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>Internal value</th>
<th>Internal value, type</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input delay</td>
<td>0.1 ms</td>
<td>0</td>
<td>BYTE</td>
<td>0.1 ms</td>
</tr>
<tr>
<td></td>
<td>1 ms</td>
<td>1</td>
<td></td>
<td>0x00</td>
</tr>
<tr>
<td></td>
<td>8 ms</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>32 ms</td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Detect short circuit at outputs</td>
<td>Off</td>
<td>0</td>
<td>BYTE</td>
<td>On</td>
</tr>
<tr>
<td></td>
<td>On</td>
<td>1</td>
<td></td>
<td>0x01</td>
</tr>
<tr>
<td>Behaviour DO at comm. error 1)</td>
<td>Off</td>
<td>0</td>
<td>BYTE</td>
<td>Off</td>
</tr>
<tr>
<td></td>
<td>Last value</td>
<td>1</td>
<td></td>
<td>0x00</td>
</tr>
<tr>
<td></td>
<td>Last value 5 sec</td>
<td>6</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Last value 10 sec</td>
<td>11</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Substitute value</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Substitute value 5 sec</td>
<td>7</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Substitute value 10 sec</td>
<td>12</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Substitute value at output</td>
<td>0...65535</td>
<td>0000h...FFFFh</td>
<td>WORD</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0x0000</td>
</tr>
<tr>
<td>Preventive voltage feedback monitoring for DC0..DC7 2)</td>
<td>Off</td>
<td>0</td>
<td>BYTE</td>
<td>Off</td>
</tr>
<tr>
<td></td>
<td>On</td>
<td>1</td>
<td></td>
<td>0x00</td>
</tr>
<tr>
<td>Detect voltage overflow at outputs 3)</td>
<td>Off</td>
<td>0</td>
<td>BYTE</td>
<td>Off</td>
</tr>
<tr>
<td></td>
<td>On</td>
<td>1</td>
<td></td>
<td>0x00</td>
</tr>
</tbody>
</table>

Remarks:
The parameter Behaviour DO at comm. error is apply to DC and DO channels and only analyzed if the Failsafe-mode is ON.

The state "externally voltage detected" appears, if the output of a channel DC0...DC7 should be switched on while an externally voltage is connected. In this case the start up is disabled, as long as the externally voltage is connected. The monitoring of this state and the resulting diagnosis message can be disabled by setting the parameters to "OFF".

The error state "voltage overflow at outputs" appears, if externally voltage at digital outputs DC0...DC7 and accordingly DO0...DO7 has exceeded the process supply voltage UP3. The according diagnosis message "Voltage overflow on outputs " can be disabled by setting the parameters on "OFF". This parameter should only be disabled in exceptional cases for voltage overflow may produce reverse voltage.

### Diagnosis

Structure of the Diagnosis Block via **PNIO_DEV_ALARM** Function Block.

<table>
<thead>
<tr>
<th>Byte Number</th>
<th>Description</th>
<th>Possible Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Diagnosis Byte, slot number</td>
<td>31 = CI502-PNIO (e. g. error at integrated 8 DI / 8 DO)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 = 1st connected S500 I/O module</td>
</tr>
<tr>
<td></td>
<td></td>
<td>...</td>
</tr>
<tr>
<td></td>
<td></td>
<td>10 = 10th connected S500 I/O module</td>
</tr>
<tr>
<td>2</td>
<td>Diagnosis Byte, module number</td>
<td>According to the I/O bus specification passed on by modules to the fieldbus master</td>
</tr>
<tr>
<td>3</td>
<td>Diagnosis Byte, channel</td>
<td>According to the I/O bus specification passed on by modules to the fieldbus master</td>
</tr>
<tr>
<td>4</td>
<td>Diagnosis Byte, error code</td>
<td>According to the I/O bus specification</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Bit 7 and bit 6, coded error class</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 = E1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 = E2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2 = E3</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3 = E4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Bit 0 to bit 5, coded error description</td>
</tr>
<tr>
<td>5</td>
<td>Diagnosis Byte, flags</td>
<td>According to the I/O bus specification</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Bit 7: 1 = coming error</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Bit 6: 1 = leaving error</td>
</tr>
</tbody>
</table>

In cases of short circuit or overload, the digital outputs are turned off. The modules performs reactivation automatically. Thus an acknowledgement of the errors is not necessary. The error message is stored via the LED.
<table>
<thead>
<tr>
<th>Class</th>
<th>Interface</th>
<th>Device</th>
<th>Module</th>
<th>Channel</th>
<th>Error-Identifier</th>
<th>Error message</th>
<th>Remedy</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Replace I/O module</td>
</tr>
<tr>
<td>3</td>
<td>-</td>
<td>31</td>
<td>31</td>
<td>31</td>
<td>19</td>
<td>Checksum error in the I/O module</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>-</td>
<td>31</td>
<td>31</td>
<td>31</td>
<td>3</td>
<td>Timeout in the I/O module</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>-</td>
<td>31</td>
<td>31</td>
<td>31</td>
<td>40</td>
<td>Different hard-/firmware versions in the module</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>-</td>
<td>31</td>
<td>31</td>
<td>31</td>
<td>43</td>
<td>Internal error in the module</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>-</td>
<td>31</td>
<td>31</td>
<td>31</td>
<td>36</td>
<td>Internal data exchange failure</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>-</td>
<td>31</td>
<td>31</td>
<td>31</td>
<td>9</td>
<td>Overflow diagnosis buffer</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>-</td>
<td>31</td>
<td>31</td>
<td>31</td>
<td>26</td>
<td>Parameter error</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>-</td>
<td>31</td>
<td>31</td>
<td>31</td>
<td>11</td>
<td>Process voltage UP too low</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>-</td>
<td>31</td>
<td>31</td>
<td>31</td>
<td>45</td>
<td>Process voltage UP gone</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>-</td>
<td>31</td>
<td>31</td>
<td>31</td>
<td>17</td>
<td>No communication with I/O device</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>-</td>
<td>1...10</td>
<td>31</td>
<td>31</td>
<td>32</td>
<td>Wrong I/O device type on socket</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>-</td>
<td>1...10</td>
<td>31</td>
<td>31</td>
<td>31</td>
<td>At least one module does not support failsafe function</td>
<td></td>
</tr>
</tbody>
</table>

Module errors:

- **Checksum error in the I/O module**: Replace I/O module
- **Timeout in the I/O module**: Replace I/O module
- **Different hard-/firmware versions in the module**: Replace I/O module
- **Internal error in the module**: Replace I/O module
- **Internal data exchange failure**: Restart
- **Overflow diagnosis buffer**: Check master
- **Parameter error**: Check master
- **Process voltage UP too low**: Check process supply voltage
- **Process voltage UP gone**: Check process supply voltage
- **No communication with I/O device**: Replace I/O module
- **Wrong I/O device type on socket**: Replace I/O module / Check configuration
- **At least one module does not support failsafe function**: Check modules and parameterization
### Classic PLC Interface

<table>
<thead>
<tr>
<th>Class</th>
<th>Interface</th>
<th>Device</th>
<th>Module</th>
<th>Channel</th>
<th>Error-Identi- fier</th>
<th>Error message</th>
<th>Remedy</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>-</td>
<td>1...10</td>
<td>31</td>
<td>5</td>
<td>8</td>
<td>I/O module removed from hot-swap terminal unit or defective module on hot-swap terminal unit ⁹)</td>
<td>Plug I/O module, replace I/O module</td>
</tr>
<tr>
<td>4</td>
<td>-</td>
<td>1...10</td>
<td>31</td>
<td>5</td>
<td>28</td>
<td>Wrong I/O module plugged on hot-swap terminal unit ⁹)</td>
<td>Remove wrong I/O module and plug projected I/O module</td>
</tr>
<tr>
<td>4</td>
<td>-</td>
<td>1...10</td>
<td>31</td>
<td>5</td>
<td>42</td>
<td>No communication with I/O module on hot-swap terminal unit ⁹)</td>
<td>Replace I/O module</td>
</tr>
<tr>
<td>4</td>
<td>-</td>
<td>1...10</td>
<td>31</td>
<td>5</td>
<td>54</td>
<td>I/O module does not support hot swap ⁹) ⁹)</td>
<td>Power off system and replace I/O module</td>
</tr>
<tr>
<td>4</td>
<td>-</td>
<td>1...10</td>
<td>31</td>
<td>6</td>
<td>42</td>
<td>No communication with hot-swap terminal unit ⁹)</td>
<td>Restart, if error persists replace terminal unit</td>
</tr>
<tr>
<td>4</td>
<td>1...6</td>
<td>255</td>
<td>2</td>
<td>0</td>
<td>45</td>
<td>The connected Communication Module has no connection to the network</td>
<td>Check cabling</td>
</tr>
<tr>
<td>4</td>
<td>-</td>
<td>31</td>
<td>31</td>
<td>31</td>
<td>45</td>
<td>Process voltage UP3 too low</td>
<td>Check process voltage</td>
</tr>
<tr>
<td>Class</td>
<td>Interface</td>
<td>Device</td>
<td>Module</td>
<td>Channel</td>
<td>Error-Identifer</td>
<td>Error message</td>
<td>Remedy</td>
</tr>
<tr>
<td>-------</td>
<td>-----------</td>
<td>--------</td>
<td>--------</td>
<td>---------</td>
<td>----------------</td>
<td>---------------</td>
<td>--------</td>
</tr>
<tr>
<td>4</td>
<td>-</td>
<td>31</td>
<td>31</td>
<td>31</td>
<td>46</td>
<td>Reverse voltage from digital outputs DO0..DO7 to UP3</td>
<td>Check terminals</td>
</tr>
<tr>
<td>4</td>
<td>-</td>
<td>31/1...10</td>
<td>31</td>
<td>31</td>
<td>34</td>
<td>No response during initialization of the I/O module</td>
<td>Replace I/O module</td>
</tr>
<tr>
<td>4</td>
<td>-</td>
<td>31</td>
<td>31</td>
<td>31</td>
<td>11</td>
<td>Process voltage UP3 too low</td>
<td>Check process supply voltage</td>
</tr>
<tr>
<td>4</td>
<td>-</td>
<td>31</td>
<td>31</td>
<td>31</td>
<td>45</td>
<td>Process voltage UP3 gone</td>
<td>Check process supply voltage</td>
</tr>
<tr>
<td>4</td>
<td>-</td>
<td>31</td>
<td>31</td>
<td>31</td>
<td>10</td>
<td>Voltage overflow at outputs (above UP3 level)</td>
<td>Check terminals / check process supply voltage</td>
</tr>
</tbody>
</table>

Channel error digital

<table>
<thead>
<tr>
<th>Class</th>
<th>Interface</th>
<th>Device</th>
<th>Module</th>
<th>Channel</th>
<th>Error-Identifer</th>
<th>Error message</th>
<th>Remedy</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>-</td>
<td>31</td>
<td>2</td>
<td>8...15</td>
<td>46</td>
<td>Externally voltage detected at digital output DO0..DO7</td>
<td>Check terminals</td>
</tr>
<tr>
<td>4</td>
<td>-</td>
<td>31</td>
<td>4</td>
<td>0...7</td>
<td>46</td>
<td>Externally voltage detected at digital output DC0..DC7</td>
<td>Check terminals</td>
</tr>
<tr>
<td>4</td>
<td>-</td>
<td>31</td>
<td>2</td>
<td>0...7</td>
<td>47</td>
<td>Short circuit at digital output</td>
<td>Check terminals</td>
</tr>
</tbody>
</table>

Remarks:
In AC500 the following interface identifier applies:
"-" = Diagnosis via bus-specific function blocks; 0...4 or 10 = Position of the Communication Module; 14 = I/O-Bus; 31 = Module itself
The identifier is not contained in the CI502-PNIO diagnosis block.

With "Device" the following allocation applies: 31 = Module itself, 1..10 = Expansion module

With "Module" the following allocation applies dependent of the master:
Module error: 31 = Module itself
Channel error: Module type (1 = AI, 2 = DO, 3 = AO)

This message appears, if externally voltages at one or more terminals DC0...DC7 oder DO0...DO7 cause that other digital outputs are supplied through that voltage (voltage feedback, see description in Electrical Connection “Electrical connection” on page 3006. All outputs of the apply digital output groups will be turned off for 5 seconds. The diagnosis message appears for the whole output group.

The voltage at digital outputs DC0...DC7 and accordingly DO0...DO7 has exceeded the process supply voltage UP3 “Electrical connection” on page 3006. Diagnosis message appears for the whole module.

This message appears, if the output of a channel DC0...DC7 or DO0...DO7 should be switched on while an externally voltage is connected. In this case the start up is disabled, as long as the externally voltage is connected. Otherwise this could produce reverse voltage from this output to other digital outputs. This diagnosis message appears per channel.

Short circuit: After a detected short circuit, the output is deactivated for 2000 ms. Then a new start up will be executed. This diagnosis message appears per channel.

In case of an I/O module doesn't support hot swapping, do not perform any hot-swap operations (also not on any other terminal units (slots)) as modules may be damaged or I/O bus communication may be disturbed.

Diagnosis for hot swap available as of version index F0.

### State LEDs

The LEDs are located at the front of module. There are 2 different groups:
- The 5 system LEDs (PWR, STA1 ETH, STA2 ETH, S-ERR and I/O-Bus) show the operation state of the module and display possible errors.
- The 29 process LEDs (UP, UP3, inputs, outputs, CH-ERR1 to CH-ERR3) show the process supply voltage and the states of the inputs and outputs and display possible errors.

#### Table 644: States of the 5 system LEDs

<table>
<thead>
<tr>
<th>LED</th>
<th>Color</th>
<th>OFF</th>
<th>ON</th>
<th>Flashing</th>
</tr>
</thead>
<tbody>
<tr>
<td>PWR/RUN</td>
<td>Green</td>
<td>Process supply voltage missing</td>
<td>Internal supply voltage OK, module ready for communication with IO Controller</td>
<td>Start-up / preparing communication</td>
</tr>
<tr>
<td></td>
<td>Yellow</td>
<td>---</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>STA1 ETH</td>
<td>Green</td>
<td>---</td>
<td>Device configured, cyclic data exchange running</td>
<td>---</td>
</tr>
<tr>
<td>(System-LED &quot;BF&quot;)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Table 645: States of the 29 process LEDs

<table>
<thead>
<tr>
<th>LED</th>
<th>Color</th>
<th>OFF</th>
<th>ON</th>
<th>Flashing</th>
</tr>
</thead>
<tbody>
<tr>
<td>DC0 to DC7</td>
<td>Yellow</td>
<td>Input/Output is OFF</td>
<td>Input/Output is ON</td>
<td>--</td>
</tr>
<tr>
<td>DI8 to DI15</td>
<td>Yellow</td>
<td>Input is OFF</td>
<td>Input is ON (the input voltage is even displayed if the supply voltage is OFF)</td>
<td>--</td>
</tr>
<tr>
<td>DO8 to DO15</td>
<td>Yellow</td>
<td>Output is OFF</td>
<td>Output is ON</td>
<td>--</td>
</tr>
<tr>
<td>UP</td>
<td>Green</td>
<td>Process supply voltage missing</td>
<td>Process supply voltage OK and initialization finished</td>
<td>--</td>
</tr>
<tr>
<td>UP3</td>
<td>Green</td>
<td>Process supply voltage missing</td>
<td>Process supply voltage OK</td>
<td>--</td>
</tr>
<tr>
<td>CH-ERR1 to CH-ERR3</td>
<td>Red</td>
<td>No error or process supply voltage missing</td>
<td>Internal error</td>
<td>Error on one channel of the corresponding group</td>
</tr>
</tbody>
</table>

### Technical data

The system data of AC500 and S500 ☞ Chapter 1.5.4.5.1 “System data AC500” on page 3058 are valid for standard version.

The system data of AC500-XC ☞ Chapter 1.5.4.6.1 “System data AC500-XC” on page 3099 are valid for the XC version.
Only additional details are therefore documented below.
The technical data are also valid for the XC version.

**Technical data of the module**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Process supply voltages UP/UP3</td>
<td></td>
</tr>
<tr>
<td>Rated value</td>
<td>24 VDC (for inputs and outputs)</td>
</tr>
<tr>
<td>Max. load for the terminals</td>
<td>10 A</td>
</tr>
<tr>
<td>Protection against reversed voltage</td>
<td>Yes</td>
</tr>
<tr>
<td>Rated protection fuse on UP/UP3</td>
<td>10 A fast</td>
</tr>
<tr>
<td>Electrical isolation</td>
<td>Ethernet interface against the rest of the module</td>
</tr>
<tr>
<td>Inrush current from UP (at power up)</td>
<td>On request</td>
</tr>
<tr>
<td>Current consumption via UP (normal operation)</td>
<td>0.15 A</td>
</tr>
<tr>
<td>Current consumption via UP3</td>
<td>0.06 A + 0.5 A max. per output</td>
</tr>
<tr>
<td>Connections</td>
<td>Terminals 1.8 and 2.8 for +24 V (UP) Terminal 3.8 for +24 V (UP3) Terminals 1.9, 2.9 and 3.9 for 0 V (ZP)</td>
</tr>
<tr>
<td>Max. power dissipation within the module</td>
<td>6 W</td>
</tr>
<tr>
<td>Number of digital inputs</td>
<td>8</td>
</tr>
<tr>
<td>Number of digital outputs</td>
<td>8</td>
</tr>
<tr>
<td>Number of configurable digital inputs/outputs</td>
<td>8</td>
</tr>
<tr>
<td>Input data length</td>
<td>12 bytes</td>
</tr>
<tr>
<td>Output data length</td>
<td>20 bytes</td>
</tr>
<tr>
<td>Reference potential for all digital inputs and outputs</td>
<td>Minus pole of the supply voltage, signal name ZP</td>
</tr>
<tr>
<td>Setting of the IO Device identifier</td>
<td>With 2 rotary switches at the front side of the module</td>
</tr>
<tr>
<td>Diagnosis</td>
<td>See Diagnosis and Displays % Chapter 1.5.3.7.5.3.8 “Diagnosis” on page 3016</td>
</tr>
<tr>
<td>Operation and error displays</td>
<td>34 LEDs (totally)</td>
</tr>
<tr>
<td>Weight (without terminal unit)</td>
<td>Ca. 125 g</td>
</tr>
<tr>
<td>Mounting position</td>
<td>Horizontal or vertical with derating (output load reduced to 50 % at 40 °C per group)</td>
</tr>
<tr>
<td>Extended ambient temperature (XC version)</td>
<td>&gt; 60 °C on request</td>
</tr>
<tr>
<td>Cooling</td>
<td>The natural convection cooling must not be hindered by cable ducts or other parts in the switch-gear cabinet.</td>
</tr>
</tbody>
</table>

**NOTICE!**

**Attention:**

All I/O channels (digital and analog) are protected against reverse polarity, reverse supply, short circuit and continuous overvoltage up to 30 VDC.
Multiple overloads

No effects of multiple overloads on isolated multi-channel modules occur, as every channel is protected individually by an internal smart high-side switch.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bus connection</td>
<td>2 x RJ45</td>
</tr>
<tr>
<td>Switch</td>
<td>Integrated</td>
</tr>
<tr>
<td>Technology</td>
<td>Hilscher netX100</td>
</tr>
<tr>
<td>Transfer rate</td>
<td>10/100 Mbit/s (full-duplex)</td>
</tr>
<tr>
<td>Transfer method</td>
<td>According to Ethernet II, IEE802.3</td>
</tr>
<tr>
<td>Ethernet</td>
<td>100 base-TX, internal switch, 2x RJ45 socket</td>
</tr>
<tr>
<td>Expandability</td>
<td>Max. 10 S500 I/O modules</td>
</tr>
<tr>
<td>Adjusting elements</td>
<td>2 rotary switches for generation of an explicit name</td>
</tr>
<tr>
<td>Supported protocols</td>
<td>RTC - real time cyclic protocol, class 1 *)</td>
</tr>
<tr>
<td></td>
<td>RTA - real time acyclic protocol</td>
</tr>
<tr>
<td></td>
<td>DCP - discovery and configuration protocol</td>
</tr>
<tr>
<td></td>
<td>CL-RPC - connectionless remote procedure Call</td>
</tr>
<tr>
<td></td>
<td>LLDP - link layer discovery protocol</td>
</tr>
<tr>
<td></td>
<td>MRP - MRP Client</td>
</tr>
<tr>
<td>Acyclic services</td>
<td>PNIO read / write sequence (max. 1024 bytes per telegram)</td>
</tr>
<tr>
<td></td>
<td>Process-Alarm service</td>
</tr>
<tr>
<td>Supported alarm types</td>
<td>Process Alarm, Diagnostic Alarm, Return of SubModule, Plug Alarm, Pull Alarm</td>
</tr>
<tr>
<td>Min. bus cycle</td>
<td>1 ms</td>
</tr>
<tr>
<td>Conformance class</td>
<td>CC A</td>
</tr>
<tr>
<td>Protective functions (according to IEC 61131-3)</td>
<td>Protected against:</td>
</tr>
<tr>
<td></td>
<td>● short circuit</td>
</tr>
<tr>
<td></td>
<td>● reverse supply</td>
</tr>
<tr>
<td></td>
<td>● overvoltage</td>
</tr>
<tr>
<td></td>
<td>● reverse polarity</td>
</tr>
<tr>
<td></td>
<td>Electrical isolation from the rest of the module</td>
</tr>
<tr>
<td>*) Priorization with the aid of VLAN-ID including priority level</td>
<td></td>
</tr>
</tbody>
</table>

Technical data of the digital inputs

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of channels per module</td>
<td>8</td>
</tr>
<tr>
<td>Distribution of the channels into groups</td>
<td>1 group of 8 channels</td>
</tr>
<tr>
<td>Terminals of the channels DI0 to DI7</td>
<td>Terminals 2.0 to 2.7</td>
</tr>
<tr>
<td>Parameter</td>
<td>Value</td>
</tr>
<tr>
<td>-----------------------------------------------</td>
<td>-----------------------------------------------------------------------</td>
</tr>
<tr>
<td>Reference potential for all inputs</td>
<td>Terminals 1.9...3.9 (Minus pole of the supply voltage, signal name ZP)</td>
</tr>
<tr>
<td>Indication of the input signals</td>
<td>1 yellow LED per channel, the LED is ON when the input signal is high (signal 1)</td>
</tr>
<tr>
<td>Input type (according EN 61131-2)</td>
<td>Type 1</td>
</tr>
<tr>
<td>Input delay (0-&gt;1 or 1-&gt;0)</td>
<td>Typ. 0.1 ms, configurable from 0.1...32 ms</td>
</tr>
<tr>
<td>Input signal voltage</td>
<td>24 VDC</td>
</tr>
<tr>
<td>Signal 0</td>
<td>-3 V...+5 V</td>
</tr>
<tr>
<td>Undefined Signal</td>
<td>&gt; +5 V...&lt; +15 V</td>
</tr>
<tr>
<td>Signal 1</td>
<td>+15 V...+30 V</td>
</tr>
<tr>
<td>Ripple with signal 0</td>
<td>Within -3 V...+5 V</td>
</tr>
<tr>
<td>Ripple with signal 1</td>
<td>Within +15 V...+30 V</td>
</tr>
<tr>
<td>Input current per channel</td>
<td></td>
</tr>
<tr>
<td>Input voltage +24 V</td>
<td>Typ. 5 mA</td>
</tr>
<tr>
<td>Input voltage +5 V</td>
<td>&gt; 1 mA</td>
</tr>
<tr>
<td>Input voltage +15 V</td>
<td>&gt; 2 mA</td>
</tr>
<tr>
<td>Input voltage +30 V</td>
<td>&lt; 8 mA</td>
</tr>
<tr>
<td>Max. cable length</td>
<td></td>
</tr>
<tr>
<td>Shielded</td>
<td>1000 m</td>
</tr>
<tr>
<td>Unshielded</td>
<td>600 m</td>
</tr>
</tbody>
</table>

### Technical data of the digital outputs

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of channels per module</td>
<td>8</td>
</tr>
<tr>
<td>Distribution of the channels into groups</td>
<td>1 group of 8 channels</td>
</tr>
<tr>
<td>Terminals of the channels DO0 to DO7</td>
<td>Terminals 3.0 to 3.7</td>
</tr>
<tr>
<td>Reference potential for all outputs</td>
<td>Terminals 1.9...3.9 (minus pole of the supply voltage, signal name ZP)</td>
</tr>
<tr>
<td>Common power supply voltage</td>
<td>For all outputs terminal 3.8 (plus pole of the supply voltage, signal name UP3)</td>
</tr>
<tr>
<td>Output voltage for signal 1</td>
<td>UP3 (-0.8 V)</td>
</tr>
<tr>
<td>Output delay (0-&gt;1 or 1-&gt;0)</td>
<td>On request</td>
</tr>
<tr>
<td>Output current</td>
<td></td>
</tr>
<tr>
<td>Rated value per channel</td>
<td>500 mA at UP3 = 24 V</td>
</tr>
<tr>
<td>Max. value (all channels together)</td>
<td>4 A</td>
</tr>
<tr>
<td>Leakage current with signal 0</td>
<td>&lt; 0.5 mA</td>
</tr>
<tr>
<td>Fuse for UP3</td>
<td>10 A fast</td>
</tr>
<tr>
<td>Demagnetization with inductive DC load</td>
<td>Via internal varistors (see figure below this table)</td>
</tr>
<tr>
<td>Output switching frequency</td>
<td></td>
</tr>
<tr>
<td>With resistive load</td>
<td>On request</td>
</tr>
</tbody>
</table>
Technical data of the configurable digital inputs/outputs

Each of the configurable I/O channels is defined as input or output by the user program. This is done by interrogating or allocating the corresponding channel.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of channels per module</td>
<td>8 inputs/outputs (with transistors)</td>
</tr>
<tr>
<td>Distribution of the channels into groups</td>
<td>1 group for 8 channels</td>
</tr>
<tr>
<td>If the channels are used as inputs</td>
<td></td>
</tr>
<tr>
<td>Channels DC0...DC07</td>
<td>Terminals 1.0...1.7</td>
</tr>
<tr>
<td>If the channels are used as outputs</td>
<td></td>
</tr>
<tr>
<td>Channels DC0...DC07</td>
<td>Terminals 1.0...1.7</td>
</tr>
<tr>
<td>Indication of the input/output signals</td>
<td>1 yellow LED per channel, the LED is ON when the input/output signal is high (signal 1)</td>
</tr>
<tr>
<td>Electrical isolation</td>
<td>From the Ethernet network</td>
</tr>
</tbody>
</table>
### Technical data of the digital inputs/outputs if used as inputs

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of channels per module</td>
<td>8</td>
</tr>
<tr>
<td>Distribution of the channels into groups</td>
<td>1 group of 8 channels</td>
</tr>
<tr>
<td>Terminals of the channels DC0 to DC7</td>
<td>Terminals 1.0 to 1.7</td>
</tr>
<tr>
<td>Reference potential for all inputs</td>
<td>Terminals 1.9...3.9 (Minus pole of the supply voltage, signal name ZP)</td>
</tr>
<tr>
<td>Indication of the input signals</td>
<td>1 yellow LED per channel, the LED is ON when the input signal is high (signal 1)</td>
</tr>
<tr>
<td>Input type (according EN 61131-2)</td>
<td>Type 1</td>
</tr>
<tr>
<td>Input delay (0-&gt;1 or 1-&gt;0)</td>
<td>Typ. 0.1 ms, configurable from 0.1...32 ms</td>
</tr>
<tr>
<td>Input signal voltage</td>
<td>24 VDC</td>
</tr>
<tr>
<td>Signal 0</td>
<td>-3 V...+5 V</td>
</tr>
<tr>
<td>Undefined Signal</td>
<td>&gt; +5 V...&lt; +15 V</td>
</tr>
<tr>
<td>Signal 1</td>
<td>+15 V...+30 V</td>
</tr>
<tr>
<td>Ripple with signal 0</td>
<td>Within -3 V...+5 V</td>
</tr>
<tr>
<td>Ripple with signal 1</td>
<td>Within +15 V...+30 V</td>
</tr>
<tr>
<td>Input current per channel</td>
<td></td>
</tr>
<tr>
<td>Input voltage +24 V</td>
<td>Typ. 5 mA</td>
</tr>
<tr>
<td>Input voltage +5 V</td>
<td>&gt; 1 mA</td>
</tr>
<tr>
<td>Input voltage +15 V</td>
<td>&gt; 2 mA</td>
</tr>
<tr>
<td>Input voltage +30 V</td>
<td>&lt; 8 mA</td>
</tr>
<tr>
<td>Max. cable length</td>
<td></td>
</tr>
<tr>
<td>Shielded</td>
<td>1000 m</td>
</tr>
<tr>
<td>Unshielded</td>
<td>600 m</td>
</tr>
</tbody>
</table>

*) Due to the direct connection to the output, the demagnetizing varistor is also effective at the input (see figure) above. This is why the difference between UPx and the input signal may not exceed the clamp voltage of the varistor. The varistor limits the voltage to approx. 36 V. Following this, the input voltage must range from -12 V to +30 V when UPx = 24 V and from -6 V to +30 V when UPx = 30 V.

### Technical data of the digital inputs/outputs if used as outputs

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of channels per module</td>
<td>8</td>
</tr>
<tr>
<td>Distribution of the channels into groups</td>
<td>1 group of 8 channels</td>
</tr>
<tr>
<td>Terminals of the channels DC0 to DC7</td>
<td>Terminals 1.0 to 1.7</td>
</tr>
<tr>
<td>Reference potential for all outputs</td>
<td>Terminals 1.9...3.9 (minus pole of the supply voltage, signal name ZP)</td>
</tr>
<tr>
<td>Common power supply voltage</td>
<td>For all outputs terminal 3.8 (plus pole of the supply voltage, signal name UP3)</td>
</tr>
<tr>
<td>Output voltage for signal 1</td>
<td>UP3 (-0.8 V)</td>
</tr>
<tr>
<td>Output delay (0-&gt;1 or 1-&gt;0)</td>
<td>On request</td>
</tr>
<tr>
<td>Output current</td>
<td></td>
</tr>
<tr>
<td>Parameter</td>
<td>Value</td>
</tr>
<tr>
<td>--------------------------------------------------</td>
<td>----------------------------------------------------------------------</td>
</tr>
<tr>
<td>Rated value per channel</td>
<td>500 mA at UP3 = 24 V</td>
</tr>
<tr>
<td>Max. value (all channels together)</td>
<td>4 A</td>
</tr>
<tr>
<td>Leakage current with signal 0</td>
<td>&lt; 0.5 mA</td>
</tr>
<tr>
<td>Fuse for UP3</td>
<td>10 A fast</td>
</tr>
<tr>
<td>Demagnetization with inductive DC load</td>
<td>Via internal varistors (see figure below this table)</td>
</tr>
<tr>
<td>Output switching frequency</td>
<td></td>
</tr>
<tr>
<td>With resistive load</td>
<td>On request</td>
</tr>
<tr>
<td>With inductive loads</td>
<td>Max. 0.5 Hz</td>
</tr>
<tr>
<td>With lamp loads</td>
<td>11 Hz max. at 5 W max.</td>
</tr>
<tr>
<td>Short-circuit-proof / overload proof</td>
<td>Yes</td>
</tr>
<tr>
<td>Overload message (I &gt; 0.7 A)</td>
<td>Yes, after ca. 100 ms</td>
</tr>
<tr>
<td>Output current limitation</td>
<td>Yes, automatic reactivation after short circuit/overload</td>
</tr>
<tr>
<td>Resistance to feedback against 24 V signals</td>
<td>Yes (software-controlled supervision)</td>
</tr>
<tr>
<td>Max. cable length</td>
<td></td>
</tr>
<tr>
<td>Shielded</td>
<td>1000 m</td>
</tr>
<tr>
<td>Unshielded</td>
<td>600 m</td>
</tr>
</tbody>
</table>

The following drawing shows the circuitry of a digital input/output with the varistors for demagnetization when inductive loads are switched off.

![Circuit diagram](image)

1. Digital input/output
2. For demagnetization when inductive loads are turned off

### Technical data of the fast counter

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Used inputs</td>
<td>Terminal 2.0 (DI8), Terminal 2.1 (DI9)</td>
</tr>
<tr>
<td>Used outputs</td>
<td>Terminal 3.0 (DO8)</td>
</tr>
<tr>
<td>Counting frequency</td>
<td>Depending on operation mode:</td>
</tr>
<tr>
<td></td>
<td>Mode 1-6: max. 200 kHz</td>
</tr>
<tr>
<td></td>
<td>Mode 7: max. 50 kHz</td>
</tr>
<tr>
<td></td>
<td>Mode 9: max. 35 kHz</td>
</tr>
<tr>
<td></td>
<td>Mode 10: max. 20 kHz</td>
</tr>
<tr>
<td>Parameter</td>
<td>Value</td>
</tr>
<tr>
<td>--------------------</td>
<td>----------------------------------------------------------------------</td>
</tr>
<tr>
<td>Detailed description</td>
<td>See Fast Counter Chapter 1.5.5.1.10 “Fast counters” on page 3194</td>
</tr>
<tr>
<td>Operating modes</td>
<td>See Operating modes</td>
</tr>
</tbody>
</table>

**Ordering data**

<table>
<thead>
<tr>
<th>Active</th>
<th>Active</th>
<th>Product life cycle phase *)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1SAP 220 700 R0001</td>
<td>CI502-PNIO (V3), PROFINET bus module, 8 DI, 8 DO and 8 DC</td>
<td>Active</td>
</tr>
<tr>
<td>1SAP 420 700 R0001</td>
<td>CI502-PNIO-XC (V3), PROFINET bus module, 8 DI, 8 DO and 8 DC, XC version</td>
<td>Active</td>
</tr>
</tbody>
</table>

*) For planning and commissioning of new installations use modules in Active status only.

### 1.5.3.8 Accessories

#### 1.5.3.8.1 AC500 (standard)

**MC502 - SD memory card**

- Secure digital card
- Solid state flash memory storage

![](SD_card.png)

**Purpose**

The SD memory card is used to back-up user data and store user programs or project source codes as well as to update the internal CPU firmware. The processor modules can be operated with and without SD memory card.

AC500/AC500-eCo processor modules are supplied without SD memory card. It therefore must be ordered separately.

The MC memory card can be read on a PC with a standard memory card reader. AC500 processor modules are equipped with an MC memory card reader.

For AC500-eCo processor modules the device must be equipped with a MC503 SD memory card adaptor.

*The SD memory card has a write protect switch. In the position "LOCK", the card can only be read.*
The use of memory cards other than the MC502 SD memory card is prohibited. ABB is not responsible nor liable for consequences resulting from use of unapproved memory cards.

NOTICE!
Removal of the SD memory card
Do not remove the SD memory card during access. Remove only when the RUN LED does not blink. Otherwise the SD memory card and/or files on it might get corrupted and/or normal PLC operation might be disturbed.

Unpack the SD memory card and insert it into the opening of the front face of the processor module until locked:

Fig. 249: Insertion: PM57x, PM58x, PM59x and PM56xx
To remove the SD memory card, push on the card until it moves forward. By this, the card is unlocked and can be removed.

Technical data

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Memory capacity</td>
<td>Up to 2 GB, for exactly size see type plate</td>
</tr>
<tr>
<td>Temperature range</td>
<td>-20 °C...+85 °C</td>
</tr>
<tr>
<td>No. of writing cycles</td>
<td>&gt; 100 000</td>
</tr>
<tr>
<td>No. of reading cycles</td>
<td>No limitation</td>
</tr>
<tr>
<td>Data safety</td>
<td>&gt; 10 years</td>
</tr>
<tr>
<td>Write Protect Switch</td>
<td>Yes, at the edge of the SD memory card</td>
</tr>
<tr>
<td>Weight</td>
<td>2 g</td>
</tr>
<tr>
<td>Dimensions</td>
<td>24 mm x 32 mm x 2.1 mm</td>
</tr>
</tbody>
</table>

It is not possible to use 100 % of a device’s memory space. About 10 % of the total available space must remain unused at any time to maintain normal device operation.

Further information on using the SD memory card in AC500 PLCs is provided in the chapter Storage Devices § Chapter 1.5.7.2 “SD memory card in AC500” on page 3513.
### Ordering data

<table>
<thead>
<tr>
<th>Part no.</th>
<th>Description</th>
<th>Product life cycle phase *)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1SAP 180 100 R0001</td>
<td>MC502, SD memory card</td>
<td>Active</td>
</tr>
</tbody>
</table>

*) For planning and commissioning of new installations use modules in Active status only.

### TA521 - Lithium battery

- Manganese dioxide lithium battery, 3 V, 560 mAh
- Non-rechargeable

The TA521 lithium battery is the only applicable battery for the AC500 processor modules and PM56xx. Chapter 1.5.3.3.1.1 “PM56xx-2ETH for AC500 V3 products” on page 2274. It cannot be recharged.

The processor modules are supplied without lithium battery. It must be ordered separately. The TA521 lithium battery is used for data (SRAM) and RTC buffering while the processor module is not powered.

See system technology - AC500 battery. Chapter 1.5.5.1.3.2 “AC500 battery” on page 3116.

The CPU monitors the discharge degree of the battery. A warning is issued before the battery condition becomes critical (about 2 weeks before). Once the warning message appears, the battery should be replaced as soon as possible.

#### Handling instructions

- Do not short-circuit or re-charge the battery! It can cause excessive heating and explosion.
- Do not disassemble the battery!
- Do not heat up the battery and not put into fire! Risk of explosion.
- Store the battery in a dry place.
- Replace the battery with supply voltage ON in order not to risk data being lost.
- Recycle exhausted batteries meeting the environmental standards.

### Battery lifetime

The battery lifetime is the time, the battery can store data while the processor module is not powered. As long as the processor module is powered, the battery will only be discharged by its own leakage current.
To avoid a short battery discharge, the battery should always be inserted or replaced while the process module is under power, then the battery is correctly recognized and will not shortly discharged.

### Technical data

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nominal voltage</td>
<td>3 V</td>
</tr>
<tr>
<td>Nominal capacity</td>
<td>560 mAh</td>
</tr>
<tr>
<td>Temperature range (index below C0)</td>
<td>Operating: 0 °C...+60 °C</td>
</tr>
<tr>
<td></td>
<td>Storage: -20 °C...+60 °C</td>
</tr>
<tr>
<td></td>
<td>Transport: -20 °C...+60 °C</td>
</tr>
<tr>
<td>Temperature range (index C0 and above)</td>
<td>Operating: -40 °C...+70 °C</td>
</tr>
<tr>
<td></td>
<td>Storage: -40 °C...+85 °C</td>
</tr>
<tr>
<td></td>
<td>Transport: -40 °C...+85 °C</td>
</tr>
<tr>
<td>Battery lifetime</td>
<td>Typ. 3 years at 25 °C</td>
</tr>
<tr>
<td>Self-discharge</td>
<td>2 % per year at 25 °C</td>
</tr>
<tr>
<td></td>
<td>5 % per year at 40 °C</td>
</tr>
<tr>
<td></td>
<td>20 % per year at 60 °C</td>
</tr>
<tr>
<td>Protection against reverse polarity</td>
<td>Yes, by mechanical coding of the plug.</td>
</tr>
<tr>
<td>Insulation</td>
<td>The battery is completely insulated.</td>
</tr>
<tr>
<td>Connection</td>
<td>Red = plus pole = above at plug, black = minus pole,</td>
</tr>
<tr>
<td>Weight</td>
<td>7 g</td>
</tr>
<tr>
<td>Dimensions</td>
<td>Diameter of the button cell: 24.5 mm</td>
</tr>
<tr>
<td></td>
<td>Thickness of the button cell: 5 mm</td>
</tr>
</tbody>
</table>

### Ordering data

<table>
<thead>
<tr>
<th>Part no.</th>
<th>Description</th>
<th>Product life cycle phase *)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1SAP 180 300 R0001</td>
<td>TA521, lithium battery</td>
<td>Active</td>
</tr>
</tbody>
</table>

*) For planning and commissioning of new installations use modules in Active status only.
Fig. 250: Insertion of the Lithium battery

To ensure proper operation and to prevent data loss, the battery insertion or replacement must be always done with the system under power. Without battery and power supply there is no data buffering possible.

1. Open the battery compartment with the small locking mechanism, press it down and slip down the door. The door is attached to the front face of the processor module and cannot be removed.

2. Remove the TA521 battery from its package and hold it by the small cable.

3. Insert the battery connector into the small connector port of the compartment. The connector is keyed to find the correct polarity (red = plus-pole = above).

4. Insert first the cable and then the battery into the compartment, push it until it reaches the bottom of the compartment.

5. Arrange the cable in order not to inhibit the door to close.

6. Pull-up the door and press until the locking mechanism snaps.

In order to prevent data losses or problems, the battery should be replaced after 3 years of utilisation or at least as soon as possible after receiving the "low battery warning" indication.

Do not use a battery older than 3 years for replacement, do not keep batteries too long in stock.
To ensure proper operation and to prevent data loss, the battery insertion or replacement must be always done with the system under power. Without battery and power supply there is no data buffering possible.

1. Open the battery compartment with the small locking mechanism, press it down and slip down the door. The door is attached to the front face of the processor module and cannot be removed.

2. Remove the old TA521 battery from the battery compartment by pulling it by the small cable. Remove then the small connector from the socket, do this best by lifting it out with a screwdriver.

3. Follow the previous instructions to insert a new battery.

CAUTION!

Risk of explosion!

Do not open, re-charge or disassemble a lithium battery. Attempts to charge lithium batteries lead to overheating and possible explosions.

Prevent them from heat and fire and store them in a dry place.

Never short-circuit or operate lithium batteries with the polarities reversed. The batteries are likely to overheat and explode. Avoid chance short circuiting and therefore do not store batteries in metal containers and do not place them on metallic surfaces. Escaping lithium is a health hazard.

In order to prevent data losses or problems, the battery should be replaced after 3 years of utilisation or at least as soon as possible after receiving the "low battery warning" indication.

Do not use a battery older than 3 years for replacement, do not keep batteries too long in stock.
TA524 is used to cover an unused communication module slot of a terminal base and TB56xx. It protects the terminal base from dust and inadvertent touch.

Handling instructions

TA524 is mounted in the same way as a common communication module.

Technical data

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Weight</td>
<td>50 g</td>
</tr>
<tr>
<td>Dimensions</td>
<td>135 mm x 28 mm x 62 mm</td>
</tr>
</tbody>
</table>

Ordering data

<table>
<thead>
<tr>
<th>Part no.</th>
<th>Description</th>
<th>Product life cycle phase *)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1SAP 180 600 R0001</td>
<td>TA524, dummy communication module</td>
<td>Active</td>
</tr>
</tbody>
</table>
For planning and commissioning of new installations use modules in Active status only.

**TA526 - Wall mounting accessory**

*Purpose*
If a terminal base TB5xx or a terminal unit TU5xx should be mounted with screws, the wall mounting accessories TA526 must be inserted at the rear side first. This plastic parts prevent bending of terminal bases and terminal units while screwing up.

*Handling instructions*
Handling of the wall mounting accessory is described in detail in the section *Mounting and Disassembling the Terminal Unit* \(\circledast\) “Mounting with screws” on page 3071 and *Mounting/Disassembling Terminal Bases and Function Module Terminal Bases* \(\circledast\) “Mounting with screws” on page 3069.

*Technical data*

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Weight</td>
<td>5 g</td>
</tr>
<tr>
<td>Dimensions</td>
<td>67 mm x 35 mm x 5,5 mm</td>
</tr>
</tbody>
</table>

*Ordering data*

<table>
<thead>
<tr>
<th>Part no.</th>
<th>Description</th>
<th>Product life cycle phase *)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1SAP 180 800 R0001</td>
<td>TA526, wall mounting accessory</td>
<td>Active</td>
</tr>
</tbody>
</table>

*) For planning and commissioning of new installations use modules in Active status only.

1.5.3.8.2 S500

**TA523 - Pluggable marker holder**

For labelling the channels of S500 I/O modules.
1 Pluggable Marker Holder TA523
2 Marking stripes to be inserted into the holder
3 Pluggable Marker Holder, snapped on an I/O module

**Purpose**

The Pluggable Marker Holder is used to hold 4 marking stripes, on which the meaning of the I/O channels of I/O modules can be written down. The holder is transparent so that after snapping it onto the module the LEDs shine through.

**Handling instructions**


**Technical data**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use</td>
<td>For labelling channels of I/O modules</td>
</tr>
<tr>
<td>Mounting</td>
<td>Snap-on to the module</td>
</tr>
<tr>
<td>Weight</td>
<td>20 g</td>
</tr>
<tr>
<td>Dimensions</td>
<td>82 mm x 67 mm x 13 mm</td>
</tr>
</tbody>
</table>

**Ordering data**

<table>
<thead>
<tr>
<th>Part no.</th>
<th>Description</th>
<th>Product life cycle phase *)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1SAP 180 500 R0001</td>
<td>TA523, Pluggable Marker Holder (10 pieces)</td>
<td>Active</td>
</tr>
</tbody>
</table>
For planning and commissioning of new installations use modules in Active status only.

TA525 - Plastic markers
Accessory to label AC500 and S500 modules.

1. Module without Plastic Marker TA525
2. Module with Plastic Marker TA525

Purpose
The Plastic Markers are suitable for labelling AC500 and S500 modules (CPUs, communication modules and I/O modules). The small plastic parts can be written with a standard waterproof pen.

Handling instructions
The Plastic Markers are inserted under a slight pressure. For disassembly, a small screwdriver is inserted at the lower edge of the module.

Technical data

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use</td>
<td>For labelling AC500 and S500 modules</td>
</tr>
<tr>
<td>Mounting</td>
<td>Insertion under a slight pressure</td>
</tr>
<tr>
<td>Disassembly</td>
<td>With a small screwdriver</td>
</tr>
<tr>
<td>Scope of delivery</td>
<td>10 pieces</td>
</tr>
</tbody>
</table>

2020/12/10
### Ordering data

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Weight</td>
<td>1 g per piece</td>
</tr>
<tr>
<td>Dimensions</td>
<td>8 mm x 20 mm x 5 mm</td>
</tr>
</tbody>
</table>

### TA525 - Set of 10 white Plastic Markers

**Part no.** 1SAP 180 700 R0001

**Description** TA525, Set of 10 white Plastic Markers

**Product life cycle phase** Active

*) For planning and commissioning of new installations use modules in Active status only.

### TA526 - Wall mounting accessory

**Part no.** 1SAP 180 800 R0001

**Description** TA526, wall mounting accessory

**Product life cycle phase** Active

*) For planning and commissioning of new installations use modules in Active status only.

### Purpose

If a terminal base TB5xx or a terminal unit TU5xx should be mounted with screws, the wall mounting accessories TA526 must be inserted at the rear side first. This plastic parts prevent bending of terminal bases and terminal units while screwing up.

### Handling instructions

Handling of the wall mounting accessory is described in detail in the section *Mounting and Disassembling the Terminal Unit* ≥ “Mounting with screws” on page 3071 and *Mounting/Disassembling Terminal Bases and Function Module Terminal Bases* ≥ “Mounting with screws” on page 3069.

### Technical data

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Weight</td>
<td>5 g</td>
</tr>
<tr>
<td>Dimensions</td>
<td>67 mm x 35 mm x 5,5 mm</td>
</tr>
</tbody>
</table>

### Ordering data

<table>
<thead>
<tr>
<th>Part no.</th>
<th>Description</th>
<th>Product life cycle phase *)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1SAP 180 800 R0001</td>
<td>TA526, wall mounting accessory</td>
<td>Active</td>
</tr>
</tbody>
</table>

*) For planning and commissioning of new installations use modules in Active status only.
TA535 - Protective caps for XC devices

Purpose
Accessory to cover unused connectors of XC devices in salt mist environments. One TA535 package includes different cap types for the following connectors:
- RJ45 connectors
- 9-pole D-sub connector
- FieldBusPlug connector

Protection should be done for all unused slots of -XC devices.

Ordering data

<table>
<thead>
<tr>
<th>Part no.</th>
<th>Description</th>
<th>Product life cycle phase *)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1SAP 182 300 R0001</td>
<td>TA535, Protective Caps for XC devices</td>
<td>Active</td>
</tr>
</tbody>
</table>

*) For planning and commissioning of new installations use modules in Active status only.

1.5.4 System assembly, construction and connection
1.5.4.1 Introduction
This chapter provides information on assembly, construction and connection of control systems of the product family AC500.

The AC500 product family consists of the sub-families:
- AC500 (standard): standard PLC that offers a wide range of performance levels and scalability.
- AC500-eCo: cost-effective PLC that offers total inter-operability with the core AC500 range.
- AC500-S: PLC for special safety requirements in all functional safety applications.

AC500 (standard) and AC500-S provide devices with -XC extension as a product variant. Those devices operate mainly identical to the appropriate AC500 product family, however, can be operated under extreme conditions "Chapter 1.5.4.6.1 "System data AC500-XC" on page 3099."

AC500 product family is characterized by functional modularity, i.e. the devices of all AC500 sub-families can be combined flexible.

As assembly, construction and connection for the devices of the AC500 product family is similar, information that is valid for all sub-families is provided within an overall section. Details that are only valid for a specific AC500 sub-family are described in separate sections.

As assembly, construction and connection for the devices of the AC500 product family is similar, information that is valid for all sub-families is provided within an overall section "Chapter 1.5.4.4 "Overall information (valid for complete AC500 product family)" on page 3044. Details that are only valid for a specific AC500 sub-family are described in separate sections."
Consider the safety instructions

In the description, special attention must be paid to designs using electrical isolation, earthing and EMC measures for the reasons stated. Consider the safety instructions for AC500 product family § Chapter 1.5.2.1 “Safety instructions” on page 2230.

1.5.4.2 Regulations

Appropriate system setup

The following regulations have to be taken into due consideration:

● DIN VDE 0100: "Regulations for the Setting up of Power Installations"
● DIN VDE 0110 Part 1 and Part 2: "The Rating of Creepage Distances and Clearances"
● DIN VDE 0160 and DIN VDE 0660 Part 500: "The Equipment of Power Installations with Electrical Components"

To ensure project success and proper installation of all systems, customers must be familiar and proficient with the following standards and must comply with their directives:

● DIN VDE 0106 Part 100: "Close proximity to dangerous voltages"
● DIN VDE 0160, DIN VDE 0110 Part 1: "Protection against direct contact"

The user has to guarantee that the devices and the components are mounted following these regulations. For operating the machines and installations, other national and international relevant regulations, concerning prevention of accidents and using technical working means, also have to be met.

AC500 devices are designed according to IEC 1131 Part 2 under overvoltage category II per DIN VDE 0110 Part 2.

For direct connection of AC Category III overvoltages provide protection measures for overvoltage category II according to IEC-Report 664/1980 and DIN VDE 0110 Part 1.

Equivalent standards:

● DIN VDE 0110 Part 1 ↔ IEC 664
● DIN VDE 0113 Part 1 ↔ EN 60204 Part 1
● DIN VDE 0660 Part 500 ↔ EN 60439-1 ↔ IEC 439-1

All rights reserved to change design, size, weight, etc.

Qualified personnel

Both the control system AC500 and other components in the vicinity are operated with dangerous contact voltages. Touching parts, which are under such voltages, can cause grave damage to health.

In order to avoid such risks and the occurrence of material damage, persons involved with the assembly, starting up and servicing must possess pertinent knowledge of the following:

● Automation technology sector
● Dealing with dangerous voltages
● Using standards and regulations, in particular VDE, accident prevention regulations and regulations concerning special ambient conditions (e.g. areas potentially endangered by explosive materials, heavy pollution or corrosive influences).

1.5.4.3 Safety instructions

The examples and diagrams in this manual are included solely for illustrative purposes. Because of the many variants and requirements associated with any particular installation, ABB cannot assume responsibility or liability for actual use based on the examples and diagrams.
No patent liability is assumed by ABB with respect to use of information, circuits, equipment or software described in this manual. No liability is assumed for the direct or indirect consequences of the improper use, improper application or inadequate maintenance of these devices. In no event will ABB be responsible or liable for indirect or consequential damages resulting from the use or application of this equipment.

The product family AC500 control system is designed according to EN 61131-2 IEC 61131-2 standards. Data, different from IEC 61131, are caused by the higher requirements of Maritime Services. Other differences are described in the technical data description of the devices.

NOTICE!

Avoidance of electrostatic charging

PLC devices and equipment is sensitive to electrostatic discharge, which can cause internal damage and affect normal operation. Observe the following rules when handling the system:

- Touch a grounded object to discharge potential static.
- Wear an approved grounding wrist strap.
- Do not touch connectors or pins on component boards.
- Do not touch circuit components inside the equipment.
- If available, use a static-safe workstation.
- When not in use, store the equipment in appropriate static-safe packaging.

NOTICE!

PLC damage due to operation conditions

Protect the devices from dampness, dirt and damage during transport, storage and operation!

NOTICE!

PLC damage due to wrong enclosures

Due to their construction (degree of protection IP 20 according to EN 60529) and their connection technology, the devices are suitable only for operation in enclosed switchgear cabinets.

Cleaning instruction

Do not use cleaning agent for cleaning the device.

Use a damp cloth instead.

Connection plans and user software must be created so that all technical safety aspects, legal regulations and standards are observed. In practice, possible shortcircuits and breakages must not be able to lead to dangerous situations. The extent of resulting errors must be kept to a minimum.

Do not operate devices outside of the specified, technical data!

Trouble-free functioning cannot be guaranteed outside of the specified data.
NOTICE!

PLC damage due to missing earthing
- Ensure to earth the devices.
- The earthing (switch cabinet earthing, PE) is supplied both by the mains connection (or 24 V supply voltage) and via DIN rail. The DIN rail must be connected to the earth before the device is subjected to any power. The earthing may be removed only if it is certain that no more power is being supplied to the control system.

In the description for the devices (operating manual or AC500 system description), reference is made at several points to earthing, electrical isolation and EMC measures. One of the EMC measures consists of discharging interference voltages into the earthing via Y-type capacitors. Capacitor discharge currents must basically be able to flow off to the earthing (in this respect, see also VBG 4 and the relevant VDE regulations).

CAUTION!

Do not obstruct the ventilation for cooling!
The ventilation slots on the upper and lower side of the devices must not be covered.

CAUTION!

Run signal and power wiring separately!
Signal and supply lines (power cables) must be laid out so that no malfunctions due to capacitive and inductive interference can occur (EMC).

WARNING!

Labels on or inside the device alert people that dangerous voltage may be present or that surfaces may have dangerous temperatures.

WARNING!

Splaying of strands can cause hazards!
During wiring of terminals with stranded conductors, splaying of strands shall be avoided.
- Ferrules can be used to prevent splaying.
WARNING!

Removal/Insertion under power

The devices are not designed for removal or insertion under power. Because of unforeseeable consequences, it is not allowed to plug or unplug devices with the power being ON.

Make sure that all voltage sources (supply and process voltage) are switched off before you

– connect or disconnect any signal or terminal block
– remove, mount or replace a module.

Disconnecting any powered devices while energized in a hazardous location could result in an electric arc, which could create a flammable ignition resulting in fire or explosion.

Make sure that power is removed and that the area has been thoroughly checked to ensure that flammable materials are not present prior to proceeding.

The devices must not be opened when in operation. The same applies to the network interfaces.

CAUTION!

Use only ABB approved lithium battery modules!

At the end of the battery’s lifetime, always replace it only with a genuine battery module.

CAUTION!

Risk of explosion!

Do not open, re-charge or disassemble a lithium battery. Attempts to charge lithium batteries lead to overheating and possible explosions.

Prevent them from heat and fire and store them in a dry place.

Never short-circuit or operate lithium batteries with the polarities reversed. The batteries are likely to overheat and explode. Avoid chance short circuiting and therefore do not store batteries in metal containers and do not place them on metallic surfaces. Escaping lithium is a health hazard.

Environment considerations

Recycle exhausted batteries. Dispose batteries in an environmentally conscious manner, in accordance to local-authority regulations.
This equipment is intended for use in a Pollution Degree 2 industrial environment, in overvoltage Category II applications (as defined in IEC publication 60664-1), at altitudes up to 2.000 meters without derating.

This equipment is considered Group 1, Class A industrial equipment according to IEC/CISPR Publication 11. Without appropriate precautions, there may be potential difficulties ensuring electromagnetic compatibility in other environments due to conducted as well as radiated disturbance.

This equipment is supplied as "open type" equipment. It must be mounted within an enclosure that is suitably designed for those specific environmental conditions that will be present and appropriately designed to prevent personal injury resulting from accessibility to live parts. The interior of the enclosure must be accessible only by the use of a tool. Subsequent sections of this publication may contain additional information regarding specific enclosure type ratings that are required to comply with certain product safety certifications.

Refer to NEMA Standards publication 250 and IEC publication 60529, as applicable, for explanations of the degrees of protection provided by different types of enclosure. Also see the appropriate sections in this manual.

1.5.4.4 Overall information (valid for complete AC500 product family)

1.5.4.4.1 Serial I/O bus

The synchronized serial I/O bus is the I/O data bus for the I/O modules connected with the processor modules or communication interface modules. Through this bus, I/O and diagnosis data are transferred.

Up to 10 I/O terminal units (for 1 I/O module each) can be added to one terminal base or to one AC500-eCo processor module. The I/O terminal units and the AC500-eCo I/O modules, have a bus input at the left side and a bus output at the right side. Thus the length of the I/O bus increases with the number of attached I/O modules.

The electrical connection of the I/O bus is performed automatically by telescoping the modules on the DIN rail. The I/O bus provides the following signals:

- Supply voltage of 3.3 VDC for feeding the electronic interface components
- 3 data lines for the synchronized serial data exchange
- several control signals
NOTICE!
The I/O bus is not designed for plugging and unplugging modules while in operation. If a module is plugged or replaced while the bus is in operation, the following consequences are possible:
- reset of the station or of the CPU
- system lockup
- damage of the module

WARNING!
Removal/Insertion under power
The devices are not designed for removal or insertion under power. Because of unforeseeable consequences, it is not allowed to plug or unplug devices with the power being ON.
Make sure that all voltage sources (supply and process voltage) are switched off before you
- connect or disconnect any signal or terminal block
- remove, mount or replace a module.
Disconnected any powered devices while energized in a hazardous location could result in an electric arc, which could create a flammable ignition resulting in fire or explosion.
Make sure that power is removed and that the area has been thoroughly checked to ensure that flammable materials are not present prior to proceeding.
The devices must not be opened when in operation. The same applies to the network interfaces.

With its fast data transmission, the I/O bus obtains very low reaction times. Depending on the device and on the version of firmware and Automation Builder, the following numbers of I/O devices can be connected to the I/O bus.

<table>
<thead>
<tr>
<th>Device</th>
<th>Version Automation Builder</th>
<th>Version firmware</th>
<th>Max. number of I/O devices</th>
</tr>
</thead>
<tbody>
<tr>
<td>CANopen bus modules CI581-CN and CI582-CN</td>
<td>As of V2.1.0</td>
<td>All</td>
<td>0</td>
</tr>
<tr>
<td>PROFINET bus modules CI501-PNIO and CI502-PNIO</td>
<td>As of V2.1.0</td>
<td>all</td>
<td>10</td>
</tr>
<tr>
<td>EtherCAT communication interface module CI511-ETHCAT and CI512-ETHCAT</td>
<td>As of V2.1.0</td>
<td>As of V2.0.x</td>
<td>10</td>
</tr>
</tbody>
</table>

Table 646: General data

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Supply voltage, signal level</td>
<td>3.3 V DC ± 10 %</td>
</tr>
<tr>
<td>Max. supply current</td>
<td>On request</td>
</tr>
<tr>
<td>Type of the data interface</td>
<td>Synchronized serial data exchange</td>
</tr>
</tbody>
</table>
### Parameter Table

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bus data transmission speed</td>
<td>1.8 Mb/s</td>
</tr>
<tr>
<td>Minimum bus cycle time</td>
<td>500 µs ¹)</td>
</tr>
<tr>
<td>Galvanic isolation</td>
<td>I/O-Bus is galvanic connected to CPU and communication interface logic circuits. Galvanic isolation of I/O-Bus is I/O module specific. See each module specification for details.</td>
</tr>
<tr>
<td>Protection against electrostatic discharge (ESD)</td>
<td>TB5xx, TB56xx: with protection diodes, no ESD discharge allowed on the port.</td>
</tr>
<tr>
<td>Max. bus length</td>
<td>1 m</td>
</tr>
</tbody>
</table>

¹) Minimum bus cycle time: This value is valid for all module combinations (from 1 to 10 I/O modules)

### Table 647: Wiring (bus connection)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bus connection</td>
<td>Left-side and right-side connection from module to module via a 10-pole HE plug (male at the left side, female at the right side)</td>
</tr>
<tr>
<td>Mechanical connection</td>
<td>Established by the Terminal Units</td>
</tr>
<tr>
<td>Max. bus length</td>
<td>1 m</td>
</tr>
</tbody>
</table>

### 1.5.4.4.2 Mechanical encoding

**Fig. 251: Possible positions for mechanical encoding (1 to 18)**

**NOTICE!**

Terminal units and terminal bases have a mechanical coding which prevents that modules are inserted to wrong places for cases that might result in dangerous parasitic voltages or if modules could be destroyed.
The coding either makes it impossible to insert the module to the wrong place or blocks its electrical function (outputs are not activated).

The following figures show the possible encodings.

Fig. 252: Encoding for processor modules with Ethernet interface

```
 18  18  18
 17  17  17
 16  □16 □16
 □15 □15 □15
 14  14  14
 13  13  13
 12  12  12
 11  11  11
 14  14  14
   9   9   9
   8   8   8
   7   7   7
   6   6   6
   5   5   5
 □  4   4   4
   3 □  3 □  3
   2   2   2
   1   1   1
```

Fig. 253: Encoding for real-time Ethernet modules

```
 18  18  18
 17 □17 □17
 16  16  16
 15 □15 □15
 14 □14 □14
 □13 □13 □13
 12  12  12
 11  11  11
 14  14  14
   9   9   9
   8   8   8
 □  7   7   7
   6   6   6
 □  5 □  5 □  5
 □  4   4   4
 □  3   3   3
 □  2 □  2 □  2
 □  1   1   1
```
Fig. 254: Encoding for communication interface modules

Fig. 255: Encoding for I/O modules (24 VDC)

Fig. 256: Encoding for communication interface modules with PROFINET interface
1.5.4.4.3 Earthing concept (Block diagrams)

NOTICE!

PLC damage due to missing earthing

- Ensure to earth the devices.
- The earthing (switch cabinet earthing, PE) is supplied both by the mains connection (or 24 V supply voltage) and via DIN rail. The DIN rail must be connected to the earth before the device is subjected to any power. The earthing may be removed only if it is certain that no more power is being supplied to the control system.
Block diagram:
Digital I/O modules
1.5.4.4 EMC-conforming assembly and construction

**General principles**

**General considerations** Electric and electronical devices have to work correctly on site. This is also valid when electromagnetic influences affect them in defined and/or expected strength. The devices themselves must not emit electro-magnetic noises.

Advant Controller components have a very high noise immunity.

When the wiring and earthing instructions are met, an error-free operation is given.

High electro-magnetic noises of nearby mounted applications must be taken in consideration during the planning phase.

An EMC compatible earthing concept will also guarantee an error-free operation here.
There are three important principles to be especially considered:

– Keep all connections as short as possible (in particular the earthing conductors)
– Use large conductor cross sections (in particular for the earthing conductors)
– Create low-impedance, i.e. good and large-sized contacts (in particular for the earthing conductors)

Pay attention to the following:

– Use vibration-resistant connections
– Clean metallic contact areas
– Use solid plug and screw-type connections
– Use earth cable shields with clips on a well-grounded metallic surface
– Do not use aluminium parts
– Do not use sheath wires
– Do not use toothed lock washers under screw connections

Fig. 259: Assembly: wrong

Fig. 260: Assembly: correct

Make a connection between the DIN rails and PE (Protective Earth). For this, use an earthing wire with a minimum conductor cross section of 10 mm².

The wire is connected to the DIN rail with an M6 screw.

A large-area contact of the DIN rail with the metallic mounting plate improves the EMC behaviour significantly, as the disturbances can be discharged more effective.

Cable routing

• Route cables meeting the standards.
• Sort the cables into cable groups:
  – Power current cables
  – Power supply cables
  – Signal cables
  – Data cables
● Rout signal cables and data cables separately from the power cables.
  – Separate cable ducts or cable bundles.
  – The distance should be 20 cm or greater.
● Lay signal and data cables close to earthed surfaces.

### Cable shields

- Use only shielded data cables. The shield should be earthed at both ends.
  A cable shield only earthed at one end can only protect from capacitively coupled interference and low-frequency disturbances (50 Hz hum).
- Avoid parasitic currents flowing through the cable shields.
  This can be done by installing current-carrying equipotential bondings.
- Use only cables with braided shields.
  Foil shields are not robust enough, cannot be contacted well and have poor HF properties.
- Use only metallic or metal-plated plugs for shielded data cables.
- Use only shielded cables for analog signals.
  For small signals earth the shield only at one end.
- Earth the cable shield directly with a clip when entering the switch-gear cabinet.
  Do not cut the shield until the cable reaches the module connected.

```
The connection between the PE bar and the shield bar must have a low impedance.
```

### Switch-gear cabinet

#### Connections

The connections between the switch-gear cabinet, the mounting plates, the PE bar and the shield bar must have a low impedance.

#### Earthing

Earth the switch-gear cabinet doors with short and highly flexible conductors.

#### Illumination

Only use filament lamps (bulbs) or fluorescent tubes with interference suppression.

#### For supplying the PC

Use the mains socket which is located inside the switch-gear cabinet.

☞ Chapter 1.5.4.5.2.1 “Switchgear cabinet assembly” on page 3062

### Reference potential

- Provide a uniform reference potential in the entire installation and earth all electrical appliances if possible.
- Route your earthing conductors in a star configuration so that no earth loops can occur.

### Equipotential bonding

The Installation of equipotential bondings are necessary if there are present or expected potential differences between parts of your application.
- The impedance of equipotential bonding must be equal or lower than 10 % of the shield impedance of the shielded signal cables between the same points.
- The conductor cross section of a equipotential bonding must be 16 mm² to withstand the maximum possible compensating current.
- Equipotential bondings and shielded signal cables should be laid close to each other.
- Equipotential bondings must be connected to PE with low impedance.

Fig. 261: AC500, equipotential bonding

1. Cabinet 1
2. Cabinet 2
3. Power supply for the CPU
4. Fuse for the CPU power
5. Power supply for the I/Os
6. Fuse for the I/O power
7. For fuses for the contacts of the relay outputs
8. 0V rail
9. Earthing of the 0V rail
10. Cabinet earthing
11. Equipotential bonding between the cabinets min. 16 mm²
12. Cable shields earthing
13. Fieldbus connection (e.g. Ethernet)

1.5.4.4.5 Power consumption of an entire station

The power consumption of a complete station consists of the sum of all individual consumptions.
● Consumers over terminals L+ and M on the AC500 terminal base/AC500-eCo CPU:
  – CPU itself
  – I/O modules attached on the I/O bus
  – Communication modules attached (AC500 terminal base)
● Consumers over the process supply voltage terminals ZP and UP of the AC500 Terminal Units / the L+/M or UP/ZP terminals of the AC500-eCo I/O modules:
  – Digital I/O modules
  – Analog I/O modules

The two supply voltages can be provided by the same power supply unit. The CPU and the I/O modules should, however, be fused separately. Of course also separate power supplies are possible.

**Calculation of the total current consumption**

**Example**

In the example, the AC500 control system consists of the following devices:

- CPU: PM5xx-ETH or PM5xxx-ETH
- 4 communication modules
- 7 I/O modules (digital and analog)
- As well as the required terminal bases and terminal units
Because of the high total current consumption of the digital I/O modules (from UP = 24 VDC), the supply is divided up into several electric circuits fused separately.

The maximum permitted total current over the supply terminals of the I/O terminal units is 8 A.

The total current can be calculated as follows:

\[ I_{\text{Total}} = I_{\text{LOGIC}} + I_{\text{UP}} \]

with the assumptions

\[ I_{\text{LOGIC}} = I_{\text{CPU}} + I_{\text{I/O bus}} + I_{C1} + I_{C2} + I_{C3} + I_{C4} \] (CPU + communication modules + I/O bus)

\[ I_{\text{I/O bus}} = \text{Number of expansion modules} \times \text{Current consumption through the I/O bus per module} \]

and

\[ I_{\text{UP}} = I_{\text{UP1}} + I_{\text{LOAD1}} + I_{\text{UP2}} + I_{\text{LOAD2}} + I_{\text{UP3}} + I_{\text{LOAD3}} + I_{\text{UP4}} + I_{\text{LOAD4}} + I_{\text{UP5}} + I_{\text{LOAD5}} + I_{\text{UP6}} + I_{\text{LOAD6}} + I_{\text{UP7}} + I_{\text{LOAD7}} \]

If one assumes that all outputs are switched on and are operated with their maximum permitted load currents (under compliance with the maximum permitted currents at the supply terminals), then the following values are the result for an example shown above:

<table>
<thead>
<tr>
<th></th>
<th>( I_{\text{CPU} ,*} )</th>
<th>( I_{\text{Cx} ,*} )</th>
<th>( I_{\text{I/O bus} ,*} )</th>
<th>( I_{\text{UPx} ,*} )</th>
<th>( I_{\text{LOADx} ,*} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU / communication module part</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CPU</td>
<td>0.110 A</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>C1</td>
<td>-</td>
<td>0.050 A</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>C2</td>
<td>-</td>
<td>0.085 A</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>C3</td>
<td>-</td>
<td>0.050 A</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>C4</td>
<td>-</td>
<td>0.050 A</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>I/O module part</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Analog1</td>
<td>-</td>
<td>-</td>
<td>0.002 A</td>
<td>0.150 A</td>
<td>-</td>
</tr>
<tr>
<td>Analog2</td>
<td>-</td>
<td>-</td>
<td>0.002 A</td>
<td>0.150 A</td>
<td>0.160 A</td>
</tr>
<tr>
<td>Analog3</td>
<td>-</td>
<td>-</td>
<td>0.002 A</td>
<td>0.100 A</td>
<td>0.080 A</td>
</tr>
<tr>
<td>Analog4</td>
<td>-</td>
<td>-</td>
<td>0.002 A</td>
<td>0.100 A</td>
<td>0.080 A</td>
</tr>
<tr>
<td>Digital1</td>
<td>-</td>
<td>-</td>
<td>0.002 A</td>
<td>0.050 A</td>
<td>8.000 A</td>
</tr>
<tr>
<td>Digital2</td>
<td>-</td>
<td>-</td>
<td>0.002 A</td>
<td>0.050 A</td>
<td>8.000 A</td>
</tr>
<tr>
<td>Digital3</td>
<td>-</td>
<td>-</td>
<td>0.002 A</td>
<td>0.050 A</td>
<td>8.000 A</td>
</tr>
<tr>
<td>( \Sigma \text{ columns} )</td>
<td>0.110 A</td>
<td>0.235 A</td>
<td>0.014 A</td>
<td>0.650 A</td>
<td>24.320 A</td>
</tr>
<tr>
<td>( \Sigma I_{\text{LOGIC}} \approx 0.4 A )</td>
<td>( \Sigma I_{\text{UP}} \approx 25 A )</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

\[ I_{\text{Total}} \approx 25.4 A \]

*) All values in this column are exemplary values

Dimensoning of the fuses

To be able to select the fuses for the station correctly, both the current consumption and the inrush currents (melting integral for the series-connected fuse) must be taken into consideration.
### 1.5.4.4.6 Decommissioning

**Secure decommissioning of a functional SPS**

1. Delete the runtime licenses ‡ Chapter 1.5.6.2.2.2.4 “Returning a license” on page 3263.
2. Delete certificates available on the CPU ‡ Chapter 1.7.2.4.1 “View ‘Security-Screen’ - ‘Devices’” on page 3599.
3. Delete applications ‡ Chapter 1.3.1.25.2.6.11 “Command ‘Reset Origin’” on page 932 ‡ Chapter 1.3.1.25.2.6.12 “Command ‘Reset Origin Device’” on page 933.
4. Delete applications from SD memory card, if available ‡ Chapter 1.3.1.15 “Copying files to/from PLC” on page 357.
5. If available, remove SD memory card and battery from CPU.
6. Delete all user accounts and user data ‡ “Tab ‘User’” on page 891.
7. Demount and dispose the hardware modules ‡ Chapter 1.5.4.5.3 “Mounting and demounting” on page 3068 ‡ Chapter 1.5.4.4.7 “Recycling and disposal” on page 3057.

**Secure decommissioning of a not functional SPS**

- The device is not functional and you cannot execute the steps for a secure decommissioning.
- Physically destroy the device.
- This ensures that the credentials that are stored in the device, can not be misused.

### 1.5.4.7 Recycling and disposal

Devices of AC500 product family must not be disposed as unsorted domestic waste. The devices contain valuable raw material which can be recycled. Remove the battery - if existing. Dispose the products according to the local regulations.

Devices of AC500 product family are free from pollutants and are no danger for the environment.
### 1.5.4.5 AC500 (Standard)

### 1.5.4.5.1 System data AC500

#### Environmental conditions

*Table 648: Process and supply voltages*

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>24 VDC</td>
<td></td>
</tr>
<tr>
<td>Voltage</td>
<td>24 V (-15 %, +20 %)</td>
</tr>
<tr>
<td>Max. ripple</td>
<td>5 % from nominal value</td>
</tr>
<tr>
<td>Protection against reverse polarity</td>
<td>Yes</td>
</tr>
<tr>
<td>120 VAC</td>
<td></td>
</tr>
<tr>
<td>Voltage</td>
<td>120 V (-15 %, +10 %)</td>
</tr>
<tr>
<td>Frequency</td>
<td>50/60 Hz (-6 %, +4 %)</td>
</tr>
<tr>
<td>230 VAC</td>
<td></td>
</tr>
<tr>
<td>Voltage</td>
<td>230 VAC (-15 %, +10 %)</td>
</tr>
<tr>
<td>Frequency</td>
<td>50/60 Hz (-6 %, +4 %)</td>
</tr>
<tr>
<td>120 VAC...240 VAC wide range supply</td>
<td></td>
</tr>
<tr>
<td>Voltage</td>
<td>120 V...240 V (-15 %, +10 %)</td>
</tr>
<tr>
<td>Frequency</td>
<td>50/60 Hz (-6 %, +4 %)</td>
</tr>
</tbody>
</table>

**Allowed interruptions of power supply, according to EN 61131-2**

<table>
<thead>
<tr>
<th>Supply</th>
<th>Interruption</th>
</tr>
</thead>
<tbody>
<tr>
<td>DC supply</td>
<td>Interruption &lt; 10 ms, time between 2 interruptions &gt; 1 s, PS2</td>
</tr>
<tr>
<td>AC supply</td>
<td>Interruption &lt; 0.5 periods, time between 2 interruptions &gt; 1 s</td>
</tr>
</tbody>
</table>

---

**NOTICE!**

Exceeding the maximum power supply voltage for process or supply voltages could lead to unrecoverable damage of the system. The system could be destroyed.

**NOTICE!**

Improper voltage level or frequency range which cause damage of AC inputs:

- AC voltage above 264 V
- Frenquency below 47 Hz or above 62.4 Hz

**NOTICE!**

Improper connection leads cause overtemperature on terminals.

PLC modules may be destroyed by using wrong cable type, wire size and cable temperature classification.
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Temperature</td>
<td></td>
</tr>
<tr>
<td>Operating</td>
<td>0 °C...+60 °C: Horizontal mounting of modules.</td>
</tr>
<tr>
<td></td>
<td>0 °C...+40 °C: Vertical mounting of modules.</td>
</tr>
<tr>
<td></td>
<td>Output load reduced to 50 % per group.</td>
</tr>
<tr>
<td>Storage</td>
<td>-40 °C...+70 °C</td>
</tr>
<tr>
<td>Transport</td>
<td>-40 °C...+70 °C</td>
</tr>
<tr>
<td>Humidity</td>
<td>Max. 95 %, without condensation</td>
</tr>
<tr>
<td>Air pressure</td>
<td></td>
</tr>
<tr>
<td>Operating</td>
<td>&gt; 800 hPa / &lt; 2000 m</td>
</tr>
<tr>
<td>Storage</td>
<td>&gt; 660 hPa / &lt; 3500 m</td>
</tr>
<tr>
<td>Ingress protection</td>
<td>IP20</td>
</tr>
</tbody>
</table>

Creepage distances and clearances

The creepage distances and clearances meet the requirements of the overvoltage category II, pollution degree 2.

Insulation test voltages, routine test

According to EN 61131-2

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>230 V circuits against other circuitry</td>
<td>2500 V</td>
</tr>
<tr>
<td></td>
<td>1.2/50 μs</td>
</tr>
<tr>
<td>120 V circuits against other circuitry</td>
<td>1500 V</td>
</tr>
<tr>
<td></td>
<td>1.2/50 μs</td>
</tr>
<tr>
<td>120 V...240 V circuits against other circuitry</td>
<td>2500 V</td>
</tr>
<tr>
<td></td>
<td>1.2/50 μs</td>
</tr>
<tr>
<td>24 V circuits (supply, 24 V inputs/outputs, analogue inputs/outputs), if they are electrically isolated against other circuitry</td>
<td>500 V</td>
</tr>
<tr>
<td></td>
<td>1.2/50 μs</td>
</tr>
<tr>
<td>COM interfaces, electrically isolated</td>
<td>500 V</td>
</tr>
<tr>
<td></td>
<td>1.2/50 μs</td>
</tr>
<tr>
<td>Ethernet</td>
<td>500 V</td>
</tr>
<tr>
<td></td>
<td>1.2/50 μs</td>
</tr>
<tr>
<td>230 V circuits against other circuitry</td>
<td>1350 V</td>
</tr>
<tr>
<td></td>
<td>AC 2 s</td>
</tr>
<tr>
<td>120 V circuits against other circuitry</td>
<td>820 V</td>
</tr>
<tr>
<td></td>
<td>AC 2 s</td>
</tr>
<tr>
<td>120 V...240 V circuits against other circuitry</td>
<td>1350 V</td>
</tr>
<tr>
<td></td>
<td>AC 2 s</td>
</tr>
</tbody>
</table>
### Parameter

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>24 V circuits (supply, 24 V inputs/outputs, analogue inputs/outputs), if they are electrically isolated against other circuitry</td>
<td>350 V \ AC 2 s</td>
</tr>
<tr>
<td>COM interfaces, electrically isolated</td>
<td>350 V \ AC 2 s</td>
</tr>
<tr>
<td></td>
<td>Not applicable \ Not applicable</td>
</tr>
<tr>
<td>Ethernet</td>
<td>350 V \ AC 2 s</td>
</tr>
</tbody>
</table>

**According to IEC 61010-2-01:2013**

The content of the following table is only valid for PM56xx and TB56xx.

### Table 649: Insulation, test voltages and continuous voltages

<table>
<thead>
<tr>
<th>Insulation</th>
<th>Test Voltage</th>
<th>Continuous Voltage</th>
</tr>
</thead>
<tbody>
<tr>
<td>COM interfaces, electrically isolated</td>
<td>1.1 mm</td>
<td>1216 V DC (60 s) 1500 V (1.2/50 µs)</td>
</tr>
<tr>
<td>CAN interface, electrically isolated</td>
<td>1.1 mm</td>
<td>1216 V DC (60 s) 1500 V (1.2/50 µs)</td>
</tr>
<tr>
<td>Ethernet</td>
<td>1.1 mm</td>
<td>1500 V rms (50-60 Hz, 60 s) 2400 V (1.2/50 µs)</td>
</tr>
</tbody>
</table>

### Power supply units

For the supply of the modules, power supply units according to PELV specifications must be used.

### Electromagnetic compatibility

#### Table 650: Range of use

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Industrial applications</td>
<td>Yes</td>
</tr>
<tr>
<td>Domestic applications</td>
<td>No</td>
</tr>
</tbody>
</table>

#### Table 651: Immunity against electrostatic discharge (ESD), according to IEC 61000-4-2, zone B, criterion B

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Electrostatic voltage in case of air discharge</td>
<td>8 kV</td>
</tr>
<tr>
<td>Electrostatic voltage in case of contact discharge</td>
<td>4 kV, in a closed switch-gear cabinet 6 kV ¹)</td>
</tr>
</tbody>
</table>
In order to prevent operating malfunctions, it is recommended, that the operating personnel discharge themselves prior to touching communication connectors or perform other suitable measures to reduce effects of electrostatic discharges.

The connectors between the Terminal Bases and Processor Modules or Communication Modules must not be touched during operation. The same is valid for the I/O-Bus with all modules involved.

1) High requirement for shipping classes are achieved with additional specific measures (see specific documentation).

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>ESD with communication connectors</td>
<td>In order to prevent operating malfunctions, it is recommended, that the operating personnel discharge themselves prior to touching communication connectors or perform other suitable measures to reduce effects of electrostatic discharges.</td>
</tr>
<tr>
<td>ESD with connectors of terminal bases</td>
<td>The connectors between the Terminal Bases and Processor Modules or Communication Modules must not be touched during operation. The same is valid for the I/O-Bus with all modules involved.</td>
</tr>
</tbody>
</table>

Table 652: Immunity against the influence of radiated (CW radiated), according to IEC 61000-4-3, zone B, criterion A

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test field strength</td>
<td>10 V/m</td>
</tr>
</tbody>
</table>

Table 653: Immunity against fast transient interference voltages (burst), according to IEC 61000-4-4, zone B, criterion B

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Supply voltage units (DC)</td>
<td>2 kV</td>
</tr>
<tr>
<td>Supply voltage units (AC)</td>
<td>2 kV</td>
</tr>
<tr>
<td>Digital inputs/outputs (24 VDC)</td>
<td>1 kV</td>
</tr>
<tr>
<td>Digital inputs/outputs (120 VAC…240 VAC)</td>
<td>2 kV</td>
</tr>
<tr>
<td>Analog inputs/outputs</td>
<td>1 kV</td>
</tr>
<tr>
<td>CS31 system bus</td>
<td>1 kV</td>
</tr>
<tr>
<td>Serial RS-485 interfaces (COM)</td>
<td>1 kV</td>
</tr>
<tr>
<td>Serial RS-232 interfaces (COM, not for PM55x and PM56x)</td>
<td>1 kV</td>
</tr>
<tr>
<td>Ethernet</td>
<td>1 kV</td>
</tr>
<tr>
<td>I/O supply (DC-out)</td>
<td>1 kV</td>
</tr>
</tbody>
</table>

Table 654: Immunity against the influence of line-conducted interferences (CW conducted), according to IEC 61000-4-6, zone B, criterion A

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test voltage</td>
<td>3V zone B, 10 V is also met.</td>
</tr>
<tr>
<td>High energy surges</td>
<td>According to IEC 61000-4-5, zone B, criterion B</td>
</tr>
<tr>
<td>Power supply DC</td>
<td>1 kV CM / 0.5 kV DM ²</td>
</tr>
<tr>
<td>DC I/O supply</td>
<td>0.5 kV CM / 0.5 kV DM ²</td>
</tr>
<tr>
<td>Communication Lines, shielded</td>
<td>1 kV CM ²</td>
</tr>
<tr>
<td>AC I/O unshielded</td>
<td>2 kV CM / 1 kV DM ²</td>
</tr>
</tbody>
</table>
Parameter | Value
--- | ---
I/O analog, I/O DC unshielded | 1 kV CM / 0.5 kV DM \(^2\)
Radiation (radio disturbance) | According to IEC 55011, group 1, class A

\(^2\) CM = Common Mode, DM = Differential Mode

### Mechanical data

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mounting</td>
<td>Horizontal</td>
</tr>
<tr>
<td>Degree of protection</td>
<td>IP 20</td>
</tr>
<tr>
<td>Housing</td>
<td>Classification V-2 according to UL 94</td>
</tr>
<tr>
<td>Vibration resistance acc. to EN 61131-2</td>
<td>all three axes&lt;br&gt;2 Hz...8.4 Hz, continuous 3.5 mm&lt;br&gt;8.4 Hz...150 Hz, continuous 1 g (higher values on request)</td>
</tr>
<tr>
<td>Shock test</td>
<td>All three axes&lt;br&gt;15 g, 11 ms, half-sinusoidal</td>
</tr>
</tbody>
</table>

### Approvals and certifications

Information on approvals and certificates can be found in the corresponding chapter of the *Main catalog, PLC Automation*.

### 1.5.4.5.2 Mechanical dimensions

#### Switchgear cabinet assembly

*Information on EMC-conforming assembly and construction is provided within the overall functions section Chapter 1.5.4.4.4 “EMC-conforming assembly and construction” on page 3051.*

#### PLC enclosure

**NOTICE!**

**PLC damage due to wrong enclosures**

Due to their construction (degree of protection IP 20 according to EN 60529) and their connection technology, the devices are suitable only for operation in enclosed switchgear cabinets.

To protect PLCs against:
• unauthorized access,
• dusting and pollution,
• moisture and wetness and
• mechanical damage,

switchgear cabinet IP54 for common dry factory floor environment is suitable.

Maintain spacing from:
• enclosure walls
• wireways
• adjacent equipment

Allow a minimum of 20 mm clearance on all sides. This provides ventilation and electrical isolation.

It is recommended to mount the modules on an earthed mounting plate, or an earthed DIN rail, independent of the mounting location.
Fig. 262: Installation of AC500/S500 modules in a switch-gear cabinet

1 Cable duct
2 Distance from cable duct ≥20 mm
3 Mounting plate, earthed

**NOTICE!**

Horizontal mounting is highly recommended.

Vertical mounting is possible, however, derating consideration should be made to avoid problems with poor air circulation and overheating (see Chapter 1.5.4.5.1.1 “Environmental conditions” on page 3058).
By vertical mounting, always place an end-stop terminal block (e.g. type BADL, P/N: 1SNA399903R0200) on the bottom and on the top of the modules to properly secure the modules.

By high vibration applications and horizontal mounting, we also recommend to place end-stop terminals at the right and left side of the device to properly secure the modules, e.g. type BADL, P/N: 1SNA399903R0200.

### Mechanical dimensions AC500

**Dimensions: terminal bases**

![Diagram of terminal bases](image)

**Fig. 263: Terminal bases, side view and front view**
Mechanical dimensions S500

Dimensions: terminal units

Fig. 264: Terminal bases with processor modules, side view and front view

Fig. 265: Terminal units, side view and front view
Fig. 266: Terminal units and S500 modules, side view and front view

Fig. 267: Terminal base (for comparison)
1.5.4.5.3 Mounting and demounting

The control system is designed to be mounted to a well-grounded mounting surface such as a metal panel. Additional grounding connections from the mounting tabs or DIN rail (if used), are not required unless the mounting surface cannot be grounded.

During panel or DIN rail mounting of all devices, be sure that all debris (metal chips, wire strands, etc.) is kept from falling into the controller. Debris that falls into the controller could cause damage while the controller is energized.

All devices are grounded through the DIN rail to chassis ground. Use zinc plated yellow-chromate stell DIN rail to assure proper grounding. The use of other DIN rail materials (e.g. aluminium, plastic, etc.) that can corrode, oxidize, or are poor conductors, can result in improper or intermittent grounding.

Mounting/Demounting terminal bases and function module terminal bases

Demounting on DIN rail

1. Mount DIN rail 7.5 mm or 15 mm.
2. Mount the Terminal Base/Function Module Terminal Base:

⇒ The Terminal Base is put on the DIN rail above and then snapped-in below.
3. The demounting is carried out in a reversed order.

**Mounting with screws**

If the Terminal Base should be mounted with screws, Wall Mounting Accessories TA526 must be inserted at the rear side first. These plastic parts prevent bending of the Terminal Base while screwing on. TB560x and TB561x need one TA526, TB562x, TB564x and TB566x need two TA526.

---

By wall mounting, the Terminal Base is earthed through the screws. It is necessary that

- the screws have a conductive surface (e.g. steel zinc-plated or brass nickel-plated)
- the mounting plate is earthed
- the screws have a good electrical contact to the mounting plate
The following procedure allows you to use the mounted modules as a template for drilling holes in the panel. Due to module mounting hole tolerance, it is important to follow these procedures:

1. On a clean work surface, mount no more than 3 modules (e.g. one Terminal Base and two Terminal Units).
2. Using the mounted modules as a template, carefully mark the center of all module-mounting holes on the panel.
3. Return the mounted modules to the clean work surface, including any previously mounted modules.
4. Drill and tap the mounting holes for the screws (M4 or #8 recommended).
5. Place the modules back on the panel and check for proper hole alignment.
6. Attach the modules to the panel using the mounting screws.
7. Repeat the steps for all remaining modules.

If mounting more modules, mount only the last one of this group and put the others aside. This reduces remounting time during drilling and tapping of the next group.

Mounting and demounting the terminal unit

Mounting on DIN rail

1. Mount DIN rail 7.5 mm or 15 mm.
2. Mount the Terminal Unit.

The Terminal Unit is snapped into the DIN rail in the same way as the Terminal Base. Once secured to the DIN rail, slide the Terminal Unit to the left until it fully locks into place creating a solid mechanical and electrical connection.

When attaching the devices, make sure the bus connectors are securely locked together to ensure proper electrical connection. Max. 10 Terminal Units can be attached.
3. Demounting: A screwdriver is inserted in the indicated place to separate the Terminal Units.

Mounting with screws

If the Terminal Unit should be mounted with screws, Wall Mounting Accessories TA526 must be inserted at the rear side first. These plastic parts prevent bending of the Terminal Base while screwing on.

![Fig. 270: Fastening with screws](image)

**Practical tip**

The following procedure allows you to use the mounted modules as a template for drilling holes in the panel. Due to module mounting hole tolerance, it is important to follow these procedures:

- the screws have a conductive surface (e.g. steel zinc-plated or brass nickel-plated)
- the mounting plate is earthed
- the screws have a good electrical contact to the mounting plate
1. On a clean work surface, mount no more than 3 modules (e.g. one Terminal Base and two Terminal Units).
2. Using the mounted modules as a template, carefully mark the center of all module-mounting holes on the panel.
3. Return the mounted modules to the clean work surface, including any previously mounted modules.
4. Drill and tap the mounting holes for the screws (M4 or #8 recommended).
5. Place the modules back on the panel and check for proper hole alignment.
6. Attach the modules to the panel using the mounting screws.

**Note:** If mounting more modules, mount only the last one of this group and put the others aside. This reduces remounting time during drilling and tapping of the next group.

7. Repeat the steps for all remaining modules.

Mounting processor modules PM57x, PM58x, PM59x and PM56xx

1. After mounting the Terminal Base on the DIN rail, mount the Processor Module.
2. Press the Processor Module into the Terminal Base until it locks in place.

3. The demounting is carried out in a reversed order. Press above and below, then remove the Processor Module.

Mounting and demounting the I/O modules

After mounting the Terminal Unit, mount the I/O Modules.
1. Press the I/O Module into the Terminal Unit until it locks in place.

2. The demounting is carried out in a reversed order. Press above and below, then remove the module.
Mounting and demounting the communication modules

Communication Modules are mounted on the left side of the Processor Module on the same Terminal Base. The electrical connection is established automatically when mounting the Communication Module.

**NOTICE!**

Risk of damaging the PLC modules!

- Overvoltages and short circuits might damage the PLC modules.
  - Make sure that all voltage sources (supply voltage and process supply voltage) are switched off before you begin with operations at the system.
  - Never connect any voltages or signals to reserved terminals (marked with ---). Reserved terminals may carry internal voltages.

After mounting the Terminal Base, mount the Communication Modules.

1. First insert the bottom nose of the Communication Module into the dedicated holes of the Terminal Base. Then, rotate the Communication Module on the dedicated Terminal Base slot until it is locked in place.

**NOTICE!**

Risk of malfunctions!

- Unused slots for communication modules are not protected against accidental physical contact.
  - Unused slots for communication modules must be covered with dummy communication modules (*TA524 § Chapter 1.5.3.8.1.3 “TA524 - Dummy communication module” on page 3034*) to achieve IP20 rating.
  - I/O bus connectors must not be touched during operation.
2. The demounting is carried out in a reversed order.
Press above and below, then rotate the Communication Module and remove it.

1.5.4.5.4 Connection and wiring

For detailed information such as technical data of your mounted devices (AC500 product family) refer to the hardware device description of the appropriate device.

---

**NOTICE!**
**Attention:**
- The devices should be installed by trained persons with knowledge of wiring electronic devices. In case of bad wiring, the following problems could occur:
  - On the Terminal Base, the terminals L+ and M are doubled. If the power supply is badly connected, a short circuit could happen and lead to a destruction of the power supply or its fuse. If no suitable fuse exists, the Terminal Base itself could be destroyed.
  - The Terminal Bases and all electronic modules and Terminal Units are protected against reverse polarity.
  - All necessary measures should be carried out to avoid damages to modules and wiring. Notice the wiring plans and connection examples.

---

**NOTICE!**
**Attention:**
- All I/O channels (digital and analog) are protected against reverse polarity, reverse supply, short circuit and continuous overvoltage up to 30 VDC.

---

**NOTICE!**
**Attention:**
- Due to possible loss of communication, the communication cables should be fixed with cable duct or bracket or clamp during application.
Power supply

AC500 system power supply

As soon as the power supply of the Processor Module (CPU) is higher than the minimum Process and supply voltage (see “Chapter 1.5.4.5.1.1 “Environmental conditions” on page 3058), the power supply detection is activated and the Processor Module is started. Power supply of Processor Module and I/O modules should be powered on the same time, otherwise the Processor Module will not switch to run after startup.

When during operation the power supply is going down lower than the minimum Process and supply voltage (see “Chapter 1.5.4.5.1.1 “Environmental conditions” on page 3058) for more than 10 ms, the Processor Module is switched to safety mode (display shows “AC500”). A restart of the Processor Module only occurs by switching the power supply off and on again.

If an I/O module is disconnected during normal operation from power supply while Processor Module is still powered, the Processor Module will continue its normal operation on all other powered peripherals (I/O modules, communication modules and communication interfaces), but freezes the input image. After recovery of I/O Module power supply it will continue normal operation and inputs and outputs were updated.

Logic Controller Supply: AC500 logic controller power supply is provided through terminals L+ / M.

Process Power Supply: S500 process power supply is provided through terminals UP / ZP.

Logic Controller Supply is galvanic isolated from Process Power Supply.

As system power supply for AC500/S500, the ABB CP power supply series can be used.

Power supply for processor modules

The supply voltage of 24 VDC is connected to a removable 5-pin terminal block. L+/M exist twice. It is therefore possible to feed e.g. external sensors (up to 8 A max. with 1.5 mm² conductor) via these terminals.

### Pin assignment

<table>
<thead>
<tr>
<th>Pin Assignment</th>
<th>Label</th>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Terminal block removed</td>
<td>L+</td>
<td>+24 VDC</td>
<td>Positive pin of the power supply voltage</td>
</tr>
<tr>
<td>Terminal block inserted</td>
<td>L+</td>
<td>+24 VDC</td>
<td>Positive pin of the power supply voltage</td>
</tr>
<tr>
<td></td>
<td>M</td>
<td>0 V</td>
<td>Negative pin of the power supply voltage</td>
</tr>
<tr>
<td></td>
<td>M</td>
<td>0 V</td>
<td>Negative pin of the power supply voltage</td>
</tr>
<tr>
<td></td>
<td>FE</td>
<td></td>
<td>Functional earth</td>
</tr>
</tbody>
</table>

### Terminals for power supply and the COM1 interface

<table>
<thead>
<tr>
<th>Terminal type: Spring terminal</th>
<th>Number of cores per terminal</th>
<th>Conductor type</th>
<th>Cross section</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Solid</td>
<td>0.08 mm² to 1.5 mm²</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>Flexible</td>
<td>0.08 mm² to 1.5 mm²</td>
<td></td>
</tr>
<tr>
<td>1 with wire end ferrule (without plastic sleeve)</td>
<td>Flexible</td>
<td>0.25 mm² to 1.5 mm²</td>
<td></td>
</tr>
<tr>
<td>1 with wire end ferrule (with plastic sleeve)</td>
<td>Flexible</td>
<td>0.25 mm² to 0.5 mm²</td>
<td></td>
</tr>
<tr>
<td>1 (TWIN wire end ferrule)</td>
<td>Flexible</td>
<td>0.5 mm²</td>
<td></td>
</tr>
</tbody>
</table>
Terminals at the terminal unit

Front terminal, conductor connection vertically with respect to the printed circuit board.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>Front terminal</td>
</tr>
<tr>
<td>Degree of protection</td>
<td>IP 20</td>
</tr>
<tr>
<td>Stripped conductor end</td>
<td>9 mm, min. 8 mm</td>
</tr>
<tr>
<td>Fastening torque</td>
<td>0.6 Nm</td>
</tr>
<tr>
<td>Needed tool</td>
<td>Slotted screwdriver</td>
</tr>
<tr>
<td>Dimensions</td>
<td>Blade diameter 3.5 mm</td>
</tr>
</tbody>
</table>

Terminal units with product index < C0 e. g. 1SAP 212 200 R0001 B0

<table>
<thead>
<tr>
<th>Number of cores per terminal</th>
<th>Conductor type</th>
<th>Cross section</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Solid</td>
<td>0.08 mm² to 2.5 mm²</td>
</tr>
<tr>
<td>1</td>
<td>Flexible</td>
<td>0.08 mm² to 2.5 mm²</td>
</tr>
<tr>
<td>1 with wire end ferrule</td>
<td>Flexible</td>
<td>0.25 mm² to 1.5 mm²</td>
</tr>
<tr>
<td>2</td>
<td>Solid</td>
<td>Not intended</td>
</tr>
<tr>
<td>2</td>
<td>Flexible</td>
<td>Not intended</td>
</tr>
<tr>
<td>2 with TWIN wire end ferrule</td>
<td>Flexible</td>
<td>2 x 0.25 mm² or 2 x 0.5 mm² or 2 x 0.75 mm², with square cross-section of the wire-end ferrule also 2 x 1.0 mm²</td>
</tr>
</tbody>
</table>

Terminal Units with Product Index ≥ C0, e. g. 1SAP 212 200 R0001 C0

<table>
<thead>
<tr>
<th>Number of cores per terminal</th>
<th>Conductor type</th>
<th>Cross section</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Solid</td>
<td>0.08 mm² to 2.5 mm²</td>
</tr>
<tr>
<td>1</td>
<td>Flexible</td>
<td>0.08 mm² to 2.5 mm²</td>
</tr>
</tbody>
</table>
### Table: Number of Cores per Terminal, Conductor Type, and Cross Section

<table>
<thead>
<tr>
<th>Number of Cores per Terminal</th>
<th>Conductor Type</th>
<th>Cross Section</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 with wire end ferrule without plastic sleeve</td>
<td>Flexible</td>
<td>0.08 mm² to 2.5 mm²</td>
</tr>
<tr>
<td>1 with wire end ferrule with plastic sleeve</td>
<td>Flexible</td>
<td>0.14 mm² to 1.5 mm²</td>
</tr>
<tr>
<td>2</td>
<td>Solid</td>
<td>0.08 mm² to 1.5 mm²</td>
</tr>
<tr>
<td>2</td>
<td>Flexible</td>
<td>0.08 mm² to 1.5 mm²</td>
</tr>
<tr>
<td>2 with TWIN wire end ferrule (length 10 mm) with plastic sleeve</td>
<td>Flexible</td>
<td>2 x 0.5 mm² to 2 x 1.0 mm²</td>
</tr>
<tr>
<td>2 with separate wire end ferrule without plastic sleeve</td>
<td>Flexible</td>
<td>0.08 mm² to 0.75 mm²</td>
</tr>
</tbody>
</table>

### Terminal Type: Spring Terminal

- Front terminal, conductor connection vertically with respect to the printed circuit board.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>Front terminal</td>
</tr>
<tr>
<td>Degree of protection</td>
<td>IP 20</td>
</tr>
<tr>
<td>Stripped conductor end</td>
<td>9 mm, min. 8 mm</td>
</tr>
<tr>
<td>Needed tool</td>
<td>Slotted screwdriver</td>
</tr>
<tr>
<td>Dimensions</td>
<td>2.5 x 0.4 to 3.5 x 0.5 mm, screwdriver must be at least 15 mm free of insulation at the tip</td>
</tr>
</tbody>
</table>

### Table: Number of Cores per Terminal, Conductor Type, and Cross Section

<table>
<thead>
<tr>
<th>Number of Cores per Terminal</th>
<th>Conductor Type</th>
<th>Cross Section</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Solid</td>
<td>0.08 mm² to 2.5 mm²</td>
</tr>
<tr>
<td>1</td>
<td>Flexible</td>
<td>0.08 mm² to 2.5 mm²</td>
</tr>
<tr>
<td>1 with wire end ferrule</td>
<td>Flexible</td>
<td>0.25 mm² to 1.5 mm²</td>
</tr>
<tr>
<td>2</td>
<td>Solid</td>
<td>Not intended</td>
</tr>
<tr>
<td>2</td>
<td>Flexible</td>
<td>Not intended</td>
</tr>
<tr>
<td>2 with TWIN wire end ferrule (length 10 mm) with plastic sleeve</td>
<td>Flexible</td>
<td>2 x 0.25 mm² or 2 x 0.5 mm² or 2 x 0.75 mm², with square cross-section of the wire-end ferrule also 2 x 1.0 mm²</td>
</tr>
</tbody>
</table>
Connection of wires at the spring terminals

Connection

Fig. 271: Connect the wire to the spring terminal (steps 1 to 3)

Fig. 272: Connect the wire to the spring terminal (steps 4 to 7)

1. Side view (open terminal drawn for illustration)
2. The top view shows the openings for wire and screwdriver
3. Insert screwdriver (2.5 x 0.4 to 3.5 x 0.5 mm) at an angle, screwdriver must be at least 15 mm free of insulation at the tip
4. While erecting the screwdriver, insert it until the stop (requires a little strength)
5. Screwdriver inserted - terminal open
6. Strip the wire for 7 mm (and put on wire end ferrule)
7. Insert wire into the open terminal
8. Done

Disconnection

![Disconnection diagram](image)

Fig. 273: Disconnect wire from the spring terminal (steps 1 to 3)

1. Terminal with wire connected
2. Insert screwdriver (2.5 x 0.4 to 3.5 x 0.5 mm) at an angle, screwdriver must be at least 15 mm free of insulation at the tip
3. While erecting the screwdriver, insert it until the stop (requires a little strength) - terminal is now open
4. Remove wire from the open terminal
5. Done

Fig. 274: Disconnect wire from the spring terminal (steps 4 to 6)
Terminals for CANopen/DeviceNet communication modules

Fig. 275: Combicon, 5-pole, female, removable plug with spring terminals

Fig. 276: Combicon, 5-pole, female, removable plug with spring terminals

<table>
<thead>
<tr>
<th>Terminal type: Spring terminal</th>
<th>Number of cores per terminal</th>
<th>Conductor type</th>
<th>Cross section</th>
<th>Stripped conductor end</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
<td>solid</td>
<td>0.2 mm² to 2.5 mm²</td>
<td>10 mm</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>flexible</td>
<td>0.2 mm² to 2.5 mm²</td>
<td>10 mm</td>
</tr>
<tr>
<td></td>
<td>1 with wire end ferrule (without plastic sleeve)</td>
<td>flexible</td>
<td>0.25 mm² to 2.5 mm²</td>
<td>10 mm</td>
</tr>
<tr>
<td></td>
<td>1 with wire end ferrule (with plastic sleeve)</td>
<td>flexible</td>
<td>0.25 mm² to 2.5 mm²</td>
<td>10 mm</td>
</tr>
</tbody>
</table>

CANopen field bus

Types of bus cables

For CANopen, only bus cables with characteristics as recommended in ISO 11898 are to be used. The requirements for the bus cables depend on the length of the bus segment. Regarding this, the following recommendations are given by ISO 11898:

<table>
<thead>
<tr>
<th>Length of segment [m]</th>
<th>Bus cable (shielded, twisted pair)</th>
<th>Max. transmission rate [kbit/s]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Conductor cross section [mm²]</td>
<td>Line resistance [Ω/km]</td>
</tr>
<tr>
<td>0...40</td>
<td>0.25...0.34 / AWG23, AWG22</td>
<td>70</td>
</tr>
<tr>
<td>40...300</td>
<td>0.34...0.60 / AWG22, AWG20</td>
<td>&lt; 60</td>
</tr>
<tr>
<td>Length of segment [m]</td>
<td>Bus cable (shielded, twisted pair)</td>
<td>Max. transmission rate [kbit/s]</td>
</tr>
<tr>
<td>-----------------------</td>
<td>-----------------------------------</td>
<td>-----------------------------</td>
</tr>
<tr>
<td></td>
<td>Conductor cross section [mm²]</td>
<td>Line resistance [Ω/km]</td>
</tr>
<tr>
<td>300...600</td>
<td>0.50...0.60 / AWG20</td>
<td>&lt; 40</td>
</tr>
<tr>
<td>600...1000</td>
<td>0.75...0.80 / AWG18</td>
<td>&lt; 26</td>
</tr>
</tbody>
</table>

NOTICE!
Risk of telegram and data errors!
The use of wrong cable type and quality could lead to limitations in cable length, causing telegram and data errors.

NOTICE!
Risk of damaging the terminating resistor!
A bus-line short-circuit to the 24 VDC power supply can cause damage by exceeding the power rating of the terminating resistor.

NOTICE!
Risk of telegram and data errors!
Miss- or unterminated data lines can cause reflections on the bus, leading to telegram and data errors. For maximum cable length and transmission rate, the bus must always be terminated on both ends with the characteristic impedance of the cable type.

NOTICE!
Verification of termination (Make sure the power supply on all CAN nodes is turned off)!
To verify the termination, the DC resistance between CAN_H and CAN_L can be measured. The value should be between 50 Ω and 70 Ω.
Check for correct resistor values, short circuits and correct number of terminating resistors, if the measurement is showing deviations.

Installation hint

Ensure that the termination and FE connection will not be removed when removing CAN modules from the bus.

Branches are not allowed in a CAN network. Stubs should be avoided or kept as short as possible (< 0.3 m).
When connecting the cable take care to use one dedicated twisted pair for the CAN signals (CAN_L and CAN_H) and another free wire for CAN_GND. CAN_GND must be connected as reference, to avoid common mode problems causing telegram errors.

Keep the CAN bus wiring away from electrical disturbance and close to earth potential to minimize interference.

**Fig. 277: CAN Bus, connection and wiring**

**Ethernet connection details**

*Ethernet is also used for PROFINET, EtherCAT and Modbus TCP connection.*

**Ethernet interface**

<table>
<thead>
<tr>
<th>Interface</th>
<th>Pin</th>
<th>Signal</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RJ45</td>
<td>1</td>
<td>TxD+</td>
<td>Transmit Data +</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>TxD-</td>
<td>Transmit Data -</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>RxD+</td>
<td>Receive Data +</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>NU</td>
<td>Not used</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>NU</td>
<td>Not used</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>RxD-</td>
<td>Receive Data -</td>
</tr>
<tr>
<td></td>
<td>7</td>
<td>NU</td>
<td>Not used</td>
</tr>
</tbody>
</table>
### Interface Pin Signal Description

<table>
<thead>
<tr>
<th>Interface</th>
<th>Pin</th>
<th>Signal</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>NU</td>
<td>Not used</td>
<td></td>
</tr>
<tr>
<td>Shield</td>
<td></td>
<td>Cable shield</td>
<td>Functional earth</td>
</tr>
</tbody>
</table>

See supported protocols and used Ethernet ports: *Ethernet Protocols and Ports* Chapter 1.5.1.3 “Ethernet protocols and ports for AC500 V3 products” on page 2224.

See communication via Modbus TCP/IP: *Modbus TCP/IP* Chapter 1.5.5.1.9 “Communication with Modbus TCP/IP” on page 3182.

See communication via Modbus RTU: *Modbus RTU* Chapter 1.5.5.1.8 “Communication with Modbus RTU” on page 3166.

### Wiring

#### Cable length restrictions

For the maximum possible cable lengths within an Ethernet network, various factors have to be taken into account. Twisted pair cables (TP cables) are used as transmission medium for 10 Mbit/s Ethernet (10Base-T) as well as for 100 Mbit/s (Fast) Ethernet (100Base-TX). For a transmission rate of 10 Mbit/s, cables of at least category 3 (IEA/TIA 568-A-5 Cat3) or class C (according to European standards) are allowed. For fast Ethernet with a transmission rate of 100 Mbit/s, cables of category 5 (Cat5) or class D or higher have to be used. The maximum length of a segment, which is the maximum distance between two network components, is restricted to 100 m due to the electric properties of the cable.

Furthermore, the length restriction for one collision domain has to be observed. A collision domain is the area within a network which can be affected by a possibly occurring collision (i.e. the area the collision can propagate over). This, however, only applies if the components operate in half-duplex mode since the CSMA/CD access method is only used in this mode. If the components operate in full-duplex mode, no collisions can occur. Reliable operation of the collision detection method is important, which means that it has to be able to detect possible collisions even for the smallest possible frame size of 64 bytes (512 bits). But this is only guaranteed if the first bit of the frame arrives at the most distant subscriber within the collision domain before the last bit has left the transmitting station. Furthermore, the collision must be able to propagate to both directions at the same time. Therefore, the maximum distance between two ends must not be longer than the distance corresponding to the half signal propagation time of 512 bits. Thus, the resulting maximum possible length of the collision domain is 2000 m for a transmission rate of 10 Mbit/s and 200 m for 100 Mbit/s. In addition, the bit delay times caused by the passed network components also have to be considered.

The following table shows the specified properties of the respective cable types per 100 m.

**Table 655: Specified cable properties:**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>10Base-T [10 MHz]</th>
<th>100Base-TX [100 MHz]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Attenuation [dB / 100m]</td>
<td>10.7</td>
<td>23.2</td>
</tr>
<tr>
<td>NEXT [dB / 100m]</td>
<td>23</td>
<td>24</td>
</tr>
<tr>
<td>ACR [dB / 100m]</td>
<td>N/A</td>
<td>4</td>
</tr>
<tr>
<td>Return loss [dB / 100m]</td>
<td>18</td>
<td>10</td>
</tr>
<tr>
<td>Wave impedance [Ohms]</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>Category</td>
<td>3 or higher</td>
<td>5</td>
</tr>
<tr>
<td>Class</td>
<td>C or higher</td>
<td>D or higher</td>
</tr>
</tbody>
</table>
The TP cable has eight wires arranged in four pairs of twisted wires. Different color codes exist for the coding of the wires, the coding according to EIA/TIA 568, version 1, being the one most commonly used. In this code, the individual pairs are coded with blue, orange, green and brown color. One wire of a pair is unicolored and the corresponding second wire is striped, the respective color alternating with white. For shielded cables, a distinction is made between cables that have one single shield around all pairs of wires and cables that have an additional individual shield for each pair of wires. The following table shows the different color coding systems for TP cables:

<table>
<thead>
<tr>
<th>Pairs</th>
<th>EIA/TIA 568 Version 1</th>
<th>EIA/TIA 568 Version 2</th>
<th>DIN 47100</th>
<th>IEC 189.2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pair 1</td>
<td>white/blue</td>
<td>blue</td>
<td>green</td>
<td>red</td>
</tr>
<tr>
<td>Pair 2</td>
<td>white/orange</td>
<td>orange</td>
<td>black</td>
<td>yellow</td>
</tr>
<tr>
<td>Pair 3</td>
<td>white/green</td>
<td>green</td>
<td>blue</td>
<td>orange</td>
</tr>
<tr>
<td>Pair 4</td>
<td>white/brown</td>
<td>brown</td>
<td>brown</td>
<td>slate</td>
</tr>
</tbody>
</table>

Two general variants are distinguished for the pin assignment of the normally used RJ45 connectors: EIA/TIA 568 version A and version B. The wiring according to EIA/TIA 568 version B is the one most commonly used.

Cable types

**Crossover cable**

*Particular use*

Crossover cables are needed only for a direct Ethernet connection without crossover functionality. In particular for AC500 modules in product life cycle phase “Classic”.

Crossover cables are for a direct Ethernet connection of two terminal devices as the simplest variant of a network. From transmission lines of the first station to the reception lines of the second station.
Fig. 279: Wiring of a crossover cable

Straight-through cable

For networks with more than two subscribers, hubs or switches have to be used additionally for distribution. These active devices already have the crossover functionality implemented which allows a direct connection of the terminal devices using straight-through cables.

Fig. 280: Wiring of a straight-through cable

CAUTION!
Risk of communication faults!
When using inappropriate cables, malfunctions in communication may occur.
Only use network cables of the categories 5 (Cat 5, Cat 5e, Cat 6 or Cat 7) or higher within PROFINET networks.

Modbus RTU connection details

The Modbus RTU protocol is implemented in the AC500 Processor Modules.

Modbus is a master-slave (client-server) protocol. The client sends a request to the server(s) and receives the response(s).

Available serial interfaces can work as Modbus interfaces simultaneously.

The Modbus client operating mode of an interface is set with the Function Block COM_MOD_MAST.

Technical data

The Modbus operating mode and the interface parameters are set in the PLC configuration Chapter 1.5.6.2.10.1 “Configuring Modbus RTU on serial interface” on page 3325.

Table 657: Description of the Modbus protocol

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Supported standard</td>
<td>See Serial interface Chapter 1.5.6.2.10.1 “Configuring Modbus RTU on serial interface” on page 3325</td>
</tr>
</tbody>
</table>
| Number of connection points | 1 client  
                          | Max. 1 server with RS-232 interface  
<pre><code>                      | Max. 31 servers with RS-485   |
</code></pre>
<p>| Protocol               | Modbus                                                               |</p>
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operating mode</td>
<td>Client/server</td>
</tr>
<tr>
<td>Address</td>
<td>Server only</td>
</tr>
<tr>
<td>Data transmission control</td>
<td>CRC16</td>
</tr>
<tr>
<td>Data transmission speed</td>
<td>Up to 187.500 baud (see Serial interface Chapter 1.5.6.2.10.1 “Configuring Modbus RTU on serial interface” on page 3325)</td>
</tr>
<tr>
<td>Encoding</td>
<td>1 start bit, 8 data bits, 1 or 2 stop bits, 1 parity bit (see Serial interface Chapter 1.5.6.2.10.1 “Configuring Modbus RTU on serial interface” on page 3325)</td>
</tr>
<tr>
<td>Max. cable length for RS-485 on COM1 for AC500 CPU</td>
<td>1.200 m at 19.200 baud</td>
</tr>
</tbody>
</table>

**Bus topology**  
Point-to-point with RS-232 or bus topology with RS-485. Modbus is a master-slave protocol. For further information on Modbus see chapter Communication with Modbus RTU Chapter 1.5.5.1.8 “Communication with Modbus RTU” on page 3166.

**1.5.4.5.5 Handling of accessories**  
This section only describes accessories that are frequently used for system assembly, connection and construction. A description of all additional accessories that can be used to supplement AC500 system can be found in the Hardware PLC device description.

**TA521 - Lithium battery**  
- Manganese dioxide lithium battery, 3 V, 560 mAh
- Non-rechargeable

**Purpose**  
The TA521 lithium battery is the only applicable battery for the AC500 processor modules and PM56xx Chapter 1.5.3.3.1.1 “PM56xx-2ETH for AC500 V3 products” on page 2274. It cannot be recharged.

The processor modules are supplied without lithium battery. It must be ordered separately. The TA521 lithium battery is used for data (SRAM) and RTC buffering while the processor module is not powered.

See system technology - AC500 battery Chapter 1.5.5.1.3.2 “AC500 battery” on page 3116.
The CPU monitors the discharge degree of the battery. A warning is issued before the battery condition becomes critical (about 2 weeks before). Once the warning message appears, the battery should be replaced as soon as possible.

**Handling instructions**

- Do not short-circuit or re-charge the battery! It can cause excessive heating and explosion.
- Do not disassemble the battery!
- Do not heat up the battery and not put into fire! Risk of explosion.
- Store the battery in a dry place.
- Replace the battery with supply voltage ON in order not to risk data being lost.
- Recycle exhausted batteries meeting the environmental standards.

**Battery lifetime**

The battery lifetime is the time, the battery can store data while the processor module is not powered. As long as the processor module is powered, the battery will only be discharged by its own leakage current.

To avoid a short battery discharge, the battery should always be inserted or replaced while the processor module is under power, then the battery is correctly recognized and will not shortly discharged.

**Technical data**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nominal voltage</td>
<td>3 V</td>
</tr>
<tr>
<td>Nominal capacity</td>
<td>560 mAh</td>
</tr>
<tr>
<td>Temperature range (index below C0)</td>
<td>Operating: 0 °C...+60 °C</td>
</tr>
<tr>
<td></td>
<td>Storage: -20 °C...+60 °C</td>
</tr>
<tr>
<td></td>
<td>Transport: -20 °C...+60 °C</td>
</tr>
<tr>
<td>Temperature range (index C0 and above)</td>
<td>Operating: -40 °C...+70 °C</td>
</tr>
<tr>
<td></td>
<td>Storage: -40 °C...+85 °C</td>
</tr>
<tr>
<td></td>
<td>Transport: -40 °C...+85 °C</td>
</tr>
<tr>
<td>Battery lifetime</td>
<td>Typ. 3 years at 25 °C</td>
</tr>
<tr>
<td>Self-discharge</td>
<td>2 % per year at 25 °C</td>
</tr>
<tr>
<td></td>
<td>5 % per year at 40 °C</td>
</tr>
<tr>
<td></td>
<td>20 % per year at 60 °C</td>
</tr>
<tr>
<td>Protection against reverse polarity</td>
<td>Yes, by mechanical coding of the plug.</td>
</tr>
<tr>
<td>Insulation</td>
<td>The battery is completely insulated.</td>
</tr>
<tr>
<td>Connection</td>
<td>Red = plus pole = above at plug, black = minus pole,</td>
</tr>
<tr>
<td>Weight</td>
<td>7 g</td>
</tr>
<tr>
<td>Dimensions</td>
<td>Diameter of the button cell: 24.5 mm</td>
</tr>
<tr>
<td></td>
<td>Thickness of the button cell: 5 mm</td>
</tr>
</tbody>
</table>
### Ordering data

<table>
<thead>
<tr>
<th>Part no.</th>
<th>Description</th>
<th>Product life cycle phase *)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1SAP 180 300 R0001</td>
<td>TA521, lithium battery</td>
<td>Active</td>
</tr>
</tbody>
</table>

*) For planning and commissioning of new installations use modules in Active status only.

---

### Insertion

**Insertion**

1. Open the battery compartment with the small locking mechanism, press it down and slip down the door. The door is attached to the front face of the processor module and cannot be removed.
2. Remove the TA521 battery from its package and hold it by the small cable.
3. Insert the battery connector into the small connector port of the compartment. The connector is keyed to find the correct polarity (red = plus-pole = above).
4. Insert first the cable and then the battery into the compartment, push it until it reaches the bottom of the compartment.
5. Arrange the cable in order not to inhibit the door to close.
6. Pull-up the door and press until the locking mechanism snaps.

---

*To ensure proper operation and to prevent data loss, the battery insertion or replacement must be always done with the system under power. Without battery and power supply there is no data buffering possible.*

---

Fig. 281: Insertion of the Lithium battery

---

**Ordering data**

**Insertion**

**PLC Automation with V3 CPUs**

**PLC integration > System assembly, construction and connection**

2020/12/10
In order to prevent data losses or problems, the battery should be replaced after 3 years of utilisation or at least as soon as possible after receiving the "low battery warning" indication.

Do not use a battery older than 3 years for replacement, do not keep batteries too long in stock.

Replacement
Replacement of the battery

To ensure proper operation and to prevent data loss, the battery insertion or replacement must be always done with the system under power. Without battery and power supply there is no data buffering possible.

1. Open the battery compartment with the small locking mechanism, press it down and slip down the door. The door is attached to the front face of the processor module and cannot be removed.

2. Remove the old TA521 battery from the battery compartment by pulling it by the small cable. Remove then the small connector from the socket, do this best by lifting it out with a screwdriver.

3. Follow the previous instructions to insert a new battery.

CAUTION!
Risk of explosion!
Do not open, re-charge or disassemble a lithium battery. Attempts to charge lithium batteries lead to overheating and possible explosions.

Prevent them from heat and fire and store them in a dry place.

Never short-circuit or operate lithium batteries with the polarities reversed. The batteries are likely to overheat and explode. Avoid chance short circuiting and therefore do not store batteries in metal containers and do not place them on metallic surfaces. Escaping lithium is a health hazard.
In order to prevent data losses or problems, the battery should be replaced after 3 years of utilisation or at least as soon as possible after receiving the "low battery warning" indication.

Do not use a battery older than 3 years for replacement, do not keep batteries too long in stock.

MC502 - SD memory card

- Secure digital card
- Solid state flash memory storage

The SD memory card is used to back-up user data and store user programs or project source codes as well as to update the internal CPU firmware. The processor modules can be operated with and without SD memory card.

AC500/AC500-eCo processor modules are supplied without SD memory card. It therefore must be ordered separately.

The MC memory card can be read on a PC with a standard memory card reader. AC500 processor modules are equipped with an MC memory card reader.

For AC500-eCo processor modules the device must be equipped with a MC503 SD memory card adaptor.

The SD memory card has a write protect switch. In the position "LOCK", the card can only be read.

The use of memory cards other than the MC502 SD memory card is prohibited. ABB is not responsible nor liable for consequences resulting from use of unapproved memory cards.

NOTICE!

Removal of the SD memory card

Do not remove the SD memory card during access. Remove only when the RUN LED does not blink. Otherwise the SD memory card and/or files on it might get corrupted and/or normal PLC operation might be disturbed.

Unpack the SD memory card and insert it into the opening of the front face of the processor module until locked:
Fig. 282: Insertion: PM57x, PM58x, PM59x and PM56xx

To remove the SD memory card, push on the card until it moves forward. By this, the card is unlocked and can be removed.

### Technical data

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Memory capacity</td>
<td>Up to 2 GB, for exactly size see type plate</td>
</tr>
<tr>
<td>Temperature range</td>
<td>-20 °C...+85 °C</td>
</tr>
<tr>
<td>No. of writing cycles</td>
<td>&gt; 100 000</td>
</tr>
<tr>
<td>No. of reading cycles</td>
<td>No limitation</td>
</tr>
<tr>
<td>Data safety</td>
<td>&gt; 10 years</td>
</tr>
<tr>
<td>Write Protect Switch</td>
<td>Yes, at the edge of the SD memory card</td>
</tr>
<tr>
<td>Weight</td>
<td>2 g</td>
</tr>
<tr>
<td>Dimensions</td>
<td>24 mm x 32 mm x 2.1 mm</td>
</tr>
</tbody>
</table>

It is not possible to use 100 % of a device's memory space. About 10 % of the total available space must remain unused at any time to maintain normal device operation.

Further information on using the SD memory card in AC500 PLCs is provided in the chapter *Storage Devices* Chapter 1.5.7.2 “SD memory card in AC500” on page 3513.

### Ordering data

<table>
<thead>
<tr>
<th>Part no.</th>
<th>Description</th>
<th>Product life cycle phase *)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1SAP 180 100 R0001</td>
<td>MC502, SD memory card</td>
<td>Active</td>
</tr>
</tbody>
</table>

*) For planning and commissioning of new installations use modules in Active status only.
TA524 - Dummy communication module

1 Type
2 Label

Purpose
TA524 is used to cover an unused communication module slot of a terminal base and *TB56xx* \(\textcircled{\text{C}}\) Chapter 1.5.3.2.1 “TB56xx for AC500 V3 products” on page 2265. It protects the terminal base from dust and inadvertent touch.

Handling instructions
TA524 is mounted in the same way as a common communication module \(\textcircled{\text{C}}\) Chapter 1.5.4.5.3.5 “Mounting and demounting the communication modules” on page 3075.

Technical data

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Weight</td>
<td>50 g</td>
</tr>
<tr>
<td>Dimensions</td>
<td>135 mm x 28 mm x 62 mm</td>
</tr>
</tbody>
</table>

Ordering data

<table>
<thead>
<tr>
<th>Part no.</th>
<th>Description</th>
<th>Product life cycle phase *)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1SAP 180 600 R0001</td>
<td>TA524, dummy communication module</td>
<td>Active</td>
</tr>
</tbody>
</table>
*) For planning and commissioning of new installations use modules in Active status only.

CP-E - Economic range

The power supplies feature series and parallel connection as well as a true redundant setup via a redundancy module.

- Wide-range input voltage
- Mounting on DIN rail
- High efficiency of up to 90 %
- Low power dissipation and low heating
- Wide ambient temperature range from -40 °C...+70 °C
- No-load-proof, overload-proof, continuous short-circuit-proof
- Power factor correction (depending on the type)
- Approved in accordance with all relevant international standards
<table>
<thead>
<tr>
<th>Order No.</th>
<th>Type</th>
<th>Input</th>
<th>Output</th>
<th>Overload capacity</th>
<th>Module width [mm]</th>
</tr>
</thead>
<tbody>
<tr>
<td>1SVR427030R0000</td>
<td>CP-E 24/0.75</td>
<td>100-240 VAC or 120-370 VDC</td>
<td>24 VDC, 0.75 A</td>
<td>-</td>
<td>22.5</td>
</tr>
<tr>
<td>1SVR427031R0000</td>
<td>CP-E 24/1.25</td>
<td>100-240 VAC or 90-375 VDC</td>
<td>24 VDC, 1.25 A</td>
<td>-</td>
<td>40.5</td>
</tr>
<tr>
<td>1SVR427032R0000</td>
<td>CP-E 24/2.5</td>
<td>100-240 VAC or 90-375 VDC</td>
<td>24 VDC, 2.5 A</td>
<td>-</td>
<td>40.5</td>
</tr>
<tr>
<td>1SVR427034R0000</td>
<td>CP-E 24/5.0</td>
<td>115/230 VAC auto select or 210-370 VDC</td>
<td>24 VDC, 5 A</td>
<td>-</td>
<td>63.2</td>
</tr>
<tr>
<td>1SVR427035R0000</td>
<td>CP-E 24/10.0</td>
<td>115/230 VAC auto select or 210-370 VDC</td>
<td>24 VDC, 10 A</td>
<td>-</td>
<td>83</td>
</tr>
<tr>
<td>1SVR427036R0000</td>
<td>CP-E 24/20.0</td>
<td>115-230 VAC or 120-370 VDC</td>
<td>24 VDC, 20 A</td>
<td>-</td>
<td>175</td>
</tr>
</tbody>
</table>
CP-C.1 - High performance range

The power supplies feature series and parallel connection as well as a true redundant setup via a redundancy module.

The CP-C.1 power supplies are ABB’s high performance and most advanced range. With excellent efficiency, high reliability and innovative functionality it is prepared for the most demanding industrial applications. These power supplies have a 50 % integrated power reserve and operate at an efficiency of up to 94 %. They are equipped with overheat protection and active power factor correction. Combined with a broad AC and DC input range and extensive worldwide approvals the CP-C.1 power supplies are the preferred choice for professional DC applications.

- Typical efficiency of up to 94 %
- Power reserve design delivers up to 150 % of the nominal output current
- Signaling outputs for DC OK and power reserve mode
- High power density leads to very compact and small devices
- No-load-proof, overload-proof, continuous short-circuit-proof
- Active power factor correction (PFC)
Table 659: Ordering data

<table>
<thead>
<tr>
<th>Order No.</th>
<th>Type</th>
<th>Input</th>
<th>Output</th>
<th>Overload capacity</th>
<th>Module width [mm]</th>
</tr>
</thead>
<tbody>
<tr>
<td>1SVR360563R1001</td>
<td>CP-C.1</td>
<td>24/5.0</td>
<td>110-240 VAC or 90-300 VDC</td>
<td>24 VDC, 5 A</td>
<td>40</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>+50 %</td>
<td></td>
</tr>
<tr>
<td>1SVR360663R1001</td>
<td>CP-C.1</td>
<td>24/10.0</td>
<td>110-240 VAC or 90-300 VDC</td>
<td>24 VDC, 10 A</td>
<td>60</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>+50 %</td>
<td></td>
</tr>
<tr>
<td>1SVR360763R1001</td>
<td>CP-C.1</td>
<td>24/20.0</td>
<td>110-240 VAC or 90-300 VDC</td>
<td>24 VDC, 20 A</td>
<td>82</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>+30 %</td>
<td></td>
</tr>
</tbody>
</table>

TA526 - Wall mounting accessory

Purpose
If a terminal base TB5xx or a terminal unit TU5xx should be mounted with screws, the wall mounting accessories TA526 must be inserted at the rear side first. This plastic parts prevent bending of terminal bases and terminal units while screwing up.

Handling instructions
Handling of the wall mounting accessory is described in detail in the section Mounting and Disassembling the Terminal Unit "Mounting with screws" on page 3071 and Mounting/Disassembling Terminal Bases and Function Module Terminal Bases "Mounting with screws" on page 3069.

Technical data

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Weight</td>
<td>5 g</td>
</tr>
<tr>
<td>Dimensions</td>
<td>67 mm x 35 mm x 5,5 mm</td>
</tr>
</tbody>
</table>

Ordering data

<table>
<thead>
<tr>
<th>Part no.</th>
<th>Description</th>
<th>Product life cycle phase *)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1SAP 180 800 R0001</td>
<td>TA526, wall mounting access- sory</td>
<td>Active</td>
</tr>
</tbody>
</table>

*) For planning and commissioning of new installations use modules in Active status only.
Assembly, construction and connection of devices of the variant AC500-XC is identical to AC500 (standard) § Chapter 1.5.4.5 “AC500 (Standard)” on page 3058. The following description provides information on general technical data of AC500-XC system.

Environmental conditions

**Table 660: Process and supply voltages**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>24 VDC</td>
<td></td>
</tr>
<tr>
<td>Voltage</td>
<td>24 V (-15 %, +20 %)</td>
</tr>
<tr>
<td>Protection against reverse polarity</td>
<td>Yes</td>
</tr>
<tr>
<td>120 VAC...240 VAC wide range supply</td>
<td></td>
</tr>
<tr>
<td>Voltage</td>
<td>120...240 V (-15 %, +10 %)</td>
</tr>
<tr>
<td>Frequency</td>
<td>50/60 Hz (-6 %, +4 %)</td>
</tr>
</tbody>
</table>

Allowed interruptions of power supply

| DC supply | Interruption < 10 ms, time between 2 interruptions > 1 s, PS2 |

**NOTICE!**

Exceeding the maximum power supply voltage for process or supply voltages could lead to unrecoverable damage of the system. The system could be destroyed.

**NOTICE!**

For the supply of the modules, power supply units according to PELV or SELV specifications must be used.

The creepage distances and clearances meet the requirements of the over-voltage category II, pollution degree 2.
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
</table>
| **Temperature**        | -40 °C...+70 °C  
-40 °C...-30 °C: Proper start-up of system; technical data not guaranteed  
-40 °C...0 °C: Due to the LCD technology, the display might respond very slowly.  
-40 °C...+40 °C: Vertical mounting of modules possible, output load limited to 50 % per group  
+60 °C...+70 °C with the following deratings:  
  - System is limited to max. 2 communication modules per terminal base  
  - Applications certified for cULus up to +60 °C  
  - Digital inputs: maximum number of simultaneously switched on input channels limited to 75 % per group (e.g. 8 channels => 6 channels)  
  - Digital outputs: output current maximum value (all channels together) limited to 75 % per group (e.g. 8 A => 6 A)  
  - Analog outputs only if configured as voltage output: maximum total output current per group is limited to 75 % (e.g. 40 mA => 30 mA)  
  - Analog outputs only if configured as current output: maximum number of simultaneously used output channels limited to 75 % per group (e.g. 4 channels => 3 channels)  |
| **Storage / Transport**| -40 °C...+85 °C  
-40 °C...+85 °C  
Operating / Storage: 100 % r. H. with condensation  
-1000 m...4000 m (1080 hPa...620 hPa)  
> 2000 m (< 795 hPa):  
  - max. operating temperature must be reduced by 10 K (e.g. 70 °C to 60°C)  
  - I/O module relay contacts must be operated with 24 V nominal only  |
| **Humidity**           | Operating / Storage: 100 % r. H. with condensation  |
| **Air pressure**       | Operating:  
-1000 m...4000 m (1080 hPa...620 hPa)  
> 2000 m (< 795 hPa):  
  - max. operating temperature must be reduced by 10 K (e.g. 70 °C to 60°C)  
  - I/O module relay contacts must be operated with 24 V nominal only  |
| **Immunity to corrosive gases** | Operating: Yes, according to:  
ISA S71.04.1985 Harsh group A, G3/GX  
IEC 60721-3-3  3C2 / 3C3  |
| **Immunity to salt mist** | Operating: Yes, horizontal mounting only, according to IEC 60068-2-52 severity level: 1 |
NOTICE!
Risk of corrosion!
Unused connectors and slots may corrode if XC devices are used in salt-mist environments.

Protect unused connectors and slots with TA535 protective caps for XC devices
TA535 § Chapter 1.5.3.8.2.4 “TA535 - Protective caps for XC devices” on page 3039.

Table 661: Electromagnetic compatibility

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Device suitable for:</td>
<td></td>
</tr>
<tr>
<td>Industrial applications</td>
<td>Yes</td>
</tr>
<tr>
<td>Domestic applications</td>
<td>No</td>
</tr>
<tr>
<td>Radiated emission (radio disturbances)</td>
<td>Yes, according to: CISPR 16-2-3</td>
</tr>
<tr>
<td>Conducted emission (radio disturbances)</td>
<td>Yes, according to: CISPR 16-2-1, CISPR 16-1-2</td>
</tr>
<tr>
<td>Electrostatic discharge (ESD)</td>
<td>Yes, according to: IEC 61000-4-2, zone B, criterion B</td>
</tr>
<tr>
<td>Fast transient interference voltages (burst)</td>
<td>Yes, according to: IEC 61000-4-4, zone B, criterion B</td>
</tr>
<tr>
<td>High energy transient interference voltages (surge)</td>
<td>Yes, according to: IEC 61000-4-5, zone B, criterion B</td>
</tr>
<tr>
<td>Influence of radiated disturbances</td>
<td>Yes, according to: IEC 61000-4-3, zone B, criterion A</td>
</tr>
<tr>
<td>Influence of line-conducted interferences</td>
<td>Yes, according to: IEC 61000-4-6, zone B, criterion A</td>
</tr>
<tr>
<td>Influence of power frequency magnetic fields</td>
<td>Yes, according to: IEC 61000-4-8, zone B, criterion A</td>
</tr>
</tbody>
</table>

In order to prevent malfunctions, it is recommended, that the operating personnel discharge themselves prior to touching communication connectors or perform other suitable measures to reduce effects of electrostatic discharges.
NOTICE!

Risk of malfunctions!

Unused slots for communication modules are not protected against accidental physical contact.

- Unused slots for communication modules must be covered with dummy communication modules ([TA524](#) Chapter 1.5.3.8.1.3 “TA524 - Dummy communication module” on page 3034) to achieve IP20 rating.
- I/O bus connectors must not be touched during operation.

### Mechanical data

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Wiring method</td>
<td>Spring terminals</td>
</tr>
<tr>
<td>Degree of protection</td>
<td>IP 20</td>
</tr>
<tr>
<td>Vibration resistance</td>
<td>Yes, according to:</td>
</tr>
<tr>
<td></td>
<td>IEC 61131-2</td>
</tr>
<tr>
<td></td>
<td>IEC 60068-2-6</td>
</tr>
<tr>
<td></td>
<td>IEC 60068-2-64</td>
</tr>
<tr>
<td>Shock resistance</td>
<td>Yes, according to:</td>
</tr>
<tr>
<td></td>
<td>IEC 60068-2-27</td>
</tr>
<tr>
<td>Assembly position</td>
<td>Horizontal</td>
</tr>
<tr>
<td></td>
<td>Vertical (no application in salt mist environment)</td>
</tr>
<tr>
<td>Assembly on DIN rail</td>
<td>According to IEC 60715</td>
</tr>
<tr>
<td></td>
<td>35 mm, depth 7.5 mm or 15 mm</td>
</tr>
<tr>
<td>Assembly with screws</td>
<td></td>
</tr>
<tr>
<td>Screw diameter</td>
<td>4 mm</td>
</tr>
<tr>
<td>Fastening torque</td>
<td>1.2 Nm</td>
</tr>
</tbody>
</table>

### Environmental tests

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Storage</td>
<td>IEC 60068-2-1 Test Ab: cold withstand test -40 °C / 16 h</td>
</tr>
<tr>
<td></td>
<td>IEC 60068-2-2 Test Bb: dry heat withstand test +85 °C / 16 h</td>
</tr>
<tr>
<td>Humidity</td>
<td>IEC 60068-2-30 Test Db: Cyclic (12 h / 12 h) damp-heat test 55 °C, 93 % r. H. / 25 °C, 95 % r. H., 6 cycles</td>
</tr>
<tr>
<td></td>
<td>IEC 60068-2-78, stationary humidity test: 40 °C, 93 % r. H., 240 h</td>
</tr>
<tr>
<td>Insulation Test</td>
<td>IEC 61131-2</td>
</tr>
<tr>
<td>Parameter</td>
<td>Value</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>----------------------------------------------------------------------</td>
</tr>
<tr>
<td>Vibration resistance</td>
<td>IEC 61131-2 / IEC 60068-26: 5 Hz...500 Hz, 2 g (with SD memory card inserted)</td>
</tr>
<tr>
<td></td>
<td>IEC 60068-2-64: 5 Hz...500 Hz, 4 g rms</td>
</tr>
<tr>
<td>Shock resistance</td>
<td>IEC 60068-2-27: all 3 axes 15 g, 11 ms, half-sinusoidal</td>
</tr>
</tbody>
</table>

Table 662: EMC immunity

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Electrostatic discharge (ESD)</td>
<td>Electrostatic voltage in case of air discharge: 8 kV</td>
</tr>
<tr>
<td></td>
<td>Electrostatic voltage in case of contact discharge: 6 kV</td>
</tr>
<tr>
<td>Fast transient interference voltages (burst)</td>
<td>Supply voltage units (DC): 4 kV</td>
</tr>
<tr>
<td></td>
<td>Digital inputs/outputs (24 VDC): 2 kV</td>
</tr>
<tr>
<td></td>
<td>Analog inputs/outputs: 2 kV</td>
</tr>
<tr>
<td></td>
<td>Communication lines shielded: 2 kV</td>
</tr>
<tr>
<td></td>
<td>I/O supply (DC-out): 2 kV</td>
</tr>
<tr>
<td>High energy transient interference voltages (surge)</td>
<td>Supply voltage units (DC): 1 kV CM *) / 0.5 kV DM *)</td>
</tr>
<tr>
<td></td>
<td>Digital inputs/outputs (24 VDC): 1 kV CM *) / 0.5 kV DM *)</td>
</tr>
<tr>
<td></td>
<td>Digital inputs/outputs (AC): 4 kV</td>
</tr>
<tr>
<td></td>
<td>Analog inputs/outputs: 1 kV CM *) / 0.5 kV DM *)</td>
</tr>
<tr>
<td></td>
<td>Communication lines shielded: 1 kV CM *)</td>
</tr>
<tr>
<td></td>
<td>I/O supply (DC-out): 0.5 kV CM *) / 0.5 kV DM *)</td>
</tr>
<tr>
<td>Influence of radiated disturbances</td>
<td>Test field strength: 10 V/m</td>
</tr>
<tr>
<td>Influence of line-conducted interferences</td>
<td>Test voltage: 10 V</td>
</tr>
<tr>
<td>Power frequency magnetic fields</td>
<td>30 A/m 50 Hz</td>
</tr>
<tr>
<td></td>
<td>30 A/m 60 Hz</td>
</tr>
</tbody>
</table>

*) CM = Common Mode, * DM = Differential Mode

1.5.4.7 AC500-S

The AC500-S safety user manual must be read and understood before using safety configuration and programming tools of Automation Builder / PS501 Control Builder Plus. Only qualified personnel shall be allowed to work with AC500-S safety PLCs.

In order to have always the latest version and due to a different lifecycle compared to Automation Builder help, the AC500-S safety user manual is only available on our website.
1.5.5 System technology for AC500 V3 products

This chapter provides advanced information on the system technology of AC500 control systems from a general perspective. It provides information to link the details from the hardware descriptions (provided in the device specifications section) with detailed information on configuring/programming a corresponding library (provided in the individual library sections).

Configuration of a specific device with Automation Builder is described in the PLC configuration section.
1.5.5.1 System technology of CPU and overall system
1.5.5.1.1 Handling of remanent variables for AC500 V3 products

All operands supported by CODESYS are described in [CODESYS documentation] Chapter 1.3.1.8 “Configuring I/O links” on page 155. For the memory sizes of the different CPUs, see [Memory Sizes] Chapter 1.5.7.1.3 “Memory sizes” on page 3510.

This part of the documentation describes the declaration of remanent variables for AC500 V3 products.

Different handling of remanent variables in AC500 FW ≥V3.0.2

- No more %R memory area (use instead %M with {no_init}) [Chapter 1.5.5.1.1.4 "Initialization of %M variables" on page 3108]
- Creating of addresses for "VAR RETAIN PERSISTENT" variables automatically by IEC-Compiler

Adding a global list of persistent/retain variables

A global list of persistent variables will be added with the standard definition for persistent variables "VAR RETAIN PERSISTENT" (see Remanent variables).

First steps:
1. Expand the object path of your PLC
2. After right click on App select Add object in the context menu.
   The window Add object below: Application appears.

3. Select Persistent Variables and click “Add object”.

The object name can be chosen freely. In the application it will be reused to reference the persistent variables.
Declaring a new variable in global list

Declare a new variable in the window "GlobPersist".

Afterwards the variable can be selected in the program.

In this way the persistent variable can be accessed directly.

Do not use the same persistent variable in different IEC tasks, to avoid problems with consistency.

Declaring a new persistent/retain variable in local POU

It is also possible to declare a persistent/retain variable in a local POU and not in the global list of persistent variables.

It is not recommended to declare a large number of persistent variables locally, due to the potentially effect to performance.
The auto-declare mechanism declares **always** a persistent variable locally and not in the global list. If the program will be executed, the following warning appears in the message window:

⚠️ C0244: No VAR_PERSISTENT list is part of the application to enter instance path for variable PLC_PRG new_var1

The locally declared persistent variable has to be added to the global list.

---

**NOTICE!**

For the initialization of a Retain/Persistent variable the value of the global list is used **NOT** the value of the local declaration.

For further information see "RetainPersistentExample.project".

---

1. Right-click in window "PersistentVars".
2. Select option "Add all instance paths".
   - Afterwards the persistent variables are added.
The application can be downloaded to the PLC

**It is NOT recommended to declare a new persistent variable in the application due to performance problems.**

For example PM5650-2ETH:

1000 DWORD ≈ 600 µs additional cycle time of task.

---

**Initialization of %M variables**

After download or restart, all %M variables will be initialized to 0. This can be prevented by setting the "no_init" attribute.

In doing so the %M variables behave similar to the "VAR RETAIN PERSISTENT" variables.

In the example above variable "ProzMivar" has the attribute "no_init". This variable will not be initialized and keeps its last value.

The attribute "no_init" is always and only valid for the next following variable (see Attributes Chapter 1.3.1.24.6.2.26 "Attribute 'noinit'" on page 623).

The following two variables "Proz MivarField" and "Proz Mivar1" will be further on initialized to 0.

---

**Behavior of retain variables**

The declaration of the retain variables strictly follows the 3S standard (see Remanent variables).

For retain variables it does not matter if they are declared locally in a program or in the global variable list.

---

**PLC shell command for import and export of retain/persistent variables**

The syntax of the command is: **sram <direction><area><path>**

<table>
<thead>
<tr>
<th>Supported options:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Direction: i=import, e=export</td>
</tr>
<tr>
<td>Area: rp=Retain/Persistent, m=%M area</td>
</tr>
<tr>
<td>Path: Any pathname</td>
</tr>
</tbody>
</table>
The file will be stored in the user partition of the PLC. This data can be imported or exported via the FTP-Server or the Files dialog in Automation Builder.

If no path is indicated, the files are saved under "PlcLogic/<ApplicationName>/<ApplicationName>.ret or .prozm".

If a path is indicated, the files are saved under or accessed via "<path>/<ApplicationName>.ret or .prozm".

A non-existing path is created with the exception of the SD card. The path for the SD card must be an existing path. On the SD card a non-existing path leads to an error message.

<table>
<thead>
<tr>
<th>Data area</th>
<th>File extension</th>
<th>Path</th>
</tr>
</thead>
</table>
| Retain/Persistent    | .ret           | PlcLogic/<ApplicationName>/<ApplicationName>.ret  
                      |                | <path>/<ApplicationName>.ret               |
| %M (memory area)     | .prozm         | PlcLogic/<ApplicationName>/<ApplicationName>.prozm  
                      |                | <path>/<ApplicationName>.prozm             |

**Examples:**

<table>
<thead>
<tr>
<th>Application</th>
<th>Command</th>
<th>File</th>
</tr>
</thead>
<tbody>
<tr>
<td>myApp</td>
<td>sram e rp</td>
<td>PlcLogic/myApp/myApp.ret</td>
</tr>
<tr>
<td></td>
<td>sram e ep data</td>
<td>data/myApp.ret</td>
</tr>
<tr>
<td>Application</td>
<td>sram i m</td>
<td>PlcLogic/Application/Application.prozm</td>
</tr>
<tr>
<td></td>
<td>sram i m data</td>
<td>data/Application.prozm</td>
</tr>
</tbody>
</table>

If the path "data" does not exist, the path is created. The path for the SD card must be an existing path. The path "sdcard/data" leads to an error message if the path "data" does not exist on the SD card.

Only if the application uses Retain or Retain/Persistent variables the command generates an output file.

**Attention!**

It is recommended to execute the PLC shell command only while PLC is in state STOP, or it is ensured that there is no write access to the %M or the Retain/Persistent area.

**Import and export of retain/persistent variables by library functions**

It is also possible to import or export the Retain/Persistent variables and the %M markers via system function calls from the PLC Application. The required system functions are implemented in the IEC library **ABB_IntUtils_AC500.library**.

It provides the following Functions or Function Blocks:

- **SRAM_IMPORT**
- **SRAM_EXPORT**
- **SRAM_CLEARED**
SRAM_IMPORT

The Function Block SRAM_IMPORT is used to import the %M markers and the Retain/Persistent variables from the specified files in the userdisk.

- Import only those %M markers and/or Retain/Persistent variables that are compatible to the application running in the PLC.

- It is recommended to import only when the %M and/or the Retain/Persistent area is not accessed by the application.
  Otherwise inconsistencies are possible.

For a complete description of the Function Block see ABB_IntUtils_AC500.library.

---

SRAM_EXPORT

The Function Block SRAM_EXPORT is used to export the %M markers and the Retain/Persistent variables to the specified files in the userdisk.

- Export only those %M markers and/or Retain/Persistent variables that are compatible to the application running in the PLC.

- It is recommended to export only when the %M and/or the Retain/Persistent area is not accessed by the application.
  Otherwise inconsistencies are possible.

For a complete description of the Function Block see ABB_IntUtils_AC500.library.

---

SRAM_CLEARED

The Function SRAM_CLEARED is used to check if the SRAM was deleted.

For a complete description of the Function see ABB_IntUtils_AC500.library.

---

1.5.5.1.2 System processing

System start-up / Program processing

- AC500-eCo Processor Modules do not have an integrated display and keyboard. All functions related to keyboard and display are not applied for those devices.
Definitions: PLC system start-up

Cold start
- A cold start is performed by switching power OFF/ON if no battery is connected.
- All RAM memory modules are checked and erased (see Command 'Reset cold' Chapter 1.3.1.25.2.6.9 “Command ‘Reset Cold’” on page 931).
- If no user program is stored in the Flash EPROM, the default values (as set on delivery) are applied to the interfaces.
- If there is a user program stored in the Flash EPROM, it is loaded into RAM.
- The default operating modes set by the PLC configuration are applied.

Warm start
- A warm start is performed by switching power OFF/ON with a battery connected.
- All RAM memory modules are checked and erased except of the buffered operand areas and the RETAIN variables (see Command 'Reset warm' Command ‘Reset Warm’ on page 931).
- If there is a user program stored in the Flash EPROM, it is loaded into RAM.
- The default operating modes set by the PLC configuration are applied.

RUN -> STOP
- RUN -> STOP means pressing the RUN function key on the PLC while the PLC is in run mode (PLC display "run").
- If a user program is loaded into RAM, execution is stopped.
- All outputs are set to FALSE or 0.
- Variables keep their current values, i.e., they are not initialized.
- The PLC display changes from "run" to "StoP".

START -> STOP
- START -> STOP means stopping the execution of the user program in the PLC's RAM using the menu item "Online/Stop" in the programming system.
- All outputs are set to FALSE or 0.
- Variables keep their current values, i.e., they are not initialized.
- The PLC display changes from "run" to "StoP".

Reset
- Performs a START -> STOP process.
- Preparation for program restart, i.e., the variables (VAR) (exception: RETAIN variables) are set to their initialization values.
- Reset is performed using the menu item "Online/Reset" in the programming system or pressing the function key RUN for ≥ 5 s in STOP mode.

Reset (cold)
- Performs a START -> STOP process.
- Preparation for program restart, i.e., the variables (VAR) (also RETAIN variables) are set to their initialization values.
- Reset (cold) is performed using the menu item "Online/Reset (cold)" in the programming system.

Reset (original)
- Resets the controller to its original state (deletion of Flash, SRAM (%M, area, %R area, RETAIN, RETAIN PERSISTENT), Communication Module configurations and user program!).
- Reset (original) is performed using the menu item "Online/Reset (original)" in the programming system.

STOP -> RUN
- STOP -> RUN means short pressing the RUN function key on the PLC while the PLC is in STOP mode (PLC display "StoP").
- If a user program is loaded into RAM, execution is continued, i.e., variables will not be set to their initialization values.
- The PLC display changes from "StoP" to "run".
STOP -> START
- STOP -> START means continuing the execution of the user program in the PLC's RAM
  using the menu item "Online/Start" in the programming system.
- If a user program is loaded into RAM, execution is continued, i.e., variables will not be set to
  their initialization values.
- The PLC display changes from "StoP" to "run".

Download
- Download means loading the complete user program into the PLC's RAM. This process is
  started by selecting the menu item "Online/Download" in the programming system or after
  confirming a corresponding system message when switching to online mode (menu item
  "Online/Login").
- Execution of the user program is stopped.
- In order to store the user program to the Flash memory, the menu item "Online/Create boot
  project" must be called after downloading the program.
- Variables are set to their initialization values according to the initialization table.
- RETAIN variables can have wrong values as they can be allocated to other memory
  addresses in the new project!
- A download is forced by the following:
  - changed PLC configuration
  - changed task configuration
  - changed library management
  - changed compile-specific settings (segment sizes)
  - execution of the commands "Project/Clean all" and "Project/Rebuild All".

Online change
- After a project has changed, only these changes are compiled when pressing the key <F11>
  or calling the menu item "Project/Build". The changed program parts are marked with a blue
  arrow in the block list.
- The term Online Change means loading the changes made in the user program into the
  PLC's RAM using the programming system (after confirming a corresponding system mes-
  sage when switching to online mode, menu item "Online/Login").
- Execution of the user program is not stopped. After downloading the program changes, the
  program is re-organized. During re-organization, no further online change command is
  allowed. The storage of the user program to the Flash memory using the command "Online/
  Create boot project" cannot be initiated until re-organization is completed.
- Online Change is not possible after:
  - changes in the PLC configuration
  - changes in the task configuration
  - changes in the library management
  - changed compile-specific settings (segment sizes)
  - performing the commands "Project/Clean all" and "Project/Rebuild All".

Data buffering
- Data buffering, i.e., maintaining data after power ON/OFF, is only possible, if a battery is
  connected. The following data can be buffered completely or in parts:
  - Data in the addressable flag area (%M area)
  - RETAIN variable
  - PERSISTENT variable (number is limited, no structured variables)
  - PERSISTENT area (%R area)
- In order to buffer particular data, the data must be excluded from the initialization process
  (see Initialization of Variables).

Start of the user program
The user program (UP) is started according to the following table. It is assumed that a valid user
program is stored to the Flash memory.
See Storage Devices Chapter 1.5.7.1.4 “Storage device details” on page 3511.
<table>
<thead>
<tr>
<th>Action</th>
<th>No SD memory card with UP installed</th>
<th>No SD memory card with UP installed</th>
<th>SD memory card with UP installed</th>
<th>SD memory card with UP installed</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Auto run = ON</td>
<td>Auto run = OFF</td>
<td>Auto run = ON</td>
<td>Auto run = OFF</td>
</tr>
<tr>
<td>Voltage ON or Warm start or Cold start</td>
<td>UP is loaded from Flash into RAM and started from Flash.</td>
<td>No UP is loaded from Flash. When logging in, the message &quot;No program available in the controller ...&quot; is displayed.</td>
<td>UP is loaded from the SD memory card into Flash memory and RAM and then started from RAM.</td>
<td>UP is loaded from the SD memory card to the Flash memory. RAM remains empty. When logging in, the message &quot;No program available in the controller ...&quot; is displayed.</td>
</tr>
<tr>
<td>STOP -&gt; RUN</td>
<td>UP in RAM is started.</td>
<td>UP in RAM is started.</td>
<td>UP in RAM is started.</td>
<td>UP in RAM is started.</td>
</tr>
<tr>
<td>STOP -&gt; START</td>
<td>UP in RAM is started.</td>
<td>UP in RAM is started.</td>
<td>UP in RAM is started.</td>
<td>UP in RAM is started.</td>
</tr>
<tr>
<td>Download *1)</td>
<td>The UP currently stored in the CPU's RAM is stopped. The built UP is loaded from the PC into the PLC's RAM.</td>
<td>The built UP is loaded from the PC into the PLC's RAM.</td>
<td>The UP currently stored in the CPU's RAM is stopped. The built UP is loaded from the PC into the PLC's RAM.</td>
<td>The built UP is loaded from the PC into the PLC's RAM.</td>
</tr>
<tr>
<td>Online Change *2)</td>
<td>Processing of the UP currently stored in the CPU's RAM is continued. The changes made to the UP are loaded from the PC into the PLC's RAM. The UP is reorganized and processed.</td>
<td>The changes made to the UP are loaded from the PC into the PLC's RAM. The UP is reorganized.</td>
<td>Processing of the UP currently stored in the CPU's RAM is continued. The changes made to the UP are loaded from the PC into the PLC's RAM. The UP is reorganized and processed.</td>
<td>The changes made to the UP are loaded from the PC into the PLC's RAM. The UP is reorganized.</td>
</tr>
</tbody>
</table>

Remarks:

*1): After the download is completed, the program is not automatically stored to the Flash memory. To perform this, create a boot project `boot` from Chapter 1.3.1.11.6 “Generating boot applications” on page 305. If the UP is not stored to the Flash memory, the UP is reloaded from the Flash memory after voltage OFF/ON. Start the program either by pressing the RUN/STOP function key or using Automation Builder.

*2): After the online change process is completed, the program is not automatically stored to the Flash memory. For this, after reorganization is completed create a boot project `boot` from Chapter 1.3.1.11.6 “Generating boot applications” on page 305. During reorganization and flashing, no further online change command is allowed. If the UP is not stored to the Flash memory, the UP is reloaded from the Flash memory after voltage OFF/ON.

**Task configuration**

The task model processes the following kind of tasks:
- Non real time system tasks: system tasks with no real time property (e.g. file access, Ethernet Communication, OPC UA, …)
- Non real time IEC tasks: IEC tasks with no real time property
- Real time system tasks: system tasks with real time property
- Real time IEC tasks: IEC tasks with real time property

The possible number of tasks depends on the type of Processor Module. How to distribute the IEC tasks over multiple CPU cores and on how to use the IEC task configuration for Automation Builder is described in detail in the CODESYS task configuration section.

**Task configuration** Chapter 1.3.1.25.1.38.1 “Tab ‘Configuration’” on page 839

---

**PLC utilization**

The parameters cpuload and plcload represent the actual CPU load or PLC load of the system.

- **cpuload:** This value represents the time the PLC requires to calculate all processes running on the PLC. For a good system performance this value should be less than 80%. In case of a higher value, the degree of utilization should be reduced by using a more powerful PLC or by reducing the amount of processes.
- **plcload:** This value represents the time the PLC requires to calculate all real time processes. Real time processes are either high priority system tasks or IEC tasks with a priority between 0 and 15. For a good system performance this value should be less than 60%. In case of a higher value, the degree of utilization should be reduced by using a more powerful PLC.

During commissioning we recommend to monitor the CPU and PLC values online with one of the following methods:

<table>
<thead>
<tr>
<th>Automation Builder</th>
<th>Commissioning via <strong>PLC shell</strong> (command 'plcload' and 'cpuload') <strong>Chapter 1.5.6.4.4 &quot;PLC shell commands&quot;</strong> on page 3473.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Automation Builder</td>
<td>Commissioning via <strong>Device Trace</strong>. In order to display the load of the CPU or PLC, create a new Device Trace object in your PLC project. Then, <strong>Upload</strong> the data into the views <strong>Chapter 1.3.1.25.2.21.19 “Command 'Upload Trace’”</strong> on page 1034.</td>
</tr>
</tbody>
</table>
| IEC applications/IEC program | To access the parameters plcload and cpuload please use system functions as follows:  
  - **plcload:** SchedGetProcessorLoad() included in library ‘CmpSchedule’.  
  - **cpuload:** SysMCGetLoad() included in library ‘SysCpuMultiCore’. |

**Setting standard configuration**

If the target setting configuration is changed, standard configuration can be restored:
1. Open CODESYS.
2. In the “Resources” tab, double-click “PLC Configuration”.
3. Select “Menu Extras ➤ Standard Configuration”.

1.5.5.1.3 Real-time clock and battery

The real-time clock is an optional function for AC500-eCo Processor Modules.

The real-time clock operates as a PC clock. It saves date and time to a DWORD in DT format (DATE AND TIME FORMAT), i.e., in seconds passed since the start time: 1 January 1970 at 00:00.

If a battery is connected and full, the real-time clock continues to run even if the control voltage is switched off.

If no battery is inserted or the battery is empty, the real-time clocks starts with the value 0 (=1970-01-01, 00:00:00).

When switching on the control voltage, the system clock of the operating system is set to the value of the real-time clock.

Real-time clock

The PLC browser/PLC shell commands `date` and `time` are used to set the real-time clock.

The commands `date <ENTER>` or `time <ENTER>` display the current date and time of the real-time clock.

The command: `date yyyy-mm-dd<ENTER>` (year-month-day) sets the date.

The command: `time hh-mm-ss<ENTER>` (hours-minutes-seconds) sets the time.

Example:
The real-time clock should be set to 22 February 2005, 16:50.

1. Enter the date:
   ```
   date 2005-02-22<ENTER>
   
   Display: date 2005-02-22 Clock set to 2005-02-22 08:01:07
   
   The time remains unchanged.
   ```

2. Enter the time:
   ```
   time 16:50<ENTER>
   
   Display: time 16:50 Clock set to 2005-02-22 16:50:00
   ```

The following Function Blocks located in the folder "Realtime clock" of the system library `ABB_ExtUtils_AC500.lib` can be used to set and display the real-time clock (RTC) with help of the user program:
### Function Block

<table>
<thead>
<tr>
<th>Function Block</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLOCK (V3) “Library Manager  ABB-AC500 Use Cases  AC500 Utils PM&lt;Version&gt; (ABB) Function Blocks Realtime clock”</td>
<td>Sets and displays the real-time clock with values for year, month, day, hours, minutes and seconds. Also the day of week is indicated (Mo=1, Tue=2, Wed=3, Thu=4, Fr=5, Sa=6, Su=0). Note: The week of day cannot be set. It is given by the real-time clock. The input DAY_SET is ignored.</td>
</tr>
<tr>
<td>CLOCK_DT (V3) “Library Manager  ABB-AC500 Use Cases  AC500 Utils PM&lt;Version&gt; (ABB) Function Blocks Realtime clock”</td>
<td>Sets and displays the real-time clock in DT format, for example DT#2005-02-17-17:15:00.</td>
</tr>
</tbody>
</table>

The Function Blocks CLOCK and CLOCK_DT are documented in the “Library Manager” for the system library PM<Version>.lib (in AC500 Utils).

### AC500 battery

The AC500 battery buffers the following data in case of "control voltage off":

- Retentive variables in SRAM (VAR_RETAIN..END_VAR) · Chapter 1.5.5.1.1 “Handling of remanent variables for AC500 V3 products” on page 3105
- Date and time of the real-time clock

Further information:

- PLC diagnosis · Chapter 1.6.3 “Diagnosis messages” on page 3565

> To prevent data loss when using the AC500 battery, the battery status should be periodically monitored by the user program.

### Battery status

The battery status can be monitored either with the help of a user program on the PLC or in Automation Builder.

In the PLC shell of Automation Builder the command "batt" · Chapter 1.5.6.4.4 “PLC shell commands” on page 3473 “batt” can be used · Chapter 1.5.6.4.4 “PLC shell commands” on page 3473. The following is output:

<table>
<thead>
<tr>
<th></th>
<th>Battery status</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Battery empty</td>
</tr>
<tr>
<td>20</td>
<td>Remaining battery charge below 20 %</td>
</tr>
<tr>
<td>100</td>
<td>Battery charge OK</td>
</tr>
</tbody>
</table>

In the user program, the battery status can be checked with the function BATT which is available in the folder "Battery" of the system library ABB_ExtUtils_AC500.lib (“Library Manager  ABB-AC500  Use Cases”). The following is output:

<table>
<thead>
<tr>
<th></th>
<th>Battery status</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Battery empty</td>
</tr>
<tr>
<td>20</td>
<td>Remaining battery charge below 20 %, battery must be replaced</td>
</tr>
<tr>
<td>100</td>
<td>Battery charge OK</td>
</tr>
</tbody>
</table>

On the device, the battery status can be checked with the function keys of a Processor Module.
1.5.5.1.4 LEDs, display and function keys on the front panel

Overview

The display of a processor module is equipped with a background-lighted 7-segment display. This display consists of 6 digits for plain text or error codes.

Some functionalities may be not yet supported by the product. Please refer to the release notes of the product at time of release.

Display indicators

- A black square (■) denotes the state/working activity of the corresponding object on the left/right side of the display. The black square flashes according to the device's activity, e.g. during data exchange on ETH1, ETH2, COM1, etc.

**MC activity**

For the activity of the memory card the black square (■) is shown as long as a file is open on memory card.

- A black triangle (▲) points to the selected item/interface on the left/right side of the display to be configured or read. Further, it acts as a cursor for the count up/count down function keys.

A black triangle (▲) at the BATT item indicates a missing or uncharged battery.
The indicators point to the following items on the left side of the display:

<table>
<thead>
<tr>
<th>No.</th>
<th>On the left Side</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>MC (memory card)</td>
<td>Refers to the memory card status.</td>
</tr>
<tr>
<td>2</td>
<td>SYS (system)</td>
<td>Refers to the system status.</td>
</tr>
<tr>
<td>3</td>
<td>BATT (battery)</td>
<td>Refers to the battery status.</td>
</tr>
<tr>
<td>4</td>
<td>I/O bus</td>
<td>Refers to I/O bus connection.</td>
</tr>
</tbody>
</table>

The indicators point to the following items on the right side of the display:

<table>
<thead>
<tr>
<th>No.</th>
<th>On the right side</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>ETH1</td>
<td>Refers to the first Ethernet interface.</td>
</tr>
<tr>
<td>6</td>
<td>ETH2</td>
<td>Refers to the second Ethernet interface.</td>
</tr>
<tr>
<td>7</td>
<td>COM1</td>
<td>Refers to COM1 interface.</td>
</tr>
<tr>
<td>8</td>
<td>CAN</td>
<td>Refers to CAN interface.</td>
</tr>
<tr>
<td>9</td>
<td></td>
<td>Function keys on front panel</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Processor Module</th>
<th>Display Variant</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PM56xx-2ETH</td>
<td>![Display Image]</td>
<td>Display of a processor module with support for 2 Ethernet interfaces, CAN and COM1.</td>
</tr>
</tbody>
</table>

Startup procedure of the PLC

Startup procedure of a new PLC from factory

<table>
<thead>
<tr>
<th>State</th>
<th>Display</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>![Display Image]</td>
<td>Display on system start (power on).</td>
</tr>
<tr>
<td>1</td>
<td>![Display Image]</td>
<td>PLC is in boot mode.</td>
</tr>
<tr>
<td>2</td>
<td>![Display Image]</td>
<td>PLC is in initialization mode.</td>
</tr>
<tr>
<td>3</td>
<td>![Display Image]</td>
<td>No system firmware (SystemFW) available. Start update firmware. PLC is waiting for a firmware download via Automation Builder or memory card. See Firmware identification and update</td>
</tr>
</tbody>
</table>

2020/12/10

3ADR010583, 1, en_US
Startup procedure of a PLC with system firmware

The startup procedure depends on the selected PLC mode.

<table>
<thead>
<tr>
<th>PLC mode</th>
<th>Display</th>
<th>Startup Behavior</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td><img src="image" alt="Mode 00" /></td>
<td>The user program will be loaded and run. PLC changes to mode „RUN“.</td>
</tr>
<tr>
<td>01</td>
<td><img src="image" alt="Mode 01" /></td>
<td>User program will not be loaded / run. PLC stay in mode „STOP“.</td>
</tr>
<tr>
<td>02</td>
<td><img src="image" alt="Mode 02" /></td>
<td>Reserved for further development (currently like Mode 00).</td>
</tr>
</tbody>
</table>

Mode 01 can be activated via function key CFG (see Further information on page 3119), or by pressing function key RUN during startup of PLC until Mode 01 is shown on display.

<table>
<thead>
<tr>
<th>State</th>
<th>Display</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td><img src="image" alt="State 0" /></td>
<td>Display on system start (power on).</td>
</tr>
<tr>
<td>1</td>
<td><img src="image" alt="State 1" /></td>
<td>PLC is in boot mode (see Further information on page 3119).</td>
</tr>
<tr>
<td>2</td>
<td><img src="image" alt="State 2" /></td>
<td>PLC is in initialization mode (see Further information on page 3119).</td>
</tr>
<tr>
<td>3</td>
<td><img src="image" alt="State 3" /></td>
<td>PLC is in STOP mode (see Further information on page 3119). Same as status “Stop” in Automation Builder.</td>
</tr>
<tr>
<td>4</td>
<td><img src="image" alt="State 4" /></td>
<td>PLC is in RUN mode (see Further information on page 3119). Switch into RUN mode is only possible if a valid boot project is available in the flash memory.</td>
</tr>
<tr>
<td>5</td>
<td><img src="image" alt="State 5" /></td>
<td>Reminder: demo license Will be displayed for 5 minutes at every license check Only in RUN mode and as of SystemFW V3.2.0</td>
</tr>
<tr>
<td>6</td>
<td><img src="image" alt="State 6" /></td>
<td>10 minutes step reminder: license was removed Will be displayed for 5 minutes Only in RUN mode and as of SystemFW V3.2.0</td>
</tr>
</tbody>
</table>

Description of LEDs

The LEDs below the display indicate the status of the processor module:
<table>
<thead>
<tr>
<th>LED</th>
<th>State</th>
<th>Color</th>
<th>LED = ON</th>
<th>LED = OFF</th>
<th>LED flashes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Power LED</td>
<td>Denotes the power supply state of the processor module</td>
<td>Green</td>
<td>Voltage is present (24 V DC)</td>
<td>Voltage is missing</td>
<td>If the LED flashes fast (4 Hz) a firmware update is finished with no errors.</td>
</tr>
<tr>
<td>(PWR)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>If the Run LED flashes fast (4 Hz), alternating with a flashing Run LED the firmware is updated.</td>
</tr>
<tr>
<td>Run LED</td>
<td>Denotes the activity state of the processor module</td>
<td>Green</td>
<td>Processor module is in RUN mode</td>
<td>Processor module is in STOP mode</td>
<td>To enforce boot mode 1, keep the RUN function key pressed during the boot procedure. In this case, the Run LED flashes slowly (1 Hz). A subsequent project download (from within Automation Builder) cancels the blinking.</td>
</tr>
<tr>
<td>(RUN)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Error LED</td>
<td>Denotes an error</td>
<td>Red</td>
<td>An error has occurred.</td>
<td>No errors or only warnings have occurred.</td>
<td>If the Error LED flashes slowly (1 Hz) a firmware update from the memory card is finished with errors.</td>
</tr>
<tr>
<td>(ERR)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>If the Error LED flashes fast with AC500 on display a fatal system error has occurred.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>If the Error LED flashes fast (4 Hz) alternating with a flashing Run LED the firmware is updated.</td>
</tr>
</tbody>
</table>

A running processor module is indicated with the state RUN on the display, a deactivated processor module is indicated with the state STOP. In both cases the display's backlight is off.

**Description of the function keys**

**Overview**

The processor module can be operated manually using the function keys on the front panel:
<table>
<thead>
<tr>
<th>Function Key</th>
<th>Description</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>RUN</strong></td>
<td>Run</td>
<td>Toggles between RUN and STOP mode. <strong>Switching into RUN mode is only possible if an error free project has been created and downloaded with Automation Builder.</strong></td>
</tr>
<tr>
<td><strong>VAL</strong></td>
<td>Value</td>
<td>Shows different state values of the processor module.</td>
</tr>
<tr>
<td><strong>ESC</strong></td>
<td>Escape</td>
<td>Quits the current menu, submenu or function without saving.</td>
</tr>
<tr>
<td><strong>OK</strong></td>
<td>OK / Acknowledgement</td>
<td>Acknowledges the current value or selects a menu/submenu. Changes that have been sent to the processor module successfully are confirmed with <strong>done</strong> on the display.</td>
</tr>
<tr>
<td><strong>DIAG</strong></td>
<td>Diagnostic</td>
<td>Allows evaluation of error messages in detail.</td>
</tr>
<tr>
<td><strong>CFG</strong></td>
<td>Configuration</td>
<td>Show/set IP configuration, PLC startup mode and Ethernet address. Enters submenus.</td>
</tr>
<tr>
<td><strong>↑</strong></td>
<td>Count up / navigate in submenu</td>
<td>Press the function key repeatedly in order to increase the value each time by 1, or navigate in submenu to previous entry. Keep the function key pressed in order to count up fast.</td>
</tr>
<tr>
<td><strong>↓</strong></td>
<td>Count down / navigate in submenu</td>
<td>Press the function key repeatedly in order to decrease the value each time by 1, or navigate in submenu to next entry. Keep the function key pressed in order to count down fast.</td>
</tr>
</tbody>
</table>

Backlight is switched on for about 20 seconds by pressing any function key.

### Start and stop PLC

**Function key RUN**

<table>
<thead>
<tr>
<th>State</th>
<th>Description Menu level 0</th>
<th>Result on pressing one of the function keys</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td><img src="image" alt="Stop" /></td>
<td><strong>RUN</strong> Short click: State 1 is displayed. Long click (&gt;5 sec): State 2 is displayed. <strong>OK</strong> No action <strong>ESC</strong> No action</td>
</tr>
<tr>
<td>State</td>
<td>Description Menu level 0</td>
<td>Result on pressing one of the function keys</td>
</tr>
<tr>
<td>-------</td>
<td>--------------------------</td>
<td>--------------------------------------------</td>
</tr>
</tbody>
</table>
| 1     | PLC only in state RUN if a correct project is in RAM of PLC | **RUN** State 0 is displayed.  
**OK** STOP - same as Online stop in Automation Builder (halt, no init of variables)  
**ESC** No RESET State 0 is displayed. |
| 2     | **RUN** LED=ON | Perform RESET same as Online reset in Automation Builder (stop and init variables)  
State 0 is displayed.  
**ESC** Refers to sub menu 1 |

**Configuration**

**Configuration CPU firmware SystemFW V3.1.x and DisplayFW V3.0**

Navigation starts with the Processor Module being in RUN/STOP mode (State 0). By pressing one of the three function keys a certain action is triggered. The result of this action is described in the result columns of the tables.

<table>
<thead>
<tr>
<th>State</th>
<th>Description - Main menu 1</th>
<th>Result on pressing one of the function keys</th>
</tr>
</thead>
</table>
| 0     | The Processor Module is in RUN/STOP mode. | **CFG** State 1 is displayed.  
**ESC** Remains in RUN/STOP mode.  
**OK** Remains in RUN/STOP mode. |
| 1     | **ETH1** | State 2 is displayed.  
**ESC** Return into RUN/STOP mode.  
**OK** Refers to sub menu 1 |
| 2     | **ETH2** | State 3 is displayed.  
**ESC** Return into RUN/STOP mode.  
**OK** Shows DONE, your settings are saved. Return into RUN/STOP mode. |
| 3     | **Adr000** | State 4 is displayed.  
**ESC** Return into RUN/STOP mode.  
**OK** Refers to sub menu 1 |
| 4     | **Adr000** | State 5 is displayed.  
**ESC** Return into RUN/STOP mode.  
**OK** Shows DONE, your settings are saved. Return into RUN/STOP mode. |
### State Description - Main menu 1

<table>
<thead>
<tr>
<th>State</th>
<th>Result on pressing one of the function keys</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td><img src="image1.png" alt="Image" /> <strong>Change the values with the Count up/Count down function keys.</strong>&lt;br&gt;See also Further information on page 3120.</td>
</tr>
</tbody>
</table>

### State Description - Main menu 2

<table>
<thead>
<tr>
<th>State</th>
<th>Result on pressing one of the function keys</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td><img src="image2.png" alt="Image" /> <strong>The Processor Module is in RUN/STOP mode.</strong>&lt;br&gt;State 1 is displayed. Remains in RUN/STOP mode. Remains in RUN/STOP mode.</td>
</tr>
<tr>
<td>1</td>
<td><img src="image3.png" alt="Image" /> <strong>State 2 is displayed.</strong>&lt;br&gt;Return into RUN/STOP mode. Refers to sub menu 1</td>
</tr>
<tr>
<td>2</td>
<td><img src="image4.png" alt="Image" /> <strong>Change the values with the Count up/Count down function keys.</strong>&lt;br&gt;State 3 is displayed. Return into RUN/STOP mode. Your settings are saved. State 2 is displayed.</td>
</tr>
<tr>
<td>3</td>
<td><img src="image5.png" alt="Image" /> <strong>Change the values with the Count up/Count down function keys.</strong>&lt;br&gt;See also Further information on page 3120. State 1 is displayed. Return into RUN/STOP mode. Shows DONE, your settings are saved. Return into RUN/STOP mode.</td>
</tr>
</tbody>
</table>

### Function key CFG main menu with ETH1 / ETH2 mode: “Switch functionality ETH1-ETH2” (see Switch functionality)

Navigation starts with the Processor Module being in RUN/STOP mode (State 0). By pressing one of the three function keys a certain action is triggered. The result of this action is described in the result columns of the tables.

### State Description - Submenu 1

<table>
<thead>
<tr>
<th>State</th>
<th>Result on pressing one of the function keys</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td><img src="image6.png" alt="Image" /> <strong>IPETH1 or IPETH2; DHCP not active</strong> State 2 is displayed. Return into RUN/STOP mode. State 1.2 is displayed.</td>
</tr>
</tbody>
</table>

Function key CFG sub menu IPETH1 or IPETH2; DHCP not active
<table>
<thead>
<tr>
<th>State</th>
<th>Description - Submenu 1</th>
<th>Result on pressing one of the function keys</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.2</td>
<td>IP Configuration (address, subnet mask, gateway)</td>
<td>State 1.3 is displayed. Aborts the menu unchanged. Return to State 1.1</td>
</tr>
<tr>
<td>1.3</td>
<td>Reset to production data (default settings)</td>
<td>State 1.4 is displayed. Aborts the menu unchanged. Return to State 1.1</td>
</tr>
<tr>
<td>1.4</td>
<td>Activate DHCP Sets a DHCP address.</td>
<td>State 1.2 is displayed. Aborts the menu unchanged. Return to State 1.1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Function key</th>
<th>CFG sub menu</th>
<th>IPETH1 or IPETH2; DHCP active</th>
</tr>
</thead>
<tbody>
<tr>
<td>State</td>
<td>Description - Submenu 2</td>
<td>Result on pressing one of the function keys</td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>2.1</td>
<td>IPETH1 or IPETH2</td>
<td>State 2 is displayed. Aborts the menu unchanged. Return to State 0.</td>
</tr>
<tr>
<td>2.2</td>
<td>DHCP active</td>
<td>State 2.3 is displayed. Aborts the menu unchanged. Return to State 2.1.</td>
</tr>
<tr>
<td>2.3</td>
<td>IP Configuration (address, subnet mask, gateway)</td>
<td>State 2.4 is displayed. Aborts the menu unchanged. Return to State 2.1.</td>
</tr>
<tr>
<td>2.4</td>
<td>Reset to production data (default settings)</td>
<td>--</td>
</tr>
</tbody>
</table>

- **CFG**
- **ESC**
- **OK**

Activate RESET to default by pressing **OK** twice.
Shows DONE, your settings are saved. Return into RUN/STOP mode.
<table>
<thead>
<tr>
<th>State</th>
<th>Description - Submenu 3</th>
<th>Result on pressing one of the function keys</th>
</tr>
</thead>
</table>
| 3.1   | IP Configuration (address, subnet mask, gateway) | \[\begin{array}{|c|c|c|} \hline
| CFG | ESC | OK  \\ \hline
| State 2.4 is displayed. Aborts the menu unchanged. Return to State 1.1 (submenu IPETH1 or IPETH2) | State 3.2 is displayed. |
| \end{array} \right. |
| 3.2   | IP address A1-A4  
★ Number is blinking if value has changed and is not yet sent to CPU | \[\begin{array}{|c|c|c|} \hline
| CFG | ESC | OK  \\ \hline
| State 3.3 is displayed. Aborts the menu unchanged. Return to State 1.1 (submenu IPETH1 or IPETH2) | Sends changed values to CPU and go to default menu RUN/STOP Displays: DONE New settings stored in CPU. or: FAIL Failed to write new settings to CPU. |
| \end{array} \right. |
| 3.3   | Subnet mask N1-N4  
★ Number is blinking if value has changed and is not yet sent to CPU | \[\begin{array}{|c|c|c|} \hline
| CFG | ESC | OK  \\ \hline
| State 3.4 is displayed. Aborts the menu unchanged. Return to State 1.1 (submenu IPETH1 or IPETH2) | Sends changed values to CPU and go to default menu RUN/STOP Displays: DONE New settings stored in CPU. or: FAIL Failed to write new settings to CPU. |
| \end{array} \right. |
| 3.4   | Gateway G1-G4  
★ Number is blinking if value has changed and is not yet sent to CPU | \[\begin{array}{|c|c|c|} \hline
| CFG | ESC | OK  \\ \hline
<p>| State 3.2 is displayed again. Aborts the menu unchanged. Return to State 1.1 (submenu IPETH1 or IPETH2) Aborts the menu unchanged. Return to State 1. | Sends changed values to CPU and go to default menu RUN/STOP Displays: DONE New settings stored in CPU. or: FAIL Failed to write new settings to CPU. |
| \end{array} \right. |</p>
<table>
<thead>
<tr>
<th>Function key</th>
<th>Description - Submenu 4</th>
<th>Result on pressing one of the function keys</th>
</tr>
</thead>
<tbody>
<tr>
<td>CFG sub menu ADR</td>
<td>State 4.1 is displayed.</td>
<td>Aborts the menu unchanged. Return to State 1</td>
</tr>
<tr>
<td></td>
<td>DHCP not active: State 1.2 is displayed</td>
<td>DHCP active: State 2.2 is displayed</td>
</tr>
<tr>
<td>CFG menu level 1, with ETH1 / ETH2 mode: “Two separate interfaces” (see Switch functionality)</td>
<td>State 4.2 is displayed.</td>
<td>Aborts the menu unchanged. Return to State 1</td>
</tr>
<tr>
<td></td>
<td>Sends changed values to CPU and go to default menu RUN/STOP</td>
<td>Displays: DONE New settings stored in CPU. or: FAIL Failed to write new settings to CPU.</td>
</tr>
<tr>
<td>CFG menu level 1, with ETH1 / ETH2 mode: “Two separate interfaces” (see Switch functionality)</td>
<td>State 2.3 is displayed.</td>
<td>Aborts the menu unchanged. Return to State 4.1</td>
</tr>
<tr>
<td></td>
<td>Sends changed values to CPU and go to default menu RUN/STOP</td>
<td>Displays: DONE New settings stored in CPU. or: FAIL Failed to write new settings to CPU.</td>
</tr>
<tr>
<td>CFG menu level 1, with ETH1 / ETH2 mode: “Two separate interfaces” (see Switch functionality)</td>
<td>State 4.1 is displayed.</td>
<td>Aborts the menu unchanged. Return to State 4.1</td>
</tr>
<tr>
<td></td>
<td>Sends changed values to CPU and go to default menu RUN/STOP</td>
<td>Displays: DONE New settings stored in CPU. or: FAIL Failed to write new settings to CPU.</td>
</tr>
</tbody>
</table>

Configuration CPU firmware SystemFW >=V3.2.0 and DisplayFW >=V4.1

Navigation starts with the Processor Module being in RUN/STOP mode (State 0). By pressing one of the five function keys a certain action is triggered. The result of this action is described in the result columns of the tables.
<table>
<thead>
<tr>
<th>State</th>
<th>Description - CFG menu level 1</th>
<th>Result on pressing one of the function keys</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>The Processor Module is in RUN/STOP mode.</td>
<td>State 1 is displayed.</td>
</tr>
<tr>
<td>1</td>
<td>Switch is OFF</td>
<td>State 2 is displayed if KNX functionality is active. State 3 is displayed if KNX functionality is inactive.</td>
</tr>
<tr>
<td>2</td>
<td>KNX program button (appears if functionality is active)</td>
<td>State 3 is displayed.</td>
</tr>
<tr>
<td>3</td>
<td>Refers to submenu level 2 &quot;Function key CFG menu level 2 (IPETH1 or IPETH2); &quot; on page 313</td>
<td>State 4 is displayed.</td>
</tr>
<tr>
<td>4</td>
<td>Refers to submenu level 2 &quot;Function key CFG menu level 2 (IPETH1 or IPETH2); &quot; on page 313</td>
<td>State 5 is displayed.</td>
</tr>
<tr>
<td>State</td>
<td>Description - CFG menu level 1</td>
<td>Result on pressing one of the function keys</td>
</tr>
<tr>
<td>-------</td>
<td>--------------------------------</td>
<td>---------------------------------------------</td>
</tr>
<tr>
<td>5</td>
<td><img src="image1.png" alt="Image" /></td>
<td><img src="image2.png" alt="Image" /></td>
</tr>
<tr>
<td></td>
<td><img src="image3.png" alt="Image" /></td>
<td><img src="image4.png" alt="Image" /></td>
</tr>
<tr>
<td></td>
<td>Note: COM1 mode RS232 (default) or RS485 can only be shown but not changed. This is a PLC boot parameter, and must be set in AB. Chapter 1.5.6.2.10.3.1 “Configuration” on page 3330. Mode is activated in PLC boot process.</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td><img src="image5.png" alt="Image" /></td>
<td><img src="image6.png" alt="Image" /></td>
</tr>
<tr>
<td></td>
<td><img src="image7.png" alt="Image" /></td>
<td><img src="image8.png" alt="Image" /></td>
</tr>
<tr>
<td></td>
<td>Refers to submenu set startup mode of PLC “Function key CFG submenu show / set startup mode of PLC” on page 3136</td>
<td></td>
</tr>
</tbody>
</table>

**Function key CFG menu level 1, with ETH1 / ETH2 mode:**

“Switch functionality ETH1-ETH2” (see Switch functionality)

Navigation starts with the Processor Module being in RUN/STOP mode (State 0). By pressing one of the five function keys a certain action is triggered. The result of this action is described in the result columns of the tables.
<table>
<thead>
<tr>
<th>State</th>
<th>Description - <strong>CFG</strong> menu level 1</th>
<th>Result on pressing one of the function keys</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>The Processor Module is in RUN/STOP mode.</td>
<td><strong>CFG</strong></td>
</tr>
<tr>
<td></td>
<td>State 1 is displayed.</td>
<td>Remains in RUN/STOP mode.</td>
</tr>
<tr>
<td>1</td>
<td>Switch is ON</td>
<td>State 2 is displayed if KNX functionality is active.</td>
</tr>
<tr>
<td>2</td>
<td>KNX program button (appears if functionality is active)</td>
<td>State 3 is displayed.</td>
</tr>
<tr>
<td>3</td>
<td>Refers to submenu level 2</td>
<td>State 4 is displayed.</td>
</tr>
<tr>
<td></td>
<td>“Function key <strong>CFG</strong> menu level 2 (IPETH1 or IPETH2): ” on page 3131</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Note: COM1 mode RS232 (default) or RS485 can only be shown but not changed. This is a <strong>PLC boot parameter</strong> and must be set in AB **Chapter 1.5.6.2.10.3.1 “Configuration” on page 3330. Mode is activated in PLC boot process.</td>
<td>State 5 is displayed.</td>
</tr>
<tr>
<td>State</td>
<td>Description - CFG menu level 1</td>
<td>Result on pressing one of the function keys</td>
</tr>
<tr>
<td>-------</td>
<td>--------------------------------</td>
<td>------------------------------------------</td>
</tr>
<tr>
<td>5</td>
<td><img src="image" alt="nod 00" /></td>
<td>Refers to submenu set startup mode of PLC</td>
</tr>
<tr>
<td></td>
<td>CFG menu level 1</td>
<td>“Function key CFG submenu show / set startup mode of PLC” on page 3136</td>
</tr>
</tbody>
</table>

**Function key CFG menu level 2 (IPETH1 or IPETH2):**

<table>
<thead>
<tr>
<th>State</th>
<th>Description - CFG menu level 2</th>
<th>Result on pressing one of the function keys</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td><img src="image" alt="IPETH1" /></td>
<td>State 1 is displayed.</td>
</tr>
</tbody>
</table>

IPETH1 or IPETH2

<table>
<thead>
<tr>
<th>State</th>
<th>Description - CFG menu level 2</th>
<th>Result on pressing one of the function keys</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td><img src="image" alt="STATIC" /></td>
<td>IP Configuration (address, subnet mask, gateway)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>State</th>
<th>Description - CFG menu level 2</th>
<th>Result on pressing one of the function keys</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td><img src="image" alt="nodHCP" /></td>
<td>Refers to submenu set DHCP “Function key CFG menu level 2 Show/set DHCP” on page 3134</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>State</th>
<th>Description - CFG menu level 2</th>
<th>Result on pressing one of the function keys</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td><img src="image" alt="id 000" /></td>
<td>Activate DHCP Sets a DHCP address.</td>
</tr>
</tbody>
</table>
### State Description - CFG menu level 2

<table>
<thead>
<tr>
<th>State</th>
<th>Description - CFG menu level 2</th>
<th>Result on pressing one of the function keys</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>Reset to production data (default settings)</td>
<td>Refers to submenu set <strong>RESET</strong>. State 1 is displayed. State 3 is displayed.</td>
</tr>
<tr>
<td>5</td>
<td><strong>APPLY</strong></td>
<td>Remain all changes. State 1 is displayed.</td>
</tr>
<tr>
<td>6</td>
<td><strong>done</strong></td>
<td>Changes applied. <strong>done</strong> is displayed for 2 sec. then return into RUN/STOP mode.</td>
</tr>
</tbody>
</table>

### Function key CFG menu level 3, show / set STATIC

#### State Description - Submenu STATIC

<table>
<thead>
<tr>
<th>State</th>
<th>Description - Submenu STATIC</th>
<th>Result on pressing one of the function keys</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>IP Configuration (address, subnet mask, gateway)</td>
<td>State 1 is displayed.</td>
</tr>
<tr>
<td>1</td>
<td>IP address A1-A4 If submenu is entered: Number is blinking if value has changed and is not yet sent to CPU</td>
<td>Refers to submenu of A1-A4 Count down A1-A4</td>
</tr>
</tbody>
</table>
### State Description - Sub-menu STATIC

<table>
<thead>
<tr>
<th>State</th>
<th>Description - Sub-menu STATIC</th>
<th>Result on pressing one of the function keys</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td></td>
<td><strong>CFG</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Refers to submenu of N1-N4</td>
</tr>
<tr>
<td></td>
<td></td>
<td><img src="image" alt="Subnet mask N1-N4" /></td>
</tr>
</tbody>
</table>

If submenu is entered: Number is blinking if value has changed and is not yet sent to CPU

### State Description - CFG menu level 4

<table>
<thead>
<tr>
<th>State</th>
<th>Description - CFG menu level 4</th>
<th>Result on pressing one of the function keys</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td><strong>CFG</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td>No action</td>
</tr>
<tr>
<td></td>
<td></td>
<td><img src="image" alt="A4 is blinking if submenu is entered" /></td>
</tr>
</tbody>
</table>

Function key
**CFG menu level 4**, Example: **I/P address A4**
<table>
<thead>
<tr>
<th>State</th>
<th>Description - Show/set DHCP</th>
<th>Result on pressing one of the function keys</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td><img src="image" alt="nodHCP" /></td>
<td><strong>CFG</strong></td>
</tr>
<tr>
<td></td>
<td><strong>State 1 is displayed.</strong></td>
<td><strong>Aborts the menu unchanged.</strong></td>
</tr>
<tr>
<td>1</td>
<td><img src="image" alt="dHCp" /></td>
<td><strong>State 2 is displayed.</strong></td>
</tr>
<tr>
<td></td>
<td><strong>IP address A1-A4</strong></td>
<td><strong>★ Value is blinking if value has changed and is not yet sent to CPU</strong></td>
</tr>
<tr>
<td>2</td>
<td><img src="image" alt="nodHCP" /></td>
<td><strong>State 1 is displayed.</strong></td>
</tr>
<tr>
<td>3</td>
<td><img src="image" alt="APPLY" /></td>
<td><strong>Remain all changes</strong></td>
</tr>
<tr>
<td>4</td>
<td><img src="image" alt="donE" /></td>
<td><strong>Changes applied donE is displayed for 2 sec. then return into RUN/STOP mode.</strong></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>State</th>
<th>Description - Show/set Id</th>
<th>Result on pressing one of the function keys</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td><img src="image" alt="d 000" /></td>
<td><strong>CFG</strong></td>
</tr>
<tr>
<td></td>
<td><strong>Count down value: 255 ... 000, starting with current value</strong></td>
<td><strong>Count up value: 000 ... 255, starting with current value</strong></td>
</tr>
</tbody>
</table>
### State 1

**Description - Show/set ID**

- **IP address A1-A4**
  - Number is blinking if value has changed and is not yet sent to CPU

<table>
<thead>
<tr>
<th>Function key</th>
<th>Description - Show/set ID</th>
<th>Result on pressing one of the function keys</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CFG</strong></td>
<td></td>
<td><strong>↓</strong> Count down value: 255 ... 000, starting with current value</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>↑</strong> Count up value: 000 ... 255, starting with current value</td>
</tr>
<tr>
<td><strong>ESC</strong></td>
<td></td>
<td><strong>Cancel</strong> Discard the changes. Stop blinking. Show previous value.</td>
</tr>
<tr>
<td><strong>OK</strong></td>
<td></td>
<td>Go to CFG menu level 2 (IPETH1 or IPETH2) state 5-6.</td>
</tr>
</tbody>
</table>

### State 2

**Description - Sub-menu show / set RESET**

- **Display is blinking if value has changed and is not yet sent to CPU**

<table>
<thead>
<tr>
<th>Function key</th>
<th>Description - Sub-menu show / set RESET</th>
<th>Result on pressing one of the function keys</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CFG</strong></td>
<td>State 1 is displayed.</td>
<td><strong>↓</strong> No action</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>↑</strong> No action</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>ESC</strong> Aborts the menu unchanged. Return to sub-menu level 1</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>OK</strong> Discard all made changes. Stop blinking. Send command &quot;reset to factory settings&quot; to CPU</td>
</tr>
</tbody>
</table>

### Function key

**CFG submenu show / set RESET**

- **State 0**
  - State 1 is displayed.
- **State 1**
  - Display is blinking if value has changed and is not yet sent to CPU
  - Aborts the menu unchanged. Return to sub-menu level 1

---

Chapter 1.5.5.1.4.2.1 "Startup procedure of a new PLC from factory" on page 3119. reset Ask confirmation. Go back to default menu RUN/STOP. State 2 is displayed.
### State Description - Submenu show / set

<table>
<thead>
<tr>
<th>State</th>
<th>Description - Submenu show / set</th>
<th>Result on pressing one of the function keys</th>
</tr>
</thead>
<tbody>
<tr>
<td>RESET</td>
<td></td>
<td>![APPLY]</td>
</tr>
<tr>
<td></td>
<td>Changes applied</td>
<td>![donE]</td>
</tr>
<tr>
<td></td>
<td>Changes failed</td>
<td>![FAILED]</td>
</tr>
</tbody>
</table>

### Function key CFG submenu show / set startup mode of PLC

<table>
<thead>
<tr>
<th>State</th>
<th>Description - Submenu show / set startup mode of PLC</th>
<th>Result on pressing one of the function keys</th>
</tr>
</thead>
<tbody>
<tr>
<td>Start</td>
<td></td>
<td>![Nod 00]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>![Nod 01]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>![Nod 02]</td>
</tr>
</tbody>
</table>

### Reading out values

#### Reading out values CPU firmware SystemFW 3.1.x and DisplayFW 3.0

The following settings of the Processor Module can be read out by pressing the function key VAL repeatedly:

1. Displays time of the Processor Module (hh.mm.ss).
2. Displays date of the Processor Module (yy.mm.dd).
3. Displays state of battery (ub 100 = 100%, ub 020 = 20% or ub 000 = empty).
4. Displays version of display firmware (e.g. d 3.0 r (= display Version 3.0 release)).
5. Displays version of CPU firmware (e.g. C 3.1.0r (= CPU Version 3.1.0 release)).
6. Displays CPU type.
7. Displays default text (RUN/STOP).
# Reading out values CPU firmware SystemFW >=V3.2.0 and DisplayFW >=V4.1

**Function key VAL**

By pressing Function Key **VAL** state 1 is displayed

<table>
<thead>
<tr>
<th>State</th>
<th>Description Menu VAL</th>
<th>Result on pressing one of the function keys</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Time of the Processor Module (hh.mm.ss).</td>
<td>![Image] No action</td>
</tr>
<tr>
<td>2</td>
<td>Date of the Processor Module (yy.mm.dd).</td>
<td>![Image] State 3 displayed</td>
</tr>
<tr>
<td>3</td>
<td>State of battery (ub 100 = 100%, ub 020 = 20% or ub 000 = empty).</td>
<td>![Image] State 4 displayed</td>
</tr>
<tr>
<td>4</td>
<td>Version of display firmware (e.g. d 4.1 r (= display Version 4.1 release)).</td>
<td>![Image] State 5 displayed</td>
</tr>
<tr>
<td>5</td>
<td>Version of CPU firmware (e.g. C 3.2.0r (= CPU Version 3.2.0 release)).</td>
<td>![Image] State 6 displayed</td>
</tr>
<tr>
<td>6</td>
<td>CPU type.</td>
<td>![Image] State 1 displayed</td>
</tr>
</tbody>
</table>
### Reading out diagnosis messages on the CPU

**Table 663: Example: no diagnosis message in status list**

<table>
<thead>
<tr>
<th>State</th>
<th>Display</th>
<th>Result on pressing one of the function keys</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>The processor module is in RUN/STOP mode.</td>
<td>State 1 is displayed - - - -</td>
</tr>
<tr>
<td>1</td>
<td><img src="image" alt="No Err" /></td>
<td>No action No action Return into RUN/STOP mode.</td>
</tr>
</tbody>
</table>

**Table 664: Example: diagnosis messages in status list**

<table>
<thead>
<tr>
<th>State</th>
<th>Display</th>
<th>Result on pressing one of the function keys</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>The processor module is in RUN/STOP mode.</td>
<td>State 1 is displayed - - - -</td>
</tr>
<tr>
<td>1</td>
<td><img src="image" alt="Err 4" /></td>
<td>Number of diagnosis messages; here 4 Go to first/next diagnosis message in status list (e.g., state 2) Go to last/previous diagnosis message in status list Return into RUN/STOP mode. Return into RUN/STOP mode.</td>
</tr>
<tr>
<td>2</td>
<td><img src="image" alt="Err 4 BAT" /></td>
<td>Diagnosis message example: Error battery empty or missing Toggling between state 2 and 3 Selects displayed diagnosis message and shows details ( \text{\textregistered} ) Table 6 65 “Example: error battery empty or missing” on page 3139 Go to first/next diagnosis message in status list Go to last/previous diagnosis message in status list Return into RUN/STOP mode. Return into RUN/STOP mode.</td>
</tr>
<tr>
<td>3</td>
<td><img src="image" alt="Id 1" /></td>
<td>Error ID example Toggling between state 2 and 3 Selects displayed diagnosis message and shows details ( \text{\textregistered} ) Table 6 65 “Example: error battery empty or missing” on page 3139 Go to first/next diagnosis message in status list Go to last/previous diagnosis message in status list Return into RUN/STOP mode. Return into RUN/STOP mode.</td>
</tr>
</tbody>
</table>
### Table 665: Example: error battery empty or missing

<table>
<thead>
<tr>
<th>State</th>
<th>Display</th>
<th>Result on pressing one of the function keys</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>DIAG</td>
</tr>
<tr>
<td>0</td>
<td><img src="Image" alt="Display" /></td>
<td>State 2 is displayed</td>
</tr>
<tr>
<td></td>
<td>E4 = error severity 4 bAt = subdevice battery Toggling between state 0 and 1</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td><img src="Image" alt="Display" /></td>
<td>State 3 is displayed</td>
</tr>
<tr>
<td></td>
<td>Error ID example Toggling between state 0 and 1</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td><img src="Image" alt="Display" /></td>
<td>State 4 is displayed</td>
</tr>
<tr>
<td></td>
<td>Error number 8 Battery is missing or empty</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td><img src="Image" alt="Display" /></td>
<td>State 5 is displayed</td>
</tr>
<tr>
<td></td>
<td>Detail 1 Subdevice 22: battery</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td><img src="Image" alt="Display" /></td>
<td>State 6 is displayed</td>
</tr>
<tr>
<td></td>
<td>Detail 2 Error type 0: device</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td><img src="Image" alt="Display" /></td>
<td>State 1 is displayed</td>
</tr>
<tr>
<td></td>
<td>Detail 3 Error type number 0: device itself</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td><img src="Image" alt="Display" /></td>
<td>State 1 is displayed</td>
</tr>
<tr>
<td></td>
<td>Detail 4 Additional information 0: none</td>
<td></td>
</tr>
</tbody>
</table>
Enable flashing of display

Blink functionality

As of SystemFW 3.1.0 and AB 2.1.0 the Blink functionality is implemented. “Blink” means – activate flashing of background light of AC500 display in Automation Builder via “IP-Configuration” tool (for details see Blink functionality Chapter 1.5.6.2.4.2.1 “Blink functionality” on page 3269).

Wink functionality

As of SystemFW 3.1.0 and AB 2.0.0 the Wink functionality is implemented. “Wink” means – activate flashing of background light of AC500 display in Automation Builder via communication settings (for details see Wink functionality).

Function blocks

PmErrLedSet

This Function Block switches the ERR-LED ON and OFF.

PmDispSetText

With this Function Block a text can be displayed on the CPU.

Further information is provided in the documentation of the AC500 Pm library. See the function blocks PmErrLedSet and PmDispSetText in the “Library Manager” of Automation Builder.

1.5.5.1.5 Onboard technologies

Ethernet

Ethernet protocols and ports for AC500 V3 products

<table>
<thead>
<tr>
<th>Description</th>
<th>PM5630-2ETH</th>
<th>PM5650-2ETH</th>
<th>PM5670-2ETH</th>
<th>PM5675-2ETH</th>
<th>≥ CPU firmware</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABB netConfig</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>V3.0.0</td>
</tr>
<tr>
<td>Online access with driver 3S UDP BlkDrvUdp</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>V3.0.0</td>
</tr>
<tr>
<td>Online access with driver 3S TCP/IP BlkDrvTcp</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>V3.0.0</td>
</tr>
<tr>
<td>Modbus TCP server</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>V3.0.0</td>
</tr>
<tr>
<td>Modbus TCP client with POU ETHx MOD MAST</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>V3.0.1</td>
</tr>
<tr>
<td>UDP out of user program with library netBaseService.lib</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>V3.0.0</td>
</tr>
<tr>
<td>UDP data exchange, Network variables</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>V3.0.0</td>
</tr>
<tr>
<td>TCP/IP out of user program with library netBaseService.lib</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>V3.0.0</td>
</tr>
<tr>
<td>Web server on PLC with web visualization</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>V3.0.0</td>
</tr>
<tr>
<td>SNTP (Simple Network Time Protocol) client with 3S licenced store package SNTPService.package. Library container: SNTPService</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>V3.0.0</td>
</tr>
<tr>
<td>IEC60870-5-104 control station incl. 2nd connection and 2nd port</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>V3.0.0</td>
</tr>
<tr>
<td>IEC60870-5-104 substation incl. 2nd port</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>V3.0.0</td>
</tr>
<tr>
<td>Description</td>
<td>PM5630 -2ETH</td>
<td>PM5650 -2ETH</td>
<td>PM5670 -2ETH</td>
<td>PM567 5-2ETH</td>
<td>≥ CPU firmware</td>
</tr>
<tr>
<td>-----------------------------------------------------------------------------</td>
<td>--------------</td>
<td>--------------</td>
<td>--------------</td>
<td>--------------</td>
<td>---------------</td>
</tr>
<tr>
<td>FTP server (see FTP server configuration Chapter 1.5.6.3.4.1 &quot;Configuration of FTP server&quot; on page 3428)</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>V3.0.0</td>
</tr>
<tr>
<td>CODESYS network variables</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>V3.0.0</td>
</tr>
<tr>
<td>OPC DA server</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>V3.0.0</td>
</tr>
<tr>
<td>OPC UA server</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>V3.0.0</td>
</tr>
<tr>
<td>ICMP – ping out of user project with POU ETHx_ICMP_PING</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>V3.0.0</td>
</tr>
<tr>
<td>DHCP client</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>V3.1.0</td>
</tr>
<tr>
<td>SNTP (Simple Network Time Protocol) client system solution (see SNTP client configuration Chapter 1.5.6.3.3.2.1 &quot;SNTP client configuration&quot; on page 3425)</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>V3.1.0</td>
</tr>
<tr>
<td>SNTP (Simple Network Time Protocol) server system solution (see SNTP server configuration Chapter 1.5.6.3.3.2.2 &quot;SNTP server configuration&quot; on page 3427)</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>V3.1.0</td>
</tr>
<tr>
<td>Maximum number of Input/output allowed variable on Ethernet for the protocol</td>
<td>2 kB /2 kB</td>
<td>4 kB /4 kB</td>
<td>5 kB /5 kB</td>
<td>5 kB /5 kB</td>
<td>V3.4.0</td>
</tr>
<tr>
<td>IEC 61850 (MMS server, GOOSE) ²)</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>V3.1.0</td>
</tr>
<tr>
<td>EthernetIP Adapter ¹, ²)</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>AB 2.4.1/FW 3.4.1</td>
</tr>
<tr>
<td>KNX - Building communication ²)</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>V3.2.x</td>
</tr>
<tr>
<td>BACnet-BC - Infrastructure communication ²)</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>V3.3.1</td>
</tr>
<tr>
<td>HTTPS – secure web server on PLC with CODESYS web visualization (see secure web server Chapter 1.5.6.3.6.3.2 &quot;Secure web server&quot; on page 3433)</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>V3.1.0</td>
</tr>
<tr>
<td>WebVisu for data visualisation on web-server HTML5</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>V3.0.0</td>
</tr>
<tr>
<td>FTPS – secure FTP (see secure FTP Chapter 1.5.6.3.6.3.3 “Secure FTP&quot; on page 3434)</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>V3.1.0</td>
</tr>
<tr>
<td>Secure online access with driver 3S UDP BlkDrvUdp</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>V3.1.0</td>
</tr>
<tr>
<td>Secure online access with driver 3S TCP/IP BlkDrvTcp</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>V3.1.0</td>
</tr>
<tr>
<td>ICMP – ping out of user project with POU ETHx_ICMP_PING or EthicmpPing (PLCopen style)</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>V3.1.0</td>
</tr>
<tr>
<td>Modbus TCP client (master) with POU ETHx_MOD_MAST or ModTcpMast (PLCopen style)</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>V3.1.0</td>
</tr>
</tbody>
</table>
Default open Ethernet ports of PM56xx-2ETH

After startup without a PLC project the PM56xx-2ETH contains the following Ethernet ports and sockets:

<table>
<thead>
<tr>
<th>Protocol</th>
<th>Port</th>
<th>Sockets</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABB NetConfig ¹)</td>
<td>UDP 24576</td>
<td>1 permanent socket per interface</td>
</tr>
<tr>
<td>Online access with driver 3S UDP BlkDrvUdp (with scan)</td>
<td>UDP 1740</td>
<td>1 socket per connection + 4 listen</td>
</tr>
<tr>
<td>Online access with driver 3S Tcp/Ip BlkDrvTcp (no scan)</td>
<td>TCP 11740</td>
<td>1 socket per connection + 1 listen</td>
</tr>
<tr>
<td>OPC UA server ²)</td>
<td>TCP 4840</td>
<td></td>
</tr>
</tbody>
</table>

Remarks:
¹): The port 24576 for ABB NetConfig protocol can be disabled via PLC configuration by deleting the protocol node from configuration tree of Ethernet interfaces ETH1 and ETH2.
²): The port 4840 for OPC UA server is closed by default as of SystemFW V3.1.0.

All other ports are closed by default.

Overview of protocols, sockets and ports

<table>
<thead>
<tr>
<th>Protocol</th>
<th>Port</th>
<th>Sockets</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABB netConfig</td>
<td>24576</td>
<td>1 permanent socket per interface</td>
</tr>
<tr>
<td>3S gateway client (e.g. CODESYS) to gateway server</td>
<td>1217</td>
<td>1 permanent socket</td>
</tr>
<tr>
<td>Online access with driver 3S UDP BlkDrvUdp (with scan)</td>
<td>1740</td>
<td>1 socket per connection + 4 listen</td>
</tr>
<tr>
<td>Online access with driver 3S block driver TCP/IP (no scan)</td>
<td>11740</td>
<td>1 socket per connection + 1 listen</td>
</tr>
<tr>
<td>Modbus TCP server</td>
<td>502 or configurable</td>
<td>1 socket listen + 1 socket per server connection, number of server connections is configurable in AB</td>
</tr>
<tr>
<td>Modbus TCP client with POU ETHx_MOD_MAST</td>
<td>Random</td>
<td>1 socket per connection with POU ETHx_MOD_MAST</td>
</tr>
<tr>
<td>UDP out of user program with library SysLibSockets.lib</td>
<td>1 ... 65535</td>
<td>1 socket per connection</td>
</tr>
<tr>
<td>TCP/IP out of user program with library SysLibSockets.lib</td>
<td>1 ... 65535</td>
<td>1 socket per connection</td>
</tr>
<tr>
<td>Web server on PLC with web visualization</td>
<td>80</td>
<td>1 listen and 1 per connection</td>
</tr>
<tr>
<td>Protocol</td>
<td>Port</td>
<td>Sockets</td>
</tr>
<tr>
<td>----------</td>
<td>------</td>
<td>---------</td>
</tr>
<tr>
<td>SNTP client</td>
<td>123</td>
<td>1 permanent socket</td>
</tr>
<tr>
<td>IEC60870-5-104 control station</td>
<td>Random</td>
<td>1 per connection</td>
</tr>
<tr>
<td>IEC60870-5-104 substation</td>
<td>2404</td>
<td>1 per connection</td>
</tr>
<tr>
<td>FTP server (see <a href="#">FTP server configuration</a> Chapter 1.5.6.3.4.1 “Configuration of FTP server” on page 3428)</td>
<td>Command port = 21 Data active mode = 20 Data passive mode = random</td>
<td>1 per session, max. 4 allowed</td>
</tr>
<tr>
<td>CODESYS network variables</td>
<td>1202</td>
<td>(UDP broadcast)</td>
</tr>
<tr>
<td>OPC DA server (default 3S block driver)</td>
<td>UDP = 1740 or TCP/IP = 11740</td>
<td>1 socket per connection</td>
</tr>
<tr>
<td>OPC UA server</td>
<td>4840</td>
<td>1 permanent socket</td>
</tr>
<tr>
<td>ICMP – ping out of user project with POU ETHx_ICMP_PING DHCP</td>
<td>none</td>
<td>No socket</td>
</tr>
<tr>
<td>DHCP</td>
<td>67</td>
<td>1 socket during startup</td>
</tr>
<tr>
<td>SNTP (Simple Network Time Protocol) client system solution (see <a href="#">SNTP client configuration</a> Chapter 1.5.6.3.3.2.1 “SNTP client configuration” on page 3425)</td>
<td>123</td>
<td>1 permanent socket</td>
</tr>
<tr>
<td>SNTP (Simple Network Time Protocol) server system solution (see <a href="#">SNTP server configuration</a> Chapter 1.5.6.3.3.2.2 “SNTP server configuration” on page 3427)</td>
<td>123</td>
<td>1 permanent socket</td>
</tr>
<tr>
<td>HTTPS – secure web server on PLC with CODESYS web visualization (see <a href="#">secure web server</a> Chapter 1.5.6.3.6.3.2 “Secure web server” on page 3433)</td>
<td>443</td>
<td>1 listen and 1 per connection</td>
</tr>
<tr>
<td>FTPS – secure FTP (see <a href="#">secure FTP</a> Chapter 1.5.6.3.6.3.3 “Secure FTP” on page 3434)</td>
<td>Command port = 21 Data active mode = 20 Data passive mode = random</td>
<td>1 per session, max. 4 allowed</td>
</tr>
<tr>
<td>Secure online access with driver 3S UDP BlkDrvUdp</td>
<td>1740</td>
<td>1 socket per connection + 1 listen</td>
</tr>
<tr>
<td>Secure online access with driver 3S TCP/IP BlkDrvTcp</td>
<td>11740</td>
<td>1 socket per connection + 1 listen</td>
</tr>
<tr>
<td>ICMP – ping out of user project with POU ETHx_ICMP_PING or EthIcmpPing (PLCopen style)</td>
<td>None</td>
<td>No socket</td>
</tr>
<tr>
<td>Modbus TCP client (master) with POU ETHx_MOD_MAST or ModTcpMast (PLCopen style)</td>
<td>Random</td>
<td>1 socket per connection with POU ETHx_MOD_MAST or ModTcpMast</td>
</tr>
</tbody>
</table>
### Limitation of connections per protocol

<table>
<thead>
<tr>
<th>Protocol</th>
<th>PM5630-2ETH</th>
<th>PM5650-2ETH</th>
<th>PM5670-2ETH</th>
<th>PM5675-2ETH</th>
<th>≥ CPU firmware</th>
</tr>
</thead>
<tbody>
<tr>
<td>Modbus TCP server (e.g. for SCADA access)</td>
<td>30</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>3.0.3</td>
</tr>
<tr>
<td></td>
<td>40</td>
<td>40</td>
<td>40</td>
<td>40</td>
<td>3.1.0</td>
</tr>
<tr>
<td></td>
<td>15</td>
<td>25</td>
<td>50</td>
<td>50</td>
<td>3.1.3</td>
</tr>
<tr>
<td>Modbus TCP client with POU ETHx_MOD_MAST</td>
<td>n/a</td>
<td>100</td>
<td>n/a</td>
<td>n/a</td>
<td>3.0.1</td>
</tr>
<tr>
<td></td>
<td>40</td>
<td>40</td>
<td>40</td>
<td>40</td>
<td>3.1.0</td>
</tr>
<tr>
<td></td>
<td>30</td>
<td>50</td>
<td>120</td>
<td>120</td>
<td>3.1.3</td>
</tr>
<tr>
<td>Modbus TCP client with POU ETHx_MOD_MAST or ModTcpMast (PLCopen style)</td>
<td>30</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>3.1.0</td>
</tr>
<tr>
<td></td>
<td>30</td>
<td>50</td>
<td>120</td>
<td>120</td>
<td>3.1.3</td>
</tr>
<tr>
<td>IEC60870-5-104 control station incl. 2(^{\text{nd}}) connection and 2(^{\text{nd}}) port</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>3.1.0</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>10</td>
<td>20</td>
<td>20</td>
<td>3.4.0</td>
</tr>
<tr>
<td>IEC60870-5-104 substation incl. 2(^{\text{nd}}) port</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>3.1.0</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>10</td>
<td>20</td>
<td>20</td>
<td>3.4.0</td>
</tr>
<tr>
<td>IEC60870-5-104: No. of free tags + additional license for extension (^1)</td>
<td>1.000</td>
<td>5.000</td>
<td>10.000</td>
<td>10.000</td>
<td>3.4.0</td>
</tr>
<tr>
<td>FTP server</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>3.1.0</td>
</tr>
<tr>
<td>Online access with driver 3S UDP BlkDrvUdp</td>
<td>n/a</td>
<td>4</td>
<td>n/a</td>
<td>n/a</td>
<td>3.0.0</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>3.1.0</td>
</tr>
<tr>
<td>Online access with driver 3S TCP/IP BlkDrvTcp</td>
<td>n/a</td>
<td>4</td>
<td>n/a</td>
<td>n/a</td>
<td>3.0.0</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>3.1.0</td>
</tr>
<tr>
<td>OPC DA server (number of connections)</td>
<td>n/a</td>
<td>4</td>
<td>n/a</td>
<td>n/a</td>
<td>3.0.0</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>3.1.0</td>
</tr>
<tr>
<td>OPC UA server (number of connections)</td>
<td>50</td>
<td>50</td>
<td>50</td>
<td>50</td>
<td>3.1.0</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>20</td>
<td>50</td>
<td>50</td>
<td>3.4.0</td>
</tr>
<tr>
<td>No. of free tags + additional license for extension (^1)</td>
<td>1.000</td>
<td>5.000</td>
<td>30.000</td>
<td>30.000</td>
<td>3.4.0</td>
</tr>
<tr>
<td>min sampling rate (limit)</td>
<td>500 ms</td>
<td>100 ms</td>
<td>50 ms</td>
<td>50</td>
<td>AB 2.4.0/ FW 3.4.0</td>
</tr>
<tr>
<td>Number of Webvisu instances in a project limitation</td>
<td>16</td>
<td>32</td>
<td>64</td>
<td>64</td>
<td>AB 2.4.0/ FW 3.4.0</td>
</tr>
<tr>
<td>Secure online access with driver 3S UDP BlkDrvUdp</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>3.1.0</td>
</tr>
<tr>
<td>Secure online access with driver 3S TCP/IP BlkDrvTcp</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>3.1.0</td>
</tr>
<tr>
<td>FTPS - secure FTP server</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>3.1.0</td>
</tr>
<tr>
<td>RTV (Remote Target Visualization)</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>3.1.0</td>
</tr>
<tr>
<td>Remarks: (^1)): in preparation</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Remarks: 
\(^1\)): in preparation
The PLC types PM5630-2ETH, PM5670-2ETH and PM5675-2ETH are available as of SystemFW 3.1.0.

**Ethernet configuration**

**Default Ethernet configuration**

<table>
<thead>
<tr>
<th>Module</th>
<th>IP Address</th>
<th>Netmask</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>PM56xx-2ETH</td>
<td>ETH1: 192.168.0.10&lt;br&gt;ETH2: 192.168.1.10</td>
<td>255.255.255.0</td>
<td>The Ethernet ports must be configured in different sub networks.</td>
</tr>
</tbody>
</table>

For changing the default addresses, see *IP-Configuration tool* Chapter 1.5.6.2.2.4.2 “Configuration of the IP settings with IP configuration tool” on page 3267, or description of function key Chapter 1.5.5.1.4.4 “Description of the function keys” on page 3121.
Online access

Preferred driver for online access: 3S UDP block driver BlkDrvUdp. This driver allows to scan and select the connected PLC’s.

Alternative: 3S TCP/IP block driver. This driver requires at least 2 sockets:
- 1x driver “BlkDrvTcp” on port 11740
- 1x listen on port 11740 if PLC has established online connection

Each established connection needs one socket. In addition one socket on port 11740 is listening.
1. Startup the PLC.
   - One socket on port 11740 (listen).
2. Login from Automation Builder via driver “BlkDrvTcp”.
   - 2 sockets on port 11740 (1x online, 1x listen)
3. Additional login out of OPC server with the same driver.
   - 3 sockets on port 11740 (2x online, 1x listen)
4. Additional connect CP600 via driver “BlkDrvTcp”.
   - 4 sockets on port 11740 (3x online, 1x listen)

SNTP client and server

As of version 3.1.0 the SystemFW provides a SNTP Protocol implementation which can be used for time synchronization of PLC clock. It can be used as SNTP Client or / and SNTP Server. But only one instance of each can be executed at the same time on one PLC.

See also, Configuration of SNTP Protocol V3

> Chapter 1.5.6.3.3.2 “Configuration of the SNTP protocol” on page 3425.

The SNTP server is listening only on the Ethernet interface, which the protocol is configured on. It is not possible to have an SNTP server on several Ethernet interfaces.

To read diagnosis information from the SNTP protocol within an IEC Application the Function Block PmSntpInfo can be used. This Function block is part of the library ABB_Pm_AC500.lib. It can also be used to determine the synchronization state of the PLC clock.

Using network variables in AC500 V3

When using network variables via UDP broadcast, the default broadcast address is set to 255.255.255.255.

This will not work on PLCs with multiple Ethernet interfaces, because of undecidable routing.

Set the broadcast address to a matching subnet broadcast address, depending on which interface should be used to send the variables into the network.
Example

- ETH1 with IP 192.168.0.10 netmask 255.255.255.0
- ETH2 with IP 192.168.1.10 netmask 255.255.255.0

If you want the network variables to be broadcast on ETH1, use broadcast address 192.168.0.255.

Onboard CAN configuration

Onboard CAN interface supports the following protocols

- CAN Open Manager: Connection of CI581 and CI582 without additional I/O modules
- CAN 2A/2B
- J1939

Configuration in Automation Builder is described in chapter “Chapter 1.5.6.2.7.1.1 “CM598-CAN - CANopen master communication module” on page 3286.

Further information can be found in chapter “Chapter 1.5.6.2.12 “CAN-based fieldbuses” on page 3332.

AC500 V3 PLCs provide the following methods for CAN integration:

- Onboard CAN interface
- Communication module CM598-CN

Differences in supported protocols can be seen in the table below.

<table>
<thead>
<tr>
<th></th>
<th>Onboard CAN</th>
<th>CM598-CN</th>
</tr>
</thead>
<tbody>
<tr>
<td>CANopen Manager</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>CAN 2A/2B</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>J1939</td>
<td>X</td>
<td></td>
</tr>
</tbody>
</table>
1.5.5.1.6 Hot swap

Preconditions for using hot swap

**Hot swap**

**WARNING!**
**Risk of explosion or fire in hazardous environments during hot swapping!**
Hot swap must not be performed in flammable environments to avoid life-threatening injury and property damage resulting from fire or explosion.

**WARNING!**
**Electric shock due to negligent behavior during hot swapping!**
To avoid electric shock:
- Make sure the following conditions apply:
  - Digital outputs are not under load.
  - Input/output voltages above safety extra low voltage/protective extra low voltage (SELV/PELV) are switched off.
  - Modules are fully interlocked with the terminal unit with both snap-fits engaged before switching on loads or input/output voltage.
  - Never touch exposed contacts (dangerous voltages).
  - Stay away from electrical contacts to avoid arc discharge.
  - Do not operate a mechanical installation improperly.

**NOTICE!**
**Risk of damage to I/O modules!**
Hot swapping is only allowed for I/O modules.
Processor modules and communication interface modules must not be removed or inserted during operation.

\[ H = \text{Hot swap} \]

**Hot swap**
System requirements for hot swapping of I/O modules:
- Types of terminal units that support hot swapping of I/O modules have the appendix TU5xx-H.
- I/O modules as of index F0.

The following I/O bus masters support hot swapping of attached I/O modules:
- Communication interface modules CI5xx as of index F0.
- Processor modules PM56xx-2ETH with firmware version as of V3.2.0.
The index of the module is in the right corner of the label.

**NOTICE!**
Risk of damage to I/O modules!
Modules with index below F0 can be damaged when inserted or removed from the terminal unit in a powered system.

**NOTICE!**
Risk of damage to I/O modules!
Do not perform hot swapping if any I/O module with firmware version lower than 3.0.14 is part of the I/O configuration.

For min. required device index see table below.

<table>
<thead>
<tr>
<th>Device</th>
<th>Min. required device index for I/O module as of FW Version 3.0.14</th>
</tr>
</thead>
<tbody>
<tr>
<td>AI523 (-XC)</td>
<td>D2</td>
</tr>
<tr>
<td>AI531</td>
<td>D4</td>
</tr>
<tr>
<td>AI531-XC</td>
<td>D2</td>
</tr>
<tr>
<td>AI561</td>
<td>B2</td>
</tr>
<tr>
<td>AI562</td>
<td>B2</td>
</tr>
<tr>
<td>AI563</td>
<td>B3</td>
</tr>
<tr>
<td>AO523 (-XC)</td>
<td>D2</td>
</tr>
<tr>
<td>AO561</td>
<td>B2</td>
</tr>
<tr>
<td>AX521 (-XC)</td>
<td>D2</td>
</tr>
<tr>
<td>AX522 (-XC)</td>
<td>D2</td>
</tr>
<tr>
<td>AX561</td>
<td>B2</td>
</tr>
<tr>
<td>CD522 (-XC)</td>
<td>D1</td>
</tr>
<tr>
<td>DA501 (-XC)</td>
<td>D2</td>
</tr>
<tr>
<td>Device</td>
<td>Min. required device index for I/O module as of FW Version 3.0.14</td>
</tr>
<tr>
<td>------------</td>
<td>---------------------------------------------------------------</td>
</tr>
<tr>
<td>DC522 (-XC)</td>
<td>D2</td>
</tr>
<tr>
<td>DC523 (-XC)</td>
<td>D2</td>
</tr>
<tr>
<td>DC532 (-XC)</td>
<td>D2</td>
</tr>
<tr>
<td>DC561</td>
<td>B2</td>
</tr>
<tr>
<td>DC562</td>
<td>A2</td>
</tr>
<tr>
<td>DI524 (-XC)</td>
<td>D2</td>
</tr>
<tr>
<td>DI561</td>
<td>B2</td>
</tr>
<tr>
<td>DI562</td>
<td>B2</td>
</tr>
<tr>
<td>DI571</td>
<td>B2</td>
</tr>
<tr>
<td>DI572</td>
<td>A1</td>
</tr>
<tr>
<td>DO524 (-XC)</td>
<td>A3</td>
</tr>
<tr>
<td>DO526</td>
<td>A2</td>
</tr>
<tr>
<td>DO526-XC</td>
<td>A0</td>
</tr>
<tr>
<td>DO561</td>
<td>B2</td>
</tr>
<tr>
<td>DO562</td>
<td>A2</td>
</tr>
<tr>
<td>DO571</td>
<td>B3</td>
</tr>
<tr>
<td>DO572</td>
<td>B2</td>
</tr>
<tr>
<td>DO573</td>
<td>A1</td>
</tr>
<tr>
<td>DX522 (-XC)</td>
<td>D2</td>
</tr>
<tr>
<td>DX531</td>
<td>D2</td>
</tr>
<tr>
<td>DX561</td>
<td>B2</td>
</tr>
<tr>
<td>DX571</td>
<td>B3</td>
</tr>
<tr>
<td>FM562</td>
<td>A1</td>
</tr>
</tbody>
</table>

**Compatibility of hot swap**

<table>
<thead>
<tr>
<th>I/O module on TU5xx-H connected to I/O bus master</th>
<th>Central I/O on V3 CPU</th>
</tr>
</thead>
<tbody>
<tr>
<td>Required version of I/O bus master</td>
<td>AC500 V3 CPU types: PM56xx-2ETH</td>
</tr>
<tr>
<td>Fieldbus master when used as remote I/O with AC500 V3</td>
<td>Firmware as of V3.2.0</td>
</tr>
<tr>
<td>When used as remote I/O on third party controller (PLC or DCS)</td>
<td>-</td>
</tr>
</tbody>
</table>
Hot swap behavior

The following table describes the behavior in case of I/O attached to the AC500 CPU with firmware supporting hot swap on the I/O bus.

<table>
<thead>
<tr>
<th>Hot Swap Behavior</th>
<th>Central I/O on V3 CPU</th>
</tr>
</thead>
<tbody>
<tr>
<td>Start-up behavior with unplugged or damaged I/O module on hot swap terminal unit TU5xx-H</td>
<td>System and I/O modules attached to the CPU are starting (except unplugged or damaged module when plugged on hot swap terminal unit). As soon as the correct and operational I/O module is plugged on the terminal unit, the module is configured and ready to start. No specific setting needed.</td>
</tr>
<tr>
<td>Start-up behavior with wrong I/O module type on any terminal unit</td>
<td>System and I/O modules are not starting</td>
</tr>
<tr>
<td>Diagnosis of presence of hot swap terminal unit</td>
<td>Diagnosis using PLC browser command &quot;io-bus desc&quot; in Automation Builder V3. The PLC browser then provides an overview of the modules on the I/O bus including the position of hot swap terminal units in the I/O bus. In the application program this can be detected with a function block &quot;IoModuleHotSwapInfo&quot; (Library: AC500_Io / Function Blocks / I/O-Bus). One instance of function block is needed per terminal unit on the I/O bus. The function block provides five outputs delivering information about slot number, hot swap capability and plugged/unplugged state of the I/O module.</td>
</tr>
<tr>
<td>Diagnosis while hot swap module is pulled or module (mounted on hot swap terminal unit) has stopped working</td>
<td>If module is pulled then diagnosis Err 9480 &quot;Module removed from Hot Swap Terminal Unit&quot; is generated</td>
</tr>
<tr>
<td>Diagnosis after plugging the I/O module on the hot swap terminal unit</td>
<td>Diagnosis Err 9480 is automatically acknowledged</td>
</tr>
</tbody>
</table>

1.5.5.1.7 KNX IP integration

This document describes the system aspects of AC500 V3 PLCs interface to KNX and its integration into the engineering tools.

It assumes - beneath basic AC500 and Automation Builder know-how at least basic experience and expertise in use of KNX and ETS (engineering software for KNX).

Additional information can be found:
- In the example projects and their documentation (C:\Users\Public\Documents\Automation-Builder\Examples\PS5604-KNX).
- In ABB products and services.
Introduction

KNX is a bus system used more on the room and floor level in buildings (e.g. for lighting, shading and local HVAC devices).

The KNX as such doesn’t necessarily need a dedicated controller for simple connection of sensor/switch to receiving/actuator devices.

The signals exchanged via the protocol are so called “group addresses” (“objects”), which are downloaded via ETS to all the thereby linked (=grouped) devices.

On the room level it typically has a serial wiring called KNX TP (twisted pair), which then is linked to floor or central building or management level via IP routers. On Ethernet it is called KNXnet/IP abbreviated also as KNX IP.

![Diagram of KNX network](image)

The AC500 V3 PLC is after the Automation Builder engineering step and download a standard KNX device, in which KNX communication is done via the IP network interface. It is arranged topologically on the area / main line of the KNX IP routers and communicates with them via the KNXnet/IP protocol.

Engineering workflow

Both engineering software systems for AC500 V3 PLC (Automation Builder) and KNX (ETS) are directly linked.

A data exchange for the group objects (hereinafter also called communication objects) from the Automation Builder to ETS (via an XML file) and received by a DCA (Device configuration APP) for ETS is available.

The AC500 V3 PLC is integrated into the ETS via a certified KNX “device” with the transferred group objects as configured in Automation Builder and a physical KNX address (transferred via ETS and KNX IP to the AC500 V3 PLC).
Programming and commissioning of the AC500 V3 PLC starts with Automation Builder:
Configuration of the AC500 V3 PLC, its communications, here KNX, and I/O modules and all
necessary parameters.
1. Configure programmable KNX controller in CODESYS by adding group objects to the
device.
2. Use group objects as inputs and outputs in the IEC application.
3. Download of the above into the AC500 V3 PLC (via the engineering interface)
The subsequent linking of the AC500 V3 PLC and the other KNX devices takes place with
the vendor independent KNX commissioning software ETS:
5. Install DCA Plugin and the AC500 V3 PLC device description in ETS
6. Connect group objects in ETS and assign group addresses.
7. KNX IP download to AC500 V3 PLC.
8. The physical KNX address of the AC500 V3 PLC must be set before or during download
of the KNX configuration.

The programming of the AC500 V3 PLC and the KNX commissioning can be done also by dif-
ferent people at different times and with same or separate engineering PCs. Both projects carry
out their own download parts of their respective configurations to the AC500 V3 PLC.
The only data exchange between the two Engineering programs are the details about the KNX
group objects defined in the ABB Automation Builder. This is done flexibly via the XML configu-
ration file.

Prerequisites
- PC(s) with Windows 7 or higher with Administrator right(s)
- At least temporary network access to the internet for downloading and installing of:
  - Automation Builder as of version AB 2.1.2, (and e.g. example .project)
  - ETS5 and the necessary additional files (DCA .etsapp, device description .knxprod) plus
    possibly an matching example .knxproj) to above Automation Builder .project
- Network access to the local network, were the AC500 V3 PLC and KNX devices are connected.
- PS5604-KNX AC500 runtime license for each dedicated AC500 V3 PLC used in KNX networks (see Chapter 1.5.5.1.7.6.1 "KNX runtime license" on page 3164).
- The current IP address of the engineering PC(s) where Automation Builder and ETS are located in same Network / masked IP range, as the AC500 V3 PLC to be used.

General settings and system behavior

The KNX interface at the AC500 V3 PLC is only active during the PLC is in RUN.
1. Download Automation Builder program.
2. Run PLC.
3. Set physical address or download KNX application Programm via ETS.
   - The bus status can be viewed in Online View of Automation Builder.

KNX communication is only working after download the matching ETS application to the AC500 V3 PLC. Until then, the AC500 V3 PLC KNX communication is deactivated and marked with a warning symbol.

![Fig. 283: KNX Interface not ready](image)

However in this state the AC500 V3 PLC can still be switched to the KNX programming mode and the physical KNX address can be programmed. Also the device info can be read by ETS.

If the KNX interface is ready, this can be recognized by the green symbol on the KNX interface in the Automation Builder.

![Fig. 284: KNX Interface ready](image)

Start-up behavior

Start/Stop PLC

- KNX bus works only in "RUN" mode.
- If the PLC is in "STOP" mode the KNX bus and the outputs are reset.

To avoid this behavior in "STOP" mode set the following preferences at the PLC_AC500_CPU:
1. Double-click PLC_AC500_V3 <...> and click PLC Settings.
2. Enable checkbox **Update IO while in stop** and select in dropdown-menu **Behavior for Outputs in Stop** “Keep current values”.

**Warm start / Cold start**
If the PLC is reset also the connected objects will be reset on the KNX bus.

**Power ON/Off**
After Power ON the KNX Interface need approximately 1 s to start after the PLC program had started. During this period no inputs will be recognized by the PLC and no outputs will be send to the bus.

### Engineering of KNX in Automation Builder

#### Creation of KNX group objects

---

**Attention**
This information refers to Automation Builder as of version 2.2.0.
---

The data exchange with the KNX bus is done via KNX group objects.

1. Double-click node “KNX” in the device tree “click General ➔ click Add”.
   - The window **Communication object** appears.
2. Enter your properties:
   - **Group Object Number:**
     The number of the KNX Group Object must match within the controller. It is displayed in the ETS and influences the display order in the ETS and the Automation Builder.
   - **Type:**
     Selection of the communication direction.
     – Input means that the controller receives values from the KNX bus.
     – Output means that the controller sends values to the KNX bus.
   - **Data Point Type:**
     Specification of the KNX data point type (DPT) of the Group Object. This determines the memory size, scaling and unit. For further information see the KNX Standard.
   - **Group Object Name:**
     The name of the KNX Group Object. It is freely selectable and is displayed in the ETS under the field name.
   - **Group Object Function:**
     The name of the function of the Group Object. It is freely selectable and is displayed in the ETS under the field Function.

Based on this selection, the flags of the KNX Group Object are set accordingly in the ETS.

You can use the [Export CSV...] button in the “General” tab menu bar to display the list of KNX group objects in a spreadsheet program such as Excel and edit and extend it flexibly. Then you can import them again via [Import CSV].

After you have created all the required KNX group objects, export them using the [Export to ETS] button. This exported file contains the configuration of the KNX group objects of the AC500 V3 PLC and is imported by ETS for linking to other KNX devices. If you have not yet created project information under main menu “Project ➔ Project Information”, the default values will be used during the export.
To use these KNX group objects in your application program, you must assign them with IEC 61131-3 variables. This additional abstraction layer of an additional variable allows you to create modular automation programs that are independent of the used bus system or input / output modules.

The assignment is possible either via the parameter page “KNX I/O Mapping” or “I/O mapping list”. Both editors offer the same function in different representations.

On the KNX I/O Mapping page, the KNX variables are shown hierarchically. Each KNX Group Object consists of several channels with additional information. These differ depending on whether it is an input or an output.

The view is structured as follows:

- **Variable:** Enter the name of the IEC 61131-3 variable that you want to assign to this channel (KNX Group Object).
- **Mapping:** Shows if the channel is already linked
- **Channel:** Name of the Channel (Channel name)
- **Address:** The memory address under which the information is stored in the memory of the AC500 V3 PLC. Inputs start with %I and outputs start with %Q.
- **Type:** Specification of the IEC 61131-3 variable type
- **Default Value:** The value used after starting the controller.
  - At a KNX Group Object input, this value is used by the automation program until a value has been received from the KNX bus.
  - At a KNX Group Object output, this value is sent to the bus when the controller is started.
- **Unit:** Specification of the KNX data point type (DPT)
- **Description:** Note text

A KNX Group Object “input” consists of a status and a control part:

The Channel name of the status part consists of: Object Number + Object Name + Object Function and include the following informations:

- **UpdateFlag:**
  This status flag is set to the value “true” for one cycle as soon as a new KNX telegram has been received. Even if the value of the telegram does not differ from the previous one.
- **ValueChanged:**
  This status flag is set to the value “true” as soon as a new KNX telegram has been received and the value differs from the previous one.
- **ValueValid:**
  This status flag is set to the value “true” as soon as a KNX telegram has been received for the first time after the controller has been started.
- **WatchdogTimeout:**
  As of Automation Builder 2.2.1 it will be possible to define a Watchdog Timeout for each input object. If a timeout occur this flag will be set to the value “true” for one cycle.

The Channel name of the control part consists of: “Control” + Object Number + Object Name + Object Function and include the following control possibilities:
● Reset status flags:
When this flag is set from “false” to “true” by the automation program then the above-mentioned status flags of the KNX Group Object are reset to the value “false”.

● Send value read:
When this flag is set from “false” to “true” by the automation program, a ValueRead telegram is sent to the KNX bus. This causes the KNX remote device to send back its current value.

A KNX Group Object “output” is represented as follows:
The Channel name of the Group Object consists of: Object Number + Object Name + Object Function

● Trigger Output:
When this flag is set from “false” to “true” by the automation program, the current value is sent immediately to the KNX bus. The sending conditions that are may be activated in the ETS (send on change and cyclic sending) will be restarted

● Disable Output:
As long this flag is set to “true” by the automation program, the sending conditions send on change and cyclic sending in the ETS are deactivated.

● Value:
The current value of the KNX Group Object that is sent to the KNX bus.

The permanently defined Program LED Status represent the function as known in other KNX devices, showing the status of the programming LED.

Create an application program
The KNX variables defined on the KNX I/O Mapping page are available programwide under IoConfig_Globals_Mapping.

These you can see if you click in to the programming window and either via right-click select “Input Assistant” or press F2.
Export XML file

To exchange the configured KNX group objects the configuration has to be exported via XML File.

If later both projects (from Automation Builder and ETS) are loaded on the PLC, the PLC checks if the two projects have the same source and fit together. This will be done by an automatically calculated Checksum. For calculating the Checksum the following information's from the Project information will be used:

- Company
- Title
- Version
- Timestamp

This Information will also be shown in the ETS after loading the XML File. If the user has not entered any project information some default values will be set.

Integration of the PLC in KNX

Insert controller

1. Start the ETS and insert the PS5604-KNX AC500 as controller from the ETS device catalog into your ETS project.

2. Assign a physical KNX address to the controller.
   - The controller is placed topologically on the IP Main Area.
Import configuration

1. Select the “PS5604-KNX AC500” in the ETS explorer tree and click on “DCA” tab in the editor window.
2. Click on Load Configuration and select the configuration .xml file.
   - The KNX group objects defined in AC500 V3 PLC in Automation Builder are displayed in the ETS.

Connect controller with KNX Devices

1. Right-click on a “PS5604-KNX AC500” group object and assign a KNX group address or drag and drop from group address window.
2. Interlink group objects by assigning the same KNX Group Address.
Parameters of the device

The following settings are possible in the ETS parameters of the PS5604-KNX AC500.

Tab General settings

- Default Gateway:
  The used KNXnet/IP broadcast address. This must match the KNX system (KNX IP router). It is the default setting that is usually not changed.
  - The entry field Default Gateway can contain another IP address for the Multicast communication. The normal Multicast IP address for KNX is 224.0.23.12.
  - If another Multicast IP address is to be used, it can be chosen in the area from 239.0.0.0 to 239.255.255.255. This alternative Multicast IP address can be defined in the input field Default Gateway.

- Telegram rate:
  The maximum transmission rate of the AC500 V3 PLC can be limited in order to prevent an excessive bus load and thus to avoid malfunction of the KNX system.
  The KNX telegrams are buffered until they have been sent. New values which have been calculated by the automation program in the meantime are updating the cached values. The old cached value is discarded and not sent.

- Project Information:
  At this point, the project information of the Automation Builder project is displayed.
For each KNX Group Object of the AC500 V3 PLC an Object entry is displayed in the device parameters. This is named after the number of the KNX Group Object.

For outputs (controller sends to the KNX bus) the KNX transmission conditions can be set:

- **Communication direction:**
  Setting of the transmission direction of the object.
  - Input (KNX to PLC): The Controller receives values from the KNX bus.
  - Output (PLC to KNX): The Controller sends values to the KNX bus.

- **Send condition (only for outputs):**
  Setting whether the Controller sends a telegram to the KNX bus automatically when the object value is changed. The following options are available
  - No automatic sending:
    No automatic sending to the KNX bus. This must be done via the program code by the Trigger Output flag.
  - Send on change:
    Every time the object value changes, a telegram is sent to the KNX bus. No matter how minor this change is.
  - Send on difference (only for group objects which are not DPT 1.* Boolean): Every time the object value changes, this value is only sent to the KNX bus if it differs from the last sent value at least by the settable difference.

- **Sending difference (only if Send on difference is active):**
  Input of the difference by which the object value must change to be send. You can enter numbers with decimal places.

- **Cyclic sending (only for outputs):**
  Setting whether in addition the object value is sent cyclically repeatedly to the bus. This also happens if this object value has not changed. Two different value ranges for the cycle time can be specified.

- **Cycle time (only when Cyclic sending is active):**
  Specification of the cycle time for the cyclic transmission.
  Input format hour:minute:second

Regardless of the set transmission conditions, the program code can trigger by the flag **Trigger Output** a sending of the value to the KNX bus at any time.

By activating the flag **Read on Init** of the KNX group objects in the right ETS properties panel, the Controller sends a value read query to the connected KNX device at startup. This then responds with its current object value.
In this properties panel you can also select the appropriate subdata point type of the KNX Group Object. This defines the unit of the value in the KNX system. For example DPT 9.001 represents temperature in °C.

If for example the response of an actuator state is needed for an input "Aktor A Status", this feature can be enabled in the parameter of the Switch Actuator (e.g. 1.1.6 SA/S4.10.1).
The current IP address as well as further information of the AC500 V3 PLC can be read via the ETS Device Info function. For this the physical KNX address is necessary. You can determine the address by the ETS function Programming Mode.

With the ETS function Group Monitor you can analyze the telegrams on the KNX bus. You can also use it to write/read KNX telegrams.

Download ETS configuration to controller

The download of the ETS configuration to the AC500 V3 PLC is done via the ETS function “Download” in the menu bar. This download happens via the KNX interface directly to the AC500 V3 PLC.

Best you select in ETS the network interface of the computer as the bus interface. Thus, a fast data exchange is possible and the data is not routed via the KNX TP bus.

At the first download, the physical KNX address of the controller is programmed. To do this, set the AC500 V3 PLC to KNX programming mode.

This can be done either via the display or functions inside the application program of the controller (e.g. connected to a Webvisu like done in the example program).

![Table and Diagram](image)

The AC500 V3 PLC must be in RUN mode.

1. Press the CFG function key.
   ⇒ Switch is OFF (S OFF) is displayed.

2. Press the Arrow Down function key
   ⇒ Pbut 0 is displayed.
   
   (Pbut is standing for programming button, the 0 (or the 1) showing the status of the programming LED (0=Off; 1 =ON))

3. Press the CFG function key.
   ⇒ The display shows Pbut 1 flashing.

Attention!
The activation of the KNX programming mode via the display only works with Automation Builder as of version 2.2.0.

Chapter 1.5.5.1.4.4.3.2 “Configuration CPU firmware SystemFW >=V3.2.0 and DisplayFW >=V4.1” on page 3127
4. Confirm this with the **OK** function key.
   ⇒ The display permanently shows *Pbut 1*. The AC500 V3 PLC is in KNX programming mode.

The AC500 V3 PLC automatically terminates the KNX programming mode after the programming of the physical KNX address.

Alternatively you can terminate the programming mode with *Pbut 0* by pressing the **CFG** function key.

You can exit the menu at any time with the **ESC** function key.

**Via AC500 application**

Please use the following variable for setting the KNX Program Button:

\[AC500_IoDrvKNX.GVL.IoDrvKNXCopyChannels.ProgramButton\]

The controller automatically terminates the programming mode after programming the physical address with the ETS.

The AC500 V3 PLC has then besides the Automation Builder configuration also the appropriate ETS configuration and starts its KNX communication.

Download all other linked KNX devices as well as the KNX IP routers. The ETS automatically creates the filter tables of the KNX IP routers so that the KNX telegrams are routed from the KNX TP lines to the IP line of the AC500 Controller.

**Make changes**

Changes can be made in the Automation Builder as well as in the ETS without the need for a change in the other software or the need for a new data exchange.

Only if changes are made to the KNX group objects in the Automation Builder, a data exchange with the ETS is again necessary. Afterwards, a download is required both in the Automation Builder and in the ETS. Only when these two configurations have been downloaded again to the AC500 V3 PLC, the KNX communication is in operation again.

The DCA detects changes to names and numbers of the KNX group objects when importing the configuration file in the ETS and keeps the already made settings and linked Group Addresses of these changed group objects.

**Remarks**

**KNX runtime license**

The standard V3 AC500 CPUs are shipped from the factory without firmware and need an installed PS5604-KNX runtime license for KNX operation in each CPU. The PS5604-KNX is a license document with activation code and needs to be purchased separately. The license can after a first download to one CPU also be transferred to another CPU via Automation Builder.

**Data conversion**

The KNX standard defines a big-endian byte order while the IEC 61131-3 is based on the little-endian byte order. Therefore, the controller automatically converts the data point types.

However, if you access the bits of the structured KNX data point types (DPT) for time, date (DPT 10.*, DPT 16.*, DPT 19.*) in your program code, you have to note the reverse byte order. Therefore, as of Automation Builder version 2.2.0, corresponding function libraries are available that provide conversion functions for these data point types.
KNXConvertFunctions
  Internal
    DPT10_IEC_to_KNX
    DPT10_KNX_to_IEC
    DPT16_IEC_to_KNX
    DPT16_KNX_to_IEC
    DPT19_IEC_to_KNX
    DPT19_KNX_to_IEC
1.5.5.1.8 Communication with Modbus RTU

Protocol description

The Modbus RTU protocol is implemented in the AC500 Processor Modules.

Modbus is a master-slave (client-server) protocol. The client sends a request to the server(s) and receives the response(s).

The Modbus operating mode of a serial interface is set in the PLC configuration Chapter 1.5.6.2.10.1 “Configuring Modbus RTU on serial interface” on page 3325.

Modbus client

In this operating mode, the telegram traffic with the server(s) is handled via the Function Block ModRtuMast.

This Function Block sends Modbus request telegrams to the server(s) via the set interface and receives Modbus response telegrams from the server(s) via this interface.

The Modbus blocks transferred by the server contain the following information:

- Modbus address of the interrogated server (1 byte)
- Function code that defines the request of the client (1 byte)
- Data to be exchanged (n bytes)
- CRC16 control code (2 bytes)

Modbus server

In this operating mode, no Function Block is required for Modbus communication. Sending and receiving Modbus telegrams is performed automatically.

The AC500 CPUs process the following Modbus operation codes:

<table>
<thead>
<tr>
<th>Function code</th>
<th>DEC</th>
<th>HEX</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>01 or 02</td>
<td>01 or 02</td>
<td>01 or 02</td>
<td>Read n bits</td>
</tr>
<tr>
<td>03 or 04</td>
<td>03 or 04</td>
<td>03 or 04</td>
<td>Read n words</td>
</tr>
<tr>
<td>05</td>
<td>05</td>
<td>05</td>
<td>Write one bit (encoded in one word)</td>
</tr>
<tr>
<td>06</td>
<td>06</td>
<td>06</td>
<td>Write one word</td>
</tr>
<tr>
<td>15</td>
<td>0F</td>
<td>0F</td>
<td>Write n bits (encoded in one byte)</td>
</tr>
<tr>
<td>16</td>
<td>10</td>
<td>10</td>
<td>Write n words</td>
</tr>
<tr>
<td>22</td>
<td>16</td>
<td>16</td>
<td>Mask write</td>
</tr>
<tr>
<td>23</td>
<td>17</td>
<td>17</td>
<td>Read/write multiple words in one telegram</td>
</tr>
</tbody>
</table>

The following restrictions apply to the length of the data to be sent:

<table>
<thead>
<tr>
<th>Function code</th>
<th>DEC</th>
<th>HEX</th>
<th>Max. length</th>
</tr>
</thead>
<tbody>
<tr>
<td>01 or 02</td>
<td>01 or 02</td>
<td>01 or 02</td>
<td>2000 bits</td>
</tr>
<tr>
<td>03 or 04</td>
<td>03 or 04</td>
<td>03 or 04</td>
<td>125 words / 62 double words</td>
</tr>
<tr>
<td>05</td>
<td>05</td>
<td>05</td>
<td>1 bit</td>
</tr>
<tr>
<td>06</td>
<td>06</td>
<td>06</td>
<td>1 word</td>
</tr>
</tbody>
</table>
### Function code

<table>
<thead>
<tr>
<th>DEC</th>
<th>HEX</th>
<th>Max. length</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>0F</td>
<td>2000 bits</td>
</tr>
<tr>
<td>16</td>
<td>10</td>
<td>123 words / 61 double words</td>
</tr>
<tr>
<td>22</td>
<td>16</td>
<td>Write: 1 word</td>
</tr>
<tr>
<td>23</td>
<td>17</td>
<td>Read: 125 words / 62 double words</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Write: 121 words / 60 double words</td>
</tr>
</tbody>
</table>

### Technical data

The Modbus operating mode and the interface parameters are set in the *PLC configuration* Chapter 1.5.6.2.10.1 “Configuring Modbus RTU on serial interface” on page 3325.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Supported standard</td>
<td>See <em>Serial interface</em> Chapter 1.5.6.2.10.1 “Configuring Modbus RTU on serial interface” on page 3325</td>
</tr>
<tr>
<td>Number of connection points</td>
<td>1 client</td>
</tr>
<tr>
<td></td>
<td>Max. 1 server with RS-232 interface</td>
</tr>
<tr>
<td></td>
<td>Max. 31 servers with RS-485</td>
</tr>
<tr>
<td>Protocol</td>
<td>Modbus</td>
</tr>
<tr>
<td>Operating mode</td>
<td>Client/server</td>
</tr>
<tr>
<td>Address</td>
<td>Server only</td>
</tr>
<tr>
<td>Data transmission control</td>
<td>CRC16</td>
</tr>
<tr>
<td>Data transmission speed</td>
<td>Up to 187.500 baud</td>
</tr>
<tr>
<td></td>
<td>(see <em>Serial interface</em> Chapter 1.5.6.2.10.1 “Configuring Modbus RTU on serial interface” on page 3325)</td>
</tr>
<tr>
<td>Encoding</td>
<td>1 start bit</td>
</tr>
<tr>
<td></td>
<td>8 data bits</td>
</tr>
<tr>
<td></td>
<td>1 or 2 stop bits</td>
</tr>
<tr>
<td></td>
<td>1 parity bit (see <em>Serial interface</em> Chapter 1.5.6.2.10.1 “Configuring Modbus RTU on serial interface” on page 3325)</td>
</tr>
<tr>
<td>Max. cable length for RS-485 on COM1 for AC500 CPU</td>
<td>1.200 m at 19,200 baud</td>
</tr>
</tbody>
</table>
# Modbus addresses for processor modules PM56xx

## Modbus address table

*Table 667: Modbus addresses (word accesses)*

<table>
<thead>
<tr>
<th>Modbus address</th>
<th>Byte</th>
<th>Bit (byte-oriented)</th>
<th>Word</th>
<th>Double word</th>
</tr>
</thead>
<tbody>
<tr>
<td>HEX</td>
<td>DEC</td>
<td>BYTE</td>
<td>BOOL</td>
<td>WORD</td>
</tr>
<tr>
<td>0000</td>
<td>0</td>
<td>%MB0</td>
<td>%MX0.0 ... %MX0.7</td>
<td>%MW0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>%MB1</td>
<td>%MX1.0 ... %MX1.7</td>
<td></td>
</tr>
<tr>
<td>0001</td>
<td>1</td>
<td>%MB2</td>
<td>%MX2.0 ... %MX2.7</td>
<td>%MW1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>%MB3</td>
<td>%MX3.0 ... %MX3.7</td>
<td></td>
</tr>
<tr>
<td>0002</td>
<td>2</td>
<td>%MB4</td>
<td>%MX4.0 ... %MX4.7</td>
<td>%MW2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>%MB5</td>
<td>%MX5.0 ... %MX5.7</td>
<td></td>
</tr>
<tr>
<td>0003</td>
<td>3</td>
<td>%MB6</td>
<td>%MX6.0 ... %MX6.7</td>
<td>%MW3</td>
</tr>
<tr>
<td></td>
<td></td>
<td>%MB7</td>
<td>%MX7.0 ... %MX7.7</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7FFE</td>
<td>3276</td>
<td>%MB65532</td>
<td>%MX65532.0 ... %MX65532.7</td>
<td>%MW3276</td>
</tr>
<tr>
<td></td>
<td></td>
<td>%MB65533</td>
<td>%MX65533.0 ... %MX65533.7</td>
<td></td>
</tr>
<tr>
<td>7FFF</td>
<td>3276</td>
<td>%MB65534</td>
<td>%MX65534.0 ... %MX65534.7</td>
<td>%MW3276</td>
</tr>
<tr>
<td></td>
<td></td>
<td>%MB65535</td>
<td>%MX65535.0 ... %MX65535.7</td>
<td></td>
</tr>
<tr>
<td>8000</td>
<td>3276</td>
<td>%MB65536</td>
<td>%MX65536.0 ... %MX65536.7</td>
<td>%MW3276</td>
</tr>
<tr>
<td></td>
<td></td>
<td>%MB65537</td>
<td>%MX65537.0 ... %MX65537.7</td>
<td></td>
</tr>
<tr>
<td>8001</td>
<td>3276</td>
<td>%MB65538</td>
<td>%MX65538.0 ... %MX65538.7</td>
<td>%MW3276</td>
</tr>
<tr>
<td></td>
<td></td>
<td>%MB65539</td>
<td>%MX65539.0 ... %MX65539.7</td>
<td></td>
</tr>
<tr>
<td>8002</td>
<td>3277</td>
<td>%MB65540</td>
<td>%MX65540.0 ... %MX65540.7</td>
<td>%MW32770</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Modbus address</td>
<td>Byte</td>
<td>Bit (byte-oriented)</td>
<td>Word</td>
<td>Double word</td>
</tr>
<tr>
<td>----------------</td>
<td>------</td>
<td>---------------------</td>
<td>------</td>
<td>-------------</td>
</tr>
<tr>
<td></td>
<td>HEX</td>
<td>BYTE</td>
<td>BOOL</td>
<td>WORD</td>
</tr>
<tr>
<td></td>
<td>DEC</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8003</td>
<td>32771</td>
<td>%MB65542</td>
<td></td>
<td>%MW32771</td>
</tr>
<tr>
<td></td>
<td></td>
<td>%MX65542.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>%MX65542.7</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>%MB65543</td>
<td></td>
<td>%MW65543</td>
</tr>
<tr>
<td></td>
<td></td>
<td>%MX65543.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>%MX65543.7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FFFE</td>
<td>65534</td>
<td>%MB131068</td>
<td></td>
<td>%MW65534</td>
</tr>
<tr>
<td></td>
<td></td>
<td>%MX131068.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>%MX131068.7</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>%MB131069</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>%MX131069.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>%MX131069.7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FFFF</td>
<td>65535</td>
<td>%MB131070</td>
<td></td>
<td>%MW65535</td>
</tr>
<tr>
<td></td>
<td></td>
<td>%MX131070.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>%MX131070.7</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>%MB131071</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>%MX131071.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>%MX131071.7</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 668: Address assignment (bit accesses)
<table>
<thead>
<tr>
<th>Modbus address</th>
<th>Byte</th>
<th>Bit (byte-oriented)</th>
<th>Word</th>
<th>Double word</th>
</tr>
</thead>
<tbody>
<tr>
<td>HEX</td>
<td>DEC</td>
<td>BOOL</td>
<td>WORD</td>
<td>DWORD</td>
</tr>
<tr>
<td>000E</td>
<td>14</td>
<td>%MX1.6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>000F</td>
<td>15</td>
<td>%MX1.7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0010</td>
<td>16</td>
<td>%MB2</td>
<td>%MW1</td>
<td></td>
</tr>
<tr>
<td>0011</td>
<td>17</td>
<td>%MX2.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0012</td>
<td>18</td>
<td>%MX2.1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0013</td>
<td>19</td>
<td>%MX2.2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0014</td>
<td>20</td>
<td>%MX2.3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0015</td>
<td>21</td>
<td>%MX2.4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0016</td>
<td>22</td>
<td>%MX2.5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0017</td>
<td>23</td>
<td>%MX2.6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0018</td>
<td>24</td>
<td>%MB3</td>
<td>%MX3.0</td>
<td></td>
</tr>
<tr>
<td>0019</td>
<td>25</td>
<td>%MX3.1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>001A</td>
<td>26</td>
<td>%MX3.2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>001B</td>
<td>27</td>
<td>%MX3.3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>001C</td>
<td>28</td>
<td>%MX3.4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>001D</td>
<td>29</td>
<td>%MX3.5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>001E</td>
<td>30</td>
<td>%MX3.6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>001F</td>
<td>31</td>
<td>%MX3.7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0020</td>
<td>32</td>
<td>%MB4</td>
<td>%MX4.0</td>
<td>%MW2</td>
</tr>
<tr>
<td>0021</td>
<td>33</td>
<td>%MX4.1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0022</td>
<td>34</td>
<td>%MX4.2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td></td>
</tr>
<tr>
<td>0FFF</td>
<td>4095</td>
<td>%MB511</td>
<td>%MX511.7</td>
<td>%MW255</td>
</tr>
<tr>
<td>1000</td>
<td>4096</td>
<td>%MB512</td>
<td>%MX512.0</td>
<td>%MW256</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td></td>
</tr>
<tr>
<td>7FFF</td>
<td>32767</td>
<td>%MB4095</td>
<td>%MX4095.7</td>
<td>%MW2047</td>
</tr>
<tr>
<td>8000</td>
<td>32768</td>
<td>%MB4096</td>
<td>%MX4096.0</td>
<td>%MW2048</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td></td>
</tr>
<tr>
<td>FFFF</td>
<td>65535</td>
<td>%MB8191</td>
<td>%MX8191.7</td>
<td>%MW4095</td>
</tr>
</tbody>
</table>

Calculation of the bit variable from the hexadecimal address:

**Formula:**

\[
\text{Bit variable (BOOL)} := \%\text{MX}_\text{BYTE}.\text{BIT}
\]

where:

<table>
<thead>
<tr>
<th>DEC</th>
<th>Decimal address</th>
</tr>
</thead>
<tbody>
<tr>
<td>BYTE</td>
<td>DEC / 8</td>
</tr>
<tr>
<td>BIT</td>
<td>DEC mod 8 (Modulo division)</td>
</tr>
</tbody>
</table>

**3ADR010583, 1, en_US**

2020/12/10
Examples:
- Address hexadecimal = 16#2002
  DEC := 8194
  BYTE := 8194 / 8 := 1024
  BIT := 8194 mod 8 := 2
  Bit variable: %MX1024.2
- Address hexadecimal = 16#3016
  DEC := 12310
  BYTE := 12310 / 8 := 1538,75 -> 1538
  BIT := 12310 mod 8 := 6
  Bit variable: %MX1538.6
- Address hexadecimal = 16#55AA
  DEC := 21930
  BYTE := 21930 / 8 := 2741,25 -> 2741
  BIT := 21930 mod 8 := 2
  Bit variable: %MX2741.2

Examples:
- Bit variable := %MX515.4
  DEC := 515 * 8 + 4 := 4124
  Address hex := 16#101C
- Bit variable := %MX3.3
  DEC := 3 * 8 + 3 := 27
  Address hex := 16#001B
- Bit variable := %MX6666.2
  DEC := 6666 * 8 + 2 := 53330
  Address hex := 16#D052

Peculiarities for accessing Modbus addresses

Peculiarities for bit access:
- A WORD in the %M area is assigned to each Modbus address 0000hex .. FFFFhex.
- Bit addresses 0000hex .. FFFFhex are contained in the word range %MW0 .. %MW4095

Areas protect from read/write access by Modbus client

As described in Configuration of Modbus TCP/IP Server - parameter disable, one write-protected and one read-protected area can be defined. As described in Chapter 1.5.6.3.2.1.1 “Configuration of Modbus TCP/IP server” on page 3422. If you try to write to a write-protected area or to read from a read-protected area, an exception response is generated.

Local data of the Modbus client

The address of the area from which data are to be read or to which data are to be written is specified in the Function Block ModRtuMast at input "Data", via the ADR operator.

For the AC500, the following areas can be accessed using the ADR operator:
- Inputs area (%I area)
- Outputs area (%Q area)
- Area of non-buffered variables (VAR .. END_VAR or VAR_GLOBAL END_VAR)
- Addressable flag area (also protected areas for %M area)
- Area of buffered variables (VAR RETAIN .. END_VAR or VAR_GLOBAL RETAIN .. END_VAR)

Modbus telegrams

The send and receive of telegrams shown in this section are not visible in the PLC. However, the complete telegrams can be made visible using a serial data analyzer connected to the connection line between server and client, if required.

The amount of user data depends on the capabilities of the server and the client.

For the following examples, it is assumed that one AC500 Modbus module is used as client and another one is used as server. There may be different properties if modules of other manufacturers are used.

**FCT 1 or 2: Read n bits**

Table 669: Client request

<table>
<thead>
<tr>
<th>Server address</th>
<th>Function code</th>
<th>Server operand address</th>
<th>Number of bits</th>
<th>CRC</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>High</td>
<td>Low</td>
<td>High</td>
</tr>
</tbody>
</table>

Table 670: Server response

<table>
<thead>
<tr>
<th>Server address</th>
<th>Function code</th>
<th>Number of Bytes</th>
<th>Data...</th>
<th>CRC</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>High</td>
</tr>
</tbody>
</table>

**FCT 3 or 4: Read n words**

Table 671: Client request

<table>
<thead>
<tr>
<th>Server address</th>
<th>Function code</th>
<th>Server operand address</th>
<th>Number of words</th>
<th>CRC</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>High</td>
<td>Low</td>
<td>High</td>
</tr>
</tbody>
</table>

Table 672: Server response

<table>
<thead>
<tr>
<th>Server address</th>
<th>Function code</th>
<th>Number of Bytes</th>
<th>Data</th>
<th>CRC</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>High</td>
</tr>
</tbody>
</table>

**FCT 3 or 4: Read n double words**

The function code "read double word" is not defined in the Modbus RTU standard. This is why the double word is composed of a low word and a high word (depending on the manufacturer)

Same tables as [Chapter 1.5.5.1.8.5.2 “FCT 3 or 4: Read n words” on page 3172](#).

**FCT 5: Write 1 bit**

For the function code "write 1 bit", the value of the bit to be written is encoded in one word.
BIT = TRUE -> Data word = FF 00 HEX  
BIT = FALSE -> Data word = 00 00 HEX

**Table 673: Client request**

<table>
<thead>
<tr>
<th>Function code</th>
<th>Server operand address</th>
<th>Number of words</th>
<th>CRC</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>High</td>
<td>Low</td>
<td></td>
</tr>
<tr>
<td></td>
<td>High</td>
<td>Low</td>
<td></td>
</tr>
</tbody>
</table>

**Table 674: Server response**

<table>
<thead>
<tr>
<th>Function code</th>
<th>Server operand address</th>
<th>Data</th>
<th>CRC</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>High</td>
<td>Low</td>
<td></td>
</tr>
<tr>
<td></td>
<td>High</td>
<td>Low</td>
<td></td>
</tr>
</tbody>
</table>

FCT 6: Write 1 word

**Table 675: Server request**

<table>
<thead>
<tr>
<th>Server address</th>
<th>Function code</th>
<th>Server operand address</th>
<th>Data</th>
<th>CRC</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>High</td>
<td>Low</td>
<td>High</td>
<td>Low</td>
</tr>
</tbody>
</table>

**Table 676: Server response**

<table>
<thead>
<tr>
<th>Server address</th>
<th>Function code</th>
<th>Server operand address</th>
<th>Data</th>
<th>CRC</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>High</td>
<td>Low</td>
<td>High</td>
<td>Low</td>
</tr>
</tbody>
</table>

FCT 15: Write n bits

**Table 677: Client request**

<table>
<thead>
<tr>
<th>Server operand address</th>
<th>Number of bits</th>
<th>Number of bytes</th>
<th>...Data...</th>
<th>CRC</th>
</tr>
</thead>
<tbody>
<tr>
<td>High</td>
<td>Low</td>
<td>High</td>
<td>Low</td>
<td></td>
</tr>
</tbody>
</table>

**Table 678: Server response**

<table>
<thead>
<tr>
<th>Server address</th>
<th>Function code</th>
<th>Server operand address</th>
<th>Number of bits</th>
<th>CRC</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>High</td>
<td>Low</td>
<td>High</td>
<td>Low</td>
</tr>
</tbody>
</table>

FCT 16: Write n words

**Table 679: Client request**

<table>
<thead>
<tr>
<th>Server operand address</th>
<th>Number of words</th>
<th>Number of bytes</th>
<th>...Data...</th>
<th>CRC</th>
</tr>
</thead>
<tbody>
<tr>
<td>High</td>
<td>Low</td>
<td>High</td>
<td>Low</td>
<td></td>
</tr>
</tbody>
</table>
**FCT 16: Write n double words**

The function code "write double word" is not defined in the Modbus RTU standard. This is why the double word is composed of a low word and a high word (depending on the manufacturer).

**Table 680: Server response**

<table>
<thead>
<tr>
<th>Function code</th>
<th>Server operand address</th>
<th>Number of words</th>
<th>CRC</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>High</td>
<td>Low</td>
<td>High</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Function code</th>
<th>Server operand address</th>
<th>Number of words</th>
<th>CRC</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>High</td>
<td>Low</td>
<td>High</td>
</tr>
</tbody>
</table>

**Table 681: Client request**

<table>
<thead>
<tr>
<th>Server operand address</th>
<th>Number of words</th>
<th>Number of bytes</th>
<th>...Data...</th>
<th>CRC</th>
</tr>
</thead>
<tbody>
<tr>
<td>High</td>
<td>Low</td>
<td>High</td>
<td>Low</td>
<td>High</td>
</tr>
</tbody>
</table>

**Table 682: Server response**

<table>
<thead>
<tr>
<th>Server address</th>
<th>Function code</th>
<th>Server operand address</th>
<th>Number of words</th>
<th>CRC</th>
</tr>
</thead>
<tbody>
<tr>
<td>High</td>
<td>Low</td>
<td>High</td>
<td>Low</td>
<td>High</td>
</tr>
</tbody>
</table>

**FCT 22: Mask write register**

**Table 683: Client request**

<table>
<thead>
<tr>
<th>Server address</th>
<th>Function code</th>
<th>Server operand address</th>
<th>AND Mask</th>
<th>OR Mask</th>
<th>CRC</th>
</tr>
</thead>
<tbody>
<tr>
<td>High</td>
<td>Low</td>
<td>High</td>
<td>Low</td>
<td>High</td>
<td>Low</td>
</tr>
</tbody>
</table>

**Table 684: Server response**

<table>
<thead>
<tr>
<th>Server address</th>
<th>Function code</th>
<th>Server operand address</th>
<th>AND Mask</th>
<th>OR Mask</th>
<th>CRC</th>
</tr>
</thead>
<tbody>
<tr>
<td>High</td>
<td>Low</td>
<td>High</td>
<td>Low</td>
<td>High</td>
<td>Low</td>
</tr>
</tbody>
</table>

**FCT 23: Read/Write n words**

**Table 685: Client request**

<table>
<thead>
<tr>
<th>Server address</th>
<th>Function code</th>
<th>Operand addr. read</th>
<th>Number of words</th>
<th>...Data...</th>
<th>CRC</th>
</tr>
</thead>
<tbody>
<tr>
<td>High</td>
<td>Low</td>
<td>High</td>
<td>Low</td>
<td>High</td>
<td>Low</td>
</tr>
</tbody>
</table>

**Table 686: Server response**

<table>
<thead>
<tr>
<th>Server address</th>
<th>Function code</th>
<th>Number of bytes read</th>
<th>...Data...</th>
<th>CRC</th>
</tr>
</thead>
<tbody>
<tr>
<td>High</td>
<td>Low</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Exception response by server

In operating mode Modbus client, the AC500 does only send requests, if the parameters at the ModRtuMast inputs are logically correct.

Nevertheless, it can happen that a server cannot process the request of the client or that the server cannot interpret the request due to transmission errors or in case it’s capabilities are exceeded in any way. In those cases, the server returns an exception response to the client. In order to identify this response as an exception response, the function code returned by the server is a logical OR interconnection of the function code received from the client and the value 80HEX.

Table 687: Server response

<table>
<thead>
<tr>
<th>Server address</th>
<th>OR 80HEX</th>
<th>Error code</th>
<th>CRC</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>High</td>
</tr>
</tbody>
</table>

Possible error codes of the client

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>01DEC</td>
<td>ILLEGAL FUNCTION</td>
</tr>
<tr>
<td></td>
<td>The server does not support the function requested by the client</td>
</tr>
<tr>
<td>02DEC</td>
<td>ILLEGAL DATA ADDRESS</td>
</tr>
<tr>
<td></td>
<td>Invalid operand address in the server or operand area exceeded</td>
</tr>
<tr>
<td>03DEC</td>
<td>ILLEGAL DATA VALUE</td>
</tr>
<tr>
<td></td>
<td>At least one value is outside the permitted value range</td>
</tr>
<tr>
<td>04DEC</td>
<td>SERVER DEVICE FAILURE</td>
</tr>
<tr>
<td></td>
<td>An unrecoverable error occurred while the server was attempting to perform the requested action</td>
</tr>
<tr>
<td>05DEC</td>
<td>ACKNOWLEDGE</td>
</tr>
<tr>
<td></td>
<td>Specialized use in conjunction with programming commands.</td>
</tr>
<tr>
<td></td>
<td>The server has accepted the request and is processing it, but a long duration of time will be required to do so. This response is returned to prevent a timeout error from occurring in the client. The client can next issue a Poll Program Complete message to determine if processing is completed</td>
</tr>
<tr>
<td>06DEC</td>
<td>SERVER DEVICE BUSY</td>
</tr>
<tr>
<td></td>
<td>Specialized use in conjunction with programming commands.</td>
</tr>
<tr>
<td></td>
<td>The server is engaged in processing a long-duration program command. The client should retransmit the message later when the server is free.</td>
</tr>
<tr>
<td>07DEC</td>
<td>NEGATIVE ACKNOWLEDGE</td>
</tr>
<tr>
<td></td>
<td>Specialized use in conjunction with programming commands.</td>
</tr>
<tr>
<td></td>
<td>The server cannot perform the programming functions. Client should request diagnostic or error information from server.</td>
</tr>
<tr>
<td>08DEC</td>
<td>MEMORY PARITY ERROR</td>
</tr>
<tr>
<td></td>
<td>Specialized use in conjunction with function codes 20 and 21 and reference type 6, to indicate that the extended file area failed to pass a consistency check. The server attempted to read record file, but detected a parity error in the memory. The client can retry the request, but service may be required on the server device.</td>
</tr>
<tr>
<td>Code</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>--------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>10DEC</td>
<td>GATEWAY PATH UNAVAILABLE</td>
</tr>
<tr>
<td></td>
<td>Specialized use in conjunction with gateways, indicates that the gateway was unable to allocate an</td>
</tr>
<tr>
<td></td>
<td>internal communication path from the input port to the output port for processing the request.</td>
</tr>
<tr>
<td></td>
<td>Usually means that the gateway is misconfigured or overloaded.</td>
</tr>
<tr>
<td>11DEC</td>
<td>GATEWAY TARGET DEVICE FAILED TO RESPOND</td>
</tr>
<tr>
<td></td>
<td>Specialized use in conjunction with gateways, indicates that no response was obtained from the target</td>
</tr>
<tr>
<td></td>
<td>device. Usually means that the device is not present on the network.</td>
</tr>
</tbody>
</table>

**Example**

Table 688: Example:

<table>
<thead>
<tr>
<th>Modbus request of the client:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Function code: 01</td>
</tr>
<tr>
<td>Server operand address: 4000HEX = 16384DEC</td>
</tr>
<tr>
<td>Area for read access disabled in server</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Modbus response of the server:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Function code: 81HEX</td>
</tr>
<tr>
<td>Error code: 03</td>
</tr>
</tbody>
</table>

**Processing bits**

Some of the Modbus function codes are used to read or write bits (coils, discrete inputs). While a variable of data type WORD can be accessed easily, accessing a stream of bits is complex.

Data type BIT must not be mixed up with data type BOOL. Variables of both types may have values 'TRUE' or 'FALSE'. But while BIT means one single bit only, BOOL requires a byte (8 bit) of memory.

**Modbus client**

When accessing bits in a Server, the local data referred to at Client Function Blocks input data is always expected to be of format BOOL.
Modbus server

Using the bit offset

The simplest way to access a certain bit within a larger variable is to directly use the bit offset (0 based; see Bit Access to Variables © Chapter 1.3.1.24.4.11 “Bit access to variables” on page 565).
Defining symbolic names for the bit offsets

A more convenient way to access bits e.g. within a word is to define a symbolic name for each single offset (see Bit Access to a variable by using a global constant © Chapter 1.3.1.24.4.11 “Bit access to variables” on page 565).

```
VAR_GLOBAL_CONSTANT
("bit offsets in awWordVariable[0]*
NameBitOffset_0_00 : INT := 0;
NameBitOffset_0_01 : INT := 1;
(*...*)
NameBitOffset_0_15 : INT := 15;
("bit offsets in awWordVariable[7]*
NameBitOffset_7_00 : INT := 0;
NameBitOffset_7_01 : INT := 1;
(*...*)
NameBitOffset_7_15 : INT := 15;
END_VAR

VAR
awWordVariable AT %MW0 : ARRAY [0..7] OF WORD;
END_VAR

awWordVariable[0].NameBitOffset_0_00 := TRUE;
awWordVariable[7].NameBitOffset_7_15 := FALSE;
```
Defining a data type

A further alternative is to define your own data type according to the requirements of your particular application (see Chapter 1.3.1.25.1.14 "Object ‘DUT’" on page 735). Bit access to BIT data types (see Chapter 1.3.1.24.4.11 "Bit access to variables" on page 565).

```plaintext
TYPE ControlWordType :
  STRUCT
    NameBitOffset_00 : BIT;
    NameBitOffset_01 : BIT;
    NameBitOffset_02 : BIT;
    NameBitOffset_03 : BIT;
    NameBitOffset_04 : BIT;
    NameBitOffset_05 : BIT;
    NameBitOffset_06 : BIT;
    NameBitOffset_07 : BIT;
    NameBitOffset_08 : BIT;
    NameBitOffset_09 : BIT;
    NameBitOffset_10 : BIT;
    NameBitOffset_11 : BIT;
    NameBitOffset_12 : BIT;
    NameBitOffset_13 : BIT;
    NameBitOffset_14 : BIT;
    NameBitOffset_15 : BIT;
  END_STRUCT
END_TYPE

VAR
atTypeVariable AT §M60 : ARRAY [0..7] OF ControlWordType;
END_VAR

atTypeVariable[0].NameBitOffset_00 := TRUE;
atTypeVariable[7].NameBitOffset_15 := FALSE;
```
Defining a complex data type

In case your application requires some more complex data types you can combine data types (DUT; see Bit access to BIT data types Chapter 1.3.1.24.4.11 “Bit access to variables” on page 565).

```plaintext
TYPE ControlWord0Type :
  STRUCT
  NameBitOffset_0_00 : BIT;
  NameBitOffset_0_01 : BIT;
  NameBitOffset_0_02 : BIT;
  NameBitOffset_0_03 : BIT;
  NameBitOffset_0_04 : BIT;
  NameBitOffset_0_05 : BIT;
  NameBitOffset_0_06 : BIT;
  NameBitOffset_0_07 : BIT;
  NameBitOffset_0_08 : BIT;
  NameBitOffset_0_09 : BIT;
  NameBitOffset_0_10 : BIT;
  NameBitOffset_0_11 : BIT;
  NameBitOffset_0_12 : BIT;
  NameBitOffset_0_13 : BIT;
  NameBitOffset_0_14 : BIT;
  NameBitOffset_0_15 : BIT;
END_STRUCT
END_TYPE

TYPE ControlWord1Type :
  STRUCT
  NameBitOffset_1_00 : BIT;
  NameBitOffset_1_01 : BIT;
  NameBitOffset_1_02 : BIT;
  NameBitOffset_1_03 : BIT;
  NameBitOffset_1_04 : BIT;
  NameBitOffset_1_05 : BIT;
  NameBitOffset_1_06 : BIT;
  NameBitOffset_1_07 : BIT;
  NameBitOffset_1_08 : BIT;
  NameBitOffset_1_09 : BIT;
  NameBitOffset_1_10 : BIT;
  NameBitOffset_1_11 : BIT;
  NameBitOffset_1_12 : BIT;
  NameBitOffset_1_13 : BIT;
  NameBitOffset_1_14 : BIT;
  NameBitOffset_1_15 : BIT;
END_STRUCT
END_TYPE

TYPE ControlWordListType :
  STRUCT
  ControlWord0 : ControlWord0Type;
  ControlWord1 : ControlWord1Type;
END_STRUCT
END_TYPE

VAR
  atWordListType AT @MSW0 : ControlWordListType;
END_VAR

atWordListType.ControlWord0.NameBitOffset_0_00 := TRUE;
atWordListType.ControlWord1.NameBitOffset_1_15 := FALSE;
```
Pack/unpack BOOL variables

In case you prefer variables of type BOOL you can use the functions for **packing** and **unpacking**.

```plaintext
VAR
abBoolVariable : ARRAY [0..15] OF BOOL;
wWordVariable0 : AT WORD;
wWordVariable1 : AT WORD;
END_VAR

VAR
Unpack : Mem.UnpackWord;
END_VAR

abBoolVariable[0] := TRUE;
abBoolVariable[15] := FALSE;

wWordVariable0 := Mem.PackBitsToWord(abBoolVariable[0], abBoolVariable[1], abBoolVariable[2], abBoolVariable[3],
abBoolVariable[4], abBoolVariable[5], abBoolVariable[6], abBoolVariable[7],
abBoolVariable[8], abBoolVariable[9], abBoolVariable[10], abBoolVariable[11],
abBoolVariable[12], abBoolVariable[13], abBoolVariable[14], abBoolVariable[15]);

wWordVariable1 := 1;
Unpack(wValue:= wWordVariable1,
    xBit0 => abBoolVariable[0], xBit1 => abBoolVariable[1], xBit2 => abBoolVariable[2], xBit3 => abBoolVariable[3],
    xBit4 => abBoolVariable[4], xBit5 => abBoolVariable[5], xBit6 => abBoolVariable[6], xBit7 => abBoolVariable[7],
    xBit8 => abBoolVariable[8], xBit9 => abBoolVariable[9], xBit10 => abBoolVariable[10], xBit11 => abBoolVariable[11],
    xBit12 => abBoolVariable[12], xBit13 => abBoolVariable[13], xBit14 => abBoolVariable[14], xBit15 => abBoolVariable[15]);
```

Function block ModRtuMast

This Function Block is only required in the operating mode Modbus client. It handles the communication (transmission of telegrams to the servers and receipt of telegrams from the servers). The Function Block can be used for the local serial interfaces of the controller. A separate instance of the Function Block has to be used for each interface.

*ModRtuMast* is contained in the library *AC500_ModRtuMast*. 
1.5.5.1.9 Communication with Modbus TCP/IP

Protocol description

The Modbus TCP protocol is implemented in the AC500 Processor Modules.

Modbus is a master-slave (client-server) protocol. The client sends a request to the server(s) and receives the response(s).

Each Ethernet interface can work as Modbus client and server interface in parallel if required.

The Modbus operating mode of an Ethernet interface is set in Modbus on TCP/IP.

Modbus client

In this operating mode, the telegram traffic with the server(s) is handled via the Function Block ETHx_MOD_MAST. This Function Block sends Modbus request telegrams to the server(s) via the set interface and receives Modbus response telegrams from the server(s) via this interface.

The Modbus Function Blocks transferred by the client contain the following information:

- Transaction identifier for synchronization between messages of server and client (2 byte)
- Protocol identifier (0 for Modbus/TCP) (2 byte)
- Length field (Number of bytes in frame) (2 byte)
- Unit identifier (1 byte)
- Function code that defines the request of the client (1 byte)
- Data to be exchanged (n bytes)

In this operating mode, no Function Block is required for Modbus communication. Sending and receiving Modbus telegrams is performed automatically.

The AC500 CPUs process the following Modbus operation codes:

<table>
<thead>
<tr>
<th>Function code</th>
<th>HEX</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>01 or 02</td>
<td>01 or 02</td>
<td>Read n bits</td>
</tr>
<tr>
<td>03 or 04</td>
<td>03 or 04</td>
<td>Read n words</td>
</tr>
<tr>
<td>05</td>
<td>05</td>
<td>Write one bit (encoded in one word)</td>
</tr>
<tr>
<td>06</td>
<td>06</td>
<td>Write one word</td>
</tr>
<tr>
<td>15</td>
<td>0F</td>
<td>Write n bits (encoded in one byte)</td>
</tr>
<tr>
<td>16</td>
<td>10</td>
<td>Write n words</td>
</tr>
<tr>
<td>22</td>
<td>16</td>
<td>Mask write</td>
</tr>
<tr>
<td>23</td>
<td>17</td>
<td>Read/write multiple words in one telegram</td>
</tr>
</tbody>
</table>

The following restrictions apply to the length of the data to be sent:

<table>
<thead>
<tr>
<th>Function code</th>
<th>HEX</th>
<th>Max. length</th>
</tr>
</thead>
<tbody>
<tr>
<td>01 or 02</td>
<td>01 or 02</td>
<td>2000 bits</td>
</tr>
<tr>
<td>03 or 04</td>
<td>03 or 04</td>
<td>125 words / 62 double words</td>
</tr>
<tr>
<td>05</td>
<td>05</td>
<td>1 bit</td>
</tr>
<tr>
<td>Function code</td>
<td>DEC</td>
<td>HEX</td>
</tr>
<tr>
<td>---------------</td>
<td>-----</td>
<td>-----</td>
</tr>
<tr>
<td>06</td>
<td>06</td>
<td>06</td>
</tr>
<tr>
<td>15</td>
<td>0F</td>
<td>15</td>
</tr>
<tr>
<td>16</td>
<td>10</td>
<td>16</td>
</tr>
<tr>
<td>22</td>
<td>16</td>
<td>17</td>
</tr>
<tr>
<td>23</td>
<td>17</td>
<td>17</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Technical data**

Configuration of Modbus on TCP/IP is described in the chapter *Modbus on TCP/IP*.

**Modbus addresses for processor modules PM56xx**

**Modbus address table**

*Table 689: Modbus addresses (word accesses)*

<table>
<thead>
<tr>
<th>Modbus address</th>
<th>Byte</th>
<th>Bit (byte-oriented)</th>
<th>Word</th>
<th>Double word</th>
</tr>
</thead>
<tbody>
<tr>
<td>HEX DEC</td>
<td>BYTE</td>
<td>BOOL</td>
<td>WORD</td>
<td>DWORD</td>
</tr>
<tr>
<td>0000 0</td>
<td>%MB0</td>
<td>%MX0.0 ... %MX0.7</td>
<td>%MW0</td>
<td>%MD0</td>
</tr>
<tr>
<td></td>
<td>%MB1</td>
<td>%MX1.0 ... %MX1.7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0001 1</td>
<td>%MB2</td>
<td>%MX2.0 ... %MX2.7</td>
<td>%MW1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>%MB3</td>
<td>%MX3.0 ... %MX3.7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0002 2</td>
<td>%MB4</td>
<td>%MX4.0 ... %MX4.7</td>
<td>%MW2</td>
<td>%MD1</td>
</tr>
<tr>
<td></td>
<td>%MB5</td>
<td>%MX5.0 ... %MX5.7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0003 3</td>
<td>%MB6</td>
<td>%MX6.0 ... %MX6.7</td>
<td>%MW3</td>
<td></td>
</tr>
<tr>
<td></td>
<td>%MB7</td>
<td>%MX7.0 ... %MX7.7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7FFE 32766</td>
<td>%MB65532</td>
<td>%MX65532.0 ... %MX65532.7</td>
<td>%MW32766</td>
<td>%MD16383</td>
</tr>
<tr>
<td></td>
<td>%MB65533</td>
<td>%MX65533.0 ... %MX65533.7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7FFF 32767</td>
<td>%MB65534</td>
<td>%MX65534.0 ... %MX65534.7</td>
<td>%MW32767</td>
<td></td>
</tr>
<tr>
<td>Modbus address</td>
<td>Byte</td>
<td>Bit (byte-oriented)</td>
<td>Word</td>
<td>Double word</td>
</tr>
<tr>
<td>----------------</td>
<td>------</td>
<td>---------------------</td>
<td>------</td>
<td>-------------</td>
</tr>
<tr>
<td></td>
<td>HEX</td>
<td>DEC</td>
<td>BOOL</td>
<td>WORD</td>
</tr>
<tr>
<td></td>
<td>%MB65535</td>
<td>%MX65535.0</td>
<td>%MW32768</td>
<td>%MD16384</td>
</tr>
<tr>
<td>8000</td>
<td>32768</td>
<td>%MB65536</td>
<td>%MX65536.0</td>
<td>%MW32769</td>
</tr>
<tr>
<td></td>
<td></td>
<td>%MB65537</td>
<td>%MX65537.0</td>
<td></td>
</tr>
<tr>
<td>8001</td>
<td>32769</td>
<td>%MB65538</td>
<td>%MX65538.0</td>
<td>%MW32770</td>
</tr>
<tr>
<td></td>
<td></td>
<td>%MB65539</td>
<td>%MX65539.0</td>
<td></td>
</tr>
<tr>
<td>8002</td>
<td>32770</td>
<td>%MB65540</td>
<td>%MX65540.0</td>
<td>%MW32771</td>
</tr>
<tr>
<td></td>
<td></td>
<td>%MB65541</td>
<td>%MX65541.0</td>
<td></td>
</tr>
<tr>
<td>8003</td>
<td>32771</td>
<td>%MB65542</td>
<td>%MX65542.0</td>
<td>%MW32772</td>
</tr>
<tr>
<td></td>
<td></td>
<td>%MB65543</td>
<td>%MX65543.0</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
<td>%MB131068</td>
<td>%MX131068.0</td>
<td>%MW65534</td>
</tr>
<tr>
<td></td>
<td></td>
<td>%MB131069</td>
<td>%MX131069.0</td>
<td></td>
</tr>
<tr>
<td>FFFE</td>
<td>65534</td>
<td>%MB131070</td>
<td>%MX131070.0</td>
<td>%MW65535</td>
</tr>
<tr>
<td></td>
<td></td>
<td>%MB131071</td>
<td>%MX131071.0</td>
<td></td>
</tr>
</tbody>
</table>
Table 690: Address assignment (bit accesses)

<table>
<thead>
<tr>
<th>Modbus address</th>
<th>Byte</th>
<th>Bit (byte-oriented)</th>
<th>Word</th>
<th>Double word</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>HEX</td>
<td>BYTE</td>
<td>BOOL</td>
<td>WORD</td>
</tr>
<tr>
<td>0000</td>
<td>0</td>
<td>%MB0</td>
<td>%MX0.0</td>
<td>%MW0</td>
</tr>
<tr>
<td>0001</td>
<td>1</td>
<td></td>
<td>%MX0.1</td>
<td></td>
</tr>
<tr>
<td>0002</td>
<td>2</td>
<td></td>
<td>%MX0.2</td>
<td></td>
</tr>
<tr>
<td>0003</td>
<td>3</td>
<td></td>
<td>%MX0.3</td>
<td></td>
</tr>
<tr>
<td>0004</td>
<td>4</td>
<td></td>
<td>%MX0.4</td>
<td></td>
</tr>
<tr>
<td>0005</td>
<td>5</td>
<td></td>
<td>%MX0.5</td>
<td></td>
</tr>
<tr>
<td>0006</td>
<td>6</td>
<td></td>
<td>%MX0.6</td>
<td></td>
</tr>
<tr>
<td>0007</td>
<td>7</td>
<td></td>
<td>%MX0.7</td>
<td></td>
</tr>
<tr>
<td>0008</td>
<td>8</td>
<td>%MB1</td>
<td>%MX1.0</td>
<td></td>
</tr>
<tr>
<td>0009</td>
<td>9</td>
<td></td>
<td>%MX1.1</td>
<td></td>
</tr>
<tr>
<td>000A</td>
<td>10</td>
<td></td>
<td>%MX1.2</td>
<td></td>
</tr>
<tr>
<td>000B</td>
<td>11</td>
<td></td>
<td>%MX1.3</td>
<td></td>
</tr>
<tr>
<td>000C</td>
<td>12</td>
<td></td>
<td>%MX1.4</td>
<td></td>
</tr>
<tr>
<td>000D</td>
<td>13</td>
<td></td>
<td>%MX1.5</td>
<td></td>
</tr>
<tr>
<td>000E</td>
<td>14</td>
<td></td>
<td>%MX1.6</td>
<td></td>
</tr>
<tr>
<td>000F</td>
<td>15</td>
<td></td>
<td>%MX1.7</td>
<td></td>
</tr>
<tr>
<td>0010</td>
<td>16</td>
<td>%MB2</td>
<td>%MX2.0</td>
<td>%MW1</td>
</tr>
<tr>
<td>0011</td>
<td>17</td>
<td></td>
<td>%MX2.1</td>
<td></td>
</tr>
<tr>
<td>0012</td>
<td>18</td>
<td></td>
<td>%MX2.2</td>
<td></td>
</tr>
<tr>
<td>0013</td>
<td>19</td>
<td></td>
<td>%MX2.3</td>
<td></td>
</tr>
<tr>
<td>0014</td>
<td>20</td>
<td></td>
<td>%MX2.4</td>
<td></td>
</tr>
<tr>
<td>0015</td>
<td>21</td>
<td></td>
<td>%MX2.5</td>
<td></td>
</tr>
<tr>
<td>0016</td>
<td>22</td>
<td></td>
<td>%MX2.6</td>
<td></td>
</tr>
<tr>
<td>0017</td>
<td>23</td>
<td></td>
<td>%MX2.7</td>
<td></td>
</tr>
<tr>
<td>0018</td>
<td>24</td>
<td>%MB3</td>
<td>%MX3.0</td>
<td></td>
</tr>
<tr>
<td>0019</td>
<td>25</td>
<td></td>
<td>%MX3.1</td>
<td></td>
</tr>
<tr>
<td>001A</td>
<td>26</td>
<td></td>
<td>%MX3.2</td>
<td></td>
</tr>
<tr>
<td>001B</td>
<td>27</td>
<td></td>
<td>%MX3.3</td>
<td></td>
</tr>
<tr>
<td>001C</td>
<td>28</td>
<td></td>
<td>%MX3.4</td>
<td></td>
</tr>
<tr>
<td>001D</td>
<td>29</td>
<td></td>
<td>%MX3.5</td>
<td></td>
</tr>
<tr>
<td>001E</td>
<td>30</td>
<td></td>
<td>%MX3.6</td>
<td></td>
</tr>
<tr>
<td>001F</td>
<td>31</td>
<td></td>
<td>%MX3.7</td>
<td></td>
</tr>
<tr>
<td>0020</td>
<td>32</td>
<td>%MB4</td>
<td>%MX4.0</td>
<td>%MW2</td>
</tr>
<tr>
<td>0021</td>
<td>33</td>
<td></td>
<td>%MX4.1</td>
<td></td>
</tr>
<tr>
<td>0022</td>
<td>34</td>
<td></td>
<td>%MX4.2</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td></td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>0FFF</td>
<td>4095</td>
<td>%MB511</td>
<td>%MX511.7</td>
<td>%MW255</td>
</tr>
<tr>
<td>1000</td>
<td>4096</td>
<td>%MB512</td>
<td>%MX512.0</td>
<td>%MW256</td>
</tr>
</tbody>
</table>
**Modbus address**

<table>
<thead>
<tr>
<th>HEX</th>
<th>DEC</th>
<th>BYTE</th>
<th>Bit (byte-oriented) BOOL</th>
<th>Word WORD</th>
<th>Double word DWORD</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>7FFF</td>
<td>32767</td>
<td>%MB4095</td>
<td>%MX4095.7</td>
<td>%MW2047</td>
<td>%MD1023</td>
</tr>
<tr>
<td>8000</td>
<td>32768</td>
<td>%MB4096</td>
<td>%MX4096.0</td>
<td>%MW2048</td>
<td>%MD1024</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>FFFF</td>
<td>65535</td>
<td>%MB8191</td>
<td>%MX8191.7</td>
<td>%MW4095</td>
<td>%MD2047</td>
</tr>
</tbody>
</table>

Calculation of the bit variable from the hexadecimal address:

**Formula:**

\[
\text{Bit variable (BOOL)} := \%M\text{XBYTE}.\text{BIT}
\]

where:

<table>
<thead>
<tr>
<th>DEC</th>
<th>BYTE</th>
<th>BIT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Decimal address</td>
<td>DEC / 8</td>
<td>DEC mod 8 (Modulo division)</td>
</tr>
</tbody>
</table>

**Examples:**

- Address hexadecimal = 16#2002
  
  
  DEC := 8194
  BYTE := 8194 / 8 := 1024
  BIT := 8194 mod 8 := 2
  Bit variable: %MX1024.2

- Address hexadecimal = 16#3016
  
  DEC := 12310
  BYTE := 12310 / 8 := 1538.75 -> 1538
  BIT := 12310 mod 8 := 6
  Bit variable: %MX1538.6

- Address hexadecimal = 16#55AA
  
  DEC := 21930
  BYTE := 21930 / 8 := 2741.25 -> 2741
  BIT := 21930 mod 8 := 2
  Bit variable: %MX2741.2

Calculation of the hexadecimal address from the bit variable:

**Examples:**

- Bit variable := %MX515.4
  
  DEC := 515 * 8 + 4 := 4124
  Address hex := 16#101C

- Bit variable := %MX3.3
  
  DEC := 3 * 8 + 3 := 27
  Address hex := 16#001B

- Bit variable := %MX6666.2
  
  DEC := 6666 * 8 + 2 := 53330
  Address hex := 16#D052
Peculiarities for accessing Modbus addresses

Peculiarities for bit access:
- A WORD in the %M area is assigned to each Modbus address 0000hex .. FFFFhex.
- Bit addresses 0000hex .. FFFFhex are contained in the word range %MW0 .. %MW4095

Areas protect from read/write access by Modbus client

As described in Configuration of Modbus TCP/IP Server - parameter disable, one write-protected and one read-protected area can be defined. If you try to write to a write-protected area or to read from a read-protected area, an exception response is generated.

Local data of the Modbus client

The address of the area from which data are to be read or to which data are to be written is specified in the Function Block ETHx_MOD_MAST or ModTcpMast at input "Data", via the ADR operator.

For the AC500, the following areas can be accessed using the ADR operator:
- Inputs area (%I area)
- Outputs area (%Q area)
- Area of non-buffered variables (VAR .. END_VAR or VAR_GLOBAL END_VAR)
- Addressable flag area (also protected areas for %M area)
- Area of buffered variables (VAR RETAIN .. END_VAR or VAR_GLOBAL RETAIN .. END_VAR)

Modbus telegrams

For a detailed description of the Modbus TCP telegrams and their elements please see the corresponding specifications on public websites.

Exception response by server

In operating mode Modbus client, the AC500 does only send requests, if the parameters at the MODMAST inputs are logically correct. Nevertheless, it can happen that a server cannot process the request of the client or that the server cannot interpret the request due to transmission errors or in case it's capabilities are exceeded in any way. In those cases, the server returns an exception response to the client. In order to identify this response as an exception response, the function code returned by the server is a logical OR interconnection of the function code received from the client and the value 80HEX.

General telegram description

Table 691: Server response

<table>
<thead>
<tr>
<th>Error code</th>
<th>CRC</th>
</tr>
</thead>
<tbody>
<tr>
<td>High</td>
<td>Low</td>
</tr>
</tbody>
</table>

Possible error codes of the client
<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>01DEC</td>
<td>ILLEGAL FUNCTION</td>
</tr>
<tr>
<td></td>
<td>The server does not support the function requested by the client</td>
</tr>
<tr>
<td>02DEC</td>
<td>ILLEGAL DATA ADDRESS</td>
</tr>
<tr>
<td></td>
<td>Invalid operand address in the server or operand area exceeded</td>
</tr>
<tr>
<td>03DEC</td>
<td>ILLEGAL DATA VALUE</td>
</tr>
<tr>
<td></td>
<td>At least one value is outside the permitted value range</td>
</tr>
<tr>
<td>04DEC</td>
<td>SERVER DEVICE FAILURE</td>
</tr>
<tr>
<td></td>
<td>An unrecoverable error occurred while the server was attempting to perform the requested action</td>
</tr>
<tr>
<td>05DEC</td>
<td>ACKNOWLEDGE</td>
</tr>
<tr>
<td></td>
<td>Specialized use in conjunction with programming commands.</td>
</tr>
<tr>
<td></td>
<td>The server has accepted the request and is processing it, but a long duration of time will be required to do so. This response is returned to prevent a timeout error from occurring in the client. The client can next issue a Poll Program Complete message to determine if processing is completed</td>
</tr>
<tr>
<td>06DEC</td>
<td>SERVER DEVICE BUSY</td>
</tr>
<tr>
<td></td>
<td>Specialized use in conjunction with programming commands.</td>
</tr>
<tr>
<td></td>
<td>The server is engaged in processing a long-duration program command. The client should retransmit the message later when the server is free.</td>
</tr>
<tr>
<td>07DEC</td>
<td>NEGATIVE ACKNOWLEDGE</td>
</tr>
<tr>
<td></td>
<td>Specialized use in conjunction with programming commands.</td>
</tr>
<tr>
<td></td>
<td>The server cannot perform the programming functions. Client should request diagnostic or error information from server.</td>
</tr>
<tr>
<td>08DEC</td>
<td>MEMORY PARITY ERROR</td>
</tr>
<tr>
<td></td>
<td>Specialized use in conjunction with function codes 20 and 21 and reference type 6, to indicate that the extended file area failed to pass a consistency check. The server attempted to read record file, but detected a parity error in the memory. The client can retry the request, but service may be required on the server device.</td>
</tr>
<tr>
<td>09DEC</td>
<td>UNDEFINED</td>
</tr>
<tr>
<td></td>
<td>Actually not defined by Modbus specification but might be used by particular servers.</td>
</tr>
<tr>
<td>10DEC</td>
<td>GATEWAY PATH UNAVAILABLE</td>
</tr>
<tr>
<td></td>
<td>Specialized use in conjunction with gateways, indicates that the gateway was unable to allocate an internal communication path from the input port to the output port for processing the request. Usually means that the gateway is misconfigured or overloaded.</td>
</tr>
<tr>
<td>11DEC</td>
<td>GATEWAY TARGET DEVICE FAILED TO RESPOND</td>
</tr>
<tr>
<td></td>
<td>Specialized use in conjunction with gateways, indicates that no response was obtained from the target device. Usually means that the device is not present on the network.</td>
</tr>
</tbody>
</table>
Example

Table 692: Example:

<table>
<thead>
<tr>
<th>Modbus request of the client:</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Function code:</td>
<td>01</td>
<td>Read n bits</td>
</tr>
<tr>
<td>Server operand address:</td>
<td>4000HEX = 16384DEC</td>
<td>Area for read access disabled in server</td>
</tr>
</tbody>
</table>

| Modbus response of the server: | | |
|---|---|
| Function code: | 81HEX |
| Error code: | 03 |

Processing bits

Some of the Modbus function codes are used to read or write bits (coils, discrete inputs). While a variable of data type WORD can be accessed easily, accessing a stream of bits is complex.

Data type BIT must not be mixed up with data type BOOL. Variables of both types may have values ‘TRUE’ or ‘FALSE’. But while BIT means one single bit only, BOOL requires a byte (8 bit) of memory.

Modbus client

When accessing bits in a Server, the local data referred to at Client Function Blocks input data is always expected to be of format BOOL.

```plaintext
VAR
  abBoolArray : ARRAY [0..15] OF BOOL;
  ModMast : ModTcpMast;
END_VAR

abBoolArray[0] := TRUE;
```

Modbus server

Using the bit offset

The simplest way to access a certain bit within a larger variable is to directly use the bit offset (0 based; see Bit Access to Variables Chapter 1.3.1.24.4.11 “Bit access to variables” on page 565).
VAR
awWordVariable AT %M0 : ARRAY [0..7] OF WORD;
END_VAR

awWordVariable[0].0 := TRUE;
awWordVariable[7].15 := FALSE;
Defining symbolic names for the bit offsets

A more convenient way to access bits e.g. within a word is to define a symbolic name for each single offset (see Bit Access to a variable by using a global constant on page 565).

```plaintext
VAR_GLOBAL_CONSTANT
  (*bit offsets in awWordVariable[0]*)
  NameBitOffset_0_00 : INT := 0;
  NameBitOffset_0_01 : INT := 1;
  (*...*)
  NameBitOffset_0_15 : INT := 15;
  (*bit offsets in awWordVariable[7]*)
  NameBitOffset_7_00 : INT := 0;
  NameBitOffset_7_01 : INT := 1;
  (*...*)
  NameBitOffset_7_15 : INT := 15;
END_VAR

VAR
  awWordVariable AT %MW0 : ARRAY [0..7] OF WORD;
END_VAR

awWordVariable[0].NameBitOffset_0_00 := TRUE;
awWordVariable[7].NameBitOffset_7_15 := FALSE;
```
Defining a data type

A further alternative is to define your own data type according to the requirements of your particular application (see Chapter 1.3.1.25.1.14 “Object ‘DUT’” on page 735). Bit access to BIT data types (see Chapter 1.3.1.24.4.11 “Bit access to variables” on page 565).

```plaintext
TYPE ControlWordType:
  STRUCT
    NameBitOffset_00 : BIT;
    NameBitOffset_01 : BIT;
    NameBitOffset_02 : BIT;
    NameBitOffset_03 : BIT;
    NameBitOffset_04 : BIT;
    NameBitOffset_05 : BIT;
    NameBitOffset_06 : BIT;
    NameBitOffset_07 : BIT;
    NameBitOffset_08 : BIT;
    NameBitOffset_09 : BIT;
    NameBitOffset_10 : BIT;
    NameBitOffset_11 : BIT;
    NameBitOffset_12 : BIT;
    NameBitOffset_13 : BIT;
    NameBitOffset_14 : BIT;
    NameBitOffset_15 : BIT;
  END_STRUCT
END_TYPE

VAR
  atTypeVariable AT M80 : ARRAY [0..7] OF ControlWordType;
END_VAR

atTypeVariable[0].NameBitOffset_00 := TRUE;
atTypeVariable[7].NameBitOffset_15 := FALSE;
```
Defining a complex data type

In case your application requires some more complex data types you can combine data types (DUT; see Bit access to BIT data types Chapter 1.3.1.24.4.11 "Bit access to variables" on page 565).

```plaintext
TYPE ControlWord0Type :
  STRUCT
  NameBitOffset_0_00 : BIT;
  NameBitOffset_0_01 : BIT;
  NameBitOffset_0_02 : BIT;
  NameBitOffset_0_03 : BIT;
  NameBitOffset_0_04 : BIT;
  NameBitOffset_0_05 : BIT;
  NameBitOffset_0_06 : BIT;
  NameBitOffset_0_07 : BIT;
  NameBitOffset_0_08 : BIT;
  NameBitOffset_0_09 : BIT;
  NameBitOffset_0_10 : BIT;
  NameBitOffset_0_11 : BIT;
  NameBitOffset_0_12 : BIT;
  NameBitOffset_0_13 : BIT;
  NameBitOffset_0_14 : BIT;
  NameBitOffset_0_15 : BIT;
  END_STRUCT
END_TYPE

TYPE ControlWord1Type :
  STRUCT
  NameBitOffset_1_00 : BIT;
  NameBitOffset_1_01 : BIT;
  NameBitOffset_1_02 : BIT;
  NameBitOffset_1_03 : BIT;
  NameBitOffset_1_04 : BIT;
  NameBitOffset_1_05 : BIT;
  NameBitOffset_1_06 : BIT;
  NameBitOffset_1_07 : BIT;
  NameBitOffset_1_08 : BIT;
  NameBitOffset_1_09 : BIT;
  NameBitOffset_1_10 : BIT;
  NameBitOffset_1_11 : BIT;
  NameBitOffset_1_12 : BIT;
  NameBitOffset_1_13 : BIT;
  NameBitOffset_1_14 : BIT;
  NameBitOffset_1_15 : BIT;
  END_STRUCT
END_TYPE

TYPE ControlWordListType :
  STRUCT
  ControlWord0 : ControlWord0Type;
  ControlWord1 : ControlWord1Type;
  END_STRUCT
END_TYPE

VAR
  atWordListType AT &MW0 : ControlWordListType;
END_VAR

atWordListType.ControlWord0.NameBitOffset_0_00 := TRUE;
atWordListType.ControlWord1.NameBitOffset_1_15 := FALSE;
```
Pack/unpack BOOL variables

In case you prefer variables of type BOOL you can use the functions for packing and unpacking.

```plaintext
VAR
  aBooleanVariable : ARRAY [0..15] OF BOOL;
  wWordVariable0 : AT WORD WORD;
  wWordVariable1 : AT WORD WORD;
END_VAR

VAR
  Unpack : Mem.UnpackWord;
END_VAR

aBooleanVariable[0] := TRUE;

wWordVariable0 := Mem.PackBitsToWord(aBooleanVariable[0], aBooleanVariable[1], aBooleanVariable[2], aBooleanVariable[3],
                                      aBooleanVariable[4], aBooleanVariable[5], aBooleanVariable[6], aBooleanVariable[7],
                                      aBooleanVariable[8], aBooleanVariable[9], aBooleanVariable[10], aBooleanVariable[11],
                                      aBooleanVariable[12], aBooleanVariable[13], aBooleanVariable[14], aBooleanVariable[15]);

Unpack(Value := wWordVariable0,  
       xBit0 => aBooleanVariable[0], xBit1 => aBooleanVariable[1], xBit2 => aBooleanVariable[2], xBit3 => aBooleanVariable[3],
       xBit4 => aBooleanVariable[4], xBit5 => aBooleanVariable[5], xBit6 => aBooleanVariable[6], xBit7 => aBooleanVariable[7],
       xBit8 => aBooleanVariable[8], xBit9 => aBooleanVariable[9], xBit10 => aBooleanVariable[10], xBit11 => aBooleanVariable[11],
       xBit12 => aBooleanVariable[12], xBit13 => aBooleanVariable[13], xBit14 => aBooleanVariable[14], xBit15 => aBooleanVariable[15]);
```

Function block ETHx_MOD_MAST and ModTcpMast

These Function Blocks are only required for the operating mode Modbus client. It handles the communication (transmission of telegrams to the servers and receipt of telegrams from the servers). The Function Block can be used for the Ethernet interfaces of the controller.

ETHx_MOD_MAST is contained in the library Ethernet_AC500_V10.lib.

ModTcpMast is contained in the library ABB_ModbusTcp_AC500.

1.5.5.1.10 Fast counters

Fast counters in AC500 devices

For AC500 devices the function "fast counter" is available in S500 I/O modules as of firmware version V1.3.

For AC500-eCo devices the function "fast counter" is available in onboard I/Os of PM55x and PM56x.

Integrated fast counters are only available for digital I/O modules.
The digital I/O modules on the I/O bus contain two fast counters each.
If the counter is used, it needs up to 2 digital inputs and one digital output.
If the fast counter is deactivated, the inputs and outputs reserved for the counter can be used for other tasks.

See Configuration Chapter 1.5.6.2.9.8 “Fast counter” on page 3311.

A fast counter is available in the following constellations:

- In digital I/O modules, connected to an AC500 processor module.
- In AC500-eCo processor modules PM55x and PM56x with onboard I/Os.
- In CS31 and CANopen communication interface modules.
- In Modbus, PROFINET and PROFIBUS communication interface modules and in the connected digital I/O modules.
- In digital I/O modules, connected to an EtherCAT communication interface module.
The following table shows the S500 modules which contain a fast counter and which of the digital inputs and outputs are reserved for the counter.

<table>
<thead>
<tr>
<th>Module</th>
<th>Assigned inputs ¹)</th>
<th>Assigned output</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Channel A</td>
<td>Channel B</td>
<td>Channel C ²) or (CF)</td>
</tr>
<tr>
<td>DA501</td>
<td>DC16</td>
<td>DC17</td>
<td>DC18</td>
</tr>
<tr>
<td>DA502</td>
<td>DC16</td>
<td>DC17</td>
<td>DC18, DC18 - in mode 1 and mode 2, DO0 - in mode 101 and mode 102 ⁴)</td>
</tr>
<tr>
<td>DC522</td>
<td>C8</td>
<td>C9</td>
<td>C10</td>
</tr>
<tr>
<td>DC523</td>
<td>C16</td>
<td>C17</td>
<td>C18</td>
</tr>
<tr>
<td>DC532</td>
<td>C24</td>
<td>C25</td>
<td>C26</td>
</tr>
<tr>
<td>DI524</td>
<td>I24</td>
<td>I25</td>
<td>No hardware output available</td>
</tr>
<tr>
<td>DX522</td>
<td>I0</td>
<td>I1</td>
<td>The counter does not activate any relay output</td>
</tr>
<tr>
<td>DC551-CS31</td>
<td>C16</td>
<td>C17</td>
<td>C18</td>
</tr>
<tr>
<td>CI501-PNIO, CI541-DP, CI581-CN, CI521-MODTCP</td>
<td>DI0</td>
<td>DI1</td>
<td>DO0</td>
</tr>
<tr>
<td>CI502-PNIO, CI542-DP, CI582-CN, CI522-MODTCP</td>
<td>DI8</td>
<td>DI9</td>
<td>DO8</td>
</tr>
<tr>
<td>CI590-CS31-HA</td>
<td>C8</td>
<td>C9</td>
<td>C10</td>
</tr>
<tr>
<td>CI592-CS31</td>
<td>DC8</td>
<td>DC9</td>
<td>DC10</td>
</tr>
</tbody>
</table>

¹) The two hardware inputs (channels A and B) are also and always available within the normal process image, irrespective of the operating mode of the counter.

²) The hardware output channel C is activated by the fast counter only in the operating modes 1 and 2.
The counter function of the CS31 communications interface module can only be activated if a bus address greater than 70 is set on the module by means of the address rotary switches. In this case, the effective bus address equals the set address minus 70, and the counter is ready for operation. Example: A set bus address of 83 means that the effective bus address = (83 - 70) = 13 and that the integrated fast counter can be used. In this case the parameter “Fast counter operating mode” may not be 0!

Especially for module DA502: The counter operating mode 101 is the same as mode 1, but the assigned output is DO0 instead of DC18. Also the counter operating mode 102 is the same as mode 2, but the assigned output is DO0 instead of DC18.

The counter function is performed within the communications interface module and, accordingly, in the digital I/O module(s). It works independently of the user program and is therefore able to respond quickly to external signals. A simultaneous counter operation of several digital I/O modules is possible.

Each module counter can be configured for one out of 10 possible modes. The desired operating mode is selected in the PLC configuration using module parameters. After that, it is activated during the initialization phase (power-on, cold start, warm start).

The data exchange to and from the user program is performed using input and output operands. While integrating a module containing a fast counter in the PLC configuration, the necessary operands are created and reserved immediately. Thus, a counter implementation carried out later on does not cause an address shift.

---

### Features independent of the fast counter operating mode

- The pulses at the fast counters' inputs or the evaluated signals of the traces A and B in case of incremental position sensors are counted.
- The counting frequencies of the communications interface modules of PROFINET, PROFIBUS and CANopen are max. 200 kHz (in modes 1 to 6), max. 50 kHz (in mode 7), max. 35 kHz (in mode 9), and max. 20 kHz (in mode 10).
- If the modules DA501, DC522, DC523, DC532, DC531-CS31, CI592-CS31 and CI590-CS31-HA are used, each counting input must be circuited externally in series with a resistor of 470 Ω / 1 W, in order to safely avoid influences from the deactivated module outputs to the connected sensors.
- The positive signal edges are counted, if not noted differently.
- By setting the operating mode 0, the counting function is switched off. In this case, the reserved inputs and outputs can be used for other tasks. Simultaneous use of these terminals for the fast counter and other signals must be avoided.
- The fast counter's actual value is provided as a double word (32 bits).
- The fast counter can count upwards in all operating modes. It counts beginning at the start value (set value) up to the end value (max. from 0 to 4,294,967,295 or hexadecimal from 00 00 00 00 to FF FF FF FF. After reaching 4,294,967,295, the counter jumps with the next pulse to 0. When the counter reaches the programmed end value, the counter output is stored permanently as CF = TRUE (end value reached). Only when the fast counter is set again (set value), CF is reset to FALSE.

---

### Further information

- Operating modes of the fast counter: Operating Modes [Chapter 1.5.6.2.9.8.1.2 “Operating modes” on page 3313](#)
- Configuration of the fast counter: Configuration [Chapter 1.5.6.2.9.8 “Fast counter” on page 3311](#)
- Operation with the library Counter_AC500_V<>:lib: Library documentation [](#)
1.5.5.2 System technology of the AC500 communication modules

1.5.5.2.1 CANopen communication modules

Triggering of event tasks with CAN-IDs

For CM598-CAN module the execution of a PLC application task can be triggered automatically by a certain event, i.e. by incoming CAN 2.0 A or CAN 2.0 B frames. For this, the PLC application task is to be configured as external event task.

**Prerequisites**

- PLC firmware version 3.2.5 and Automation Builder as of version 2.2.5.
- Only one PLC application task can be assigned to a communication module.
- Triggering of event tasks is only supported for the communication module CM598-CAN.

Every incoming CAN frame on a CM598-CAN module processes an event in the AC500 PLC. If the parameter "Trigger PLC Task" is set to TRUE, the CAN protocol task checks via the receive buffer configuration and the corresponding CAN-ID of the CAN frame whether a CAN frame is to be executed or not. Only those CAN-IDs that are configured in the protocol configuration will be processed. All other CAN frames will be rejected. If a CAN frame is to be processed, the CAN frame data is copied to the receive buffer and an event on the IEC event task is triggered.

The IEC event task will be executed for one cycle.

The IEC event task will be triggered continuously until all associated receive buffers have been emptied. Hence, ensure that the buffers are emptied by the task, otherwise the task will run into a loop.

Within the task the function block Cm598CanMsgRecEvt must be used to read the CAN frames from the receive buffers. The function block Cm598CanMsgRec is not suitable as it requires several task cycles for execution.

**CAN frame processing**

The following picture shows the sequence CAN frames processing when the triggering of event task is used.
– Only one external event task can be assigned to a CM598-CAN.
– There is only one common event for an external event task and all selected CAN-IDs. It must be evaluated which CAN-IDs have been received.
– It is possible that CAN frames are lost when necessary system resources are in use or when the CAN frames could not be processed in time due to high system load. So, the PLC application must monitor the task which consumes the events of the CAN protocol with a watchdog mechanism or something similar.
– Received CAN frames of the same CAN-ID are internally stored in FIFO buffers. Reading and writing of the FIFO buffers is not possible at the same time.
– Within an external event task the function block Cm598CanMsgRecEvt must be used to read the received CAN frames. The function block Cm598CanMsgRec is not suitable since its execution needs more than one task cycle.
– The CAN-IDs that are enabled to trigger an external event task must be read by the associated task. Otherwise the task is triggered again and again, and the CPU load will be high.

**Event task configuration**

Add the external event task that should be executed to the task configuration of the PLC application:
1. Right-click on “Task Configuration”. Enter a name for the task and click “Add object”.

2. Right-click on the new task and append a “Program Call”. This contains the program code that is executed by the task.

3. Double-click on the task and setup the task parameters.

A parameter description is given in the CODESYS task configuration § Chapter 1.3.1.25.1.38.1 “Tab ’Configuration”’ on page 839. Deviations are described in the following:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Default</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Priority</td>
<td>16</td>
<td>0..16</td>
<td>Priority of the task</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Value '0' indicates the highest priority</td>
<td></td>
</tr>
<tr>
<td>Type</td>
<td>n.a.</td>
<td>External</td>
<td>Specifies the task type.</td>
</tr>
<tr>
<td>External Event</td>
<td>n.a.</td>
<td>CouplerEvent&lt;slot index of the CM&gt;_CAN</td>
<td>Specifies the event that triggers execution of the task.</td>
</tr>
<tr>
<td>Interval</td>
<td>n.a.</td>
<td>Cycle time</td>
<td>Not used</td>
</tr>
</tbody>
</table>

Configuration of a CM598-CAN module is described in the configuration chapter CM598-CAN § Chapter 1.5.6.2.7 “Communication modules” on page 3286.
1.5.5.3 System technology of the communication interface modules

1.5.5.3.1 Modbus communication interface module

Overview

The Modbus TCP bus module CI52x-MODTCP is used as decentralized I/O module in Modbus TCP networks. The network connection is performed via 2 RJ45 connectors which are integrated in the terminal unit.

CI521-MODTCP I/O channels properties:

- 4 analog inputs (1.0...1.3)
- 2 analog outputs (1.5...1.6)
- 8 digital inputs 24 VDC in 1 group (2.0...2.7)
- 8 digital outputs 24 VDC in 1 group (3.0...3.7)

Functionality

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Interface</td>
<td>Ethernet</td>
</tr>
<tr>
<td>Protocol</td>
<td>Modbus TCP</td>
</tr>
<tr>
<td>Power supply</td>
<td>from the process supply voltage UP</td>
</tr>
<tr>
<td>Supply of the electronic circuitry of the I/O expansion modules attached</td>
<td>Through the expansion bus interface (I/O bus)</td>
</tr>
<tr>
<td>Rotary switches</td>
<td>For setting the last BYTE of the IP (00h to FFh)</td>
</tr>
<tr>
<td>Analog inputs</td>
<td>4 (configurable via software)</td>
</tr>
<tr>
<td>Analog outputs</td>
<td>2 (configurable via software)</td>
</tr>
<tr>
<td>Digital inputs</td>
<td>8 (24 VDC; delay time configurable via software)</td>
</tr>
<tr>
<td>Digital outputs</td>
<td>8 (24 VDC, 0.5 A max.)</td>
</tr>
<tr>
<td>LED displays</td>
<td>For system displays, signal states, errors and power supply</td>
</tr>
<tr>
<td>External supply voltage</td>
<td>Via terminals ZP, UP and UP3 (process supply voltage 24 VDC)</td>
</tr>
</tbody>
</table>

CI522-MODTCP I/O channels properties:

- 8 digital configurable inputs/outputs in 1 group (1.0...1.7)
- 8 digital inputs 24 VDC in 1 group (2.0...2.7)
- 8 digital outputs 24 VDC in 1 group (3.0...3.7)
### Functionality

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Interface</td>
<td>Ethernet</td>
</tr>
<tr>
<td>Protocol</td>
<td>Modbus TCP</td>
</tr>
<tr>
<td>Power supply</td>
<td>from the process supply voltage UP</td>
</tr>
<tr>
<td>Supply of the electronic circuitry of the I/O expansion modules attached</td>
<td>Through the expansion bus interface (I/O bus)</td>
</tr>
<tr>
<td>Rotary switches</td>
<td>For setting the last BYTE of the IP (00h to FFh)</td>
</tr>
<tr>
<td>Configurable digital inputs/outputs</td>
<td>8 (configurable via software)</td>
</tr>
<tr>
<td>Digital inputs</td>
<td>8 (24 VDC; delay time configurable via software)</td>
</tr>
<tr>
<td>Digital outputs</td>
<td>8 (24 VDC, 0.5 A max.)</td>
</tr>
<tr>
<td>LED displays</td>
<td>For system displays, signal states, errors and power supply</td>
</tr>
<tr>
<td>External supply voltage</td>
<td>Via terminals ZP, UP and UP3 (process supply voltage 24 VDC)</td>
</tr>
</tbody>
</table>

The inputs/outputs are electrically isolated from the Ethernet network. There is no potential separation between the channels.

The configuration of the inputs/outputs is performed by software.

For usage in enhanced ambient conditions (e.g. wider temperature and humidity range), a special XC version of the device is available.

### Modbus TCP registers

**Register layout for CI52x-MODTCP**

The registers can be divided in 4 sections:

- Information data section 0x0000 to 0x0D50 (for acyclic use)
- I/O data and diagnosis section 0x0FFA to 0x2B00 (for cyclic use)
- Parameter data section 0x3000 to 0x3B00 (for acyclic use)
- Special functionality section 0x5A00 to 0x6A00 (for acyclic use)
Information data section (Acyclic data)

The information data section can be used to read out common and module specific information. This section is read only.

<table>
<thead>
<tr>
<th>Register (hex)</th>
<th>Description</th>
<th>Readable by Modbus function code</th>
<th>Writeable by Modbus function code</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Device and FW information CI</td>
<td>3</td>
<td>x</td>
</tr>
<tr>
<td>50</td>
<td>Production data CI</td>
<td>3</td>
<td>x</td>
</tr>
<tr>
<td>100</td>
<td>Device and FW information 1. EXP</td>
<td>3 *)</td>
<td>x</td>
</tr>
<tr>
<td>125</td>
<td>Device and FW information 1. Hot swap terminal unit</td>
<td>3 *)</td>
<td>x</td>
</tr>
<tr>
<td>150</td>
<td>Production data 1. EXP</td>
<td>3</td>
<td>x</td>
</tr>
<tr>
<td>175</td>
<td>Production data 1. Hot swap terminal unit</td>
<td>3 *)</td>
<td>x</td>
</tr>
<tr>
<td></td>
<td>...</td>
<td></td>
<td></td>
</tr>
<tr>
<td>A00</td>
<td>Device and FW information 10. EXP</td>
<td>3</td>
<td>x</td>
</tr>
<tr>
<td>A25</td>
<td>Device and FW information 10. Hot swap terminal unit</td>
<td>3 *)</td>
<td>x</td>
</tr>
<tr>
<td>A50</td>
<td>Production data 10. EXP</td>
<td>3</td>
<td>x</td>
</tr>
<tr>
<td>A75</td>
<td>Production data 10. Hot swap terminal unit</td>
<td>3 *)</td>
<td>x</td>
</tr>
<tr>
<td>D00</td>
<td>Common device information</td>
<td>3</td>
<td>x</td>
</tr>
</tbody>
</table>

*) supported from CI52x firmware version V3.2.0 (device index F0)

This section can be divided again in two sections:

- The module specific section (containing information for each module CI52x-MODTCP and expansion modules and hot swap terminal units)
- The common device information block

Module specific information registers

For each module (CI52x device, expansion modules and hot swap terminal units) the following data can be read out:

- Device and FW information
  This section consists of 20 WORDs per module and contains information on each module using the following structure:

<table>
<thead>
<tr>
<th>Data</th>
<th>DATA TYPE</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Module ID</td>
<td>WORD</td>
<td>The module ID of the requested module</td>
</tr>
<tr>
<td>Module name</td>
<td>ARRAY [1..10] OF BYTE</td>
<td>The module name of the requested module</td>
</tr>
<tr>
<td>Version 1(^{st}) processor</td>
<td>ARRAY [1..4] OF BYTE</td>
<td>The version of the 1(^{st}) processor of the requested module</td>
</tr>
<tr>
<td>Version 2(^{nd}) processor</td>
<td>ARRAY [1..4] OF BYTE</td>
<td>The version of the 2(^{nd}) processor of the requested module</td>
</tr>
<tr>
<td>Version 3(^{rd}) processor</td>
<td>ARRAY [1..4] OF BYTE</td>
<td>The version of the 3(^{rd}) processor of the requested module</td>
</tr>
<tr>
<td>Data</td>
<td>DATA TYPE</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>-------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Version 4&lt;sup&gt;th&lt;/sup&gt; processor</td>
<td>ARRAY [1..4] OF BYTE</td>
<td>The version of the 4&lt;sup&gt;th&lt;/sup&gt; processor of the requested module</td>
</tr>
<tr>
<td>Hardware version 1)</td>
<td>ARRAY [1..4] OF BYTE</td>
<td>The hardware version of the 4 processors</td>
</tr>
<tr>
<td>Reserved</td>
<td>ARRAY [1..8] OF BYTE</td>
<td>Reserved</td>
</tr>
<tr>
<td>Number input data</td>
<td>WORD</td>
<td>Number of input data of the requested module in BYTES</td>
</tr>
<tr>
<td>Number output data</td>
<td>WORD</td>
<td>Number of output data of the requested module in BYTES</td>
</tr>
</tbody>
</table>

1) supported from CI52x firmware version V3.2.0 (device index F0)

2) from CI52x firmware version V3.2.0 (device index F0) “Reserved” is ARRAY [1..4] OF BYTE

- **Production / Traceability data:**
  - This section consists of 25 WORDs per module and contains the traceability data for each module using following structure:
    - Article number: Byte 01..15
    - Index: Byte 16..17
    - Name: Byte 18..29
    - Production date: Byte 30..33
    - Key number: Byte 34..38
    - Site: Byte 39..40
    - Year: Byte 41..42
    - Serial number: Byte 41..50 (The serial number implies the year)

- **Production / Traceability data from CI5x2 firmware version V3.2.0 (device index F0):**
  - This section consists of 26 WORDs per module and contains the traceability data for each module using following structure:
    - Article number: Byte 01..15
    - Index: Byte 16..17
    - Name: Byte 18..31
    - Production date: Byte 32..35
    - Key number: Byte 36..40
    - Site: Byte 41..42
    - Year: Byte 43..44
    - Serial number: Byte 42..52 (The serial number implies the year)

---

**Common device information registers**

**Common device information block**

This section consists of 80 WORDs (90 WORDs from CI52x firmware version V3.2.0 (device index F0)) and contains cluster wide information (CI52x device and connected expansion modules using the following structure:
<table>
<thead>
<tr>
<th>Data</th>
<th>DATA TYPE</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Device state</td>
<td>BYTE</td>
<td>The actual state of the device:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0: STATE_PREOP (device booting)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1: STATE_OPERATION (device in operational, no bus supervision active)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2: STATE_ERROR (device detected a bus error, bus supervision active)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3: STATE_IP_ERROR (the device has a IP address error)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4: STATE_CYCLIC_OPERATION (device in operational, bus supervision active)</td>
</tr>
<tr>
<td>Parameter state</td>
<td>BYTE</td>
<td>The actual parameter state of the device:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0: PARA_STATE_NO_PARA (the device has no parameters)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1: PARA_STATE_PARA_ACTIVE (parameterization process running)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2: PARA_STATE_PARA_DONE (the uses valid parameters)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3: PARA_STATE_ERROR (The device has invalid</td>
</tr>
<tr>
<td>Module ID CI device</td>
<td>WORD</td>
<td>Module ID of the CI52x device itself</td>
</tr>
<tr>
<td>Module ID 1st expansion</td>
<td>WORD</td>
<td>Module ID of the 1st connected expansion module</td>
</tr>
<tr>
<td>Module ID 2nd expansion</td>
<td>WORD</td>
<td>Module ID of the 2nd connected expansion module</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Module ID 10th expansion</td>
<td>WORD</td>
<td>Module ID of the 10th connected expansion module</td>
</tr>
<tr>
<td>Expansion bus error count</td>
<td>DWORD</td>
<td>Global telegram error count over all expansion modules</td>
</tr>
<tr>
<td>Good count onboard I/O</td>
<td>DWORD</td>
<td>Telegram good count onboard I/Os</td>
</tr>
<tr>
<td>Good count 1st expansion</td>
<td>DWORD</td>
<td>Telegram good count 1st expansion module</td>
</tr>
<tr>
<td>Good count 2nd expansion</td>
<td>DWORD</td>
<td>Telegram good count 2nd expansion module</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Good count 10th expansion</td>
<td>DWORD</td>
<td>Telegram good count 10th expansion module</td>
</tr>
<tr>
<td>Error count onboard I/O</td>
<td>DWORD</td>
<td>Telegram error count onboard I/Os</td>
</tr>
<tr>
<td>Error count 1st expansion</td>
<td>DWORD</td>
<td>Telegram error count 1st expansion module</td>
</tr>
<tr>
<td>Error count 2nd expansion</td>
<td>DWORD</td>
<td>Telegram error count 2nd expansion module</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Error count 10th expansion</td>
<td>DWORD</td>
<td>Telegram error count 10th expansion module</td>
</tr>
<tr>
<td>Input address onboard I/O</td>
<td>WORD</td>
<td>Modbus TCP register address for inputs of the onboard I/Os</td>
</tr>
<tr>
<td>Input address 1st expansion</td>
<td>WORD</td>
<td>Modbus TCP register address for inputs of the 1st expansion module</td>
</tr>
<tr>
<td>Input address 2nd expansion</td>
<td>WORD</td>
<td>Modbus TCP register address for inputs of the 2nd expansion module</td>
</tr>
<tr>
<td>Data</td>
<td>DATA TYPE</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------------------------</td>
<td>-----------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Input address 10th expansion</td>
<td>WORD</td>
<td>Modbus TCP register address for inputs of the 10th expansion module</td>
</tr>
<tr>
<td>Output address onboard I/O</td>
<td>WORD</td>
<td>Modbus TCP register address for outputs of the onboard I/Os</td>
</tr>
<tr>
<td>Output address 1st expansion</td>
<td>WORD</td>
<td>Modbus TCP register address for outputs of the 1st expansion module</td>
</tr>
<tr>
<td>Output address 2nd expansion</td>
<td>WORD</td>
<td>Modbus TCP register address for outputs of the 2nd expansion module</td>
</tr>
<tr>
<td>Output address 10th expansion</td>
<td>WORD</td>
<td>Modbus TCP register address for outputs of the 10th expansion module</td>
</tr>
<tr>
<td>Module ID 1st hot swap terminal unit *)</td>
<td>WORD</td>
<td>Module ID of the 1st connected hot swap terminal unit *)</td>
</tr>
<tr>
<td>Module ID 2nd hot swap terminal unit *)</td>
<td>WORD</td>
<td>Module ID of the 2nd connected hot swap terminal unit *)</td>
</tr>
<tr>
<td>Module ID 10th hot swap terminal unit *)</td>
<td>WORD</td>
<td>Module ID of the 10th connected hot swap terminal unit *)</td>
</tr>
</tbody>
</table>

*) supported from CI52x firmware version V3.2.0 (device index F0)

I/O / Process data and diagnosis section (Cyclic data)

Table 693: The cyclic data section for CI52x-MODTCP

<table>
<thead>
<tr>
<th>Register (hex)</th>
<th>Description</th>
<th>Readable by Modbus function code</th>
<th>Writeable by Modbus function code</th>
</tr>
</thead>
<tbody>
<tr>
<td>FCE *)</td>
<td>Module state</td>
<td>3, 4, 23</td>
<td>x</td>
</tr>
<tr>
<td>FFA</td>
<td>Diagnosis</td>
<td>3, 4, 23</td>
<td>x</td>
</tr>
<tr>
<td>1000</td>
<td>Inputs CI</td>
<td>3, 4, 23</td>
<td>x</td>
</tr>
<tr>
<td>1100</td>
<td>Inputs 1.EXP</td>
<td>3, 4, 23</td>
<td>x</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1A00</td>
<td>Inputs 10.EXP</td>
<td>3, 4, 23</td>
<td>x</td>
</tr>
<tr>
<td>2000</td>
<td>Outputs CI</td>
<td>3, 23</td>
<td>6, 16, 23</td>
</tr>
<tr>
<td>2100</td>
<td>Outputs 1.EXP</td>
<td>3, 23</td>
<td>6, 16, 23</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2A00</td>
<td>Outputs 10.EXP</td>
<td>3, 23</td>
<td>6, 16, 23</td>
</tr>
<tr>
<td>2B00</td>
<td>Dummy output</td>
<td>3, 23</td>
<td>6, 16, 23</td>
</tr>
</tbody>
</table>

*) supported from CI52x firmware version V3.2.0 (device index F0)

This section can be divided again in three sections:

- Module state (containing the state of connected expansion modules and hot swap terminal units)
- Diagnosis data (containing diagnosis data in AC500 specific format)
- Process data (containing I/O data)
### Module state

The module state section consists of 44 WORDs and contains the module state of connected expansion modules and hot swap terminal units using the following structure:

<table>
<thead>
<tr>
<th>Data</th>
<th>DATA TYPE</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Module ID</td>
<td>WORD</td>
<td>Module ID of the CI52x</td>
</tr>
<tr>
<td>Expected module ID</td>
<td>WORD</td>
<td>Expected (configured) module ID of the CI52x</td>
</tr>
</tbody>
</table>
| Module state        | BYTE      | The current module state of the CI52x:  
0: NO_MOD (no module detected)  
1: MOD_INIT (module detected, module is in initialization phase)  
2: MOD_RUN (module detected and running or in failsafe state, input data are valid)  
3: WRONG_MOD (wrong module detected, module ID doesn’t match expected module ID)  
4: MOD_REMOVED (module removed or defective on hot swap terminal unit, no communication to module possible)  
5: MOD_ERROR (module defective on hot swap terminal unit, no communication to module possible)  
6: MOD_LOST (lost communication to module on not hot swap capable terminal unit)  
7: UNKNOWN (module detected but not configured) |
| Diagnosis flag      | BYTE      | Diagnosis flag for the CI52x:  
0: NO_DIAG (no diagnosis available from CI52x I/O cards)  
1: DIAGAVAILABLE (diagnosis available for CI52x I/O cards) |
| Terminal unit state | BYTE      | Terminal unit state for the CI52x:  
0: NO_HOTSWAP_TU (not hot swap terminal unit detected)  
1: HOTSWAP_TU_RUNNING (hot swap terminal unit detected and working)  
2: HOTSWAP_TU_ERROR (hot swap terminal unit detected, but communication errors for hot swap terminal unit detected) |
| Parameter state     | BYTE      | Parameter state of the CI52x:  
0: NO_PARA (module is in initialization phase and not ready for parameterization)  
1: WAIT_PARA (module awaits parameterization)  
2: PARA_RUN (parameterization running)  
3: LEN_ERR (length of parameters not correct)  
4: ID_ERR (module ID inside parameters not correct)  
5: PARA_DONE (parameterization finished without errors) |
<p>| Module ID           | WORD      | Module ID of the 1st connected expansion module |</p>
<table>
<thead>
<tr>
<th>Data</th>
<th>DATA TYPE</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Expected module ID</td>
<td>WORD</td>
<td>Expected (configured) module ID of the 1&lt;sup&gt;st&lt;/sup&gt; connected expansion module</td>
</tr>
<tr>
<td>Module state</td>
<td>BYTE</td>
<td>The current module state of the 1&lt;sup&gt;st&lt;/sup&gt; connected expansion module</td>
</tr>
<tr>
<td>Diagnosis flag</td>
<td>BYTE</td>
<td>Diagnosis flag for the 1&lt;sup&gt;st&lt;/sup&gt; connected expansion module</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0: NO_DIAG (no diagnosis available for expansion module)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1: DIAG_AVAILABLE (diagnosis available for expansion module)</td>
</tr>
<tr>
<td>Terminal unit state</td>
<td>BYTE</td>
<td>Terminal unit state for the 1&lt;sup&gt;st&lt;/sup&gt; connected expansion module</td>
</tr>
<tr>
<td>Parameter state</td>
<td>BYTE</td>
<td>Parameter state of the 1&lt;sup&gt;st&lt;/sup&gt; connected expansion module</td>
</tr>
<tr>
<td></td>
<td></td>
<td>...</td>
</tr>
<tr>
<td>Module ID</td>
<td>WORD</td>
<td>Module ID of the 10&lt;sup&gt;th&lt;/sup&gt; connected expansion module</td>
</tr>
<tr>
<td>Expected module ID</td>
<td>WORD</td>
<td>Expected (configured) module ID of the 10&lt;sup&gt;th&lt;/sup&gt; connected expansion module</td>
</tr>
<tr>
<td>Module state</td>
<td>BYTE</td>
<td>The current module state of the 10&lt;sup&gt;th&lt;/sup&gt; connected expansion module</td>
</tr>
<tr>
<td>Diagnosis flag</td>
<td>BYTE</td>
<td>Diagnosis flag for the 10&lt;sup&gt;th&lt;/sup&gt; connected expansion module</td>
</tr>
<tr>
<td>Terminal unit state</td>
<td>BYTE</td>
<td>Terminal unit state for the 10&lt;sup&gt;th&lt;/sup&gt; connected expansion module</td>
</tr>
<tr>
<td>Parameter state</td>
<td>BYTE</td>
<td>Parameter state of the 10&lt;sup&gt;th&lt;/sup&gt; connected expansion module</td>
</tr>
</tbody>
</table>
Diagnosis data

The diagnosis data section contains one diagnostic message with the following structure (according to AC500 diagnosis):

<table>
<thead>
<tr>
<th>Byte Number</th>
<th>Description</th>
<th>Possible Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Diagnosis Byte, slot number</td>
<td>31 = CI52x-MODTCP (e. g. error at integrated 8 DI / 8 DO)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 = 1st connected S500 I/O Module</td>
</tr>
<tr>
<td></td>
<td></td>
<td>...</td>
</tr>
<tr>
<td></td>
<td></td>
<td>10 = 10th connected S500 I/O Module</td>
</tr>
<tr>
<td>2</td>
<td>Diagnosis Byte, module number</td>
<td>According to the I/O bus specification passed on by modules to the fieldbus master</td>
</tr>
<tr>
<td>3</td>
<td>Diagnosis Byte, channel</td>
<td>According to the I/O bus specification passed on by modules to the fieldbus master</td>
</tr>
<tr>
<td>4</td>
<td>Diagnosis Byte, error code</td>
<td>According to the I/O bus specification Bit 7 and Bit 6, coded error class</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 = E1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 = E2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2 = E3</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3 = E4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Bit 0 to Bit 5, coded error description</td>
</tr>
<tr>
<td>5</td>
<td>Diagnosis Byte, flags</td>
<td>According to the I/O bus specification</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Bit 7: 1 = coming error</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Bit 6: 1 = leaving error</td>
</tr>
<tr>
<td>6</td>
<td>Reserved</td>
<td>0</td>
</tr>
</tbody>
</table>

If a diagnosis message is read out, the next one will be automatically filled in.
If no more diagnosis messages are available the buffer will be reset to zero.
This ensures that each diagnosis message can be delivered to the Modbus TCP client/slave and no diagnosis will be lost.

I/O data

The I/O data section can use two different formats according to the module parameter “I/O Mapping Structure” (see hardware description Chapter 1.5.3 “Device specifications” on page 2265 for details).

- Fixed I/O mapping
  In case of fixed I/O mapping each module has a predefined register range for each Inputs and Outputs.

- Dynamic I/O mapping
  In case of dynamic I/O mapping the mapping is build according to the actual configuration.

The dummy output at the end of the I/O data section can be used to retrigger the bus supervision and has no effect on the HW outputs.
In case of fixed I/O mapping the following predefined register table is used:

<table>
<thead>
<tr>
<th>Register (hex)</th>
<th>Description</th>
<th>Readable by Modbus function code</th>
<th>Writeable by Modbus function code</th>
</tr>
</thead>
<tbody>
<tr>
<td>1000</td>
<td>Inputs CI</td>
<td>3, 4, 23</td>
<td>x</td>
</tr>
<tr>
<td>1100</td>
<td>Inputs 1.EXP</td>
<td>3, 4, 23</td>
<td>x</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1A00</td>
<td>Inputs 10.EXP</td>
<td>3, 4, 23</td>
<td>x</td>
</tr>
<tr>
<td>2000</td>
<td>Outputs CI</td>
<td>3, 23</td>
<td>6, 16, 23</td>
</tr>
<tr>
<td>2100</td>
<td>Outputs 1.EXP</td>
<td>3, 23</td>
<td>6, 16, 23</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2A00</td>
<td>Outputs 10.EXP</td>
<td>3, 23</td>
<td>6, 16, 23</td>
</tr>
<tr>
<td>2B00</td>
<td>Dummy output</td>
<td>3, 23</td>
<td>6, 16, 23</td>
</tr>
</tbody>
</table>

If a certain expansion module has no inputs or outputs the corresponding registers remain empty.

In case of dynamic mapping only the start addresses of inputs and outputs are predefined:

<table>
<thead>
<tr>
<th>Register (hex)</th>
<th>Description</th>
<th>Readable by Modbus function code</th>
<th>Writeable by Modbus function code</th>
</tr>
</thead>
<tbody>
<tr>
<td>1000</td>
<td>Inputs CI</td>
<td>3, 4, 23</td>
<td>x</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2000</td>
<td>Outputs CI</td>
<td>3, 23</td>
<td>6, 16, 23</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2B00</td>
<td>Dummy output</td>
<td>3, 23</td>
<td>6, 16, 23</td>
</tr>
</tbody>
</table>

The register addresses of the connected expansion modules are calculated dynamically based on the number of inputs and outputs of the previous modules (each module starts directly on the next register after the previous module).

The register addresses of each module can be read out via the common device register (see Chapter 1.5.5.3.1.2.2.2 “Common device information registers” on page 3203).
The difference between fixed I/O mapping and dynamic I/O mapping is shown in the following table.

For this comparison a cluster with CI522, AX522, DC532, AX521, DC523, AO523, AI523, DI524, AX522 and DC523 is used.

<table>
<thead>
<tr>
<th>Register (hex)</th>
<th>Description</th>
<th>Type</th>
<th>Data</th>
<th>Register (hex)</th>
<th>Description</th>
<th>Type</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>1000</td>
<td>Inputs CI</td>
<td>8 DC, 8 DI, FC</td>
<td>4 BYTE + 4 WORD</td>
<td>1000</td>
<td>Inputs CI</td>
<td>8 DC, 8 DI, FC</td>
<td>4 BYTE + 4 WORD</td>
</tr>
<tr>
<td>1100</td>
<td>Inputs AX522</td>
<td>8 AI</td>
<td>8 WORD</td>
<td>1006</td>
<td>Inputs AX522</td>
<td>8 AI</td>
<td>8 WORD</td>
</tr>
<tr>
<td>1200</td>
<td>Inputs DC532</td>
<td>16 DI, 16 DC</td>
<td>4 BYTE</td>
<td>100E</td>
<td>Inputs DC532</td>
<td>16 DI, 16 DC</td>
<td>4 BYTE</td>
</tr>
<tr>
<td>1300</td>
<td>Inputs AX521</td>
<td>4 AI</td>
<td>4 WORD</td>
<td>1010</td>
<td>Inputs AX521</td>
<td>4 AI</td>
<td>4 WORD</td>
</tr>
<tr>
<td>1400</td>
<td>Inputs DC523</td>
<td>24 DC</td>
<td>3 BYTE</td>
<td>1014</td>
<td>Inputs DC523</td>
<td>24 DC</td>
<td>3 BYTE</td>
</tr>
<tr>
<td>1500</td>
<td>Inputs DC532</td>
<td>16 DI, 16 DC</td>
<td>4 BYTE</td>
<td>1016</td>
<td>Inputs DC532</td>
<td>16 DI, 16 DC</td>
<td>4 BYTE</td>
</tr>
<tr>
<td>1600</td>
<td>Inputs AO523</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>Inputs AO523</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>1700</td>
<td>Inputs AI523</td>
<td>16AI</td>
<td>16 WORD</td>
<td>1018</td>
<td>Inputs AI523</td>
<td>16AI</td>
<td>16 WORD</td>
</tr>
<tr>
<td>1800</td>
<td>Inputs DI524</td>
<td>32 DI</td>
<td>4 BYTE</td>
<td>1028</td>
<td>Inputs DI524</td>
<td>32 DI</td>
<td>4 BYTE</td>
</tr>
<tr>
<td>1900</td>
<td>Inputs AX522</td>
<td>8 AI</td>
<td>8 WORD</td>
<td>102A</td>
<td>Inputs AX522</td>
<td>8 AI</td>
<td>8 WORD</td>
</tr>
<tr>
<td>1A00</td>
<td>Inputs DC523</td>
<td>24 DC</td>
<td>3 BYTE</td>
<td>1032</td>
<td>Inputs DC523</td>
<td>24 DC</td>
<td>3 BYTE</td>
</tr>
<tr>
<td>2000</td>
<td>Outputs CI</td>
<td>8 DC, 8DO, FC</td>
<td>4 BYTE + 8 WORD</td>
<td>2000</td>
<td>Outputs CI</td>
<td>8 DC, 8DO, FC</td>
<td>4 BYTE + 8 WORD</td>
</tr>
<tr>
<td>2100</td>
<td>Outputs AX522</td>
<td>8 AO</td>
<td>8 WORD</td>
<td>200A</td>
<td>Outputs AX522</td>
<td>8 AO</td>
<td>8 WORD</td>
</tr>
<tr>
<td>2200</td>
<td>Outputs DC532</td>
<td>16 DC</td>
<td>2 BYTE</td>
<td>2012</td>
<td>Outputs DC532</td>
<td>16 DC</td>
<td>2 BYTE</td>
</tr>
<tr>
<td>2300</td>
<td>Outputs AX521</td>
<td>4 AO</td>
<td>4 WORD</td>
<td>2013</td>
<td>Outputs AX521</td>
<td>4 AO</td>
<td>4 WORD</td>
</tr>
<tr>
<td>2400</td>
<td>Outputs DC523</td>
<td>24 DC</td>
<td>3 BYTE</td>
<td>2017</td>
<td>Outputs DC523</td>
<td>24 DC</td>
<td>3 BYTE</td>
</tr>
<tr>
<td>2500</td>
<td>Outputs DC532</td>
<td>16 DC</td>
<td>2 BYTE</td>
<td>2019</td>
<td>Outputs DC532</td>
<td>16 DC</td>
<td>2 BYTE</td>
</tr>
<tr>
<td>2600</td>
<td>Outputs AO523</td>
<td>16 AO</td>
<td>16 WORD</td>
<td>201A</td>
<td>Outputs AO523</td>
<td>16 AO</td>
<td>16 WORD</td>
</tr>
<tr>
<td>2700</td>
<td>Outputs AI523</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>Outputs AI523</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>2800</td>
<td>Outputs DI524</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>Outputs DI524</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>2900</td>
<td>Outputs AX522</td>
<td>8 AO</td>
<td>8 WORD</td>
<td>202A</td>
<td>Outputs AX522</td>
<td>8 AO</td>
<td>8 WORD</td>
</tr>
<tr>
<td>2A00</td>
<td>Outputs DC523</td>
<td>24 DC</td>
<td>3 BYTE</td>
<td>2032</td>
<td>Outputs DC523</td>
<td>24 DC</td>
<td>3 BYTE</td>
</tr>
</tbody>
</table>
When commissioning a CI521 module with byte order "big endian" in combination with a V3 PLC.

Table 694: I/O data (Inputs 19 BYTEs)

<table>
<thead>
<tr>
<th>Signal</th>
<th>DATA TYPE</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AI0</td>
<td>WORD</td>
<td>Input value of the 1st analogue input</td>
</tr>
<tr>
<td>AI1</td>
<td>WORD</td>
<td>Input value of the 2nd analogue input</td>
</tr>
<tr>
<td>AI2</td>
<td>WORD</td>
<td>Input value of the 3rd analogue input</td>
</tr>
<tr>
<td>AI3</td>
<td>WORD</td>
<td>Input value of the 4th analogue input</td>
</tr>
<tr>
<td>Additional reserve byte</td>
<td>BYTE</td>
<td>reserved, not used</td>
</tr>
<tr>
<td>DI</td>
<td>BYTE</td>
<td>Input value of the DI channels</td>
</tr>
<tr>
<td>Fast counter actual value</td>
<td>DWORD</td>
<td>% Chapter 1.5.5.1.10.1 “Fast counters in AC500 devices” on page 3194</td>
</tr>
<tr>
<td>counter 1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Fast counter actual value</td>
<td>DWORD</td>
<td></td>
</tr>
<tr>
<td>counter 2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Fast counter state counter</td>
<td>BYTE</td>
<td></td>
</tr>
<tr>
<td>counter 1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Fast counter state counter</td>
<td>BYTE</td>
<td></td>
</tr>
<tr>
<td>counter 2</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 695: I/O data (Outputs 23 BYTEs)

<table>
<thead>
<tr>
<th>Signal</th>
<th>DATA TYPE</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AO0</td>
<td>WORD</td>
<td>Output value of the 1st analogue output</td>
</tr>
<tr>
<td>AO1</td>
<td>WORD</td>
<td>Output value of the 2nd analogue output</td>
</tr>
<tr>
<td>Additional reserve byte</td>
<td>BYTE</td>
<td>reserved, not used</td>
</tr>
<tr>
<td>DO</td>
<td>BYTE</td>
<td>Output value of the DO channels</td>
</tr>
<tr>
<td>Fast counter start value</td>
<td>DWORD</td>
<td>% Chapter 1.5.5.1.10.1 “Fast counters in AC500 devices” on page 3194</td>
</tr>
<tr>
<td>counter 1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Fast counter end value</td>
<td>DWORD</td>
<td></td>
</tr>
<tr>
<td>counter 1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Fast counter start value</td>
<td>DWORD</td>
<td></td>
</tr>
<tr>
<td>counter 2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Fast counter end value</td>
<td>DWORD</td>
<td></td>
</tr>
<tr>
<td>counter 2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Fast counter control counter</td>
<td>BYTE</td>
<td></td>
</tr>
<tr>
<td>Fast counter control counter</td>
<td>BYTE</td>
<td></td>
</tr>
</tbody>
</table>
When commissioning a CI522 module with byte order "big endian" in combination with a V3 PLC.

Table 696: I/O data (Inputs 12 BYTEs)

<table>
<thead>
<tr>
<th>Signal</th>
<th>DATA TYPE</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DI</td>
<td>BYTE</td>
<td>Input value of the DI channels</td>
</tr>
<tr>
<td>DC</td>
<td>BYTE</td>
<td>Input value of the DC channels</td>
</tr>
<tr>
<td>Fast counter actual value counter 1</td>
<td>DWORD</td>
<td>% Chapter 1.5.5.1.10.1 “Fast counters in AC500 devices” on page 3194</td>
</tr>
<tr>
<td>Fast counter actual value counter 2</td>
<td>DWORD</td>
<td></td>
</tr>
<tr>
<td>Fast counter state counter 1</td>
<td>BYTE</td>
<td></td>
</tr>
<tr>
<td>Fast counter state counter 2</td>
<td>BYTE</td>
<td></td>
</tr>
</tbody>
</table>

Table 697: I/O data (Outputs 20 BYTEs)

<table>
<thead>
<tr>
<th>Signal</th>
<th>DATA TYPE</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DO</td>
<td>BYTE</td>
<td>Output value of the DO channels</td>
</tr>
<tr>
<td>DC</td>
<td>BYTE</td>
<td>Output value of the DC channels</td>
</tr>
<tr>
<td>Fast counter start value counter 1</td>
<td>DWORD</td>
<td>% Chapter 1.5.5.1.10.1 “Fast counters in AC500 devices” on page 3194</td>
</tr>
<tr>
<td>Fast counter end value counter 1</td>
<td>DWORD</td>
<td></td>
</tr>
<tr>
<td>Fast counter start value counter 2</td>
<td>DWORD</td>
<td></td>
</tr>
<tr>
<td>Fast counter end value counter 2</td>
<td>DWORD</td>
<td></td>
</tr>
<tr>
<td>Fast counter control counter 1</td>
<td>BYTE</td>
<td></td>
</tr>
<tr>
<td>Fast counter control counter 2</td>
<td>BYTE</td>
<td></td>
</tr>
</tbody>
</table>

Parameter data (Acyclic data)

<table>
<thead>
<tr>
<th>Register (hex)</th>
<th>Description</th>
<th>Readable by Modbus function code</th>
<th>Writeable by Modbus function code</th>
</tr>
</thead>
<tbody>
<tr>
<td>3000</td>
<td>Parameters CI</td>
<td>3</td>
<td>6, 16</td>
</tr>
<tr>
<td>3080</td>
<td>Stored parameters CI</td>
<td>3</td>
<td>x</td>
</tr>
<tr>
<td>3100</td>
<td>Parameters 1. EXP</td>
<td>3</td>
<td>6, 16</td>
</tr>
<tr>
<td>3180</td>
<td>Stored parameters 10. EXP</td>
<td>3</td>
<td>x</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3A00</td>
<td>Parameters 10. EXP</td>
<td>3</td>
<td>6, 16</td>
</tr>
<tr>
<td>3A80</td>
<td>Stored parameters 10. EXP</td>
<td>3</td>
<td>x</td>
</tr>
<tr>
<td>3B00</td>
<td>controlword/statusword</td>
<td>3</td>
<td>6, 16</td>
</tr>
</tbody>
</table>
For each connected module the following parameter data are defined (the parameters are represented as ARRAY OF BYTE):

- **Actual used parameter for each module**
  In these sections the actual parameters are stored. This section is also used to write parameters to the module (For a description on how to parameterize see Chapter 1.5.5.3.1.3.2 “Parameterization” on page 3220).

- **Stored parameters for each module**
  If the module has stored nonvolatile parameters these can be read out using the corresponding registers.

The controlword/statusword is used to trigger a parameterization process. The single bits have the following meaning:

<table>
<thead>
<tr>
<th>Bit</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>End of parameterization use parameters</td>
</tr>
<tr>
<td>1</td>
<td>store parameters temporarily, use stored parameters after bus reconnect</td>
</tr>
<tr>
<td>2</td>
<td>store parameters in flash, use stored parameters after power cycle</td>
</tr>
<tr>
<td>3</td>
<td>reserved</td>
</tr>
<tr>
<td>4</td>
<td>delete stored parameters in flash</td>
</tr>
<tr>
<td>5</td>
<td>ignore parameter errors for saving</td>
</tr>
<tr>
<td>6</td>
<td>reserved</td>
</tr>
<tr>
<td>7</td>
<td>reserved</td>
</tr>
<tr>
<td>8</td>
<td>new diagnosis available</td>
</tr>
<tr>
<td>9</td>
<td>new parameters are available</td>
</tr>
<tr>
<td>10</td>
<td>reserved</td>
</tr>
<tr>
<td>11</td>
<td>reserved</td>
</tr>
<tr>
<td>12</td>
<td>reserved</td>
</tr>
<tr>
<td>13</td>
<td>reserved</td>
</tr>
<tr>
<td>14</td>
<td>reserved</td>
</tr>
<tr>
<td>15</td>
<td>reserved</td>
</tr>
</tbody>
</table>

The direction of the first 8 bits is client to server (master to slave).

The direction of the second 8 bits is server to client (slave to master). A description of the bits can be found in chapter behavior Chapter 1.5.5.3.1.3.2 “Parameterization” on page 3220.

The parameter register sections (actual and stored parameters) have the structure as explained in the hardware description of the corresponding module Chapter 1.5.3 “Device specifications” on page 2265.

**Short description of the CI521-MODTCP parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Single parameter index</th>
<th>Description</th>
<th>Additional Info</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td>Module ID (high Byte)</td>
<td>Fixed, must be 16#1C</td>
</tr>
<tr>
<td>1</td>
<td></td>
<td>Module ID (low Byte)</td>
<td>Fixed, must be 16#E8</td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>Ignore Module</td>
<td>Reserved, must be 0</td>
</tr>
<tr>
<td>3</td>
<td></td>
<td>Length of following parameter block</td>
<td>Fixed, must be 16#3F</td>
</tr>
<tr>
<td>Parameter</td>
<td>Single parameter index</td>
<td>Description</td>
<td>Additional Info</td>
</tr>
<tr>
<td>-----------</td>
<td>------------------------</td>
<td>-------------</td>
<td>-----------------</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>Error LED / Failsafe</td>
<td>See hardware description * Chapter 1.5.3 &quot;Device specifications&quot; on page 2265</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>Master IP Byte 0</td>
<td>IP Address for write restrictions ( * &quot;Configurable write restriction&quot; on page 3222)</td>
</tr>
<tr>
<td>6</td>
<td>Master IP Byte 1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>Master IP Byte 2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>Master IP Byte 3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>2</td>
<td>Master IP 1 Byte 0</td>
<td>IP Address for write restrictions ( * &quot;Configurable write restriction&quot; on page 3222)</td>
</tr>
<tr>
<td>10</td>
<td>Master IP 1 Byte 1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>Master IP 1 Byte 2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>Master IP 1 Byte 3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>3</td>
<td>Master IP 2 Byte 0</td>
<td>IP Address for write restrictions ( * &quot;Configurable write restriction&quot; on page 3222)</td>
</tr>
<tr>
<td>14</td>
<td>Master IP 2 Byte 1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>Master IP 2 Byte 2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>Master IP 2 Byte 3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>4</td>
<td>Master IP 3 Byte 0</td>
<td>IP Address for write restrictions ( * &quot;Configurable write restriction&quot; on page 3222)</td>
</tr>
<tr>
<td>18</td>
<td>Master IP 3 Byte 1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>Master IP 3 Byte 2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>Master IP 3 Byte 3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>21</td>
<td>5</td>
<td>Master IP 4 Byte 0</td>
<td>IP Address for write restrictions ( * &quot;Configurable write restriction&quot; on page 3222)</td>
</tr>
<tr>
<td>22</td>
<td>Master IP 4 Byte 1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>23</td>
<td>Master IP 4 Byte 2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>24</td>
<td>Master IP 4 Byte 3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>25</td>
<td>6</td>
<td>Master IP 5 Byte 0</td>
<td>IP Address for write restrictions ( * &quot;Configurable write restriction&quot; on page 3222)</td>
</tr>
<tr>
<td>26</td>
<td>Master IP 5 Byte 1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>27</td>
<td>Master IP 5 Byte 2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>28</td>
<td>Master IP 5 Byte 3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>29</td>
<td>7</td>
<td>Master IP 6 Byte 0</td>
<td>IP Address for write restrictions ( * &quot;Configurable write restriction&quot; on page 3222)</td>
</tr>
<tr>
<td>30</td>
<td>Master IP 6 Byte 1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>31</td>
<td>Master IP 6 Byte 2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>32</td>
<td>Master IP 6 Byte 3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>33</td>
<td>8</td>
<td>Master IP 7 Byte 0</td>
<td>IP Address for write restrictions ( * &quot;Configurable write restriction&quot; on page 3222)</td>
</tr>
<tr>
<td>34</td>
<td>Master IP 7 Byte 1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>36</td>
<td>Master IP 7 Byte 2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>36</td>
<td>Master IP 7 Byte 3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>37</td>
<td>9</td>
<td>Timeout</td>
<td>Timeout for bus supervision in 10ms steps if set to 0 no bus supervision is active</td>
</tr>
<tr>
<td>Parameter</td>
<td>Single parameter index</td>
<td>Description</td>
<td>Additional Info</td>
</tr>
<tr>
<td>-----------</td>
<td>------------------------</td>
<td>-------------</td>
<td>-----------------</td>
</tr>
<tr>
<td>38</td>
<td>10 (read only)</td>
<td>I/O Mapping Structure</td>
<td>See hardware description Chapter 1.5.3 “Device specifications” on page 2265</td>
</tr>
<tr>
<td>39</td>
<td>11</td>
<td>Reserved</td>
<td>Reserved, must be 0</td>
</tr>
<tr>
<td>40</td>
<td>12</td>
<td>Reserved</td>
<td>Reserved, must be 0</td>
</tr>
<tr>
<td>41</td>
<td>13</td>
<td>Reserved</td>
<td>Reserved, must be 0</td>
</tr>
<tr>
<td>42</td>
<td>14</td>
<td>Check supply</td>
<td>See hardware description Chapter 1.5.3 “Device specifications” on page 2265</td>
</tr>
<tr>
<td>43</td>
<td>15</td>
<td>Analogue data format</td>
<td>Reserved, must be 0</td>
</tr>
<tr>
<td>44</td>
<td>16</td>
<td>Input delay</td>
<td>See hardware description Chapter 1.5.3 “Device specifications” on page 2265</td>
</tr>
<tr>
<td>46</td>
<td>17</td>
<td>Fast counter</td>
<td>“Device specifications” on page 2265</td>
</tr>
<tr>
<td>46</td>
<td>18</td>
<td>Short circuit detection</td>
<td>“Device specifications” on page 2265</td>
</tr>
<tr>
<td>47</td>
<td>19</td>
<td>Behavior binary outputs at com. fault</td>
<td>“Device specifications” on page 2265</td>
</tr>
<tr>
<td>48</td>
<td>20</td>
<td>Substitute value binary outputs</td>
<td>“Device specifications” on page 2265</td>
</tr>
<tr>
<td>49</td>
<td>21</td>
<td>Overvoltage monitoring</td>
<td>“Device specifications” on page 2265</td>
</tr>
<tr>
<td>50</td>
<td>22</td>
<td>Behavior analogue outputs</td>
<td>“Device specifications” on page 2265</td>
</tr>
<tr>
<td>51</td>
<td>23</td>
<td>Channel Config AI0</td>
<td>“Device specifications” on page 2265</td>
</tr>
<tr>
<td>52</td>
<td>24</td>
<td>Check Channel AI0</td>
<td>“Device specifications” on page 2265</td>
</tr>
<tr>
<td>53</td>
<td>25</td>
<td>Channel Config AI1</td>
<td>“Device specifications” on page 2265</td>
</tr>
<tr>
<td>54</td>
<td>26</td>
<td>Check Channel AI1</td>
<td>“Device specifications” on page 2265</td>
</tr>
<tr>
<td>55</td>
<td>27</td>
<td>Channel Config AI2</td>
<td>“Device specifications” on page 2265</td>
</tr>
<tr>
<td>56</td>
<td>28</td>
<td>Check Channel AI2</td>
<td>“Device specifications” on page 2265</td>
</tr>
<tr>
<td>57</td>
<td>29</td>
<td>Channel Config AI3</td>
<td>“Device specifications” on page 2265</td>
</tr>
<tr>
<td>58</td>
<td>30</td>
<td>Check Channel AI3</td>
<td>“Device specifications” on page 2265</td>
</tr>
<tr>
<td>59</td>
<td>31</td>
<td>Channel Config AO0</td>
<td>“Device specifications” on page 2265</td>
</tr>
<tr>
<td>60</td>
<td>32</td>
<td>Check Channel AO0</td>
<td>“Device specifications” on page 2265</td>
</tr>
<tr>
<td>61</td>
<td>33</td>
<td>Substitute value AO0 (high Byte)</td>
<td>“Device specifications” on page 2265</td>
</tr>
<tr>
<td>62</td>
<td>34</td>
<td>Substitute value AO0 (low Byte)</td>
<td>“Device specifications” on page 2265</td>
</tr>
<tr>
<td>63</td>
<td>34</td>
<td>Channel Config AO1</td>
<td>“Device specifications” on page 2265</td>
</tr>
<tr>
<td>64</td>
<td>35</td>
<td>Check Channel AO1</td>
<td>“Device specifications” on page 2265</td>
</tr>
<tr>
<td>65</td>
<td>36</td>
<td>Substitute value AO1 (high Byte)</td>
<td>“Device specifications” on page 2265</td>
</tr>
<tr>
<td>66</td>
<td>36</td>
<td>Substitute value AO1 (low Byte)</td>
<td>“Device specifications” on page 2265</td>
</tr>
</tbody>
</table>
## Short description of the CI522-MODTCP parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Single parameter index</th>
<th>Description</th>
<th>Additional Info</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td>Module ID (high Byte)</td>
<td>Fixed, must be 16#1C</td>
</tr>
<tr>
<td>1</td>
<td></td>
<td>Module ID (low Byte)</td>
<td>Fixed, must be 16#ED</td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>Ignore Module</td>
<td>Reserved, must be 0</td>
</tr>
<tr>
<td>3</td>
<td></td>
<td>Length of following parameter block</td>
<td>Fixed, must be 16#2F</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>Error LED / Failsafe</td>
<td>See hardware description § Chapter 1.5.3 &quot;Device specifications&quot; on page 2265</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>Master IP Byte 0</td>
<td>IP Address for write restrictions ( § &quot;Configurable write restriction&quot; on page 3222)</td>
</tr>
<tr>
<td>6</td>
<td></td>
<td>Master IP Byte 1</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td>Master IP Byte 2</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
<td>Master IP Byte 3</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>2</td>
<td>Master IP 1 Byte 0</td>
<td>IP Address for write restrictions ( § &quot;Configurable write restriction&quot; on page 3222)</td>
</tr>
<tr>
<td>10</td>
<td></td>
<td>Master IP 1 Byte 1</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td></td>
<td>Master IP 1 Byte 2</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td></td>
<td>Master IP 1 Byte 3</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>3</td>
<td>Master IP 2 Byte 0</td>
<td>IP Address for write restrictions ( § &quot;Configurable write restriction&quot; on page 3222)</td>
</tr>
<tr>
<td>14</td>
<td></td>
<td>Master IP 2 Byte 1</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td></td>
<td>Master IP 2 Byte 2</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td></td>
<td>Master IP 2 Byte 3</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>4</td>
<td>Master IP 3 Byte 0</td>
<td>IP Address for write restrictions ( § &quot;Configurable write restriction&quot; on page 3222)</td>
</tr>
<tr>
<td>18</td>
<td></td>
<td>Master IP 3 Byte 1</td>
<td></td>
</tr>
<tr>
<td>19</td>
<td></td>
<td>Master IP 3 Byte 2</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td></td>
<td>Master IP 3 Byte 3</td>
<td></td>
</tr>
<tr>
<td>21</td>
<td>5</td>
<td>Master IP 4 Byte 0</td>
<td>IP Address for write restrictions ( § &quot;Configurable write restriction&quot; on page 3222)</td>
</tr>
<tr>
<td>22</td>
<td></td>
<td>Master IP 4 Byte 1</td>
<td></td>
</tr>
<tr>
<td>23</td>
<td></td>
<td>Master IP 4 Byte 2</td>
<td></td>
</tr>
<tr>
<td>24</td>
<td></td>
<td>Master IP 4 Byte 3</td>
<td></td>
</tr>
<tr>
<td>25</td>
<td>6</td>
<td>Master IP 5 Byte 0</td>
<td>IP Address for write restrictions ( § &quot;Configurable write restriction&quot; on page 3222)</td>
</tr>
<tr>
<td>26</td>
<td></td>
<td>Master IP 5 Byte 1</td>
<td></td>
</tr>
<tr>
<td>27</td>
<td></td>
<td>Master IP 5 Byte 2</td>
<td></td>
</tr>
<tr>
<td>28</td>
<td></td>
<td>Master IP 5 Byte 3</td>
<td></td>
</tr>
<tr>
<td>29</td>
<td>7</td>
<td>Master IP 6 Byte 0</td>
<td>IP Address for write restrictions ( § &quot;Configurable write restriction&quot; on page 3222)</td>
</tr>
<tr>
<td>30</td>
<td></td>
<td>Master IP 6 Byte 1</td>
<td></td>
</tr>
<tr>
<td>31</td>
<td></td>
<td>Master IP 6 Byte 2</td>
<td></td>
</tr>
<tr>
<td>32</td>
<td></td>
<td>Master IP 6 Byte 3</td>
<td></td>
</tr>
<tr>
<td>33</td>
<td>8</td>
<td>Master IP 7 Byte 0</td>
<td>IP Address for write restrictions ( § &quot;Configurable write restriction&quot; on page 3222)</td>
</tr>
<tr>
<td>34</td>
<td></td>
<td>Master IP 7 Byte 1</td>
<td></td>
</tr>
<tr>
<td>Parameter</td>
<td>Single parameter index</td>
<td>Description</td>
<td>Additional Info</td>
</tr>
<tr>
<td>-----------</td>
<td>------------------------</td>
<td>--------------------------------------------------</td>
<td>------------------------------------------------------</td>
</tr>
<tr>
<td>36</td>
<td>Master IP 7 Byte 2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>36</td>
<td>Master IP 7 Byte 3</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
| 37        | 2                      | Timeout                                          | Timeout for bus supervision in 10ms steps
if set to 0 no bus supervision is active |
| 38        | 3 (read only)          | I/O Mapping Structure                            | See [hardware description](#) Chapter 1.5.3
“Device specifications” on page 2265 |
| 39        | 4                      | Reserved                                         | Reserved, must be 0                                  |
| 40        | 5                      | Reserved                                         |                                                      |
| 41        | 6                      | Reserved                                         |                                                      |
| 42        | 7                      | Check supply                                     | See [hardware description](#) Chapter 1.5.3
“Device specifications” on page 2265 |
| 43        | 8                      | Input delay                                      | See [hardware description](#) Chapter 1.5.3
“Device specifications” on page 2265 |
| 44        | 9                      | Fast counter                                      | See [hardware description](#) Chapter 1.5.3
“Device specifications” on page 2265 |
| 46        | 10                     | Short circuit detection                          |                                                      |
| 46        | 11                     | Behavior binary outputs at com. fault            |                                                      |
| 47        | 12                     | Substitute value binary outputs (high byte)      |                                                      |
| 48        | Substitute value binary outputs (low byte)      |                                                      |
| 49        | 13                     | Voltage feedback monitoring                      |                                                      |
| 50        | 14                     | Overvoltage monitoring                           |                                                      |
Parameters of connected expansion modules

The parameters of the connected expansion modules are represented as byte array (the parameters valid for "CPU" in the hardware description of the corresponding module are used $\forall$ Chapter 1.5.3 “Device specifications” on page 2265):

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Additional Info</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Module ID (high byte)</td>
<td>Fixed, see hardware description of corresponding module (the module ID of FBP is used) $\forall$ Chapter 1.5.3 “Device specifications” on page 2265</td>
</tr>
<tr>
<td>1</td>
<td>Module ID (low byte)</td>
<td>Fixed, see hardware description of corresponding module (the module ID of FBP is used) $\forall$ Chapter 1.5.3 “Device specifications” on page 2265</td>
</tr>
<tr>
<td>2</td>
<td>Ignore module</td>
<td>Reserved must be 0</td>
</tr>
<tr>
<td>3</td>
<td>Length of following parameter block</td>
<td>Fixed, see hardware description of corresponding module $\forall$ Chapter 1.5.3 “Device specifications” on page 2265</td>
</tr>
<tr>
<td>4...</td>
<td>The rest of the parameter are described in the corresponding module</td>
<td></td>
</tr>
</tbody>
</table>

Special functionality

This section contains special services like firmware update or single parameterization.

<table>
<thead>
<tr>
<th>Register (hex)</th>
<th>Description</th>
<th>Readable by Modbus function code</th>
<th>Writeable by Modbus function code</th>
</tr>
</thead>
<tbody>
<tr>
<td>4000</td>
<td>Firmware download</td>
<td>3</td>
<td>16</td>
</tr>
<tr>
<td>4100</td>
<td>Firmware download state</td>
<td>3</td>
<td>x</td>
</tr>
<tr>
<td>5000</td>
<td>Write single parameterization of CI</td>
<td>x</td>
<td>16</td>
</tr>
<tr>
<td>5100</td>
<td>Write single parameterization of 1. EXP</td>
<td>x</td>
<td>16</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5A00</td>
<td>Write single parameterization of 10. EXP</td>
<td>x</td>
<td>16</td>
</tr>
<tr>
<td>6000</td>
<td>Read single parameterization of CI</td>
<td>3</td>
<td>16</td>
</tr>
<tr>
<td>6100</td>
<td>Read single parameterization of 1. EXP</td>
<td>3</td>
<td>16</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6A00</td>
<td>Read single parameterization of 10. EXP</td>
<td>3</td>
<td>16</td>
</tr>
</tbody>
</table>
Behavior

IP address assignment

The delivery IP address of the CI52x-MODTCP is 192.168.0.xx (xx is the hardware address switch position of the device.

The devices support BOOTP, DHCP and fixed IP address setting (these can be set individual or together). If BOOTP and DHCP are enabled the following priority takes place:

- If DHCP configuration fails, the device will fall back to BOOTP.
- In case of a BOOTP failure, the fixed IP address will be used.

A new IP address (or changing of BOOTP and DHCP) can be set in two different ways:

- With the address switches of the corresponding module
- With the IP Configuration Tool

Using the address switches

With the address switches only the last byte of the IP address can be changed. The IP address can only be set via the address switches in case of factory default or in case of the last byte of the IP address is set to zero with the IP Configuration Tool. The not allowed IP addresses are mapped as followed:

- Address switch position 255 is mapped to fixed IP 192.168.0.254 independent of other stored settings (by IP Configuration Tool). This is a backup so the module can always get a valid IP address and can be configured by the IP Configuration Tool.
- Address switch position 0 is mapped to last byte equal 1 and DHCP enabled.

Using the IP configuration tool

With the IP Configuration Tool a network scan can be executed, and the found devices can be assigned with new settings, e.g. enable BOOTP or DHCP and set a new fixed IP. If the last byte of the IP address of the CI52x-MODTCP devices is set to 0 with the IP Configuration Tool the address switch position is used instead (see Chapter 1.5.5.3.1.3.1.1 “Using the address switches” on page 3219).
Parameterization

The parameterization is done via the corresponding registers explained in the Modbus TCP registers “Parameter data (Acyclic data)” on page 3212.

In addition to that the parameters can be directly transferred via Automation Builder (see documentation of Automation Builder for that).

There are two different parameter sections with different behaviour.

**Actual used parameters**

After startup this section contains the following data:

- Default parameters (only module id and parameter length set all others zero) if no valid stored parameters are available (no or invalid parameters stored).
- Actual used / stored parameters if valid parameters are stored nonvolatile.

These parameters can be read out and changed by reading or writing of the corresponding registers, but will not be used automatically after writing them, the use of new written parameters has to be triggered by writing the parameter control word with the corresponding bits set (see below).

**Stored parameters**

This section always contains a copy of the nonvolatile stored parameters, if no parameters are stored nonvolatile this sections will be 0.

**Controlword/statusword parameter**

This parameter can be used to trigger and save new parameters.

The direction of the first 8 bit is client to server (master to slave). The direction of the second 8 bits is server to client (slave to master).
<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Use parameters / start parameterization</td>
</tr>
<tr>
<td>1</td>
<td>Store parameters volatile</td>
</tr>
<tr>
<td>2</td>
<td>Store parameters nonvolatile</td>
</tr>
<tr>
<td>3</td>
<td>Reserved</td>
</tr>
<tr>
<td>4</td>
<td>Delete nonvolatile stored parameters</td>
</tr>
<tr>
<td>5</td>
<td>Ignore parameter error for nonvolatile parameter storage</td>
</tr>
<tr>
<td>6</td>
<td>Reserved</td>
</tr>
<tr>
<td>7</td>
<td>Reserved</td>
</tr>
<tr>
<td>8</td>
<td>New diagnosis available</td>
</tr>
<tr>
<td>9</td>
<td>New parameters available</td>
</tr>
<tr>
<td>10...15</td>
<td>Reserved</td>
</tr>
</tbody>
</table>

**Cyclic I/O data exchange**

The I/O data can be exchanged cyclic by the master by reading, writing the corresponding registers.

I/O data exchange is only possible after successful parameterization of the device.

For writing of outputs **bus failure detection** can be activated by setting the corresponding parameter. This bus failure detection is described in the following chapter.

**Bus failure detection**

If the parameter "*timeout*" in the module parameters of the CI52x-MODTCP is set, the module will supervise the Modbus TCP "write telegrams".

After the first "write telegram" the bus will be supervised. If no new "write telegram" arrives at the CI52x-MODTCP within the configured time, the module will detect a bus failure and switch off its outputs or switch them to the configured failsafe state (see module parameter "Failsafe CI521" and "Failsafe CI522" for details § Chapter 1.5.3.7.4.1.7 “Parameterization” on page 2918 § Chapter 1.5.3.7.4.2.7 “Parameterization” on page 2947).
With the module parameters “Master IP”- “Master IP 7” it is possible to set write restrictions on the CI52x-MODTCP device. If none of the parameters is set, all masters / clients in the network have read and write rights on the CI52x-MODTCP device and its connected expansion modules. If at least one parameter is set only the configured masters / clients have write rights on the CI52x-MODTCP device. All other masters / clients still have read access to the CI52x-MODTCP device.

Diagnosis behavior

Each diagnosis message signals if this error is coming or going, so it is possible to create a list in the master of actual pending diagnosis. Diagnosis messages will be transferred again after a bus failure detection and reconnection. Diagnosis messages can be read out with function code 3,4,23. Function codes 3 and 4 can always read out diagnosis messages, function code 23 can only read out after successful parameterization of the device. See also table “Chapter 1.5.5.3.1.2.3.2 “Diagnosis data” on page 3208.

Single parameterization

The single parameterization services can be used to read or write parameters during runtime of device without the need of triggering a new parameterization process. For indexes used for single parameterization services see parameter lists in section Modbus TCP registers of this document. The read and write parameterization services are explained below, for each module (CI52x-MODTCP and connected expansion modules) a different section for read and write is defined (see chapter Modbus TCP registers in this document). Both services are using the following data structure:

The length of the read / write service depends on the count of parameters that should be transferred (length = 4+ count*8).

Reading of single parameters

The read single parameterization works in two steps:

- Writing of a request list containing the indexes that should be read using the structure explained above.
  Only CNT and PARA_IDX has to be set.
  Up to 5 parameters can be requested with one telegram.
  The length of the write service depends on the count of parameters that should be transferred (length = 4+ count*8).
- Reading of the parameters list with the same length then the previous write request.
  If the internal reading process inside the CI52x-MODTCP device is done the data will be read out.
  If the internal reading process inside the CI52x-MODTCP device is not yet finished the read service will be rejected with Modbus TCP exception code 6 (device busy).
Writing of single parameters

For writing of single parameters only one step is necessary, the parameters are transferred with one write request using the structure described above.

The length of the write service depends on the count of parameters that should be transferred (length = 4 + count*8).

In case of write of single parameters the following values have to be set:

- CNT: number of parameters to be set
- And for each parameter:
  - Parameter index
  - Parameter length
  - New parameter value

Written single parameters are not stored volatile and not stored nonvolatile. That means after a bus reconnection or power cycle the written parameters will be discarded.

Commissioning example

Set IP Address:

- The setting of the IP address is the first step to integrate the CI52x-MODTCP devices into a running system.
- The setting of the IP address of the CI52x-MODTCP devices is described in the chapter "Chapter 1.5.5.3.1.3.1 "IP address assignment" on page 3219 in this document.

Set Parameters (optional read parameters):

- The second step in configuring the CI52x-MODTCP devices is to set the module and channel parameters.
- A read of parameters is optional but can be used to get the module IDs and the parameter length.
- The reading and or writing of parameters is described in chapter "Chapter 1.5.5.3.1.3.2 "Parameterization" on page 3220.

Set Control Word:

- After setting the parameter data these have to be activated by writing the control word.
- The meaning and usage of the control word is described in chapter "Chapter 1.5.5.3.1.3.2 "Parameterization" on page 3220.

Exchange data:

- After setting and activating the parameters the CI52x-MODTCP device is ready for data exchange.
- The registers for data exchange are described in chapter "Chapter 1.5.5.3.1.2.3 "I/O / Process data and diagnosis section (Cyclic data)" on page 3205.

Hot swap

With hot swap for AC500 and S500 it is possible to exchange expansion modules (with same type) during runtime.

Preconditions for using hot swap

Information about preconditions for using hot swap see "Hot swap" on page 2312.
## Compatibility of hot swap

<table>
<thead>
<tr>
<th>I/O module on TU5xx-H connected to I/O bus master</th>
<th>Modbus remote I/O</th>
</tr>
</thead>
<tbody>
<tr>
<td>Required version of I/O bus master</td>
<td>CI521-MODTCP or CI522-MODTCP</td>
</tr>
<tr>
<td>Fieldbus master when used as remote I/O with AC500 V3</td>
<td>Any AC500 V3 CPU with on-board Ethernet</td>
</tr>
<tr>
<td>When used as remote I/O on third party controller (PLC or DCS)</td>
<td>No limitation known</td>
</tr>
</tbody>
</table>

### Hot swap behavior

The following table describes the behavior in case of I/O attached to communication interface module for Modbus TCP, CI521-MODTCP or CI522-MODTCP.

<table>
<thead>
<tr>
<th>Hot Swap Behavior</th>
<th>Modbus TCP remote I/O</th>
</tr>
</thead>
<tbody>
<tr>
<td>Start-up behavior with missing or damaged I/O module on hot swap terminal unit TU5xx-H</td>
<td>Remote I/O station is not starting</td>
</tr>
<tr>
<td>Start-up behavior with wrong I/O module type on any terminal unit</td>
<td>Remote I/O station is not starting</td>
</tr>
<tr>
<td>Diagnosis of presence of hot swap terminal unit</td>
<td>Information is available in Modbus registers of the communication interface module which can be accessed by the application program</td>
</tr>
<tr>
<td>Diagnosis of hot swap capability of I/O module mounted on hot swap terminal unit</td>
<td>Information can be obtained by reading Modbus registers in the communication interface module. Those Modbus registers contain: ● Diagnosis in case that a not hot-swap-pable I/O module is plugged on a hot swap terminal unit ● Diagnosis In case that in a mixed configuration with at least one hot swap terminal unit an I/O module, that must not be used in a hot swap configuration, is mounted on any terminal unit of the configuration ● Production data and version index of the modules</td>
</tr>
<tr>
<td>Diagnosis while hot swap module is pulled or module (mounted on hot swap terminal unit) has stopped working</td>
<td>Diagnosis is available in Modbus registers in the communication interface module</td>
</tr>
<tr>
<td>Input state in process image of controller while module is pulled or module is not operational</td>
<td>Input = ZERO</td>
</tr>
<tr>
<td>Diagnosis after plugging the I/O module on the hot swap terminal unit</td>
<td>Diagnose &quot;diagnosis gone&quot; is available in Modbus registers in the communication interface module</td>
</tr>
</tbody>
</table>
System behavior

If an expansion module is removed or defective during run time, the input data of this module will be set to “0” and the module state will be set to the corresponding value (see Chapter 1.5.5.3.1.3 “I/O / Process data and diagnosis section (Cyclic data)” on page 3205). A diagnosis message will be created in that case (see hardware description of CI521-MODTCP / CI522-MODTCP for diagnosis messages Chapter 1.5.3.7.4.1 “CI521-MODTCP” on page 2898 Chapter 1.5.3.7.4.2 “CI522-MODTCP” on page 2938).

In case a module is replaced, the new module will automatically be parameterized with the last parameters of the removed module (if single parameters were written to the previously removed module, this parameters will be ignored).

During pulling or plugging of a certain module, all other module will continue to operate with one limitation: The reaction time of modules connected to the right of the affected module will be bigger in that case (up to 50 ms).

If the bus failure detection is active for CI52x and failsafe is configured (see Chapter 1.5.5.3.1.3.3 “Cyclic I/O data exchange” on page 3221) the following behavior applies if a module is removed and replugged during failsafe condition:

- Last value configured for output:
  - After a bus failure is detected, failsafe will be activated and the output will remain at its last value.
  - If the module is removed and plugged again, the output will remain off, and not be kept its last value, as the last value of the new module is “0” in that case.

- Substitute value configured for output:
  - After a bus failure is detected, failsafe will be activated and the output will be according to the configured substitute value.
  - If the module is removed and plugged again now, the output will be set according to the configured substitute value again.

- Substitute value for x seconds configured for output:
  - After a bus failure is detected, failsafe will be activated and the output will be according to the configured substitute value for the configured time.
  - If the module is removed and plugged again now, the output will be set according to the configured substitute value again, and the configured time starts again.

Mandatory rules for hot swapping

Mandatory rules for hot swapping:

- Between two pull and / or plug operations of I/O modules a pause of at least 1 second must be observed.
  - That means if a module is pulled or plugged there has to be at least a break of 1 second before the next module is pulled or plugged.

- At boot up of CI52x all configured expansion modules have to be physically available.
  - Start up with missing modules is not supported.

- In the application program it is possible to detect if a hot swap terminal unit is mounted in a specific position on the I/O bus. The information is available in the common device information registers. These can be accessed when the version of the communication interface module supports hot swap.
  - This has to be checked by application:
    Best way for checking if a hot swap terminal unit is available or not, is reading out the common device information registers (see Chapter 1.5.5.3.1.2.2 “Information data section (Acyclic data)” on page 3202). If the CI52x rejects this read out the CI52x doesn’t support hot swap at all.
1.5.5.3.2 PROFINET communication interface module

Hot swap

With hot swap for AC500 and S500 it is possible to exchange expansion modules (with same type) during runtime.

Preconditions for using hot swap

Information about preconditions for using hot swap see “Hot swap” on page 2312.

Compatibility of hot swap

<table>
<thead>
<tr>
<th>I/O module on TU5xx-H connected to I/O bus master</th>
<th>PROFINET remote I/O</th>
</tr>
</thead>
<tbody>
<tr>
<td>CI501-PNIO or CI502-PNIO</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Required version of I/O bus master</th>
<th>Module index as of F0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Firmware as of V3.2.10</td>
<td></td>
</tr>
</tbody>
</table>

| Fieldbus master when used as remote I/O with AC500 V3 | Not supported |

<table>
<thead>
<tr>
<th>When used as remote I/O on third party controller (PLC or DCS)</th>
<th>Note: alarms must be acknowledged by fieldbus master.</th>
</tr>
</thead>
<tbody>
<tr>
<td>GSDML as of version</td>
<td></td>
</tr>
<tr>
<td>GSDML-V2.3-ABB-S500-CI501-PNIO-20180822.xml or</td>
<td></td>
</tr>
<tr>
<td>GSDML-V2.3-ABB-S500-CI502-PNIO-20180822.xml</td>
<td></td>
</tr>
<tr>
<td>needed for full scope of vendor specific diagnosis.</td>
<td></td>
</tr>
</tbody>
</table>
Hot swap behavior

The following table describes the behavior in case of I/O attached to communication interface module for PROFINET, CI501-PNIO or CI502-PNIO.

<table>
<thead>
<tr>
<th>Hot Swap Behavior</th>
<th>PROFINET remote I/O with third party controller</th>
</tr>
</thead>
<tbody>
<tr>
<td>Start-up behavior with missing or damaged I/O module on hot swap terminal unit TU5xx-H</td>
<td>Remote I/O station is not starting</td>
</tr>
<tr>
<td>Start-up behavior with wrong I/O module type on any terminal unit</td>
<td>Remote I/O station is not starting</td>
</tr>
<tr>
<td>Diagnosis of presence of hot swap terminal unit</td>
<td>Information is available either:</td>
</tr>
<tr>
<td>Diagnosis of hot swap capability of I/O module mounted on hot swap terminal unit</td>
<td>Diagnosis is transmitted as vendor specific PROFINET channel diagnosis:</td>
</tr>
<tr>
<td>Diagnosis while hot swap module is pulled or module (mounted on hot swap terminal unit) has stopped working</td>
<td>PROFINET channel diagnosis is generated together with standard &quot;pull alarm&quot; which must be acknowledged</td>
</tr>
<tr>
<td>Input state in process image of controller while module is pulled or module is not operational</td>
<td>Input = ZERO</td>
</tr>
<tr>
<td>Diagnosis after plugging of the I/O module on the hot swap terminal unit</td>
<td>PROFINET channel diagnosis is generated together with standard &quot;plug alarm&quot; which must be acknowledged</td>
</tr>
</tbody>
</table>

System behavior

If an expansion module is removed or defective during run time, the input data of this module will be set to “0” and the module state will be set to the corresponding value. A diagnosis message will be created in that case (see hardware description of CI501-PNIO / CI502-PNIO for diagnosis messages Chapter 1.5.3.7.5.2 “CI501-PNIO” on page 2965).

In case a module is replaced, the new module will automatically be parameterized with the last parameters of the removed module (if single parameters were written to the previously removed module, this parameters will be ignored).

During pulling or plugging of a certain module, all other module will continue to operate with one limitation: The reaction time of modules connected to the right of the affected module will be bigger in that case (up to 50 ms).

If the bus failure detection is active for CI50x and failsafe is configured the following behavior applies if a module is removed and replugged during failsafe condition:
● Last value configured for output:
  – After a bus failure is detected, failsafe will be activated and the output will remain at its last value.
  – If the module is removed and plugged again, the output will remain off, and not be kept its last value, as the last value of the new module is “0” in that case.
● Substitute value configured for output:
  – After a bus failure is detected, failsafe will be activated and the output will be according to the configured substitute value.
  – If the module is removed and plugged again now, the output will be set according to the configured substitute value again.
● Substitute value for x seconds configured for output:
  – After a bus failure is detected, failsafe will be activated and the output will be according to the configured substitute value for the configured time.
  – If the module is removed and plugged again now, the output will be set according to the configured substitute value again, and the configured time starts again.

Mandatory rules for hot swapping

Mandatory rules for hot swapping:
● Between two pull and / or plug operations of I/O modules a pause of at least 1 second must be observed.
  – That means if a module is pulled or plugged there has to be at least a break of 1 second before the next module is pulled or plugged.
● At boot up of CI50x all configured expansion modules have to be physically available.
  – Start up with missing modules is not supported.
● In the application program it is possible to detect if a hot swap terminal unit is mounted in a specific position on the I/O bus. The information is available in the process data area or can be read out via acyclic read. These can be accessed when the version of the communication interface module supports hot swap.
  – This has to be checked by application:

Best way for checking if a hot swap terminal unit is available or not, is checking the corresponding information inside the process image.

1.5.6 Configuration in Automation Builder for AC500 V3 products

1.5.6.1 General settings

This chapter describes the device configuration of AC500 product family with Automation Builder. Basic information on Automation Builder handling can be found in the Getting Started § Chapter 1.1 “Getting started” on page 14.

1.5.6.1.1 Project handling

What is a project?
● A project contains the objects that are necessary for the creation of a control program (‘application’):
  – Pure POU's, for example programs, function blocks, functions, GVLs.
  – Objects that are additionally required in order to be able to execute the application on a PLC. For example, task configuration, library manager, symbol configuration, device configuration, visualizations, external files.
● You can program several applications and connect several control devices in a project.
● CODESYS manages device-specific and application-specific POU’s in the “Devices” view (‘Device tree’), project-wide available POU’s in the “POU’s” view.
● For the creation of projects there are templates that already contain certain objects.
Basic configurations and information for the project are defined in the “Project Settings” and the “Project Information”. For example:

- Compiler settings
- User management
- Author
- Data for the project file.

There are settings for the version compatibility of the project in the configuration dialog boxes for the “Project Environment”.

- You save a project as a file in the file system. Optionally you can pack it together with project-relevant files and information in a project archive. Saving in a source control system such as SVN is also possible.

- Each project contains the information regarding the CODESYS version with which it was created. If you open it in another version, CODESYS will draw your attention to possible or necessary updates with regard to file format, library versions, etc.

- You can compare, export or import projects and create documentation for them.

- You can protect a project against change and completely, i.e. also against reading. By using a user management system you can purposefully control access to the project and even to individual objects in the project.

Creating a new project

1. Select “File → New Project”.
   - If the used Automation Builder version is not the latest version, an information is displayed.

2. • Select “Change to newest installed version” to create a project with the latest installed version of Automation Builder.
   • Select “Continue to work with version: XXX” to create a project in the current software version.

3. Select “AC500 project”, enter a project name and specify the storage location for the new project.

   With “Empty project” a project without a PLC is created.
4. Select the device type for the new project and click [Add device].

The new project is created and can be configured.

Opening and updating an existing project

**NOTICE!**
Risk of damaging Automation Builder projects!
Projects created with Automation Builder are incompatible with CODESYS V2.3.9.x. Do not open projects with CODESYS V2.3.9.x as this can cause corrupted Automation Builder projects.

Automation Builder performs an integrity check for the PLC configuration before generating the configuration.

Opening an existing project

1. Select “File ➔ Open Project”.
   ⇒ The “Open Project” dialog appears.
2. Select the existing project from the file system.
   ⇒ Automation Builder switches to the version of the project and opens the project.

**Updating an existing project**

With opening an existing project Automation Builder automatically detects the project version. In case of an outdated project version you are asked, whether the project shall be updated.

- If you confirm the update, the project is automatically updated to the latest Automation Builder version.
- If you decline the update, the outdated version is kept. To update the project at a later date, select “Project ⇒ Update Project”.
  Automation Builder updates the complete project (complete device tree) to the latest version. Success messages, warnings and errors are described in the section “All messages”.

> To keep an older project, it must be opened with the same Control Builder Plus/Automation Builder version the project has been created. For this, the appropriate Control Builder Plus/Automation Builder profil must be selected.

> In this mode, new Automation Builder features cannot be used.

> It is not possible to downgrade a project to an earlier Automation Builder version.

> Automation Builder performs an integrity check for the PLC configuration before generating the configuration.

To update all devices of an existing PLC project, right-click the PLC node and select “Update objects”. In the dialog enable “update subtree” option to update all sub-objects. Otherwise only the Processor Module object is updated.

To update a specific device only, the command “Update objects” can be executed individually at the specific node.

**Exporting and importing project**

Configuration of a complete PLC or single devices can be reused within the same project by copy-and-paste the desired nodes in the device tree.

> Imported project parts can only be imported in the same Automation Builder version.

> Automation Builder performs an integrity check for the PLC configuration before generating the configuration.

**Project export**

From the menu, select “Project ⇒ Export ⇒ Project”. Select the objects that should be added to the export file.
“One file per subtree”: If this option is activated, all objects belonging to the same subtree will be exported into the same export file, otherwise a separate file will be created for each particular object.

Project import

In order to compare projects, check differences and only adapt the necessary changes, a command for importing and comparing projects has been introduced in Automation Builder.

The project comparison is a premium feature. Users with a basic or standard licence see no comparison.

Project import without comparison function

From the menu, select “Project ➔ Import ➔ Project”.

In the dialog, select the *.export file from the file system and press [Open] to import the project file.

If the version of the export file is not identically to the version of the project an import is not possible. An error message appears.

Project import with comparison function

Below are scenarios/behavior of the command providing a project is already exported using Automation Builder export command.
1. Import already exported (*.export) file using import command.

2. Once file is imported, a compare view is shown. If current project is same as imported then projects are equal message is shown to the user. Left pane is for current project and right pane is for imported.

3. If current project is empty, then imported project is compared with empty project only. Left pane would be blank in this case.
4. If there is a difference between current and imported projects, then difference is highlighted in red.

5. If anything is added in current project (/not available in imported) is highlighted in green.

6. If imported project is having any extra module which is highlighted in blue (other way indicates that extra module is deleted from current project).
7. If user wants to adapt only changed nodes from imported project then the user can select only that node and accept it (“Accept Block”/“Accept Single”) and this operation updates left pane project (which is current project). The “Project Comparison - Differences” is completed and the changes are accepted when the dialog is closed with the small yellow marked “x”. The changes will be saved and reflected in device tree.
8. If any safety modules are present in imported project then, a login dialog box is appeared, and user has to enter credentials.

If correct credentials are entered and click “OK”, safety modules are added. On click of “Cancel”, safety modules are omitted.

I/O mapping export and import

Export I/O mapping
To exchange information on I/O mapping only, data can be exported as .csv file. This allows maintenance of I/O data outside Automation Builder, e.g. in MS Excel.

Right-click the “Processor Module” node or “I/O Bus” node in the device tree and select “Export -> I/O mapping”. To export the I/O Mapping for the complete project, e.g. with more than one configured Processor Modules, I/O data of the complete project can be exported “Project -> Export -> I/O mapping”.

Import I/O mapping
A previously exported .csv file can be imported to the project: “Project -> Import -> I/O mapping”.

Comparing projects

You can compare the currently open project with another project – a reference project. The differences in contents, properties, or access rights are detected and shown in a comparison view.

Clicking “Project ➤ Compare” opens the “Project Compare” dialog for you to configure and run the comparison. Then the result is shown in the comparison view “Project Compare - Differences” where the objects are aligned in a tree structure. Objects that indicate differences from the respective reference object are identified by colors and symbols. This is how you detect whether or not the contents, properties, or access rights are different.

For differences in the contents, you can also open the detailed compare view “Project Compare - <object name> Differences” in order to zoom into the object. In the detailed compare view, the contents of the object and reference object are displayed or their source code aligned. The detected differences are marked. Previously opened views are not closed. In this way, you can have any number of comparison views open and read them, in addition to the project compare view.

You can accept the detected differences from the reference project into the current project. This is possible only from the reference project into the open project. To do this, you activate differences (for example in the code) that should be accepted in the current project with the commands ✅, ✗, or ☐ in the active comparison view for accepting. These positions are highlighted in yellow. Make sure that any other open compare views are inactive (write-protected, read-only), therefore, you can activate differences to be accepted in exactly one comparison view only. When exiting the active compare view, if you confirm that the differences that are activated for acceptance are actually accepted into the current project, then the current project is modified.

In order to exit the project comparison completely, close the project compare view.

Creating a comparison view

Requirement: You have made changes in your current project and wish, for example, to compare it with the last-saved version. In the meantime, for example, you have added further POU’s, removed a POU, changed single lines of code or the object properties in function blocks.

1. Select the command “Project ➤ Compare”.
   ⇒ The “Project Comparison” dialog box opens.
2. Enter the path to the reference project, for example the path to the last-saved version of your current project.
3. Leave the activation of the comparison option “Ignore Spaces” as it is.
4. Click on “OK”.
   ⇒ CODESYS opens the comparison view. Title: “Project Comparison – Differences”.
   The Device trees of the current project and the reference project are displayed alongside each other and the changed objects are marked in color.
5. Select an object marked in blue in the tree of the reference project (right). The current project no longer contains this object.
   Click on ✅“Accept Single”
   ⇒ CODESYS adds the object to the tree of the current project (left). The line has a yellow background. ✗ appears in the middle column.
6. Select an object marked in green in the tree of the current project (left). The reference project does not contain this object.
   Click on ✗“Accept Single”
   ⇒ CODESYS removes the object again from the tree of the current project (left). The line has a yellow background. ☐ appears in the middle column.
7. If CODESYS detects changes in the content of an object that is contained in both the current project and the reference project, this is indicated by red lettering. You can then switch to the detailed comparison view for the object by double-clicking on the object.
8. Close the comparison view and answer the query whether the changes made are to be saved with “Yes”.
   ⇒ The changes become effective in the project.

Opening the Detailed Compare View

Requirement: For example, a user modified the code in a POU of the current project. You have performed the project comparison by clicking “Project ➤ Compare”. The project compare view shows this POU highlighted in red in the aligned in the project tree.

1. Double-click the line of the aligned POU versions.
   ⇒ The compare view switches to the detailed compare view of the POU. The modified code lines are highlighted in gray and written in red.

2. Click .
   ⇒ Code lines with changes (red) are extended by two lines: an line with insert (left, green) and a line with delete (right, blue).

3. Click again.
   ⇒ The code line is marked again as modified.

4. Move the mouse pointer to the code line marked as modified and click ✓ “Accept Single”.
   ⇒ The code line from the reference project is activated for acceptance into the current project.

5. Click .
   ⇒ The project compare view opens for the entire project. It is write-protected (read-only) to prevent you from activating differences for acceptance. The link highlighted in yellow above the tree view also indicates this.

6. Click the link: “Project compare view is read only because there are uncommitted changes in another view. Click here to switch to the modified view.”
   ⇒ The detailed compare view opens again. The unconfirmed changes are highlighted in yellow.

7. Click × in the tab of the view and confirm that the changes should be saved.
   ⇒ The detail project view is closed and the POU is overwritten. Now it corresponds to the POU of the reference project. The project view is active again so that you can continue working with project compare.

If you do not click the link, but click × instead to close the editor of the project compare view, then you will also confirm the acceptance of changes into the current project. The detail changes are accepted and then the project compare is closed completely.

See also
- ☞ Chapter 1.3.1.5 “Comparing projects” on page 137
- ☞ Chapter 1.3.1.25.2.4.18 “Command ‘Compare’” on page 905
- ☞ Chapter 1.3.1.5.1 “Creating a comparison view” on page 137

Project archive

Automation Builder supports the creation and the import of project archive files. Archive files contain all relevant project data including the PLC configuration, the project files of the CODESYS and all device descriptions. This allows exchanging Automation Builder projects without taking care of the target environment.
Creation of an archive

☐ The following steps describe the creation archive file from an Automation Builder project:
1. Select "File ➔ Project Archive ➔ Save/Send Archive".
2. Select the information which should be included in the archive file from the list box.

<table>
<thead>
<tr>
<th>Section/Control</th>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Information</td>
<td>Options</td>
<td>Not supported</td>
</tr>
<tr>
<td>selection list box</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Referenced devices</td>
<td></td>
<td>The referenced devices can be selected by expanding the &quot;Referenced devices&quot; item of the list box. It is strongly recommended to include all devices in the project archive to maintain consistency.</td>
</tr>
<tr>
<td>Additional files</td>
<td>-</td>
<td>Not supported</td>
</tr>
<tr>
<td>Comment</td>
<td>-</td>
<td>Opening a control window which allows the input of a comment to the project archive.</td>
</tr>
<tr>
<td>Save</td>
<td>-</td>
<td>Opening a dialog window to determine the path and the file name of the project archive and storing it to the file system.</td>
</tr>
<tr>
<td>Send</td>
<td>-</td>
<td>Not supported</td>
</tr>
<tr>
<td>Cancel</td>
<td>-</td>
<td>Canceling the operation and closing the dialog window.</td>
</tr>
</tbody>
</table>

With [Comment] additional information can be added to the project archive, for example to add a brief description or some information concerning the project.

3. Proceed with [Save...].

It is strongly recommended to keep the default settings.

Section “Options” of the list box is not support. Do not enable this option.

Extraction of an archive

The currently loaded project will be closed automatically when extracting the selected project archive. It is recommended to open a new instance of Automation Builder before starting the extraction process.
The following steps describe the extraction of an archive file and the import to Automation Builder.

1. Select “File ➔ Project Archive ➔ Extract Archive”.
2. Select the desired project file and click [Open].

<table>
<thead>
<tr>
<th>Section/Control</th>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Locations</td>
<td>Extract into the same folder where the archive is located</td>
<td>The project archive will be extracted to the same path where the archive is located.</td>
</tr>
<tr>
<td></td>
<td>Extract into the following folder</td>
<td>Path to which the project archive should be extracted.</td>
</tr>
<tr>
<td>Button ...</td>
<td>Opening a folder selection dialog which allows selecting the desired path.</td>
<td></td>
</tr>
<tr>
<td>Contents</td>
<td>Items</td>
<td>Select the items which should be extracted.</td>
</tr>
<tr>
<td>Comment</td>
<td></td>
<td>Displaying comments included inside the Project archive file.</td>
</tr>
<tr>
<td>Extract</td>
<td></td>
<td>Triggering the extraction process. Automation Builder extracts the archive and creates a project from out the archive. After creating the project Automation Builder checks the version of the project. If the project version and the activated Automation Builder version is not identical the workflow is the same as described in &quot;Opening an Existing Project&quot;.</td>
</tr>
<tr>
<td>Cancel</td>
<td></td>
<td>Closing the “Extract Project Archive” dialog and canceling the extraction process.</td>
</tr>
</tbody>
</table>

### 1.5.6.1.2 User and access rights management

#### User and access rights

The 'User Management' provide functions for defining user accounts and configure the access rights within a project. The rights to access project objects via specified actions are assigned only to user groups, not to a single user account. So each user must be member of a group.

#### User management

Before setting up users and user groups, notice the following: The configuration of users and groups is done in the Project Settings dialog > Chapter 1.5.6.1.2.3 “Project Settings - Users and groups” on page 3243.

- Automatically there is always a group "Everyone" and by default primarily each defined user or other groups are members of this group. Thus each user account at least automatically is provided with defined default settings. Group "Everyone" cannot be deleted, just renamed, and no members can be removed from this group.
- Also automatically there is always a group "Owner" containing one user "Owner". Users can be added to or removed from this group, but at least one user must remain. This group also cannot be deleted and always has all access rights. Thus it is not possible to make a project unusable by denying the respective rights to all groups. Both group and user "owner" might be renamed.
- When starting the programming system resp. a project, primarily no user is logged on the project. But then the user optionally might log on via a defined user account with user name and password in order to have a special set of access rights.
Notice that each project has its own user management!

So, for example to get a special set of access rights for a library included in a project, the user must separately log on to this library. Also users and groups, set up in different projects, are not identical even if they have identical names.

CAUTION!

The user passwords are stored irreversibly!

If a password gets lost, the respective user account gets unusable. If the "Owner"-password gets lost, the entire project might get unusable!

Access right management

User management in a project is only useful in combination with the access right management. Notice the following:

- In a new project basically all rights are not yet defined explicitly but set to a default value. This default value usually is: "granted".
- In the further run of working on the project each right can be explicitly granted or denied resp. set back to default. The access right management of a project is done in the Permissions dialog “Permissions” on page 3242.
- Access rights on objects get "inherited". If an object has a "father" object (example: if an action is assigned to a program object, that is inserted in the structure tree below the program, then the program is the "father" of the action object), the current rights of the father automatically will become the default settings of the child. Father-child relations of objects concerning the access rights usually correspond with the relations shown in the POUs or Devices tree and are indicated in the Permissions dialog by the syntax "<father object>.<child object>".

Example

Action ACT is assigned to POU object PLC_PRG. So in the POUs window ACT is shown in the objects tree indented below PLC_PRG. In the Permissions dialog ACT is represented by "PLC_PRG.ACT" indicating that PLC_PRG is the "father" of ACT. If the "modify" right would be denied explicitly for PLC_PRG and a certain user group, the default value of the "modify" right for ACT automatically also would be "denied".

User management commands

The 'User Management UI' plug-in provides commands for command category 'User Management'.

These are used for:

- Configuration of access rights on the project objects
- Logging on or off to/from the project via a defined user account in order to get the access rights which are associated to this account

The configuration of user accounts and groups is done in the Project Settings subdialog User Management “Chapter 1.5.6.1.2.3 “Project Settings - Users and groups” on page 3243.

By default the following commands are part of submenu 'User Management' in the 'Project' menu: Logon, Logoff, Permissions.

Logon

Symbol: 

This command opens the Logon dialog for logging on to a project or library via a defined user account.
Logging on with a certain user account means to log on with those object access rights which are granted to the group which the user belongs to. The configuration of user accounts and groups is done in the Project Settings subdialog User Management.

To log on select the project or an included library from the selection list in the Project/Library field. Enter User name and Password of a valid user account, noticing that each project or library has an own user and access rights management. Log on with OK.

If already another user is logged on the project, this one will be logged out automatically by the new log-on action.

When you are logged on to a project or library and try to perform an action for which you have no right, automatically a Logon dialog will be opened, giving the possibility to log on with another user account provided with the appropriate rights.

The status bar always displays which user currently is logged on the project.

Logoff

Symbol: 

This command logs off the currently logged on user. If no user had been logged on to the currently opened project or to a referenced library an appropriate message will appear when trying to log off.

If the user currently is logged on to more than one project or referenced library (not necessarily with the same user account) a Logoff dialog will appear when trying to log off.

From the Project/Library selection list choose those project/library for which you want to log off.

The name of the Current user is displayed just for information.

The status bar always displays which user currently is logged on the project.

Permissions

This command opens the Permissions dialog, where the rights to work on objects or to perform commands in the current project can be configured.

Any changes made in this dialog will be applied immediately.
The Actions window displays all possible rights, that is, all actions which might be performed on any object of the current project.

The tree is structured in the following way:

- **Top-level** see the names of some categories, which have been set up just for the purpose of optical structuring the rights management. They are grouping concerning the execution of Commands, the configuration of User accounts and Groups, the creation of Object Types, the viewing, editing, removing and handling of child objects of Project Objects.

- **Below each category node** there are nodes for the particular actions which might be performed on the command, user account, group, object type or project object. These nodes also only have optical function. Possible Actions:
  - execute (execution of a menu command)
  - create (creating a new object in the current project)
  - add or remove children (adding or removing of "child" objects to an existing object)
  - modify (editing an object in an editor)
  - remove (deleting or cutting an object)
  - view (viewing an object in an editor)

Below each action node find the possible targets, that is project objects, of the respective action.

The Permissions window provides a list of all currently available user groups (except the "Owner" group) and a toolbar for configuring rights to a group.

Select the group and configure it’s permissions.

Left to each group name one of the following icons indicates the currently assigned permission concerning the target which is currently selected in the Actions window:

- The action(s) for the target(s) currently selected in the Actions window are granted for the selected group.
- The action(s) for the target(s) currently selected in the Actions window are denied for the selected group.
- The right to perform the action(s) which are currently selected for the selected target(s) in the Actions window, has not been granted explicitly, but is granted by default, for example because the corresponding right has been granted to the "father" object. (Example: The group has got the right for object "myplc", thus it by default also has got it for object "myplc.pb_1".) Basically this is the default setting for all rights which not explicitly have been configured.
- The right to perform the action(s) which are currently selected for the selected target(s) in the Actions window, has not been denied explicitly, but is denied by default, for example in case because the corresponding right has been assigned to the "father" object.

If currently multiple actions are selected in the Actions window, which do not have unique settings referring to the currently selected group, no icon will be displayed.

To configure the rights for a group select the desired action(s) and target in the Actions window and the desired group in the Permissions window. Then use the appropriate button in the toolbar of the Permissions window:

- Grant: Explicit granting.
- Deny: Explicit denying.
- Clear: The currently granted right for the action(s) currently selected in the Actions window will be deleted, that is set back to the default.

**Project Settings - Users and groups**

The Project Settings dialog in category 'Users and Groups' provides three subdialogs for the user management for the current project: Users, Groups, Settings. For a general description on users and access rights management see help page "Chapter 1.5.6.1.2.1 “User and access rights” on page 3240."
Users dialog

The currently registered users are listed in a tree structure. The ownerships of each user is displayed and each user is a member of a group by default. Chapter 1.5.6.1.2.1 "User and access rights" on page 3240.

Define a new user account

1. Click “Add” to open the “Add User” dialog.
2. Define the user credentials and click OK to set up the new user. If there are incorrect entries (no login name, password mismatch, user already existing) you will get an appropriate error message.

Modify a user account

Click “Edit” to open the “Edit User” dialog. The entry fields are the same as in the “Add User” dialog. The password fields however - for security reasons - will show 32 * characters. After having modified the desired entries close the dialog with OK to get applied the new settings.

Remove user accounts

Enable the entries to be removed in the Users list and click “Remove”. Note that you will get no further inquiry! An error message appears if you try to delete all users from a group. At least one entry must remain.

Groups dialog

Add a group

The currently available groups are displayed in a tree structure. A member also might be a group.

1. Click “Add” to open the “Add Group” dialog.
2. Define a name for the new group and enable all entries (single users or groups) which should be members of the new group.
3. Click OK to set up the new group. If there are incorrect entries (no name defined, group already existing, in Members having selected a group which would cause a "group cycle", you will get an appropriate error message.

Modify a group

Click “Edit” to open the “Edit User” dialog. The entry fields are the same as in the ‘Add Group’ dialog (see above). After having modified the desired entries close the dialog with OK to get applied the new settings.

Remove groups

Enable the entries to be removed in the groups tree and click “Remove”. Note that you will get no further inquiry! The members of the deleted groups will remain unmodified. An error message appears if you try to delete the groups “Everyone” and/or "Owner”.

Settings dialog

The following basic options and settings concerning the user accounts can be made:

- Maximum number of authentication trials: If activated, the user account will be set invalid after the specified number of trials to log in with a wrong password. If not activated, the number of erroneous trials is unlimited. Default: option activated, number of trials: 3; permissible values: 1-10.
- Automatically log out after time of inactivity: If activated, the user account will be logged out automatically after the specified number of minutes of inactivity (no user actions via mouse or keyboard registered in the programming system). Default: option activated, time: 10 minutes; permissible time values: 1-180 minutes.
1.5.6.1.3 Later change-over of target system

Target change from AC500 V2 to AC500 V3

Customer libraries

CODESYS for AC500 V2 products contains different types of libraries:

- Standard CODESYS libraries
- Specific AC500 libraries
- Customer libraries

The Standard CODESYS libraries and the AC500 libraries are automatically converted during a target change from AC500 V2 to AC500 V3. The customer libraries have to be converted manually using the Library Converter integrated into the Automation Builder installation:

1. In Automation Builder click “File” ➔ “Open project”.
2. Select the CODESYS library for AC500 V2 products which has to be converted.
3. After conversion of the library, open the view POU in the device navigator and double-click “Project Information”.
4. In the dialog enter “Company”, “Title” and “Version” in the specific fields, then, open the “Properties” tab. For the target change the new “Key” “CoDeSysV2Library” has to be added. Under “value”, enter the name of the CODESYS library and click the “Add” button.

Click “File” ➔ “Save project” and install into the library repository.

1.5.6.1.4 Firmware identification and update

Version information

Information on the firmware versions of the processor modules or communication modules, is provided on the “Version information” tab.
Remarks:

- The “Version information” tab displays the version identified on the device and the version provided with Automation Builder.
- The firmware on the devices must match to the Automation Builder version. Upgrade or downgrade to version supplied with Automation Builder is recommended (especially for CPUs) to ensure correct functionality.
- The firmware type can be changed to the type required by the hardware configuration for devices that support changing the firmware type. E.g., the onboard field bus communication modules of PM959 that may be used as PROFINET, Ethernet or EtherCAT communication module.

### State icons

<table>
<thead>
<tr>
<th>Icon</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="Green" /></td>
<td>Firmware version on device matches version supplied with Automation Builder.</td>
</tr>
<tr>
<td><img src="image" alt="Yellow" /></td>
<td>Firmware version (or type) on device is different from version supplied with Automation Builder. Upgrade/downgrade to version supplied with Automation Builder is recommended.</td>
</tr>
<tr>
<td><img src="image" alt="Yellow" /></td>
<td>Only for communication modules if CPU firmware must be updated first. This happens when CPU firmware has version below 2.5.0.0. Firmware version (or type) on device is different from version supplied with Automation Builder. Upgrade/downgrade to version supplied with Automation Builder is recommended.</td>
</tr>
<tr>
<td><img src="image" alt="Red" /></td>
<td>Identified device is different from configured device, thus no firmware update is possible. Happens only for Communication Modules.</td>
</tr>
<tr>
<td><img src="image" alt="No icon" /></td>
<td>Firmware of device is not updateable or no newer firmware than the initial version is available.</td>
</tr>
</tbody>
</table>

The [Update Firmware] button to download the new firmware is only enabled if there is updateable firmware.

### AC500 V3 firmware installation and update

The PLC firmware can be updated via Automation Builder.
This is also necessary for commissioning V3 CPUs.

A very new CPU has no pre-installed firmware. To guarantee the authenticity of delivered AC500 firmware, V3 CPUs are delivered with a boot loader only. You need to download a valid firmware to the CPU. After download, the functionality of the CPU is given.

- An Automation Builder project with an AC500 V3 CPU is open.
- CPU is in "stop" mode or shows uPdAtE (update) on the display.
- After update the CPU shows either donE or StoP on the display
- For new modules: IP address is set. (The default IP address is 192.168.0.10)

1. Double-click CPU “PLC_AC500_V3”.
2. Select “Version information”.

3. Select [Update Firmware].
   - While the update process is running, the RUN and ERR LEDs are toggling, i.e., they are flashing alternating.
4. Wait for the PLC to finish the update.
   A completed update is indicated by a message on the display. Either donE, or StoP.

**NOTICE!**
Do not disconnect the power supply during the update process! The PLC could be damaged.

- StoP indicates a restart has been performed by the CPU. When donE is displayed sometimes it is necessary to re-boot the CPU manually, e.g., by powering-off. Manual re-boot might be, e.g., for some older CPU versions or if downgrading to an older firmware version according to application settings.

The CPU display shows "stop" after re-boot. The update process is finished.
5. If necessary, refresh the version information by switching to another tab and back.

⇒ Successful firmware update:

![Firmware Update Status](image)

**Behavior of LEDs during firmware update**

<table>
<thead>
<tr>
<th>LED</th>
<th>LED flashes</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>RUN and ERR</td>
<td>Toggling</td>
<td>Update pending</td>
</tr>
<tr>
<td>RUN</td>
<td>Flashing slow</td>
<td>Done successful</td>
</tr>
<tr>
<td>ERR</td>
<td>Flashing slow</td>
<td>Done failed</td>
</tr>
</tbody>
</table>

**Update CI52x-modbus firmware**

Requirement: A firmware update file is available, e.g. AC500_CI52x_Firmware_V3.2.8.bin.

*The CI52x Modbus firmware update is only available in the Automation Builder IP Configuration Tool.*
Installation of the IP configuration tool

1. In Automation Builder click “Tools ➤ Installation Manager” to start the Installation Manager.
2. Close any other running instances of Automation Builder. Then, click “Modify” in the Installation Manager.
3. Select the option “IP Configuration Tool” from the list and start the installation of the IP Configuration Tool.

Firmware update procedure

1. In the IP Configuration Tool click “Scan” to initialize a device scan.
2. From the list select the CI52x-MODTCP device(s) which shall be updated and click “FW Update”.
3. Select the firmware update file (e.g. AC500_CI52x_Firmware_V3.2.8.bin) to initialize a signature check and start the update procedure.
4. After the update, click “Scan” again to retrieve the firmware version of the device.

Troubleshooting

After the IP Configuration Tool has been installed, the firmware update of the CI devices can be initialized. If the CI firmware update fails, check the troubleshooting hints and follow the instructions.

General hints

- Close all unused applications on the update PC and do not open Automation Builder or any other applications during the firmware update.
- Stop the communication between AC500 PLC and the CI52x devices and disconnect the Ethernet connection of the update PC and the CI Modbus device(s).
- Do not close the IP Configuration Tool during a firmware update and do not switch off a CI Modbus device during the firmware update.

During a firmware update the operation of the device(s) is stopped. After the update, all outputs are set to zero.

Erroneous firmware update

<table>
<thead>
<tr>
<th>Error</th>
<th>Solution</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Error 1: Package Timeout</strong></td>
<td>Locate the PC on which the update is performed as near as possible to the stationed CI Modbus devices. Avoid network switches.</td>
</tr>
<tr>
<td>Due to a primitive firmware update protocol a fast and stable network connection is required. Otherwise the update packages cannot be transferred within the requested time and a timeout occurs.</td>
<td></td>
</tr>
<tr>
<td><strong>Error 2: Unable to read device status</strong></td>
<td>Rescan and repeat the update. If this doesn’t work, power cycle the device and retry the update.</td>
</tr>
<tr>
<td>After the firmware update the IP Configuration Tool reads out the status of the updated device in order to check if the update was successful.</td>
<td></td>
</tr>
</tbody>
</table>
### Error 3: IP is not unique
If more than one device holds the same IP address, a firmware update is not possible as the update command is IP based. Correct the IP address, rescan and repeat the update. If this doesn't work, power cycle the device and retry the update.

### Error 4: Internal Error
An internal error on the CI52x Modbus device occurred during the firmware update. Rescan and repeat the update. If this doesn't work, power cycle the device and retry the update.

### Error 5: Cannot connect to device
The TCP communication is not sufficient for a connection. Increase the connection quality. See Error 1: Package Timeout.

---

### Signature check failed
After the selection of the firmware file (*.bin) a signature check is performed. If either the firmware file or the signature file is corrupt, the signature check fails. In the event of an erroneous signature check, perform the following steps:

- Ensure the signature file is stored in the same directory as the firmware file.
- Check the file names. The name of the signature file must be the same as the firmware file + attached ".sig".

#### File names
- Name of the firmware file: c:\AC500\AC500_CI52x_Firmware_V3.2.8.bin
- Correct name of the signature file: c:\AC500\AC500_CI52x_Firmware_V3.2.8.bin.sig
- Wrong name of the signature file: c:\AC500\AC500_CI52x_Firmware_V3.2.8.sig

### Indeterminate device firmware version
If the firmware version of the device cannot be determined, an error occurs. In this case, check that the device and the update PC are located in the same subnet and ping the device. If the ping is successful you can use the IP Configuration Tool to retrieve the device firmware version.

<table>
<thead>
<tr>
<th>PC</th>
<th>Device</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>192.168.10.71 / 255.255.255.0</td>
<td>192.168.14.10 / 255.255.255.0</td>
<td>ERROR</td>
</tr>
<tr>
<td>192.168.10.71 / 255.255.0.0</td>
<td>192.168.14.10 / 255.255.0.0</td>
<td>OK</td>
</tr>
</tbody>
</table>

---

### Migration of third party devices
After an update of Automation Builder the device repository contains only ABB devices. The third party devices which were installed into previous versions of Automation Builder are not automatically installed in the newest version profile. This has to be triggered by the user.

*The feature "Migrate third party devices" is available as of Automation Builder 2.1.1.*
1. Click “Tools” in the main menu of Automation Builder.
2. Click “Migrate third party devices” in the drop-down list.
   ⇒ The window Version profile selection appears.
3. Select a version profile in the drop-down list containing previous Automation Builder / Control Builder Plus profiles. The active profile does not appear in the list.

After selection of a previous version profile, all the third party devices which have been installed inside this version profile are listed.

![Version profile selection window](image)

It is not possible to select or deselect some third party devices. Importing will affect all the third party devices which are listed in the list view.

- Click [Import].
During the migration the message window displays success or failure of device migration.

In case of failure during the migration the affected third party device description has to be installed manually via main menu “Tools ➔ Device Repository ➔ Install”.

In the status bar, the third party device which is on Migrating: <...> is displayed on the left side. The import operation can be cancelled by clicking the “Click here to CANCEL this operation” link on the right side of the status bar. This becomes effective when the migration of the just migrating third party device is finished.

- Click [Close] button of the Version profile selection dialog to close the dialog.

### 1.5.6.1.6 Advanced IO device handling

Automation Builder provides the Advanced IO Device Handling feature for configuring identical IO device types at multiple instances.

This feature is supported by the following commands that works with IO devices only.

- Generate DUT
- Map to Existing DUT
- Release DUT mapping

These commands work on individual nodes and on CI (communication interface) level nodes.

**Generating DUT**

Each device generates two DUTs. One for the input and one for the output. Some devices contain only input or output type. In such cases, the device generates only one DUT of the relevant type.
- Right-click on the desired IO device and select “Generate DUT” to generate a DUT for an IO device.

The following example shows how to generate DUTs at CI level node.

- In the device tree, right-click on a master node such as PNIO_Controller and select “Generate DUT” to create DUTs for the child nodes.
- The DUTs of child nodes are generated in “Applicationressing DUT Generated Items” folder.
- Generated DUT considers channels with BYTE datatype as members. If channels with BYTE datatype are not present in the given hierarchy, it adds the members with another higher datatype.
- Channels with BOOL datatype are not considered.

Mapping to existing DUT

This command is enabled for the IO device when the IO device is not mapped and when DUTs of matching size (calculated based on device channel list) are available in “Applicationressing DUT Generated Items” folder.

1. Right-click on an IO device and select “Map to Existing DUT”.
   ➜ Enter Instance Name dialog is displayed.
2. Enter the instance name which satisfies IEC naming validations and unique name in global scope.
3. Click “OK” to create a global variable associated with the mappings in DI (PRG).
   If you want to view mapped instances, double-click “DI (PRG)”.

With the ‘Map to Existing DUT’ command:

- Any device can be mapped only to one input DUT and one output DUT. If you have already mapped an input DUT, only the output DUT is shown in the options list and vice-versa.
- Mapping is also supported at CI level nodes. To create global variables for CI level nodes, the address of the first child is considered.

Releasing DUT mapping

This command is enabled on an IO device only when an IO device is mapped either to input, output or both DUTs. You can use this command to release (or revert) mappings and to delete global variables created during ‘Map to Existing DUT’.

Right-click on an IO device and select “Release DUT Mapping”. The mapped DUT instance is deleted.

Using DUT variables in CODESYS application

1. In the Automation Builder project, double-click “Application” to launch CODESYS application.
   ➜ CODESYS application is launched. CODESYS application contains mapped DUT instances.
2. Double-click “PLC_PRG” to create DUT variables.
3. Add DUT variables based on mapped DUTs.

For further information on mapping DUTs, see section Chapter 1.5.6.1.6.2 “Mapping to existing DUT” on page 3253.
For example, in the PLC_PRG, add analog I/O and digital I/O. If you insert a dot at a position where an identifier should be inserted, then a selection list is open, offering all the input and output variables which are found in the project.

After adding DUT variables, rebuild the program in CODESYS application using “Project ➔ Rebuild”.

**Support for CI level node**

The user can create DUTs for the entire hierarchy of CI level node (for example, IO_BUS), by right-clicking on the desired CI level node and by selecting “Generate DUT”. Further, all the DUTs are generated in “Application ➔ App ➔ IO_Device_Generated_Items” folder.

- The command generates DUT for the node itself and also for all child nodes.
- The DUT generated for the CI level node contains generated DUTs for the child nodes as their members.
- For every execution, the command checks, if any new child node is added and generates DUT.

If you delete child nodes in CI level node (for example, IO_BUS), the DUTs generated for these child nodes are not deleted automatically. You should delete the DUTs manually in the “Application ➔ App ➔ IO_Device_Generated_Items” folder if desired.

**Configuration check**

Configuration check for size is enabled to ensure that all devices are mapped with DUTs of the correct size. In case of any changes in the mapped DUT, configuration check verifies the size of the DUT. If it fails, an error message is displayed in Automation Builder messages window and does not allow to launch the application. This check can be performed in “Create configuration data”.

**1.5.6.2 PLC devices and components**

**1.5.6.2.1 Device repository**

The Device Repository of Automation Builder manages the pool of devices that can be used in the PLC configuration.

You install or uninstall devices in the “Device Repository” dialog box. The system installs a device by reading the device description files, which define the device properties for configurability, programmability, and possible connections to other devices.

You can use the devices provided in the device repository by adding them to the device tree of your project.
1. Click “Tools ➔ Device Repository”.
   ⇒ The “Device Repository” dialog box opens.

![Device Repository dialog box]

- **[Edit Locations]**: Changes the default repository location. The devices can be managed at different locations.
- **[Install]** / **[Uninstall]**: Installs or uninstalls devices.
- **[Renew device repository]**: Updates the device list, e.g. after uninstallation of a device.
- **[Details]**: Provides technical details on the selected device.

2. Select the install location. “System Repository” is set by default.

**Installing devices**

The device repository cannot be changed manually, e.g. by copying or deleting files. Use always the Device Repository dialog to add or remove devices.
1. Click [Install] and select the appropriate file format.  
⇒ The “Install Device Description” dialog box opens.

2. Select the file path of the device description.

3. Select the file type filter of the required device description.  
⇒ All device descriptions of the selected file type are listed.

4. Select the required device description and click “Open”.  
⇒ Automation Builder adds the device description to the matching category of your device repository.

   If errors occur during installation (for example, missing files that are referenced by the device description), then Automation Builder displays them in the lower part of the device repository dialog box.

   During the installation the device description files and all additional files referenced by that description will be copied to an internal location. Altering the original files will have no further effects to an internal location.

   The changes take only effect after reinstalling the corresponding device(s). The version number shown in the information section of the device should be verified.

Uninstalling devices

Select the device you want to remove and click [Uninstall].

The device is removed from the list.

Uninstalled devices which are used in existing projects are indicated by the symbol 🛑. The device will not be configured properly.
1.5.6.2.2 PLC start-up

A fast online program modification of the user program is possible without interrupting the run-
ing operation. If data areas should be saved during power OFF/ON, they can be stored in the
Flash EPROM. An optional battery saves data in the RAM.

Initialization of AC500 V3 CPU

To initialize an AC500 V3 CPU, you need to download the firmware in advance.

A very new CPU has no pre-installed firmware. To guarantee the authenticity of delivered
AC500 firmware, V3 CPUs are delivered with a boot loader only. You need to download a valid
firmware to the CPU. After download, the functionality of the CPU is given.

An Automation Builder project with an AC500 V3 CPU is open.

1. Switch on the CPU.
   ⇒ On the CPU display "Init" is shown.

   In Automation Builder the activated PLC is not displayed under “Tools ➔ IP-Configuration”.

2. Automation Builder: Double-click the node “PLC_AC500_V3” and select the IP address
   "192.168.0.10":

3. Click “Execute update”.
   ⇒ Device: The Run and Error LEDs are still blinking.
   
   Wait until only the Run LED is blinking and the device display shows "Stop".
   Then, reboot the PLC.
   After Power Cycle the PLC ends in Stop mode.

4. Automation Builder: Click “Tools ➔ IP-Configuration” and click “Scan”.


5. Double-click the node “PLC_AC500_V3” and enter the IP address:

- Click “Enter” to start the scan process.
- After a successful scan, the IP address is enhanced to "192.168.0.10 (active)".

6. Login to the PLC: “Online ➔ Login”.

7. To read out firmware information, double-click the node “PLC_AC500_V3”. In the tab PLC Shell navigate to the command line and type "rtsinfo".

**PLC runtime licensing**

The use of some libraries and devices require the PLC to have a runtime license. If you purchased such a license, activate the license in Chapter 1.5.6.2.2.2.1 “Activating a runtime license via license key” on page 3258.

If you want to test device functionality or library features in advance, you can activate a demo license in advance in Chapter 1.5.6.2.2.2.2 “Activating a demo license” on page 3262.

The license status of a PLC can be displayed at any time in Chapter 1.5.6.2.2.2.5 “View license information” on page 3265.

**NOTICE!**

After removing a Wibu SD memory card (which holds the AC500 runtime license), the PLC system moves into 'Stop' mode after 24 h.

Ensure to insert the Wibu SD memory card at the time.

**Activating a runtime license via license key**

The use of some libraries and devices require the PLC to have a runtime license.

- PC and PLC are connected. In case of no connection, perform the activation via SD memory card in Chapter 1.5.6.2.2.2.3 “Licensing via SD memory card” on page 3262.
There is a connection to the Internet. In case of no connection, perform the activation on another PC with internet connection. Chapter 1.5.6.2.2.1.1 “Activation without internet connection” on page 3259.

1. Right-click on the PLC and select “PLC runtime licensing” from the “Runtime Licensing” menu.
   ⇨ A wizard starts. Follow the instructions.

2. Enter the license activation key and select “Next” to finish the licensing procedure.
   ⇨ The license is activated on the PLC device.

   If the license shall be used on another PLC device, the installed license can be returned. Chapter 1.5.6.2.2.2.4 “Returning a license” on page 3263.

Activation without internet connection

1. If an error occurs when communicating with the ABB license server, or if Automation Builder is running on a PC without internet connection, then it is possible to manually complete the ABB license server interaction by using another PC (with internet connection).

2. In the error dialog select “Next” and save the license activation request file to a storage location the other PC can access, e.g. a file share.
3. In the dialog the web address of the ABB license server is displayed ([http://lc.codemeter.com/32838/depot/index.php](http://lc.codemeter.com/32838/depot/index.php)). From the PC with internet connection, upload the license activation request file.

4. After the upload, download and save the license activation file from the ABB license server. Transfer this file to the PC without Internet connection.

5. Select “Next” to continue the license activation process. Click “Cancel” to continue the license activation process at a later time (see Offline activation [Further information on page 3261](#)).
6. Select “Browse” and select the license activation file (*.WibuCmRaU) from the defined storage location.
   ⇒ The license is validated by the ABB license server and afterwards activated on the PLC device.
   
   If the license shall be used on another PLC device, the installed license can be returned © Chapter 1.5.6.2.2.2.4 “Returning a license” on page 3263.

7. To complete the licensing process, a license receipt file must be uploaded to the ABB license server.
   
   Save the license receipt file and upload it manually from a PC with internet connection to
   ⇒ A license confirmation is returned.

---

**Offline activation**

If the runtime licensing process was closed between saving the license activation request file and obtaining the license activation file from the ABB license server, perform an offline activation:

1. Right-click on the PLC node and select “PLC runtime licensing” from the “Runtime Licensing” menu.
   ⇒ A wizard starts. Follow the instructions.

2. Select the option “Complete offline licensing process”.

3. Select “Browse” and select the license activation file (*.WibuCmRaU) from the defined storage location.
   ⇒ The license is activated on the PLC device.
   
   If the license shall be used on another PLC device, the installed license can be returned © Chapter 1.5.6.2.2.2.4 “Returning a license” on page 3263.

4. To complete the licensing process, a license receipt file must be uploaded to the ABB license server.
   
   Save the license receipt file and upload it manually from a PC with internet connection to
   ⇒ A license confirmation is returned.
Activating a demo license

It is possible to try out device features or library features by using a Demo license on the PLC. With this, you can use the features for a limited time period.

☐ PC and PLC device are connected. In case of no connection, perform the activation via SD memory card ➔ Chapter 1.5.6.2.2.2.3 “Licensing via SD memory card” on page 3262.

☐ There is a connection to the Internet. In case of no connection, perform the activation on another PC with Internet connection ➔ Chapter 1.5.6.2.2.2.1.1 “Activation without internet connection” on page 3259.

1. Right-click on the PLC node and select “PLC runtime licensing” from the “Runtime Licensing” menu.
   ➔ A wizard is started. Follow the instructions.

2. Select the option “Create a demo license” and click “Next” to finish the licensing procedure.
   ➔ The demo license is validated by the ABB license server and afterwards activated on the PLC device.

Licensing via SD memory card

When you have no connection between your PC and the PLC device the licensing procedure can be effected via an SD memory card.

On the PC:
Create a license request

☐ There is a connection to the internet.

☐ The SD memory card can be used with AC500 V3 products.

![NOTICE!]

If an SDCard.ini file is stored on the SD memory card, the file will be overwritten.

1. Place the SD memory card in the PC.
2. Right-click on the PLC node and select “Prepare PLC license SD memory card” from the “Runtime Licensing” menu.
3. From the filesystem select the root folder of the SD memory card.
   ➔ A success message is displayed when the creation of the SD memory card files is completed.
   
   The license request files are stored to the selected folder.

On the PLC:
Transfer the license data

1. Insert the SD memory card into the PLC device and reboot the PLC.
   ➔ When the license request file is successfully created by the PLC, “done” is shown on the display of the PLC.
2. Remove the SD memory card from the PLC.
For this action, internet connection is required.

1. Place the SD memory card into the PC.
2. Open the PLC project in Automation Builder. Ensure the PLC is logged out.
3. Right-click on the PLC node and select “PLC runtime licensing” from the “Runtime Licensing” menu.
   - A wizard is started. Follow the instructions.
4. Enter the license activation key.
5. From the filesystem, select the root folder of the SD memory card.
   - The previously created license request files are sent to the ABB license server. A license activation is created on the SD memory card.
6. Remove the SD memory card from the PC.

On the PLC: Complete license activation for the PLC

1. Insert the SD memory card into the PLC device and reboot the PLC.
   - done is displayed on the PLC if license activation was successful.
2. Remove the SD memory card from the PLC

On the PC: Complete license activation on the license server

To complete the licensing process, the license receipt file must be uploaded to the ABB license server.
1. Place the SD memory card into the PC.

The license receipt on the SD memory card is located in the subfolder license

- A license confirmation is returned.

Returning a license

NOTICE!
After returning a AC500 runtime license, the PLC system moves into ‘Stop’ mode after 24 h.
A license which has been installed on a PLC device can be returned and installed on another PLC device.

PC and PLC device are connected. In case of no connection, perform the activation via SD memory card “Returning a license via SD memory card” on page 3264.

1. Right-click on the PLC node and select “Return active license” from the “Runtime Licensing” menu.
   ⇒ A wizard is started. Follow the instructions.
2. Enter the license activation key and click “Return license”.
   ⇒ The results of the return process will be displayed in the dialog.

The license from the PLC device is removed and can be used now for another PLC device.

When the PLC is not connected to the PC (PLC logged out) it is possible to return a license via SD memory card.

1. Place the SD memory card into the PC.
2. Right-click on the PLC node and select “Return active license” from the “Runtime Licensing” menu.
   ⇒ A wizard is started. Follow the instructions.
3. Enter the license activation key and click “Return license”.
4. Click “Browse” and select the root folder of the SD memory card.
   ⇒ Returning of the license is started.
5. Place the SD memory card in the PLC device and reboot the PLC.
   ⇒ The license from the PLC device is removed and can be used now for another PLC device.

6. To complete the licensing process, a license receipt file must be uploaded to the ABB license server.
   
   Save the license receipt file and upload it manually from a PC with internet connection to
   ⇒ A license confirmation is returned.

View license information

To view the license information of AC500 V3 products, right-click on the PLC node and select “Show license information” from the “Runtime Licensing” menu. The project is scanned for required licenses. If you are logged in to a PLC, then the licenses available on the PLC are displayed. A missing required license is highlighted.

![License Information](image)

Connection of devices

All installed devices that are available in Automation Builder are listed in the Device Repository Chapter 1.1.9 “Device repository” on page 31.

Configuring devices

Modify your Automation Builder project by adding device objects. Preset items can be replaced in the same way.
1. In the device tree, right-click an item node. Select “Add object”.

![Image of Add object dialog]

2. Select the desired object and click [Add object].

3. Double-click the new object in the device tree to configure the device settings. Depending on the selected item different configuration tabs are available.

![Image of A523 Parameters]

**Update of AC500 devices**

Perform a firmware update to update AC500 V3 devices. § Chapter 1.17.2.6 “AC500 V3 firmware installation and update” on page 59.

**Comparing objects**

To compare similar objects within a project (such as the project configuration) select both objects. Right-click and select Compare Objects to see the differences.
IP settings

Configuration of the IP settings with the LED display

The IP settings for the PLC can be set directly on the processor module via keypad and LED display.

See LED Display % Chapter 1.5.5.1.4.4.3 “Configuration” on page 3123

Configuration of the IP settings with IP configuration tool

With the IP configuration tool a network scan can be executed.

Further, the IP settings of found PLC devices can be changed if required.
IP Configuration Tool
1. From the menu, select “Tools → IP-Configuration”.
2. Click [Scan] to trigger the scan process for devices in the network.
   ⇒ A progress bar shows the progress.
3. To change the configured IP settings, select a connected device from the list.
   ⇒ The IP settings are displayed below the list and can be edited.
4. Click [Send Configuration] to transmit the IP settings to the corresponding device.

Processor modules with 2 onboard Ethernet interfaces:
- Configure only 1 interface as default gateway.
- In case of 2 configured interfaces only the latter will be executed.

**Blink functionality**
Activates flashing of background light of AC500 LED display in Automation Builder via IP configuration tool.
1. From the menu, select “Tools → IP-Configuration”.
2. Click [Scan] to trigger the scan process for devices in the network.
   ⇒ A progress bar shows the progress. The IP settings of a selected device is displayed below the list and can be edited.
3. Adjust your desired time and click [Blink] to activate flashing.
4. Click “Save Configuration” to transmit the IP settings to the corresponding device.
Configuration of communication via Ethernet (TCP/IP)

Programming via Ethernet is only possible on a PC with Ethernet board and installed network. Programming can be done via the internal Ethernet communication module.

Programming via internal (onboard) Ethernet communication module:

1. Right-click the top node “PLC_AC500 <...>” and select “Communication Settings” from the context menu.
   ⇒ Dialog box Communication Settings <...> appears.

2. Enter your PLC IP Address and click [OK].
Enter PLC IP address by scanning devices

1. Right-click the top node “PLC_AC500 <...>” and select “Communication Settings” from the context menu.
   ⇒ Dialog box Communication Settings <...> appears.

2. Click [...].
   ⇒ Dialog box Pick IP Address <...> appears.

3. Click [Scan], select your desired PLC and click [OK].
   ⇒ Entry is transferred to the dialog box Communication Settings <...>.
   
   Click [OK].

4. Click to log in the “PLC_AC500_V3” project.

Enter PLC IP address by [Advanced Settings...]

If a remote gateway instead of a local one has to be used it can be configured in the [Advanced Settings...].
1. Right-click the top node “PLC_AC500 <...>” and select “Communication Settings” from the context menu.
   ⇒ Dialog box Communication Settings <...> appears.

2. Enable checkbox Use advanced settings and click [Advanced Settings...].
   ⇒ Tab “Communication Settings” opens.

3. Check gateway or change if required.
   ⇒ Successful connection is indicated by green dot on the gateway icon.

4. **Manual entry of the IP address.**
   Check IP address or change if required.

5. Press ENTER to confirm changed IP address.
   ⇒ Successful communication is indicated by green dot on the PLC icon.
6. Or instead of the last two steps:

Set the IP address via a scan.

Click [Scan Network], select your desired PLC and click [OK].

Successful connection is indicated by green dot on the gateway icon.

7. Click to log in the “PLC_AC500_V3” project.

Troubleshooting Hints for IP Configuration Tool/Plugin

On a standard Windows 7 installation without third party firewall or security tools installed the IP Configuration Tool/Plugin should work as designed.

The Automation Builder setup installs rules or exceptions for the built-in Windows firewall to allow IPConfig to receive the responses for the IPConfig scan.

To check the Windows firewall is set correctly check the firewall settings in the control panel.

Windows 7 On the network that is used for communication with the PLC, set “Incoming connections” to “Block all connections to programs that are not on the list of allowed programs”.

2020/12/10 3ADR010583, 1, en_US 3273
If a third party firewall is used these exceptions must be configured manually.

Either exceptions for applications can be entered: Automation Builder and IP Configuration Tool must be added as application.

Or the protocol and the port number must be given (for IPConfig: UDP protocol and port number 24576).

### 1.5.6.2.3 Processor modules

**Configure a processor module in the device tree**

1. Add a Processor Module to your project.
2. In the device tree, double-click the “PLC_AC500_V3 <...>” node.
3. Select the “PM56xx Parameters” tab to configure the parameters for the Processor Module (see Chapter 1.5.6.2.3.4.1 “Floating point values” on page 3276).

Use the “PM56xx Hardware” tab for later on changes of “Terminal Base Type” or “Processor Module Type” (see Chapter 1.5.6.2.3.2 “Changing the processor module type” on page 3274).

### Changing the processor module type

If it is required, to change the type of Processor Module after the configuration has been finished already, it is possible to change the type of Processor Module later on. By this, all configurations of fieldbusses and interfaces that are identical on the new Processor Module are kept.
1. Close CODESYS.

2. Double-click the PLC_AC500_V3 <...> node.

2. Open the “PM56xx Hardware” tab and select the new PM5xx Type from the drop-down list.

3. Click [Change PM / TB type].

Changing the terminal base type

1. Open the “PM56xx Hardware” tab and select the new Terminal Base Type from the drop-down list.

2. Click [Change PM / TB type].

Diagnosis History

The 'Diagnosis History' view provides an overview of the current and past system events that resulted in a diagnosis event.

- Incoming diagnosis events are indicated with +.
  After the problem that causes a diagnosis event has been resolved, this diagnosis event is indicated automatically with -. 
- Alarm events, e.g. PROFINET alarms are indicated with ⚠. 
  In the 'Diagnosis' view the user can acknowledge an alarm. Note that an alarm event can be acknowledged though the problem that causes the alarm still persists. 
  The acknowledge action is indicated with ✔ on the concerning event entry. If the icon changes to ✔, the acknowledge action has been completed by the PLC.

The following buttons are available in the 'Diagnosis History' view:

- Start/Stop refresh: Enables or disables the automatic refresh mode. In refresh mode new diagnosis events will be displayed automatically. Only the last 100 entries are shown in this view, the latest events on top of the list.
- Get next entries: Adds the previous (older) 100 diagnosis events at the bottom of the list.
- Export complete history: Creates a csv file with all events from the diagnosis history (not only the visible ones).
Parameters of the processor module

Floating point values

A calculation with floating points can lead to the following values:

0 (zero)

If a calculation results in an underrun, the value is set to 0 (result near 0, but not presentable). Depending on the sign bit, it can be a positive zero or a negative zero. The operator "=" of -0 and 0 returns TRUE.

Infinity

If a calculation results in an overrun, the value is set to Infinity (the result is not presentable). Depending on the sign bit, it can be a positive infinity (Infinity) or negative infinity (-Infinity).

If Infinity is converted into another data type it results in the maximum value of the other data type (e.g., conversion into DWORD with REAL_TO_DWORD: 16#FFFFFFFF, into DINT with REAL_TO_DINT: 16#7FFFFFFF).

If -Infinity is converted into another data type it results in the maximum value of the other data type (e.g., conversion into DWORD with REAL_TO_DWORD: 16#00000000, into DINT with REAL_TO_DINT: 16#80000000).

Except for:

TRUE := REAL_TO_BOOL(Infinity);

'#Inf' := REAL_TO_STRING(Infinity);

'-#Inf' := REAL_TO_STRING(-Infinity);
Examples:

<table>
<thead>
<tr>
<th>Infinity</th>
<th>-Infinity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Infinity := 1.0 / 0.0</td>
<td>-Infinity := -1.0 / 0.0</td>
</tr>
<tr>
<td>Infinity := Infinity + Infinity</td>
<td>-Infinity := -Infinity -Infinity</td>
</tr>
<tr>
<td>Infinity := Infinity + 1.0</td>
<td>-Infinity := -Infinity + 1.0</td>
</tr>
<tr>
<td>Infinity := LREAL_TO_REAL(Infinity)</td>
<td>-Infinity := LREAL_TO_REAL(-Infinity)</td>
</tr>
</tbody>
</table>

**NaN**

If a calculation results in an undefined value the result is set to NaN (Not a Number). The result of each calculation with NaN is NaN. The operators "<", "<=", ">" and ">=" return FALSE if either or both operands are NaN.

Operator "=" returns FALSE if one operand is NaN.

Operator "<>" returns TRUE if one operand is NaN.

If NaN is converted into another data type the result is 0.

Except for:

- TRUE := REAL_TO_BOOL(NaN);
- '#NaN' := REAL_TO_STRING(NaN);

Examples:

NaN := SQRT(-2.0)
NaN := 0.0 / 0.0
NaN := Infinity -Infinity
NaN := 0.0 * Infinity
NaN := Infinity / Infinity

The result of an operation can be checked with the following program parts:

**Check for NaN (REAL):**

```plaintext
rX: REAL;
IF (rX <> rX) THEN (* rX is a NaN *)
   ...
END_IF;
```

**Check for NaN (LREAL):**

```plaintext
lrX: LREAL;
IF (lrX <> lrX) THEN (* lrX is a NaN *)
   ...
END_IF;
```
Check for Infinity (REAL):

<table>
<thead>
<tr>
<th>Infinity is represented with sign bit 0, exponent of all 1s and a fraction of all 0s.</th>
</tr>
</thead>
<tbody>
<tr>
<td>-Infinity is represented with sign bit 1, exponent of all 1s and a fraction of all 0s.</td>
</tr>
</tbody>
</table>

```
rX: REAL;
prX: POINTER TO REAL;
pdX: POINTER TO DWORD;
prX := ADR(rX);
pdX := prX;
IF (pdX^ = 16#7F800000) THEN (* rX is Infinity *)
  ...;
END_IF;
IF (pdX^ = 16#FF800000) THEN (* rX is -Infinity *)
  ...;
END_IF;
```

Check for Infinity (LREAL):

```
lrX: LREAL;
plrX: POINTER TO LREAL;
plwX: POINTER TO LWORD;
plrX := ADR(lrX);
plwX := plrX;
IF (plwX^ = 16#7FF0000000000000) THEN (* lrX is Infinity *)
  ...;
END_IF;
IF (plwX^ = 16#FFF0000000000000) THEN (* lrX is -Infinity *)
  ...;
END_IF;
```

1.5.6.2.4 Terminal bases

Change terminal base type

☞ “Changing the terminal base type” on page 3275

1.5.6.2.5 Onboard Ethernet configuration

Onboard Ethernet is provided for device types with -ETH extension.

Configuration of the IP settings with IP configuration tool

With the IP configuration tool a network scan can be executed.
Further, the IP settings of found PLC devices can be changed if required.
IP Configuration Tool
1. From the menu, select “Tools ➤ IP-Configuration”.
2. Click [Scan] to trigger the scan process for devices in the network.
   ➞ A progress bar shows the progress.
3. To change the configured IP settings, select a connected device from the list.
   ➞ The IP settings are displayed below the list and can be edited.
4. Click [Send Configuration] to transmit the IP settings to the corresponding device.

**Processor modules with 2 onboard Ethernet interfaces:**
- Configure only 1 interface as default gateway.
- In case of 2 configured interfaces only the latter will be executed.

**Blink functionality**
Activates flashing of background light of AC500 LED display in Automation Builder via IP configuration tool.

1. From the menu, select “Tools ➤ IP-Configuration”.
2. Click [Scan] to trigger the scan process for devices in the network.
   ➞ A progress bar shows the progress. The IP settings of a selected device is displayed below the list and can be edited.
3. Adjust your desired time and click [Blink] to activate flashing.
4. Click “Save Configuration” to transmit the IP settings to the corresponding device.
Configuration of communication via Ethernet (TCP/IP)

Programming via Ethernet is only possible on a PC with Ethernet board and installed network. Programming can be done via the internal Ethernet communication module.

Programming via internal (onboard) Ethernet communication module:

1. Right-click the top node “PLC_AC500 <...>” and select “Communication Settings” from the context menu.

   ⇒ Dialog box Communication Settings <...> appears.

2. Enter your PLC IP Address and click [OK].
Enter PLC IP address by scanning devices

1. Right-click the top node “PLC_AC500 <...>” and select “Communication Settings” from the context menu.
   ⇒ Dialog box Communication Settings <...> appears.

2. Click [...].
   ⇒ Dialog box Pick IP Address <...> appears.
3. Click [Scan], select your desired PLC and click [OK].
   ⇒ Entry is transferred to the dialog box Communication Settings <...>.
   Click [OK].

4. Click to log in the “PLC_AC500_V3” project.

Enter PLC IP address by [Advanced Settings...]

If a remote gateway instead of a local one has to be used it can be configured in the [Advanced Settings...].
1. Right-click the top node “PLC_AC500 <...>” and select “Communication Settings” from the context menu.
   ⇒ Dialog box Communication Settings <...> appears.

2. Enable checkbox Use advanced settings and click [Advanced Settings...].
   ⇒ Tab “Communication Settings” opens.

3. Check gateway or change if required.
   ⇒ Successful connection is indicated by green dot on the gateway icon.

4. **Manual entry of the IP address.**
   Check IP address or change if required.

5. Press ENTER to confirm changed IP address.
   ⇒ Successful communication is indicated by green dot on the PLC icon.
6. Or instead of the last two steps:

Set the IP address via a scan.

Click [Scan Network], select your desired PLC and click [OK].

![Communication Settings](image)

⇒ Successful connection is indicated by green dot on the gateway icon.

7. Click to log in the “PLC_AC500_V3” project.
Switch functionality of Ethernet interfaces ETH1/ETH2

As of SystemFW 3.1.0 the Ethernet interfaces ETH1/ETH2 can be configured as an Ethernet switch.

The default setting is "Two separate interfaces".

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ETH1 / ETH2 mode</td>
<td>&quot;Two separate interfaces&quot;</td>
<td>Two separate Ethernet interfaces ETH1 and ETH2</td>
</tr>
<tr>
<td></td>
<td>&quot;Switch functionality ETH1-ETH2&quot;</td>
<td>Switch between ETH1 and ETH2</td>
</tr>
</tbody>
</table>

If the Switch functionality ETH1-ETH2 is active, only the Ethernet interface ETH1 is available (see ETH1 / ETH2 mode) Chapter 1.5.5.1.4.4.3 “Configuration” on page 3123. Any protocols configured under Ethernet interface ETH2 must be deleted. Otherwise a compile error will be created.

The setting of ETH1 / ETH2 mode can be checked on LED display with soft key <CFG> (see ETH1 / ETH2 mode) Chapter 1.5.5.1.4.4.3 “Configuration” on page 3123.

1.5.6.2.6 Onboard CAN configuration

Onboard CAN interface supports the following protocols

- CAN Open Manager: Connection of CI581 and CI582 without additional I/O modules
- CAN 2A/2B
- J1939

Configuration in Automation Builder is described in chapter Chapter 1.5.6.2.7.1.1 “CM598-CAN - CANopen master communication module” on page 3286.

Further information can be found in chapter Chapter 1.5.6.2.12 “CAN-based fieldbuses” on page 3332.
AC500 V3 PLCs provide the following methods for CAN integration:

- Onboard CAN interface
- Communication module CM598-CN

Differences in supported protocols can be seen in the table below.

<table>
<thead>
<tr>
<th></th>
<th>Onboard CAN</th>
<th>CM598-CN</th>
</tr>
</thead>
<tbody>
<tr>
<td>CANopen Manager</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>CAN 2A/2B</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>J1939</td>
<td>X</td>
<td></td>
</tr>
</tbody>
</table>

### 1.5.6.2.7 Communication modules

**CANopen**

**CM598-CAN - CANopen master communication module**

#### Configuration of the communication module

- Click menu “Tools ➔ Options” and select “Device editor” in the Options window.
- Enable first checkbox Show generic device configuration views and click [OK].

#### Append a CM598-CAN

1. Right-click on your desired Slot below node “Extension_Bus” and click “Add object”.
2. Click CM598_CAN in the list and click [Replace object].
3. Double-click “CM598_CAN (CM598-CAN)” to get the “CM598-CAN Parameters” in the editor window.

The following parameters are available:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Default value</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Run on config fault</td>
<td>No</td>
<td>No</td>
<td>In case of a configuration error, the user program is not started.</td>
</tr>
<tr>
<td></td>
<td>Yes</td>
<td></td>
<td>The user program is started independent of a faulty configuration of the CM598-CAN communication module.</td>
</tr>
<tr>
<td>Bus behavior</td>
<td>Asynchronous (IEC bus cycle)</td>
<td>Asynchronous (IEC bus cycle)</td>
<td>Not yet supported.</td>
</tr>
<tr>
<td></td>
<td>Synchronous (start of bus cycle)</td>
<td>Synchronous (start of bus cycle)</td>
<td>Not yet supported.</td>
</tr>
<tr>
<td>Node ID</td>
<td>1</td>
<td>1 - 127</td>
<td>Identifier of the device within CANopen.</td>
</tr>
</tbody>
</table>
The tab “CAN Bus” contains the basic settings of the CAN bus and special settings for the CAN 2.0 B protocol.

The settings at “29 Bit COB-ID” are only valid for CAN 2.0 B protocol. Ensure the option “Enable 29 Bit COB-ID” is enabled. Otherwise no CAN 2.0 B frames can be received. With the other parameters at “Enable 29 Bit COB-ID” the receive filter is configured.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Default value</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bus parameters</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Transmission rate</td>
<td>250 kBit/s</td>
<td>10 kBit/s</td>
<td>Transmission speed in [kBit/s]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>20 kBit/s</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>50 kBit/s</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>100 kBit/s</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>125 kBit/s</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>250 kBit/s</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>500 kBit/s</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>800 kBit/s</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>1000 kBit/s</td>
<td></td>
</tr>
<tr>
<td>Node settings</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Stop in case of moni-</td>
<td>Disabled</td>
<td>Disabled</td>
<td>The manager does not stop in case of a monitoring error (Node Guarding or</td>
</tr>
<tr>
<td>toring error</td>
<td></td>
<td></td>
<td>Heartbeat Error). A loss of communication to one node has no influence to</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>other nodes. The manager tries to reestablish the communication to the</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>error affected nodes.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Enabled</td>
<td>If this function is enabled, the manager will also stop the communication</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>to all responding and active nodes. Not yet supported.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Send &quot;Global Start Node&quot;</td>
<td>Enabled</td>
<td>Disabled</td>
<td>No &quot;Global Start Node&quot; message is sent after configuring the nodes.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Enabled</td>
<td>A &quot;Global Start Node&quot; message is sent after configuring the nodes. This</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>synchronize all Nodes again. Not yet supported.</td>
</tr>
</tbody>
</table>

29 Bit COB-ID
### Configuration of the protocols CAN 2.0 A / CAN 2.0 B

The Communication Module CM598-CAN can be used to realize CAN bus based networks in combination with library ABB_CM598Can_AC500.library.

To enable the support for the desired protocol it must be appended to CM598-CAN.

1. Right-click “CM598_CAN (CM598-CAN)” in the device tree and select “Add device” in the context menu.
   
   ⇒ Window Add object below: CM598_CAN_1 appears.

2. Select “CAN 2.0 A” or “CAN 2.0 B” from the list.

### Parameterization

The CAN data transmission requires a buffer for the incoming data that can be read with function blocks of library ABB_CM598Can_AC500.library.
1. Right-click “CAN_2_0A_11_bit_identifier_ (CAN 2.0A)” or “CAN_2_0B_29_bit_identifier_ (CAN 2.0B)” and select “Add object”.

2. Select “Buffer for CAN 2A” for CAN 2.0A. Or select “Buffer for CAN 2B” for CAN 2.0B from the list.

3. Double-click on “Buffer_for_CAN_2A (Buffer for CAN2A)” or “Buffer_for_CAN_2B (Buffer for CAN2B)” in the device tree to open the Buffer configuration in the editor window.

The following parameters are available:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Default value</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Identifier</td>
<td>0</td>
<td>CAN 2A: 0 ... 2047 CAN 2B: 0 ... 536870911</td>
<td>The value of the CAN identifier that is compared with the identifier of the incoming telegrams. The telegrams will be added to the buffer if the identifier matches.</td>
</tr>
<tr>
<td>Number of receive buffers</td>
<td>1</td>
<td>1 ... 16</td>
<td>The size of the buffer in number of telegrams.</td>
</tr>
<tr>
<td>Behaviour on receive buffer overflow</td>
<td>Overwrite</td>
<td>Overwrite</td>
<td>The oldest telegram in the buffer is overwritten by the incoming telegram. Discard</td>
</tr>
<tr>
<td>Enable triggering of IEC task</td>
<td>No</td>
<td>No</td>
<td>Disables the triggering of the execution of the related IEC task. Yes</td>
</tr>
</tbody>
</table>

Configuration of the CANopen master

☞ Chapter 1.5.6.2.12.1.1 “CANopen manager (master)” on page 3332

PROFINET
CM579-PNIO - PROFINET IO controller
For Automation Builder < 2.2.0

Configuration in Automation Builder is described in the PROFINET IO configuration.
For Automation Builder >= 2.2.0

PROFINET IO

CM579-PNIO – PROFINET IO communication module

Configuration of the communication module

Configuration is valid as of CPU FW 3.2.0.

Append a CM579-PNIO

1. Right-click on your desired Slot below node “Extension_Bus” and click “Add object”.
   ⇨ Dialog Replace object: appears.
2. Click CM579_PNIO in the list and click [Replace object].
3. Double-click “CM579_PNIO (CM579-PNIO)” to get the “CM579-PNIO Parameters” in the editor window.

The following parameters are available:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Default value</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Run on config fault</td>
<td>No</td>
<td>No</td>
<td>In case of a configuration error, the user program is not started.</td>
</tr>
<tr>
<td></td>
<td>Yes</td>
<td></td>
<td>The user program is started independent of a faulty configuration of the CM579-PNIO communication module.</td>
</tr>
<tr>
<td>Bus behavior</td>
<td>Asynchronous (IEC bus cycle)</td>
<td>Asynchronous (IEC bus cycle)</td>
<td>The bus cycle and the IEC Application are running asynchronously. The IO update rate between the Profinet IO Controller and the IEC Application is defined with the bus cycle task.</td>
</tr>
</tbody>
</table>

Configuration of the PROFINET IO controller

The PROFINET IO Controller node appears automatically below the added Communication Module CM579-PNIO.

PROFINET IO controller - Configuration

Double-click on “PNIO_Controller” and open the tab “General” in the editor window.

The following parameters are available:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Default</th>
<th>Value</th>
<th>Description</th>
<th>Parameter</th>
</tr>
</thead>
<tbody>
<tr>
<td>Station name</td>
<td>CM579</td>
<td>Up to 240 characters</td>
<td>Network name of the PROFINET IO controller station. Must be a valid hostname.</td>
<td>Station name</td>
</tr>
<tr>
<td>IP parameters</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>IP-Address</td>
<td>192.168.0.1</td>
<td>Valid IP address</td>
<td>IP address of the PROFINET IO controller station.</td>
<td>IP address</td>
</tr>
</tbody>
</table>
### PROFINET IO controller - Parameters

The tab “PROFINET-IO-Controller Parameters” is a generic view of all PROFINET IO controller parameters. It is normally hidden and is normally not needed for configuration.

- **Use tab “PROFINET-IO-Controller Parameters” only, if you need to change a parameter, which is not visible in other dialogs.**

#### Activating tab

1. Click “Tools  Options” and select “Device editor”.
   - The Device editor dialog opens.
2. Enable checkbox Show generic device configuration views and click [OK]
   - The tab is now available.

### PROFINET IO controller - I/O mapping

In this tab the bus cycle task can be specified. It is possible to select a particular task of the IEC application by its name or to use the option “Use the parent bus cycle setting”. In the latter case the setting of the Bus cycle options in “PLC_AC500_V3  PLC settings” are used.
Configuration of PROFINET IO devices

Add PROFINET IO device

1. Right-click on node “PNIO_Controller (PROFINET-IO-controller)” and click “Add object.”
   ⇒ A list with all installed PROFINET IO Devices appears.
2. Select the desired device and click [Add object].
   ⇒ The device is added to the Profinet IO Controller in the device tree.

PROFINET IO device - Configuration

Double-click on “PNIO-Device” to open the device configuration in the editor window.

The following parameters are available:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Default</th>
<th>Value</th>
<th>Description</th>
<th>Parameter</th>
</tr>
</thead>
<tbody>
<tr>
<td>Identification</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Station name</td>
<td>Device-specific</td>
<td>Up to 240</td>
<td>This is a system wide unique name for addressing the device. Must be a valid hostname.</td>
<td>Slave parameters -&gt; Identification -&gt; Station name</td>
</tr>
<tr>
<td>Communication Parameter</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Send clock (ms)</td>
<td>Device-specific</td>
<td>0.25</td>
<td>Parameter Send clock determines the SendCycle.</td>
<td>Slave parameters -&gt; Reduction ratio</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.5</td>
<td>SendCycle = Send clock x Reduction ratio &lt;= 512ms x</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Reduction ratio</td>
<td>Device-specific</td>
<td>1...16384</td>
<td>The Reduction ratio determines the factor for calculating the cycle time.</td>
<td>Slave parameters -&gt; Reduction ratio</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Cycle time = Send clock x Reduction ratio</td>
<td></td>
</tr>
<tr>
<td>Phase</td>
<td>1</td>
<td>1...Reduction ratio</td>
<td>Defines the part of the SendCycle at which an IO frame is sent.</td>
<td>Phase</td>
</tr>
<tr>
<td>Watchdog factor</td>
<td>3</td>
<td>1...65535</td>
<td>The Watchdog time is calculated as Watchdog time = SendCycle * Watchdog factor. The transfer of a IO telegram is always checked of the consumer side. Within this time the next IO telegram must be received by a consumer. Otherwise it is checked if the Datahold has been expired too.</td>
<td>Watchdog interval</td>
</tr>
<tr>
<td>RT Class</td>
<td>RT Class 1 Data-RTC-PDU</td>
<td>RT Class 1 Data-RTC-PDU</td>
<td>Defines the Realtime Class of cyclic data. Currently only RT Class 1 (legacy) and RT Class 1 are supported.</td>
<td>Slave parameters -&gt; RT Class</td>
</tr>
</tbody>
</table>

Add PROFINET IO device

PLC Automation with V3 CPUs
PLC integration > Configuration in Automation Builder for AC500 V3 products
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Default</th>
<th>Value</th>
<th>Description</th>
<th>Parameter</th>
</tr>
</thead>
<tbody>
<tr>
<td>RT Class 2</td>
<td></td>
<td></td>
<td>In case of VLAN usage the parameter VLAN ID represents the ID of the virtual network.</td>
<td></td>
</tr>
<tr>
<td>RT Class 3</td>
<td></td>
<td></td>
<td>For VLAN type 802.1Q the range is 0..4095 while VLAN type ISL accepts values from 0 to 32767.</td>
<td></td>
</tr>
<tr>
<td>RT Class</td>
<td></td>
<td></td>
<td>The supported type depends on the used device.</td>
<td></td>
</tr>
<tr>
<td>UDP-RTC-PDU</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>VLAN ID</td>
<td>0</td>
<td>0..4095 or</td>
<td>In case of VLAN usage the parameter VLAN ID represents the ID of the virtual network.</td>
<td>Slave parameters -&gt; VLAN ID</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0..32767</td>
<td>For VLAN type 802.1Q the range is 0..4095 while VLAN type ISL accepts values from 0 to 32767.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The supported type depends on the used device.</td>
<td></td>
</tr>
</tbody>
</table>

**IP Parameter**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Default</th>
<th>Value</th>
<th>Description</th>
<th>Parameter</th>
</tr>
</thead>
<tbody>
<tr>
<td>IP-Address</td>
<td>192.168.0.8</td>
<td>Valid IP address</td>
<td>IP address of the PROFINET IO Controller station.</td>
<td>Slave parameters -&gt; Identification -&gt; IP address</td>
</tr>
<tr>
<td>Subnetmask</td>
<td>255.255.255.0</td>
<td>Valid subnet mask</td>
<td>Network mask of the PROFINET IO Controller station.</td>
<td>Slave parameters -&gt; Identification -&gt; Subnet mask</td>
</tr>
<tr>
<td>Default gateway</td>
<td>0.0.0.0</td>
<td>Valid gateway address</td>
<td>Default gateway address of the PROFINET IO Controller station.</td>
<td>Slave parameters -&gt; Identification -&gt; Default gateway address</td>
</tr>
</tbody>
</table>

**PROFINET IO device – Timing parameters**

In the current implementation are 2 Communication Relations (CR) between the controller and the device defined.

One describes the I/O telegram from the controller to the device (outputs), the other the I/O telegram from the device to the controller (inputs).

The timing of the corresponding I/O telegrams can be defined separately for each device.

Editable timing parameters are:

- Send clock
- Reduction ratio
- Phase

The relation between these parameters is shown in the following drawing.
For each device a SendCycle must be configured, which determines the sending interval of I/O frames. It is based on a time base of 31.25 µs and is calculated as:

\[ \text{SendClock [ms]} = \text{SendClockFactor} \times 31.25 \mu\text{s} / 1000. \]

The cycle time of an I/O telegram is defined by the SendCycle. It's calculated as:

\[ \text{SendCycle [ms]} = \text{SendClock [ms]} \times \text{Reduction Ratio}. \]

The values of the individual parameters are limited by the maximum value 512 ms of the SendCycle. The following table summarizes the relation of the timing parameters.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Relation</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>SendCycle</td>
<td>Is the cycle time of a RT telegram.</td>
<td>SendCycle = SendClock \times \text{Reduction ratio}.</td>
<td>1ms..512 ms</td>
</tr>
<tr>
<td>SendClock</td>
<td>The SendCycle is divided into several time slots. The SendClock defines the size of a time slot within the SendCycle.</td>
<td>SendClock = SendClock factor \times 31.25 \mu\text{s};</td>
<td>SendClock \times \text{Reduction ratio} \leq 512 ms</td>
</tr>
<tr>
<td>SendClock factor</td>
<td>Is multiplied with the time base 31.25 \mu\text{s to calculate the SendClock.}</td>
<td>SendClock factor = SendClock / 31.25 \mu\text{s}</td>
<td>1..128</td>
</tr>
<tr>
<td>Reduction ratio</td>
<td>The reduction ratio defines the number of time slots within the SendCycle.</td>
<td>SendClock \times \text{Reduction factor} \leq 512 ms</td>
<td>1..16384</td>
</tr>
<tr>
<td>Phase</td>
<td>The time slot in which the IO frame is sent.</td>
<td>A integer value of the range 1 ... Reduction ratio</td>
<td>1..16384</td>
</tr>
</tbody>
</table>
PROFINET IO device – PNIO parameters

The tab “PNIO Parameters” is a generic view of all PROFINET IO device parameters. It is normally hidden and is normally not needed for configuration.

Use tab “PNIO Parameters” only, if you need to change a parameter, which is not visible in other dialogs.

Activating tab

1. Click “Tools ➔ Options” and select “Device editor”.
   ⇒ The Device editor dialog opens.
2. Enable checkbox Show generic device configuration views and click [OK]
   ⇒ The tab is now available.

Configuration of non-ABB PROFINET IO devices

Before a Non-ABB PROFINET IO device can be used, the provided GSDML file has to be installed in the Device Repository.

Installation

Go to “Tools ➔ Device Repository ➔ Install”.

Configuration

See Chapter 1.5.6.2.7.2.1.2.1.1.3 “Configuration of PROFINET IO devices” on page 3292.

I/O mapping of the PROFINET IO devices

Open I/O mapping list

1. Double-click on the “PNIO_Controller” or below on the “<...>PNIO-Device” or below on the “I/O-Module” in the device tree.
2. Select tab “I/O mapping list” to show the list of I/O channels.

The content of the list depends on the selected node.

For instance:
- When the “PNIO_Controller” node is selected all I/O channels of all configured devices are shown.
- When a “PNIO-Device” is selected all I/O channels of the configured modules are shown.

An IEC variable for an I/O channel that is available in the Application can be defined by double-clicking in column Variable.
CM589-PNIO - PROFINET IO slave

Configuration in Automation Builder is described in the PROFINET IO Slave configuration.

EtherCAT
CM579-ETHCAT - EtherCAT IO master

Configuration in Automation Builder is described in the EtherCAT Master configuration.

EtherCAT-Master - ABB functionality for sync units

With the EtherCAT sync units, several slaves are configured into groups and subdivided into smaller units. For each group, the working counter can be monitored for granular input data validation. As soon as a slave is missing in a sync unit group, the input data of all other slaves in the same group becomes invalid.

Detection occurs immediately in the next bus cycle, as the working counter is continuously checked. Unaffected groups remain operable without any interference.

Right click on the “Application” node and press “Create configuration data”.

Automation Builder creates a set of global variables defining the working counter state of a SyncUnit command.

The variables use the following naming scheme:

"SLOT_" + "CouplerSlot_ " + "SyncUnitName" + "_CMD_" + "LogicalAccess" + "_FRAME_" + "FrameID CouplerSlot".

CouplerSlot
The coupler slot is the ID of the slot where the coupler is plugged in.

SyncUnitName
The sync unit name is as defined in the “Sync Unit Assignment” tab.

LogicalAccess
The logical access defines the command List of logical access commands:

- Read = 10;
- Write = 11
- Read/Write = 12

FrameID
The frame ID starts with 1 and increments if the cyclic exchanged data is larger than the maximum Ethernet frame boundary.

Values
- FALSE : Working counter is as expected (data from slaves is valid)
- TRUE : Working counter is different to expected value (data from slaves is invalid)

The variables can be used by conditional consumption of slave data in the application:

```c
IF NOT SLOT_1_default_CMD_10_FRAME_1 THEN
    (* Consume SyncUnit default slave data *)
END_IF;
```

See also 3S Sync Unit Assignment % Chapter 1.5.6.2.13.1.2 “Tab 'EtherCAT Master - Sync Unit Assignment' ” on page 3349.

EtherCAT diagnosis (V2 PLC and V3 PLC)

Automation Builder 2.3 provides an enhanced diagnosis interface for the EtherCAT fieldbus. The user can get EtherCAT diagnosis information from four different editor views. All these views are accessible within the EtherCAT master device editor and provide information about the master and all configured or connected slaves. The main diagnosis overview is given in the EtherCAT master view “Diagnostics main”.

“Diagnostics main” shows EtherCAT state “Operate”.

“Diagnostics main” shows EtherCAT state “Topology error”.

If the EtherCAT bus state shows “Operate”, the user does not need to check for any more information.

If the “Diagnostics main” shows any error, like “Topology mismatch detected”, the user can continue to the next level of information by opening editor view “Master State Control”.
“Master State Control” shows EtherCAT state “Operate”.

In editor view “Master State Control” the user can request a master state change or get information about configured parameters as well as events and latest communication errors. In case of any topology error (e.g. slaves are configured in a different order than they exist in hardware) the Automation Builder shows a hint to the user that it might be helpful to execute a bus scan in editor view “Diagnostics live list” to compare the scan result of the real hardware with the configurations slaves in the Automation Builder project.

Bus scan result in editor view “Diagnostics live list” shows the connected hardware.
The bus scan result list shows the following information for each connected slave:

<table>
<thead>
<tr>
<th>“ID”</th>
<th>Position of the found slave device</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Type”</td>
<td>Slave identification (name or Vendor/Device ID number)</td>
</tr>
<tr>
<td>“State”</td>
<td>The connection/link state of all ports (0-3) of the given slave</td>
</tr>
<tr>
<td></td>
<td>Connected =&gt; Cable is plugged in</td>
</tr>
<tr>
<td></td>
<td>+ Link =&gt; Physically connected to another slave</td>
</tr>
<tr>
<td></td>
<td>+ Communication =&gt; Communication works fine</td>
</tr>
<tr>
<td>“Name”</td>
<td>Not used for EtherCAT</td>
</tr>
<tr>
<td>“Address”</td>
<td>Not used for EtherCAT</td>
</tr>
<tr>
<td>“Details”</td>
<td>E.g. revision number of the slave device</td>
</tr>
</tbody>
</table>

The bus scan shows information about the real connected hardware.

Note that a bus scan will always restart the EtherCAT bus.

This should not be a problem during commissioning but it might not be applicable in a running system.

For runtime diagnosis the Automation Builder provides cyclic information of all configured slaves and their states in the editor view “Slave diagnosis”.

Slave diagnosis information shows that configured slaves are ok.

“Slave diagnosis” view shows wrong slave at position 1.

The editor view “Slave diagnosis” shows information about the configured slaves. If these slaves are found in hardware, the states of all slaves are ok. If there is a mismatch between hardware and configuration the view shows at which position that mismatch was detected.
The “Slave diagnosis” shows the following information for each configured slave:

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Topology Position”</td>
<td>Position of the configured slave device.</td>
</tr>
<tr>
<td>“Configured Station Address”</td>
<td>Address that is defined by configuration. This address is not topology dependent.</td>
</tr>
<tr>
<td>“Slave Name”</td>
<td>Configured name of the slave device.</td>
</tr>
<tr>
<td>“Slave State”</td>
<td>The state of the slave. Possible slave states are:</td>
</tr>
<tr>
<td></td>
<td>NOT CONNECTED</td>
</tr>
<tr>
<td></td>
<td>INIT</td>
</tr>
<tr>
<td></td>
<td>PREOP</td>
</tr>
<tr>
<td></td>
<td>SAFEOP</td>
</tr>
<tr>
<td></td>
<td>OP</td>
</tr>
<tr>
<td></td>
<td>INIT ERR</td>
</tr>
<tr>
<td></td>
<td>PREOP ERR</td>
</tr>
<tr>
<td></td>
<td>SAFEOP ERR</td>
</tr>
<tr>
<td>“Port State”</td>
<td>The state of all ports (0-3) of the given slave. Shows how many connections this slave has to other slaves and if the connections are working fine:</td>
</tr>
<tr>
<td>Connected</td>
<td>=&gt; Cable is plugged in</td>
</tr>
<tr>
<td>+ Link</td>
<td>=&gt; Physically connected to another slave</td>
</tr>
<tr>
<td>+ Communication</td>
<td>=&gt; Communication works fine</td>
</tr>
<tr>
<td>“Last Error”</td>
<td>The last error that occurred in this slave. As text, if available, and error number. If this is any topology error, the editor view will show a hint to perform a bus scan.</td>
</tr>
<tr>
<td>“Emergency” [CAN application protocol over EtherCAT (CoE)]</td>
<td>This column contains up to 5 CoE emergency entries. Each entry has</td>
</tr>
<tr>
<td>Error code</td>
<td></td>
</tr>
<tr>
<td>Address of the error register</td>
<td></td>
</tr>
<tr>
<td>Error data (1 byte)</td>
<td>If there are more than five emergencies reported by the slave, the columns show a hint that some emergency entries have been lost. The column is empty, if no CoE emergencies exist.</td>
</tr>
<tr>
<td>“Frame Error Counters”</td>
<td>Counts transmission errors on frame layer, detected by CRC check of frames. Fast growing values show a serious problem. Possible root causes include damaged cables, high electromagnetic noise or misbehavior of EtherCAT slave devices. Four counter values are shown, one for each port 0-3. Column has red background in case of any value other than 0.</td>
</tr>
<tr>
<td>“Physical Layer Error Counters”</td>
<td>Counts transmission errors on physical layer. Possible root causes include electromagnetic disturbance or faulty devices. Four counter values are shown, one for each port 0-3. Column has red background in case of any value other than 0.</td>
</tr>
</tbody>
</table>
“Link Lost Counters" 2) Optional feature of EtherCAT slave devices, not supported by every device.

Counts loss of physical connection (no link, LED off). Even short interruptions can be detected. Possible root causes include power dips, device reset, poor cables or connectors, loose contact. Four counter values are shown, one for each port 0-3. Column has red background in case of any value other than 0.

1) Note that columns “Port State”, “Frame Error Counters”, “Physical Layer Error Counters” and “Link Lost Counters” show “LLD: Timeout”, if this state is NOT CONNECTED, because this information is not accessible.

2) Note that this column contains “LLD: Timeout", if slave state is NOT CONNECTED.

General note on the counters
Please note that this kind of errors will be detected by devices when power state changes, e.g. when the device itself or a neighboring device is powered on, caused by switching artifacts on the cable. This does not signal an issue, only counters increasing during normal operation should trigger deeper analysis. Counters can be reset by the PLC program using corresponding function blocks.

1.5.6.2.8 Communication interface modules

Configuration of communication interface modules
Automation Builder can be used to configure the parameters of CI5xx devices.

Adding CI5xx device to the device tree

1. Right click in the device tree on the node “Slot1” or “Slot2” of the “Extension_Bus” and click “Add object”.
   ⇨ The window Replace object : Slot <...> opens.
2. Select your CM5xx master module and click [Add object].
   ⇨ The CM5xx master appears in the Slot.
3. Right click on the CM5xx master module and click “Add object”.
   ⇨ The window Add object below : <...>_Master opens.
4. Select your “CI5xx” device and click [Add object].
   ⇨ The “CI5xx” device appears in your device tree.

Adding S500 I/O modules

1. Right click on your “CI5xx” device and click [Add object].
   ⇨ The window Add object below: opens.
2. Select your I/O module and click [Add object].
   ⇒ The I/O module is added.

Configure parameters

⇒ Double-click the “CI5xx” device to open editors and select the “CI5xx_IO Parameters” tab.

This editor shows the parameters that can be set for each device. For more information see Parameters of the Communication Interface Modules Chapter 1.5.3.7 “Communication interface modules (S500)” on page 2785, and Parameters of the I/O Modules Chapter 1.5.3.6 “I/O modules” on page 2327.

CI521-MODTCP/CI522-MODTCP

Unbundled CI52x-MODTCP configuration

Automation Builder can be used to configure the parameters of CI52x-MODTCP devices.

A direct Ethernet connection is required between the PC running Automation Builder and the CI52x-MODTCP module.

Configuration of S500 I/O modules can be performed without CI52x-MODTCPs modules connected.

Start a project from template

1. Select “New Project” in menu item “File”.
   ⇒ The window “New Project” appears.

2. Select the “CI52x-MODTCP Configuration Project” and click “OK”.
   ⇒ The window “Select PLC” opens.

3. Select a “CI52x-MODTCP” device and click “Add device”.
   ⇒ A project is created. More modules can be added.

Add CI52x-MODTCP to a project

1. Right click in the device tree on the root of the “Project” and click “Add object”.
   ⇒ The window “Add object below” opens.

2. Select “Modbus devices” and click “Add object”.
   ⇒ The node “Modbus_devices” appears in your device tree.

3. Right click on the node “Modbus_devices” and click “Add object”.
   ⇒ The window “Select PLC” opens.
4. Select your “CI52x-MODTCP” device and click “Add device”.
   ⇨ The “CI52x-MODTCP” device appears in your device tree.

Add S500 I/O modules

1. “Add object” to your “CI52x-MODTCP” device.
   ⇨ The window “Add object below: CI52x-MODTCP” opens.

2. Select your I/O module and click “Add object”.
   ⇨ The I/O module is added.

Configure parameters

▷ Double-click the device to open editors and select the “CI52x-MODTCP Parameters” tab.

This editor shows the parameters that can be set for each device. For more information see
Parameters of the Module CI521 % Chapter 1.5.3.7.4.1.7.1 “Parameters of the module”
on page 2918, Parameters of the Module CI522 % Chapter 1.5.3.7.4.2.7.1 “Parameters of the module”
on page 2947 and Parameters of the I/O Modules % Chapter 1.5.3.6 “I/O modules”
on page 2327.

Connect to device

To read or write parameters, the CI52x-MODTCP module must be connected to the PC with an
Ethernet connection.

See Addressing CI521 % Chapter 1.5.3.7.4.1.5 “Addressing” on page 2917 and
Addressing CI522 % Chapter 1.5.3.7.4.2.5 “Addressing” on page 2947 of the CI52x-MODTCP
hardware documentation for information on configuring the IP address of the device.

On the CI52x-MODTCP device editor, the “Connection Settings” tab allows the IP address of
the device to be entered.
Read  Reads the parameters from the CI52x-MODTCP and also for the attached S500 I/O modules.

Write  Sends the parameters from the editors to the CI52x-MODTCP and also the S500 I/O modules.

Device checking  The CI52x-MODTCP module knows which I/O modules are attached. While reading and writing parameters, the project must match the physical hardware. Otherwise an error will be given. Communication errors will also result in error messages. When the parameters have been read or written correctly, a message is seen in the “All messages” window:

```
All messages

<table>
<thead>
<tr>
<th>Description</th>
<th>0 error(s)</th>
<th>0 warning(s)</th>
<th>2 message(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parameters read successfully</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Parameters stored successfully</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

Attached S500 modules  It is possible to read and write parameters when the S500 I/O modules are not attached to the CI52x-MODTCP module.

To perform a read, the project structure must still match the configuration of CI52x-MODTCP.

A warning will be shown if an I/O module is not detected:

```
All messages

<table>
<thead>
<tr>
<th>Description</th>
<th>0 error(s)</th>
<th>1 warning(s)</th>
<th>1 message(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>I/O module A562 not detected</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Parameters read successfully</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

When writing parameters, the CI52x-MODTCP configuration is overwritten so the current configuration of missing (unplugged) modules does not matter. If the I/O modules are attached, then the project must match the hardware, otherwise an error will be given.

Firmware update  As of Automation Builder 2.2.1, the IP Configuration Tool can be used to perform firmware updates for CI52x-MODTCP devices.
IP Configuration Tool
1.5.6.2.9 I/O-bus and I/O modules

Parameterization of the I/O bus

Double-click the “IO_Bus” node in the device tree to open the I/O-Bus configuration.

The following parameters are available:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Default</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Run on config fault</td>
<td>No</td>
<td>No</td>
<td>In case of configuration fault the user program will not be launched.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Yes</td>
<td>The user program will be also launched in case of configuration error on the I/O Bus.</td>
</tr>
<tr>
<td>Max wait run</td>
<td>3000</td>
<td>0...120000</td>
<td>Maximum waiting time for valid inputs.</td>
</tr>
</tbody>
</table>

In case of a digital I/O Module, the channels are provided as WORD, BYTE and BOOL. Because the analog inputs can also be configured as digital inputs, bit 0 of each channel is also available as BOOL.

The symbolic name of a channel can be entered in front of the string "AT" in the channel declaration.

All channels should have a symbolic name and only symbolic names should be used in the program code. If the hardware configuration has changed or if you want to download the project to a PLC with another hardware configuration and thus the PLC configuration has to be changed, the addresses of the inputs and outputs can change. In case of symbolic programming (i.e., symbolic names are used), the program code does not have to be changed.

Parameter 'Ignore module'

All I/O devices provide the parameter "Ignore module". This parameter can be used for simulation purposes and determines whether an I/O device is considered or ignored during a PLC configuration check.

This allows to use an existing Automation Builder project/PLC configuration though some hardware devices are not physically available in a hardware installation.

Example

The Automation Builder project for machine A shall be used for machine B. However, the second DC523 device is missing in the hardware installation of machine B. Hence, for machine B the value for 'Ignore module' is set to 'YES'.

---

3ADR010583, 1, en_US

2020/12/10

3306
I/O-bus - Bus cycle task

As a rule, for each IEC task the used input data is read at the start of each task (1) and the written output data is transferred to the I/O driver at the end of the task (3). The implementation in the I/O driver is decisive for further transfer of the I/O data. The implementation is therefore responsible for the timeframe and the specific time when the actual transmission occurs on the respective bus system.

The bus cycle task of the PLC can be defined globally for all fieldbuses in the PLC settings. For some fieldbuses, however, you can change this independent of the global setting. The task with the shortest cycle time is used as the bus cycle task (setting: "unspecified" in the PLC settings). In this task, the messages are normally transferred on the bus.

Other tasks copy only the I/O data from an internal buffer that is exchanged only with the physical hardware in the bus cycle task.

(1) Read inputs from input buffer       (2) IEC task
(3) Write outputs to output buffer      (4) Bus cycle
(5) Input buffer                        (6) Output buffer
(7) Copy data to/from bus               (4) Bus cycle
(9) Bus cycle task, priority 1, 1 ms
(10) Bus cycle task, priority 5
(11) Bus cycle task, priority 10, interrupted by task 5

Using tasks

The “Task Deployment” provides an overview of used I/O channels, the set bus cycle task, and the usage of channels.

WARNING!
If an output is written in various tasks, then the status is undefined, as this can be overwritten in each case.

When the same inputs are used in various tasks, the input could change when a task is processed. This happens if the task is interrupted by a task with a higher priority and causes the process map to be read again. Solution: At the beginning of the IEC task, copy the input variables to variables and then work only with the local variables in the rest of the code.

Conclusion: Using the same inputs and outputs in several tasks does not make any sense and can lead to unexpected reactions in some cases.
Insertion of S500 I/O devices

1. Right-click “IO_Bus” in device tree and select [Add object].
   - The Add Device dialog window where all available S500 I/O Devices are listed will open.
2. Append the S500 I/O Devices in the same order as they are mounted on the hardware.
   - Input and output modules connected to the I/O bus occupy the I/O following area: %IB0 .. %IB999 or %QB0 .. %QB999.

AC500 (Standard): PM56xx support up to 10 S500 I/O Devices.

Configuring the input and output modules and channels

The I/O channel configuration depends on the corresponding S500 I/O Device. See hardware documentation of the I/O Device for more information.

The individual configuration parameters can be opened in the editor window via double-click on the corresponding module and are listed in tab [S500 I/O device name] Configuration.

Symbolic names for variables, inputs and outputs

The IEC naming rules are not checked during input in Automation Builder.

Input and output mapping

Devices with I/Os provide an I/O Mapping tab in their configuration editor where the available I/O channels can directly be mapped to a global variable.

The corresponding variable declarations are automatically available in the project.

All available I/O channels can easily be assigned to a variable.

<table>
<thead>
<tr>
<th>Variable</th>
<th>Mapping</th>
<th>Channel</th>
<th>Address</th>
<th>Type</th>
<th>Default Value</th>
<th>Unit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>b_Input_IB0</td>
<td></td>
<td>Digital inputs 0 - 17</td>
<td>%IB0</td>
<td>BYTE</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>x_Input_IB0_s0</td>
<td></td>
<td>Digital input 0</td>
<td>%DI0.0</td>
<td>BOOL</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>x_Input_IB0_s1</td>
<td></td>
<td>Digital input 1</td>
<td>%DI0.1</td>
<td>BOOL</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>x_Input_IB0_s2</td>
<td></td>
<td>Digital input 2</td>
<td>%DI0.2</td>
<td>BOOL</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>x_Input_IB0_s3</td>
<td></td>
<td>Digital input 3</td>
<td>%DI0.3</td>
<td>BOOL</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>x_Input_IB0_s4</td>
<td></td>
<td>Digital input 4</td>
<td>%DI0.4</td>
<td>BOOL</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>x_Input_IB0_s5</td>
<td></td>
<td>Digital input 5</td>
<td>%DI0.5</td>
<td>BOOL</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>x_Input_IB0_s6</td>
<td></td>
<td>Digital input 6</td>
<td>%DI0.6</td>
<td>BOOL</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>x_Input_IB0_s7</td>
<td></td>
<td>Digital input 7</td>
<td>%DI0.7</td>
<td>BOOL</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Digital inputs 10 - 15</td>
<td>%IB1</td>
<td>BYTE</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Digital inputs 16 - 23</td>
<td>%IB2</td>
<td>BYTE</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Digital inputs 24 - 31</td>
<td>%IB3</td>
<td>BYTE</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Fast counter</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
AC500 uses Intel Byte Order (Little Endian).

Only entries with a data type set in column "Type" can be mapped. These entries can be expanded to show the available I/O channels.

If the project has been imported from a previous Automation Builder version, all variables should be checked to avoid inconsistencies concerning the I/O mapping.

I/O mapping list

Automation Builder contains an I/O mapping list feature for creating mapping variables with better usability support compared to the tree structured view. Details on the tree structured view is provided in the CODESYS Development System Configuring I/O Links Chapter 1.3.1.8.3 “Configuring devices and I/O mapping” on page 162

![Image of I/O mapping list]

Functionalities of the I/O mapping list:

- Displays I/O mappings for current node and all valid subsequent child nodes.
- Displays channel information with additional columns.
- Supports keyboard functions such as cut, copy, paste, delete, and select all within the editor and within Excel spreadsheet (for bulk editing).
- Contains a toolbar for various actions, e.g. filtering, undo/redo and clear mappings.
- Supports single click edit and easy navigation using arrow keys.
- Improvised error handling:
  - Allows to enter invalid mapping variables. This provides flexibility in bulk editing. Only when saving the project, the errors - according to IEC 61131 standard - are displayed.
  - In the message window, the error log is visible. The user can track the errors to their corresponding channel in the editor.
- Allows multi-selection of rows and columns. (Random selection is not allowed.)
Configuring I/O mapping list

Automation Builder supports tree and list based editors for creating I/O mapping variables.

1. From the Tools menu, select Options.
2. Under Automation Builder, select the Editors tab.
3. Choose your desired mapping dialog and click OK.
   - Choose tree based to display the I/O mapping in tree structure.
   - Choose list based to display the I/O mapping as list with the functionalities of the ToolBar.
   - Choose both to display both the tree structure (I/O Mapping tab) and the list view (I/O mapping list tab).

Available channel information

The I/O mapping list displays the channel information in offline and online mode. In online mode, all columns are read-only. In offline mode, some columns are editable.

The order of the devices in I/O mapping list is synchronized with the order in the device tree.

The channels of a device are ordered by the device description file. If channels have a section, the channel information is represented in a specific format.

Example: Fast counter: Actual value 1. These channels are listed at last position of a device.

Editing I/O mapping list

1. In the device tree, double-click IO_Bus to configure entire I/O mapping list of different I/O devices.
2. Enter the variables and descriptions to map the I/O devices.
   - Do not start variable names with a number or a special character. When saving the project, this generates an error. Example: 12input3, @input4.
3. Click Save Project to save the I/O mapping changes.

Toolbar

Filtering

Especially in case of long I/O mapping lists, it might be helpful to filter the I/O mappings. For this, click the “Filter” icon to display all available criteria for filter options.

Undo, redo and clear

- **Undo**: Cancels the last change.
- **Redo**: Repeats the last change.
- **Clear mappings**: Deletes all variables and descriptions.
Fast counter

Configuration for S500 I/O devices

1. In the device tree, add a digital I/O Module to the “IO-Bus” node.
2. Double-click the node for the I/O Module, open the “Parameters” tab and set the counting mode.  
   See Chapter 1.5.6.2.9.8.2.1 “Counting modes” on page 3318 of the “Fast counter” parameter.
3. In the “I/O Mapping” tab channel configuration is displayed.  
   See Chapter 1.5.6.2.9.8.3 “Control of the fast counter” on page 3322.

Operands

Table 698: Input information

<table>
<thead>
<tr>
<th>Description of the input information</th>
<th>Output information of the user program</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Start value 1</td>
<td>Output double word 0</td>
<td>Double word</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Set values for the counters 1 and 2: Each counter can be set to a start value. Start values are loaded into the counter by the user program. Using the set signal (depending on the operating mode either via a terminal or the bit SET within the control byte 1 or 2), the values of the double word variables are loaded into the counter 1 or 2.</td>
</tr>
<tr>
<td>Start value 2</td>
<td>Output double word 1</td>
<td>Double word</td>
</tr>
<tr>
<td></td>
<td></td>
<td>End value for the counters 1 and 2:</td>
</tr>
<tr>
<td>End value 1</td>
<td>Output double word 2</td>
<td>Double word</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The end values for the two counters are stored as comparison values into the module by the user program. Both counters compare continuously whether or not their programmed end value is equal to their actual value. When the counter (actual value) reaches its programmed end value, the binary output CF of the status byte is set permanently.</td>
</tr>
<tr>
<td>End value 2</td>
<td>Output double word 3</td>
<td>Double word</td>
</tr>
<tr>
<td>Description of the input information</td>
<td>Output information of the user program</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------------------</td>
<td>---------------------------------------</td>
<td>-------------</td>
</tr>
</tbody>
</table>
| Control byte 1  
(see 1) | Output byte 0 | Byte:  
Bit 0 = UP/DWN  
Bit 1 = EN  
Bit 2 = SET  
Bit 3 = CF_HW  
Bit 4 to Bit 7 free | Control bytes for the counter 1:  
**UP/DWN**: In some operating modes, the counter can count downwards, too. If counting down is desired, set the bit UP/DWN to TRUE and the bit SET to 1. When doing so, the counter starts counting downwards from the start value (set value) to the end value (max. from 4,294,967,295 to 0 or hexadecimal from FF FF FF FF to 00 00 00 00). After reaching 0 the counter jumps to 4,294,967,295.  
**EN**: Processing of the counter signals must be enabled. Depending on the operating mode, enabling is done via a terminal or by the bit EN = TRUE within the control byte.  
**SET**: The counter can be set to a start value (see the description of the set values for the counters 1 and 2 at the beginning of this table).  
**CF_HW**  
0 = state of CF is set to hardware channel (only for mode 1 and 2)  
1 = normal output is set to hardware channel  
Bit 3 is evaluated only in control byte of counter 1. |
| Control byte 2  
(see 1) | Output byte 0 | Byte:  
Bit 0 = UP/DWN  
Bit 1 = EN  
Bit 2 = SET  
Bit 3 to Bit 7 free | Control bytes for the counter 2:  
**UP/DWN**: In some operating modes, the counter can count downwards, too. If counting down is desired, set the bit UP/DWN to TRUE and the bit SET to 1. When doing so, the counter starts counting downwards from the start value (set value) to the end value (max. from 4,294,967,295 to 0 or hexadecimal from FF FF FF FF to 00 00 00 00). After reaching 0 the counter jumps to 4,294,967,295.  
**EN**: Processing of the counter signals must be enabled. Depending on the operating mode, enabling is done via a terminal or by the bit EN = TRUE within the control byte. |
Description of the input information | Output information of the user program | Description
--- | --- | ---

**SET**: The counter can be set to a start value (see the description of the set values for the counters 1 and 2 at the beginning of this table.

1) Only for CI581-CN/CI582-CN: Control bytes 1 and 2 are available twice on grounds of data consistency. Hence, a Start and End evaluation is only effected if the signals "Control Byte1_0" and "Control Byte1_1" or "Control Byte2_0" and "Control Byte2_1" (process image) are identical.

<table>
<thead>
<tr>
<th>Table 699: Output information</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Output information</strong></td>
</tr>
<tr>
<td>Actual Value 0</td>
</tr>
<tr>
<td>Actual Value 1</td>
</tr>
<tr>
<td>Status Byte 0</td>
</tr>
<tr>
<td>Status Byte 1</td>
</tr>
</tbody>
</table>

**Operating modes**

Inputs and outputs which are not used by the counters, are available for other tasks.

Legend:
- A refers to input channel A
- B refers to input channel B
- C refers to output channel C

<table>
<thead>
<tr>
<th>Operating mode</th>
<th>Function</th>
<th>Used inputs and outputs</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>No counter</td>
<td>none</td>
<td>This operating mode is selected if the integrated fast counter is not necessary.</td>
</tr>
<tr>
<td>1</td>
<td>One up-counter</td>
<td>A = Counting input C = End value reached</td>
<td>The counting input and the output &quot;End value reached) are enabled by the bit EN = TRUE within the control byte.</td>
</tr>
<tr>
<td>Operating mode</td>
<td>Function</td>
<td>Used inputs and outputs</td>
<td>Notes</td>
</tr>
<tr>
<td>----------------</td>
<td>----------</td>
<td>-------------------------</td>
<td>-------</td>
</tr>
</tbody>
</table>
| 2              | One up-counter with enable input via terminal | A = Counting input  
B = Enable input  
C = End value reached | The enable input enables the counting input and the output "end value reached". The counter is only enabled if the enable input = TRUE (signal 1) AND the bit EN = TRUE within the control byte. |
| 3              | Two up/down counters | A = Counting input 0  
B = Counting input 1 | With this operating mode, two counters exist, which are independent of each other. The state "End value reached" is only readable from the two status bytes. It is not readable from output terminals. The counting direction is defined by the bit UP/DWN within the control byte. |
| 4              | Two up/down counters (1 counting input inverted) | A = Counting input 0  
B = Counting input 1 | This operating mode equals operating mode 3 with one exception: The counting input B (of counter 1) is inverted. It counts the TRUE/FALSE edges at input B. |
| 5              | One up/down counter with a dynamic set input via terminal | A = Counting input  
B = Dynamic set input | With this operating mode, one up/down counter is available which has a dynamic set input. Dynamic means that the set operation is performed at the FALSE/TRUE signal edge (0/1 edge) of the set input and not while the signal is TRUE. The state "End value reached" is only readable from the status byte, not from an output terminal. |
| 6              | One up/down counter with a dynamic set input via terminal | A = Counting input  
B = Dynamic set input | This operating mode equals operating mode 5 with one exception: The dynamic set input operates at the TRUE/FALSE edge (1-0 edge). |
<table>
<thead>
<tr>
<th>Operating mode</th>
<th>Function</th>
<th>Used inputs and outputs</th>
<th>Notes</th>
</tr>
</thead>
</table>
| 7             | One up/down counter for position sensors | A = Trace A of the position sensor
B = Trace B of the position sensor                                                 | With this operating mode, incremental position sensors can be used which interchange their counting signals on tracks A and B in a 90° phase sequence. Depending on the sequence of the signals at A and B, the counter counts up or down. There is no pulse-multiplier function (e.g. x2 or x4). The position sensor must provide 24 V signals. Signals of 5 V sensors must be converted. Zero traces are not processed. The state "End value reached" is only readable from the state byte 0, not from an output terminal. The bit UP/DWN within the control byte must be FALSE. Otherwise, a parameter error occurs. In this operating mode, the maximum counting frequency is: I/O modules 35 kHz. Communication interface modules 50 kHz. |
| 8             | Reserved                       |                                                                                         |                                                                                                                                                                                                     |

PLC Automation with V3 CPUs
PLC integration > Configuration in Automation Builder for AC500 V3 products
<table>
<thead>
<tr>
<th>Operating mode</th>
<th>Function</th>
<th>Used inputs and outputs</th>
<th>Notes</th>
</tr>
</thead>
</table>
| 9              | One up/down counter for position sensors (pulse multiplier x2) | A = Trace A of the position sensor  
B = Trace B of the position sensor | This operating mode equals operating mode 7 with one exception: There is a pulse multiplication x2 with the evaluation of the counting inputs. This means, that the counter counts both the positive edges and the negative edges of trace A. This results in the double number of counting pulses. The precision increases correspondingly.  
In this operating mode, the maximum counting frequency is:  
I/O modules 30 kHz.  
Communication interface modules 35 kHz. |
| 10             | One up/down counter for position sensors (pulse multiplier x4) | A = Trace A of the position sensor  
B = Trace B of the position sensor | This operating mode equals operating mode 7 with one exception: There is a pulse multiplication x4 with the evaluation of the counting inputs. This means, that the counter counts the positive and negative edges of the traces A and B. This results in the fourfold number of counting pulses. The precision increases correspondingly.  
In this operating mode, the maximum counting frequency is:  
I/O modules 15 kHz.  
Communication interface modules 20 kHz. |
Configuration for onboard I/Os

1. In the device tree, double-click the “Onboard I/O” node (OBIO).

2. In the “Parameters” tab set the counting mode (Chapter 1.5.6.2.9.8.2.1 “Counting modes” on page 3318) for the fast counter.

3. In the “I/O Mapping” tab channel configuration is displayed. (Chapter 1.5.6.2.9.8.3 “Control of the fast counter” on page 3322)

Configuring the fast counter

The parameter of the Fast Counter channels of the Onboard I/O must be configured before they can be used. User should take these steps to configure the fast counter:

<table>
<thead>
<tr>
<th>Channel</th>
<th>Direction</th>
<th>Width</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Actual value X</td>
<td>Input</td>
<td>DWORD</td>
<td>Current value of the Fast Counter.</td>
</tr>
</tbody>
</table>
| State byte X  | Input     | BYTE  | Bit 0 = CF
If the counter reaches the programmed end value, the counter output is stored permanently as CF = TRUE (end value reached). Only, if the counter is set again (set value), CF is reset to FALSE. **Bit 1 to Bit 7 free** |
<p>| Start value X | Output    | DWORD | Each counter can be set to a start value. Start values are loaded into the counter by the user program. Using the set signal (dependent on the operating mode either via a terminal or the bit SET within the control byte X), the values of the double word variables are loaded into the counter X. |
| End value X   | Output    | DWORD | The end values for the two counters are stored as comparison values into the module by the user program. Both counters compare continuously, whether or not their programmed end value is equal to their actual value. If the counter (actual value) reaches its programmed end value, the binary output CF of the status byte is set permanently. |</p>
<table>
<thead>
<tr>
<th>Channel</th>
<th>Direction</th>
<th>Width</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Control byte 1</td>
<td>Output</td>
<td>BYTE</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td><strong>Bit 0 = UP/DWN</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>In some operating modes, the counter can count downwards, too. If counting down is desired, the bit UP/DWN must be set to TRUE. When doing so, the counter starts counting downwards at the start value (set value) to the end value (max. from 4,294,967,295 to 0 or hexadecimal from FF FF FF FF to 00 00 00 00). After reaching 0, the counter jumps to 4,294,967,295.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td><strong>Bit 1 = EN</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The processing of the counter signals must be enabled. Depending on the operating mode, enabling is done via a terminal or by the bit EN = TRUE within the control byte.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td><strong>Bit 2 = SET</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The counter can be set to a start value (see the description of the set values for the counters 1 and 2 at the beginning of this table).</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td><strong>Bit 3 = CF_HW</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0 = state of CF is set to hardware channel (only for mode 1 and 2)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1 = normal output is set to hardware channel</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Bit 3 is evaluated only in control byte of counter 1.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td><strong>Bit 4 to Bit 7 free</strong></td>
</tr>
<tr>
<td>Control byte 2</td>
<td>Output</td>
<td>BYTE</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td><strong>Bit 0 = UP/DWN</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>In some operating modes, the counter can count downwards, too. If counting down is desired, the bit UP/DWN must be set to TRUE. When doing so, the counter starts counting downwards at the start value (set value) to the end value (max. from 4,294,967,295 to 0 or hexadecimal from FF FF FF FF to 00 00 00 00). After reaching 0, the counter jumps to 4,294,967,295.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td><strong>Bit 1 = EN</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The processing of the counter signals must be enabled. Depending on the operating mode, enabling is done via a terminal or by the bit EN = TRUE within the control byte.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td><strong>Bit 2 = SET</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The counter can be set to a start value (see the description of the set values for the counters 1 and 2 at the beginning of this table).</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td><strong>Bit 3 to Bit 7 free</strong></td>
</tr>
</tbody>
</table>

**Counting modes**

The fast counter can be configured as one mode out of 10 possible modes. The desired operating mode is selected in the PLC configuration using configuration parameters. Inputs and outputs which are not used by the counter are available for other tasks. In the following table, A means input channel A, B means input channel B and C means output channel C.
<table>
<thead>
<tr>
<th>CPUs</th>
<th>Integrated fast counter</th>
<th>Assigned inputs</th>
<th>Assigned Outputs</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>PM55x, PM56x</td>
<td>Yes</td>
<td>Input channel 0</td>
<td>Input channel 1</td>
<td>Output channel 0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Only 1 fast counter is available on the module. Input channel 0 is the default channel for fast counter. Input channel 1 can be used as another fast counter channel depending on fast counter mode.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Operating Mode</th>
<th>Function</th>
<th>Input channels</th>
<th>Description</th>
<th>Counting frequency (max.) for PM5x4-T and PM5x4-R</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>No counter</td>
<td>None</td>
<td>Fast counter is disabled</td>
<td>-</td>
</tr>
<tr>
<td>1</td>
<td>1 Up counter</td>
<td>A = Counter input C = End value reached</td>
<td>Counting up A from 0 to 0xFFFFFFFF When the end value is reached, C will be set to high.</td>
<td>30 kHz (before firmware V2.0.6) 50 kHz (since firmware V2.0.6)</td>
</tr>
<tr>
<td>2</td>
<td>1 Up counter with release input</td>
<td>A = Counter input B = Enable input C = End value reached</td>
<td>Counting up A from 0 to 0xFFFFFFFF The counter is enabled if B is high When the end value is reached, C will be set to high.</td>
<td>30 kHz (before firmware V2.0.6) 50 kHz (since firmware V2.0.6)</td>
</tr>
<tr>
<td>Operating Mode</td>
<td>Function</td>
<td>Input channels</td>
<td>Description</td>
<td>Counting frequency (max.) for PM5x4-T and PM5x4-R</td>
</tr>
<tr>
<td>----------------</td>
<td>----------</td>
<td>----------------</td>
<td>-------------</td>
<td>---------------------------------------------------</td>
</tr>
</tbody>
</table>
| 3              | 2 Up/Down counters          | A = Counter input 1  
B = Counter input 2 | 2 independent counters. Status “End value reached” is only readable from the 2 status bytes, not from output terminals. The counting direction is defined by the boolean parameters UD1 and UD2 of Function Block ONB_IO_CNT (Handle Fast Counter on Onboard I/O) | 30 kHz (before firmware V2.0.6)  
50 kHz (since firmware V2.0.6) |
| 4              | 2 Up/Down counters (2nd on falling edges) | A = Counter input 1  
B = Counter input 2 | Same as operating mode 3, but counting input B is inverted (counts at TRUE/FALSE edges at input B). | 30 kHz (before firmware V2.0.6)  
50 kHz (since firmware V2.0.6) |
| 5              | 1 Up/Down counter with dynamic set/ rising edge | A = Counter input 1  
B = Dynamic set input | 1 Up/Down counter is available which counts on the rising edge of A and has a dynamic set input on B. Dynamic set input will set the start value at the rising edge of B. | 30 kHz (before firmware V2.0.6)  
50 kHz (since firmware V2.0.6) |
| 6              | 1 Up/Down counter with dynamic set/ falling edge | A = Counter input 1  
B = Dynamic set input | 1 Up/Down counter is available which counts on the rising edge of A and has a dynamic set input on B. Dynamic set input will set the start value at the falling edge of B. | 30 kHz (before firmware V2.0.6)  
50 kHz (since firmware V2.0.6) |
<table>
<thead>
<tr>
<th>Operating Mode</th>
<th>Function</th>
<th>Input channels</th>
<th>Description</th>
<th>Counting frequency (max.) for PM5x4-T and PM5x4-R</th>
</tr>
</thead>
</table>
| 7              | 1 UpDown directional discriminator | A = Phase A  
B = Phase B | With this mode, incremental encoders can be used which give their counting signals on phase A and B in a 90° phase sequence to each other.  
Dependent on the sequence of the signals at A and B, the counter counts up or down.  
There is no pulse multiplier function. | 12 kHz (before firmware V2.0.6)  
35 kHz (since firmware V2.0.6) |
<p>| 8              | Reserved | -              | -           | -                                                 |</p>
<table>
<thead>
<tr>
<th>Operating Mode</th>
<th>Function</th>
<th>Input channels</th>
<th>Description</th>
<th>Counting frequency (max.) for PM5x4-T and PM5x4-R</th>
</tr>
</thead>
<tbody>
<tr>
<td>9</td>
<td>1 UpDown directional discriminator X2</td>
<td>A = Phase A B = Phase B</td>
<td>This mode is the same as mode 7 with one exception: There is a pulse multiplication x2 with the evaluation of the counting inputs. This means that the counter counts both the positive edges and the negative edges of phase A. This results in the double number of counting pulses. The precision increases correspondingly.</td>
<td>11 kHz (before firmware V2.0.6) 30 kHz (since firmware V2.0.6)</td>
</tr>
<tr>
<td>10</td>
<td>1 UpDown directional discriminator X4</td>
<td>A = Phase A B = Phase B</td>
<td>This mode is the same as mode 7 with one exception: There is a pulse multiplication x4 with the evaluation of the counting inputs. This means that the counter counts both the positive edges and the negative edges of phase A and B. This results in the fourfold number of counting pulses. The precision increases correspondingly.</td>
<td>10 kHz (before firmware V2.0.6) 15 kHz (since firmware V2.0.6)</td>
</tr>
</tbody>
</table>

If channel 0 is configured as fast counter, the other channels 1, 2 and 3 cannot be configured as interrupt inputs. Otherwise, a configuration error will appear and the CPU will be stopped.

Control of the fast counter

To control the fast counter configuration open the “I/O Mapping” tab.

The channels can be mapped as described in Symbolic Names for Variables, Inputs and Outputs and have the following meaning in Chapter 1.1.12.2 “Symbolic names for variables, inputs and outputs” on page 36:
<table>
<thead>
<tr>
<th>Channel</th>
<th>Direction</th>
<th>Width</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Actual value X</td>
<td>Input</td>
<td>DWORD</td>
<td>Current value of the fast counter</td>
</tr>
<tr>
<td>State byte X</td>
<td>Input</td>
<td>BYTE</td>
<td>Bit 0 = CF If the counter reaches the programmed end value, the counter output is stored permanently as CF = TRUE (end value reached). Only if the counter is set again (set value), CF is reset to FALSE. Bit 1 to Bit 7 free</td>
</tr>
<tr>
<td>Start value X</td>
<td>Output</td>
<td>DWORD</td>
<td>Each counter can be set to a start value. Start values are loaded into the counter by the user program. Using the set signal (dependent on the operating mode either via a terminal or the bit SET within the control byte X), the values of the double word variables are loaded into the counter X.</td>
</tr>
<tr>
<td>End value X</td>
<td>Output</td>
<td>DWORD</td>
<td>The end values for the 2 counters are stored as comparison values into the module by the user program. Both counters compare continuously whether or not their programmed end value is equal to their actual value. When the counter (actual value) reaches its programmed end value, the binary output CF of the status byte is set permanently.</td>
</tr>
<tr>
<td>Channel</td>
<td>Direction</td>
<td>Width</td>
<td>Description</td>
</tr>
<tr>
<td>--------------</td>
<td>-----------</td>
<td>-------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
</tbody>
</table>
| Control byte 1 | Output    | BYTE  | Bit 0 = UP/DWN  
In some operating modes, the counter can count downwards, too. If counting down is desired, the bit UP/DWN must be set to TRUE. If doing so, the counter starts counting downwards at the start value (set value) to the end value (max. from 4,294,967,295 to 0 or hexadecimal from FF FF FF FF to 00 00 00 00). After reaching 0 the counter jumps to 4,294,967,295. |
|              |           |       | Bit 1 = EN  
The processing of the counter signals must be enabled. Depending on the operating mode, enabling is done via a terminal or by the bit EN = TRUE within the control byte. |
|              |           |       | Bit 2 = SET  
The counter can be set to a start value (see the description of the set values for the counters 1 and 2 at the beginning of this table. CF = 0 |
|              |           |       | Bit 3 = CF_HW  
0 = state of CF is set to hardware channel (only for mode 1 and 2)  
1 = normal output is set to hardware channel  
Bit 3 is evaluated only in control byte of counter 1. |
|              |           |       | Bit 4 to Bit 7 free |
| Control byte 2 | Output    | BYTE  | Bit 0 = UP/DWN  
In some operating modes, the counter can count downwards, too. If counting down is desired, the bit UP/DWN must be set to TRUE. If doing so, the counter starts counting downwards at the start value (set value) to the end value (max. from 4,294,967,295 to 0 or hexadecimal from FF FF FF FF to 00 00 00 00). After reaching 0 the counter jumps to 4,294,967,295. |
|              |           |       | Bit 1 = EN  
The processing of the counter signals must be enabled. Depending on the operating mode, enabling is done via a terminal or by the bit EN = TRUE within the control byte. |
<table>
<thead>
<tr>
<th>Channel</th>
<th>Direction</th>
<th>Width</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>Bit 2 = SET</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The counter can be set to a start value (see the description of the set values for the counters 1 and 2 at the beginning of this table).</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Bit 3 to Bit 7 free</td>
</tr>
</tbody>
</table>

### 1.5.6.2.10 Serial interface

**Configuring Modbus RTU on serial interface**

To enable Modbus RTU on a serial interface the protocol setup per default has to be replaced by either Modbus RTU Client or Server, depending on required operation mode.

A serial interface supports only one protocol/operation mode at once.

![Configuration screen](image)

**Replace object “CAA_SerialCom”**

1. Right-click node “CAA_SerialCom” and click “Add object”.
2. Select “Modbus RTU Client” or “Modbus RTU Server” and click “Add object”.
   ⇒ “CAA_SerialCom” is replaced by your selection.

**Parameters**

**Serial**

Serial parameters to be set selecting the interfaces node “COM_1”. They are common for both operating modes client and server.
The parameter “Data bits” always has to be set to “8” for Modbus.

Modbus RTU server

Server specific parameters to be set selecting the protocol’s node “Modbus_RTU_Server”.

Address
Bus address of the PLC as Modbus RTU Server on that interface

Byte Order
Format/Endianess for the transmission of WORD values (register) within the request/response telegram (default: Big Endian)

Disable

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Default</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Disable write to %MB from</td>
<td>0</td>
<td>0 ... 65535</td>
<td>Disable write access starting at %MBx</td>
</tr>
<tr>
<td>Disable write to %MB to</td>
<td>0</td>
<td>0 ... 65535</td>
<td>Disable write access up to %MBx</td>
</tr>
<tr>
<td>Disable read from %MB from</td>
<td>0</td>
<td>0 ... 65535</td>
<td>Disable read access starting at %MBx</td>
</tr>
<tr>
<td>Disable read from %MBx to</td>
<td>0</td>
<td>0 ... 65535</td>
<td>Disable read access up at %MBx</td>
</tr>
</tbody>
</table>

It is possible to disable read and/or write access to individual segments. Reading/writing is disabled beginning at the set start address and is valid up to the set end address (inclusive).

Modbus RTU client

“Modbus RTU Client” does not have any protocol parameters.

Configuring CAA SerialCom on serial interface

The protocol CAA SerialCom represents the standard serial protocol provided by 3S and allows the users to implement their own custom protocol.

For details on CAA SerialCom, refer to standard 3S V3 documentation (see, CAA SerialCom library).
When creating a new project, the protocol “CAA SerialCom” is automatically attached to the “COM_1” port of a V3 PLC.

Right-click on the node attached to “COM_1” node in the device tree and click “Delete”. The node is switched back to the “CAA SerialCom” protocol.

Since CAA SerialCom doesn’t represent a “real” protocol, there are no specific parameters required. All common settings can be found at the Tab “COM_1” after double-click on the “COM_1” node (see also Chapter 1.5.6.2.10.3 “Setting up a serial interface” on page 3330).

Activate particular configuration parameters

The parameters set up in the Automation Builder device tree are NOT automatically taken over in the PLC.
It is still required to use the 3S IEC POUs to activate the particular configuration parameters.

ABB provides the library AC500_Com ("ABB - AC500 / Use Cases / Serial Communication") which contains a POU called “ComGetCaaSerialComConfig”.

The Function Block can be used to obtain the configuration data which is set up in Automation Builder to directly pass it to CAA SerialCom-POU Open. This avoids manual creation of a parameter list.

The following code snippet shows, how the COM port is identified by its node name and how the parameter list for the Function Block is read from the configuration data of the currently loaded IEC application:

FUNCTION_BLOCK GET_CAA_COM_CFG
VAR_INPUT
END_VAR
VAR_OUTPUT
END_VAR
VAR

  ComGetCaaSerialComConfig:    ComGetCaaSerialComConfig;
  bExecGetCfg:              BOOL := FALSE;
  bDoneGetCfg:              BOOL := FALSE;
  bBusyGetCfg:              BOOL := FALSE;
  bErrorGetCfg:             BOOL := FALSE;
  ErrorIdGetCfg:         AC500_Com.ERROR_ID := AC500_Com.ERROR_ID.NO_ERROR;
  asParamList:              ARRAY[0..31] OF AC500_Com.Serial_Communication.COM.PARAMETER;
  uiNumParams:              UINT := 32;
  uiStep:                   UINT := 0;
  szNodeName:               STRING(80) := 'COM1';
  ComID:                       AC500_Com.COM_PORT_ID;
  bSuccess:                 BOOL := FALSE;
  bError:                   BOOL := FALSE;
END_VAR
VAR CONSTANT
STEP_INIT: UINT := 0;
STEP_GET_ID: UINT := 1;
STEP_FAILED_GET_ID: UINT := NOT STEP_GET_ID;
STEP_GET_CFG_CAA: UINT := (STEP_GET_ID + 1);
STEP_FAILED_GET_CFG_CAA: UINT := NOT STEP_GET_CFG_CAA;
STEP_DONE_SUCCESS: UINT := (STEP_GET_CFG_CAA + 1);
END_VAR

IF uiStep = STEP_GET_ID THEN
    ComId := ComGetIdByName(szNodeName);
    IF ComId = AC500_Com.COM_PORT.COM_ID_INVALID THEN
        uiStep := STEP_FAILED_GET_ID;
    ELSE
        bExecGetCfg := TRUE;
        uiStep := STEP_GET_CFG_CAA;
    END_IF
END_IF

IF uiStep = STEP_GET_CFG_CAA THEN
    ComGetCaaSerialComConfig(
        Execute:= bExecGetCfg,
        Done=> bDoneGetCfg,
        Busy=> bBusyGetCfg,
        Error=> bErrorGetCfg,
        ComID:= ComID,
        pCaaParamList:= ADR(asParamList[0]),
        NumParams:= uiNumParams,
        ErrorID=> ErrorIdGetCfg);
    IF bDoneGetCfg THEN
        uiStep := STEP_DONE_SUCCESS;
    ELSIF bErrorGetCfg THEN
        uiStep := STEP_FAILED_GET_CFG_CAA;
    END_IF
END_IF

IF uiStep = STEP_DONE_SUCCESS THEN
    bSuccess := TRUE;
END_IF

IF uiStep = STEP_FAILED_GET_ID THEN
    bError := TRUE;
END_IF

IF uiStep = STEP_FAILED_GET_CFG_CAA THEN
    bError := TRUE;
END_IF
Setting up a serial interface

General

The configuration for serial interfaces and their protocols is done via two nodes:

- One node represents the common serial parameters related to the hardware port.
- The node below represents the parameterization for the particularly attached protocol.

Protocols supported by AC500 V3 PLCs

- 3S CAA SerialCom (common serial communication, send/receive data  Chapter 1.5.6.2.10.2 “Configuring CAA SerialCom on serial interface” on page 3326)
- Modbus RTU (client & server  Chapter 1.5.6.2.10.1 “Configuring Modbus RTU on serial interface” on page 3325)

How to switch between the protocols, see “Default setting” on page 3327.

Configuration

The following parameters are available in the configuration view of the COM port node:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value ranges</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Run on config fault</td>
<td>No, Yes</td>
<td>If this parameter is set to “Yes” the IEC application will not be prevented from switching to RUN state, independent from possibly existing configuration errors of the particular COM port.</td>
</tr>
<tr>
<td>Transmission rate</td>
<td>9600 baud/sec, 19200, 38400, 57600, 115200</td>
<td>Sets up the transmission rate to use for the COM port.</td>
</tr>
<tr>
<td>Parity</td>
<td>None, Odd, Even</td>
<td>Sets up the parity to use for the COM port.</td>
</tr>
<tr>
<td>Data bits</td>
<td>5 data bits, 6 data bits, 7 data bits, 8 data bits</td>
<td>Sets up the number of data bits to use for the COM port.</td>
</tr>
<tr>
<td>Stop bits</td>
<td>1 stop bits, 2 stop bits</td>
<td>Sets up the number of stop bits to use for the COM port.</td>
</tr>
<tr>
<td>Parameter</td>
<td>Value ranges</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------</td>
<td>-------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Flow control</td>
<td>No flow control</td>
<td>Allows to switch between different flow control modes (either RTS/CTS hardware or Xon/Xoff software or none). This setting is only valid for RS-232 serial interface mode. In case RS-485 is used for parameter “Serial interface”, flow control must set to “No flow control”. Otherwise a configuration error is triggered.</td>
</tr>
<tr>
<td></td>
<td>Hardware RTS/CTS</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Software Xon/Xoff</td>
<td></td>
</tr>
<tr>
<td>Boot parameter</td>
<td>RS-232</td>
<td>Allows to switch between RS-232 and RS-485. Due to technical reasons, it’s not possible to dynamically switch between the modes. This means, a reboot (or power cycle) of the PLC is required to activate the particular setting once changed.</td>
</tr>
<tr>
<td>Serial interface</td>
<td>RS-485</td>
<td></td>
</tr>
</tbody>
</table>

Comparison to V2

The following table shows the differences between V2 and V3 PLCs regarding the parameter set for serial interfaces:

<table>
<thead>
<tr>
<th>V2 Parameter</th>
<th>Representation in V3</th>
<th>Remark</th>
</tr>
</thead>
<tbody>
<tr>
<td>Run on config fault</td>
<td>Run on config fault</td>
<td>Exactly the same</td>
</tr>
<tr>
<td>RTS control</td>
<td>Flow control (partially)</td>
<td>Special modes which allow to use PLC as modem and mode implicitly setting RS-485 will not be taken over. Flow control settings will be limited and only support hardware, software or none.</td>
</tr>
<tr>
<td>Transmission rate</td>
<td>Transmission rate</td>
<td>For V3, the transmission rate range will be 9600 to 115200. Low modes will not be supported due to lack of support in Linux. High rates were only realized in V2 to support field bus plug as well as CS31 field bus. Both protocols are not supported anymore in V3, so these transmission rates won’t be available in V3. Approach: Only support most common transmission rates</td>
</tr>
<tr>
<td>Parity</td>
<td>Parity (subset)</td>
<td>A500 V3 doesn’t allow to configure parity modes “mark” and “space”. This means, only “none”, “odd” and “even” are configurable.</td>
</tr>
<tr>
<td>Data Bits</td>
<td>Data Bits</td>
<td>Exactly the same</td>
</tr>
<tr>
<td>Stop Bits</td>
<td>Stop Bits</td>
<td>Exactly the same</td>
</tr>
</tbody>
</table>

1.5.6.2.11 Gateway configuration

1. In the Automation Builder project, right-click the topmost PLC tree node and select “Communication Settings”.
   ➤ The dialog window Communication Settings appears.

2. Click “Advanced Settings” to open the Communication Parameters ➤ Chapter 1.3.1.25.1.19.2 “Tab ‘Communication Settings’” on page 741 dialog.
   ➤ This information will be stored in the project file.
3. Click “Gateway” and select the desired action from the Gateway menu either to change the local gateway (see Configure the local gateway Chapter 1.3.1.25.2.18.2 “Command ‘Configure the local gateway’” on page 1013) or to add a new gateway channel (see Add new gateway Chapter 1.3.1.25.2.18.1 “Command ‘Add gateway’” on page 1012).

Confirm your settings with “OK”.

Gateway settings on windows server 2012

Gateway as a service

To allow multiple concurrent users from different user sessions on the server to connect to PLCs, the user has to run CODESYS gateway as a system service. This is managed by a service called "CoDeSys V2.3 Gateway Service Wrapper". The service starts on system start-up and launch the gateway.

If you want to restart the gateway, use "Services management console" to restart "CoDeSys V2.3 Gateway Service Wrapper".

Gateway settings

You can set the communication settings in the Automation Builder project for every PLC. Otherwise, an error message is displayed while trying to open CODESYS.

See the description for Chapter 1.5.6.2.11 “Gateway configuration” on page 3331 and select “TCP/IP” under “Connection”.

1.5.6.2.12 CAN-based fieldbuses

CANopen

CANopen manager (master)

Tab 'CANopen Manager - General'

Table 700: “General”

<table>
<thead>
<tr>
<th>“Node ID”</th>
<th>The node number identifies the CANopen manager as unique and if corresponds to the number set on the module (value between 1 and 127).</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Check and Fix Configuration”</td>
<td>Opens the dialog of the same name. See below for details.</td>
</tr>
</tbody>
</table>
| “Autostart CANopen Manager” | ✔: The CANopen manager starts automatically (switches to OPERATIONAL mode) after all required slaves are ready.  
☐: You have to start the CANopen manager from the application. Use the CiA405 NMT function block to do this.  
Hint: As long as the CANopen manager is not in OPERATIONAL mode, no PDOs are sent (outputs refreshed). |
| “Polling of optional slaves” | ✔: When a slave does not respond during the boot sequence, the CANopen manager interrogates it every second until it does respond.  
Constantly polling the slave increases the bus cycle time, which can interfere with the application (especially motion applications). You can deactivate polling to avoid this behavior. If polling is deactivated, then a slave is detected again when it sends a bootup message. |
| “Start Slaves” | ✔: The CANopen manager is responsible for starting the slaves.  
☐: You have to start the slaves from the application. Use the CiA405 NMT function block to do this. |
**NMT start all (if possible)**: If the “Start slaves” option is activated, then the CANopen manager starts all slaves with an "NMT Start All" command. The "NMT Start All" command is not executed as long as optional slaves are not yet ready to be started. In this case, the CANopen manager starts each slave individually. The "NMT Start All" command can be guaranteed only in a project without optional slaves.

**NMT error behavior**:
- "Restart Slave". If an error occurs during slave monitoring (NMT Error Event), then the slave is restarted automatically by the stack (NMT Reset + SDO Configuration + NMT Start).
- "Stop Slave". If an error occurs during slave monitoring (NMT Error Event), then the slave is stopped. Then you must reset the slave from the application, using the CiA405 NMT function block.

---

**Table 701: “Guarding”**

Working with heartbeat messages is an alternative method of monitoring. It can be executed from both master and slave nodes, as opposed to node-guarding. Normally the master sends heartbeat messages to the slaves.

<table>
<thead>
<tr>
<th><strong>Enable heartbeat producing</strong></th>
<th>The master sends heartbeat messages that define the time interval in the “Producer Time”. When the slaves are provided with the heartbeat function, a heartbeat consuming entry from the slave is created for the master. Then the node ID and the 1.5x heartbeat interval of the master are applied.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Node ID</strong></td>
<td>Unique identification (1-127) of the heartbeat producer on the bus.</td>
</tr>
<tr>
<td><strong>Producer time (ms)</strong></td>
<td>Interval length between successive heartbeats (in milliseconds).</td>
</tr>
<tr>
<td><strong>Redundancy node ID</strong></td>
<td>Requirement: A “Redundancy Configuration” object is inserted below the application. Unique identification (1-127) of the redundant heartbeat producer on the bus.</td>
</tr>
<tr>
<td><strong>Redundancy wait time (µs)</strong></td>
<td>Requirement: A “Redundancy Configuration” object is inserted below the application. Duration of how long the passive controller waits for the heartbeat of the active controller. If this time is exceeded, then the passive controller takes on the active role.</td>
</tr>
</tbody>
</table>

---

**Table 702: “SYNC”**

<table>
<thead>
<tr>
<th><strong>Enable SYNC producing</strong></th>
<th>(disabled by default) The CANopen manager sends SYNC telegrams. The synchronous PDOs are sent directly after the SYNC telegram.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>COB ID (Hex)</strong></td>
<td>CAN ID of the SYNC telegram. Range of possible values: [1...2047].</td>
</tr>
<tr>
<td><strong>Cycle period (µs)</strong></td>
<td>Interval length (in microseconds) after which the SYNC telegram is sent.</td>
</tr>
<tr>
<td><strong>Window length (µs)</strong></td>
<td>Length of the time frame for synchronous PDOs (in microseconds).</td>
</tr>
<tr>
<td><strong>Enable SYNC consuming</strong></td>
<td>(disabled by default) Another device must produce the SYNC telegrams that are received by the CANopen manager.</td>
</tr>
</tbody>
</table>

---

**NOTICE!**

If SYNC producing is enabled for the CANopen manager, then you are not permitted to activate the “Enable Sync Producing” option for all other bus devices.
Table 703: “TIME”

<table>
<thead>
<tr>
<th>“Enable TIME producing”</th>
<th>[✓] (disabled by default) The CANopen manager sends TIME messages.</th>
</tr>
</thead>
<tbody>
<tr>
<td>“COB ID (Hex)”</td>
<td>(Communication Object Identifier): identifies the time stamp of the message. Default values: [0...2047], preset 16#100.</td>
</tr>
<tr>
<td>“Producer time (ms)”</td>
<td>Interval (in milliseconds) when the time stamp is sent. This value must be a multiple of the task cycle time. Possible values [0...65535].</td>
</tr>
</tbody>
</table>

The runtime system must support high resolution timestamps. If not, then an error message is displayed.

See also
- Chapter 1.5.6.2.12.1.2.1.2.1 “Tab ‘CANopen Remote Device - General’” on page 3335

Dialog ‘Check and Fix Configuration’

If you insert several devices below the CANopen manager, then error messages may report multiple assigned node IDs or invalid COB IDs. The “Check and Fix Configuration” button opens a dialog for solving these conflicts.

For conflicts with node IDs or PDO COB IDs, you can click “Edit Conflicts” to open a dialog with detailed information.

Table 704: “Node ID and COB ID conflicts”

<table>
<thead>
<tr>
<th>“Doubled node number”</th>
<th>List of all devices with identical IDs. In the field of the “Node ID” column, you can enter new node numbers for the affected devices.</th>
</tr>
</thead>
</table>
| “Incorrect and double assignment of PDO COB IDs” | The COB IDs that are generated automatically from the device description files may not be permitted. All incorrect entries are listed with the respective device names, node IDs, and indexes. There are three options for correcting invalid COB IDs:  
  - Correct the displayed formula for calculating the COB IDs so that a valid COB ID results. You can change the formula in the respective table element.  
  - Accept the automatic suggestion for the COB ID by clicking the respective button.  
  - Accept all automatic suggestions by clicking the “Used suggested COB ID” button. |

Corrected entries are removed from the displayed list automatically.

You can solve timing problems automatically by using the “Automatic Repair”. The command adjusts all timing values to compatible values. (The time should be a multiple of the task time.)

CANopen remote device (Slave)

In CODESYS, a CANopen remote device is a slave device that you insert below a CANopen manager in the device tree of a project. A distinction is made between modular and non-modular slaves:

- **Modular slaves**: You can insert CANopen modules (submodules) below a modular slave. These modules provide a dialog for mapping their inputs and outputs. Modular slaves can also have fixed I/Os. Then these devices also provide the “I/O Mapping” dialog. Modular devices provide the “Configure PDO mapping automatically” option, which we recommend for standard applications. You find this option in the “CANopen Remote Device” dialog in the “General” section.
- **Non-modular slaves**: You cannot insert additional modules below a non-modular device. The inputs and outputs of these devices are mapped in the “I/O Mapping” dialog. Automatic mapping is not possible here.
See also

- § Chapter 1.5.6.2.12.1.1.2.1 "Tab 'CANopen Remote Device - General'" on page 3335

Tab 'CANopen Remote Device - General'

The general settings of the CANopen slave are defined in this dialog box of a CANopen remote device (slave).

Table 705: “General”

<table>
<thead>
<tr>
<th>Setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Node ID</strong></td>
<td>The node number identifies the CAN module as unique and it corresponds to the number set on the module (value between 1 and 127). You provide the node ID as a decimal.</td>
</tr>
<tr>
<td><strong>Enable expert settings</strong></td>
<td>☑ All settings are displayed that are predefined by the device description (EDS file) for the device.</td>
</tr>
<tr>
<td><strong>SDO channels</strong></td>
<td>Click this button to open a dialog for activating the SDO channels that are predefined in the EDS file. Service data objects (SDOs) allow access to all entries in the CANopen object directory. An SDO creates a peer-to-peer communication channel between two devices (SDO server and client channel).</td>
</tr>
<tr>
<td><strong>Optional device</strong></td>
<td>☑ The slave is optional and not required for starting the CAN network.</td>
</tr>
<tr>
<td><strong>Sync producing</strong></td>
<td>Available only when the “Enable sync producing” option is cleared in the CANopen manager.</td>
</tr>
<tr>
<td><strong>No initialization</strong></td>
<td>☑ The I/O transmission is synchronized on the bus. The slave works as a sync producer. The parameters of the sync interval are defined in the settings of the CANopen manager.</td>
</tr>
<tr>
<td><strong>Presettings</strong></td>
<td>The availability of this option depends on the content of the device description file.</td>
</tr>
<tr>
<td></td>
<td>☑ Selected by default. The slave nodes are reset to the default parameters before the configuration is loaded to the device or always when the slave is configured. Which parameters can be set is device-specific. The concrete task is performed from the subindex of the drop-down list.</td>
</tr>
<tr>
<td></td>
<td>- “Sub:001”: All parameters are reset.</td>
</tr>
<tr>
<td></td>
<td>- “Sub:002”: Communication parameters (index 1000h - 1FFFh manufacturer-specific communication parameters) are reset.</td>
</tr>
<tr>
<td></td>
<td>- “Sub:003”: Application parameters (index 6000h - 9FFFh manufacturer-specific application parameters) are reset.</td>
</tr>
<tr>
<td></td>
<td>- “Sub:004” - “Sub:127”: Manufacturer-specific, individual selection of parameters is reset.</td>
</tr>
<tr>
<td></td>
<td>- “Sub:128” - “Sub254”: Reserved for future purposes</td>
</tr>
<tr>
<td><strong>Autoconfig PDO mapping</strong></td>
<td>This option is available for modular devices only.</td>
</tr>
<tr>
<td></td>
<td>☑ PDO mapping is generated automatically from the definitions in the device description and then cannot be changed in the two mapping dialogs. If the automatically generated mapping does not match your application, then you can deactivate the option and configure the mapping manually. We recommend that this option is activated for standard applications.</td>
</tr>
</tbody>
</table>
Node guarding is an outdated monitoring method and should not be used anymore because it uses RTR frames. You should always use heartbeats when possible. In some exceptions, such as for older slaves, you can use only node guarding.

**“Enable node guarding”**

- The CANopen manager sends a message to the slave in the “Guard time (ms)” interval. If the slave does not respond with the given “Guard COB ID” (Communication Object Identifier), then the CANopen manager resends this message as many times as defined in “Life time factor” or until the slave responds.
- If the slave does not respond, then it is marked as "unavailable".

**“Guard time (ms)”**

Interval for sending messages (default: 200 ms)

**“Life time factor”**

When the slave does not respond, a node-guarding error is established according to the “Life time Factor” time multiplied by the “Guard Time”.

**“Enable heartbeat producing”**

- The module sends heartbeats in the time intervals as given in “Producer time (ms)”.

**“Producer time (ms)”**

The default setting is 200 as long as there is no special entry or the entry in the device description file is 0.

**“Heartbeat consuming”**

Opens the “Heartbeat Consuming Properties” dialog box where you activate the slaves that you will monitor.

The number of possible slaves to be monitored is defined in the EDS file. To do this, you must select the “Enable” check box and enter the node ID of the slave and the required values in the “Heartbeat time” field (in milliseconds). Then the slave monitors the heartbeats that are sent from the affected slaves (defined by the node ID). When no more heartbeats are received, the slave switches off the I/Os.

While a slave is monitoring, a green check mark is shown on the “Heartbeat consuming” switch.

**Note:** When you insert a device with the heartbeat function, its heartbeat settings are harmonized automatically with the master (CANopen manager).

**“Enable emergency”**

- When internal errors occur, the slave sends emergency messages with a unique COB ID. You can read these messages by using the function block from the CiA405 library. (RECV_EMCY_DEF, RECV_EMCY).

**“COB ID”**

CAN ID of the EMCY message. Range of possible values: [1...2047].

**“Enable TIME producing”**

- The device sends TIME messages.

**“COB ID (Hex)”**

(Communication Object Identifier): identifies the time stamp of the message.

**“Enable TIME consuming”**

- The device processes TIME messages.

**“Checks at Startup”**

The respective information is read from the firmware of the CANopen slave (0x1018 identity object) and compared to the information from the EDS file. In case of disparities, the configuration is stopped and the slaves are not started.

**“Vendor code”**

- Check of the vendor ID at startup

**“Product number”**

- Check of the product number at startup

**“Revision number”**

- Check of the revision number at startup
See also

- Chapter 1.5.6.2.12.1.1 “Tab 'CANopen Manager - General’” on page 3332

Tab 'CANopen Remote Device - PDOs'

Object: CANopen Remote Device, CANopen Local Device

This dialog is available only in the device editor of a CANopen slave of version V3.5.6.0 or later. It shows all PDOs and their default settings. In this dialog, you can add new objects and delete or edit existing objects.

On the left side, there are the PDOs that the slave receives from the master. On the right side, there are the PDOs that the slave sends to the master.

<table>
<thead>
<tr>
<th>“Add PDO”</th>
<th>Opens the “Select PDO” dialog box where all available PDOs are displayed. In this dialog, you select the PDOs to be added to the “Receive PDOs” or “Transmit PDOs”.</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Add Mapping”</td>
<td>Opens the “Select Item from Object Directory” dialog box. Objects are listed there that you can add to the PDO mapping.</td>
</tr>
<tr>
<td>“Edit”</td>
<td>Opens the “PDO Properties” (when a PDO is selected). When a PDO mapping is selected, the “Select Item from Object Directory” dialog box opens.</td>
</tr>
<tr>
<td>“Delete”</td>
<td>Deletes the selected objects from the list</td>
</tr>
<tr>
<td>“Move Up”</td>
<td>Moves the selected object upwards by one entry.</td>
</tr>
<tr>
<td>“Move Down”</td>
<td>Moves the selected object downwards by one entry.</td>
</tr>
</tbody>
</table>

Dialog 'PDO Properties'

| “COB ID” | Every PDO message must have a COB ID (Communication Object Identifier). You can input explicit values (example: 16#201) or formulas (example: $NODEID+16#200). |
| “RTR” | Remote Transmission Request. This option is available for transmit PDOs only. You can use an RTR frame for interrogating the PDO externally. |
| “Inhibit time (x 100µs)” | You can edit this field only if the device supports this functionality. The inhibit time is the minimum time between two messages of a specific PDO. You can use this setting for preventing PDOs from being sent too often when their values are edited. Default: “0”. Possible values: 0...65535. |
| “Transmission type” | - “acyclic - synchronous”: When a change is made, the PDO is transmitted synchronously, but not periodically. (default) - “cyclic - synchronous”: The PDO is transmitted every nth sync. - “synchronous – RTR only”: Available for transmit PDOs only. After a synchronization message, the PDO is updated, but not transmitted. Transmission is by explicit request only (Remote Transmission Request). - “asynchronous – RTR only”: Available for transmit PDOs only. The PDO is updated and transmitted by explicit request only (Remote Transmission Request). - “asynchronous – manufacturer-specific”: The PDO is transmitted after specific events. - “asynchronous – device profile”: The PDO is transmitted according to the CiA device profile. |
"Number of syncs"  
For transmission type “cyclic - synchronous” only.  
Indicate the interval for transmitting the PDOs. The value is a multiple of the "Cycle period (µs)" of the CANopen manager. Default: 1. Possible values: 1...240.  
Example: Number of syncs = 4; cycle period = 1000 µs → transmission interval = 4000 µs.

"Event time (x 1ms)"  
Only for transmission types “asynchronous - manufacturer-specific” and “asynchronous - device profile”.  
You can edit this field only if the device supports this functionality. Indicate the time frame that should be between two PDO transmissions PDOs (in milliseconds). Default: "0". Possible values: 0...65535.

"Processing by CANopen Manager"  
☑: Default.  
☐: The CANopen manager does not process the PDO any longer. It is no longer transmitted or received.

Dialog 'Select Item from Object Directory'  
For modular slave, you must deactivate the “Autoconfig PDO mapping” option to be able to configure the mapping manually.

The table shows all object directory entries from the EDS file of the device. For receive PDOs, CODESYS provides only the objects here with write permission (flag = w); for transmit PDOs, read permission.

<table>
<thead>
<tr>
<th>&quot;Name&quot;</th>
<th>COB ID of the PDO or the name of the mapped object as it is used in the device description and in the object directory.</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Index&quot;</td>
<td>Index of the object</td>
</tr>
<tr>
<td>&quot;Subindex&quot;</td>
<td>Subindex of the object</td>
</tr>
<tr>
<td>&quot;Access type&quot;</td>
<td></td>
</tr>
</tbody>
</table>
  ● “RW”: read-write  
  ● “RO”: read only  
  ● “WO”: write only  
  ● “RWW”: Read/write per SDO; write permission per PDO (==> RPDO, output from the master viewpoint, input from the slave viewpoint).  
  ● “RWR”: Read/write per SDO; read permission per PDO (==> TPDO, input from the master viewpoint, output from the slave viewpoint).  
  ● CONST=constant |
| "Data type"       | Data type of the object                                                                                     |
| "Default value"   | Default value of the object                                                                                  |
| "Bit length"      | Length of the object                                                                                        |

Tab 'CANopen Remote Device - SDOs'  
When initializing the CANbus, CODESYS transmits the current configuration settings by using SDOs (Service Data Objects).

In this tab, you configure the necessary SDOs and determine the transmission order of the objects and the actions taken in case of a transmission error.

The object order in this list corresponds to the transmission order of SDOs to the module.

NOTICE!  
If the “Expert settings” option is not activated for the current device, then only the user-defined SDOs are shown here.
“Add SDO”  Opens the “Select Item from Object Directory” dialog box where all available SDOs are displayed. The chosen object is inserted after the selected object.

“Edit”  Opens the “Select Item from Object Directory” dialog box and highlights the respective object. You can modify the object parameters or replace the object with another one.

“Delete”  Deletes the selected objects from the list.

“Move Up”  Moves the selected object upwards by one entry.

“Move Down”  Moves the selected object downwards by one entry.

“Abort If Error”  If an error is detected for this SDO, then the stack stops the configuration phase of the current slave. The slave remains in PREOPERATIONAL mode.

“Jump to Line If Error”  The transmission is continued with the SDO that you indicated in the “Next Line” column.

“Next Line”  Line number where processing continues if there an error is detected

“SDO Timeout (ms)”  Timeout for the SDO transmission. If the slave does not respond to the SDO request within this time, then the transmission is canceled with a timeout.

“Create all SDOs”  Creates an SDO for all writeable objects starting at index 16#2000 for which a default value is given in the EDS. Only experts should use this option. It should be deactivated for standard use.

“Write complete PDO configuration”  This option forces the writing of all PDO configuration objects. In this way, you make sure that the settings in the project correspond to those of the slave. PDOs are not deactivated explicitly. The requirement is that the “Default settings” option is activated in the common settings of the slave and the PDOs are also deactivated in the EDS. If the default values in the EDS do not match the default settings of the slave firmware, then this procedure may cause problems. In this case, you should activate this option.

Dialog ‘Select Item from Object Directory’

This table shows all object directory entries from the device EDS file for each SDO that are writeable and not larger than 4 bytes. Before you add an SDO for selection in the SDO dialog, you can modify its parameters in the fields below the table. In this way, you can also created an SDO that is not writeable in the EDS file by entering a new index/subindex value.

| “Name” | COB ID of the PDO or the name of the mapped object as it is used in the device description and in the object directory. |
| “Index” | Index of the object |
| “Subindex” | Subindex of the object |
| “Access type” | “RW”: Read and write  “WO”: Write only  “RWW”: Read/write per SDO; write permission per PDO (==> RPDO, output from the master viewpoint, input from the slave viewpoint).  “RWR”: Read/write per SDO; read permission per PDO (==> TPDO, input from the master viewpoint, output from the slave viewpoint). |
| “Data type” | Data type of the object |
| “Default value” | Default value of the object |
| “Bit length” | Length of the object |
| “Value” | Value of the object |
CANopen module

CANopen modules are components that you insert below a CANopen remote device.

J1939

J1939 is a CAN-based protocol (CAN stands for "Controller Area Network"). It was developed for serial data transmission between electronic control units (ECU) in heavy goods vehicles. The CODESYS plug-in 'DeviceEditorJ1939' provides dialogs for configuring J1939 devices according to SAE J1939 standards.

J1939 - Bus cycle task

(1) Receive single package PG             (4) Receive multi-packages
P GS, Transmit PGs

See also

- Tab '<device name> I/O Mapping'® Chapter 1.3.1.25.1.19.11 “Tab '<device name> I/O Mapping’” on page 755

J1939 manager

You add the J1939 manager below the CANbus node in the device tree. It supports the CANbus J1939 configuration by means of internal functions. You insert the ECUs below the J1939 manager. You add the devices by means of the "Add Device” command.

The “Scan Devices” command is not available for J1939.
Tab 'J1939 Manager - General'

The J1939 manager provides a database that contains all global J1939 standard signals and parameter groups (PG).

The CODESYS installation includes a standard database with default settings. This database is write-protected to prevent losing this catalog due to unintended changes. You should make a writeable copy that you can extend or modify.

All J1939 databases are located in "C:\ProgramData\CODESYS\J1939 Databases".

This database is a SQLite database that can be opened and modified with an appropriate database tool.

Table 710

<table>
<thead>
<tr>
<th>&quot;Database&quot; : Name of the database (extension *.db)</th>
<th>Name of the database (extension *.db)</th>
</tr>
</thead>
<tbody>
<tr>
<td>J1939 ECU</td>
<td>Tab 'J1939 ECU - General'.................. 3341</td>
</tr>
<tr>
<td>1.5.6.2.12.2.3.2</td>
<td>Tab 'J1939 ECU - TX Signals'.................. 3343</td>
</tr>
<tr>
<td>1.5.6.2.12.2.3.3</td>
<td>Tab 'J1939-ECU - P2P RX Signals'.................. 3345</td>
</tr>
</tbody>
</table>

Tab 'J1939 ECU - General'

In this dialog box of the J1939 ECU editor, the general parameters of a J1939 ECU can be displayed and modified.
### Table 711: “General”

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Preferred address”</td>
<td>Address of the ECU. If more than one ECU with the same address is present in the network, then all ECUs that permit an address change (“Arbitrary address capable” activated), receive a new address.</td>
</tr>
<tr>
<td>“Local device”</td>
<td>CODESYS is also an ECU in the J1939 network. For local devices, an additional “RX Signals (P2P)” dialog is provided for configuring received signals. In CODESYS, you can configure any number of local devices.</td>
</tr>
</tbody>
</table>

### Table 712: “ECU NAME”

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>“NAME (64 bit): 16#”</td>
<td>Hexadecimal 64-bit code that contains complete information about the parameter settings. Each time this code is modified, the respective parameter is also modified. The same is true for the other direction.</td>
</tr>
<tr>
<td>“Arbitrary address capable”</td>
<td>If an address conflict occurs, then an address change can be performed for the ECU.</td>
</tr>
<tr>
<td>“Industry group”</td>
<td>List of industry groups according to the definition from SAE J1939.</td>
</tr>
<tr>
<td>“Vehicle system instance”</td>
<td>The parameter depends on the “Vehicle system”. The 4-bit value assigns a number to each instance of the “Vehicle system”.</td>
</tr>
<tr>
<td>“Vehicle system”</td>
<td>The value is defined in the SAE J1939 standard.</td>
</tr>
<tr>
<td>“Reserved”</td>
<td>Always deactivated and reserved for future SAE definitions.</td>
</tr>
<tr>
<td>“Function”</td>
<td>The parameter is defined and assigned by SAE. The value range is 0...255, however not all values are assigned.</td>
</tr>
<tr>
<td></td>
<td>The interpretation of values that are greater than or equal to 127 depends on the “Industry” selection. For example, the value “133” means “Product Flow” in the “Agricultural and Forestry Equipment” industry. If “Construction Equipment” is selected for “Industry”, then the same value means “Land Leveling System Display”.</td>
</tr>
<tr>
<td></td>
<td>If the value is less than 128 (0 - 127), then there is no dependency to other parameters.</td>
</tr>
<tr>
<td>“Function instance”</td>
<td>The parameter is related to the “Function” field. A J1939 network can consist of several ECUs with the same “Function”. The 5-bit “Function instance” assigns a number to each instance of the “Function”, where 0 is assigned to the first instance.</td>
</tr>
<tr>
<td>“ECU instance”</td>
<td>A J1939 network can include several ECUs that have the same task. For example, a vehicle can have two identical ECUs, where one measures vehicle speed and the other measures the trailer speed.</td>
</tr>
<tr>
<td>“Manufacturer code”</td>
<td>The 11-bit manufacturer code is assigned by SAE and indicates the company that manufactured the ECU. This code is defined in the SAE J1939 document.</td>
</tr>
<tr>
<td>“Identity number”</td>
<td>The 21-bit identity number is assigned by the manufacturer and should be used for assuring unique names in a product line. The manufacturer can also add more information to the identity number, such as serial number and date of manufacture.</td>
</tr>
</tbody>
</table>

### Table 713: “Communication Watchdog”

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Enable communication watchdog”</td>
<td>The stack checks whether the ECU transmits data within the given “Watchdog time”. If this does not happen, then the device is classified as “not available” and highlighted in red in the device tree.</td>
</tr>
</tbody>
</table>
Tab 'J1939 ECU - TX Signals'

This dialog shows the parameter groups that you defined for transmitting them to all other ECUs (broadcast) or to a specific ECU (P2P). In this dialog box, you can activate and deactivate individual groups and modify their parameters. You can also add new groups or signals to the list.

| “Enable” | ☑: The parameter group is transmitted. ☐: The parameter group is not transmitted for a local device. For a remote device, the PLC does not process this group. |
| “Type” | • “Broadcast”: The parameter group is transmitted to all ECUs. • “Peer-to-Peer (P2P)”: The parameter group is transmitted to a specific ECU. |
| “Add PG” | Opens the “Add Parameter Groups” dialog box. |
| “Add Signal” | Opens the “Add Signal” dialog box. The button is enabled only if you have selected a parameter group. |

Parameter group properties

Table 714: “General”

| “PGN” | The “parameter group number” is a unique number for addressing a parameter group. |
| “Name” | Name of the parameter group |
| “Description” | Description of the parameter group |
| “Length” | The length of the message data (0…1785 bytes). Due to the maximum data field length of 8 bytes, messages with over 8 bytes are transmitted as multipackages. |

Table 715: “Transmission settings”

These settings are provided only for the TX parameter groups of local devices.

| “Priority” | Priority of the parameter group (0..7). Priority 0 is the highest and 7 is the lowest. |
| “Destination address” | The target address is needed for P2P parameter groups only. |
| “Transmission mode” | Determines the time when a parameter group is transmitted (for local devices). • “Mode change”: The PG is transmitted when the value of the signal changes. • “Cyclic” The PG is transmitted after a specified number of PLC cycles (see cycle time factor). • “On request”: The PG is transmitted on request of another device. • “Application-controlled”: The PG is transmitted when triggered by the application. |
| “Cycle time factor” | Number of PLC cycles after which the parameter group is transmitted. Only applies for cyclic transmission. |

Signal parameters

Table 716: “General”

| “SPN” | (Suspect Parameter Number) One of the numbers assigned by the SAE for a specific parameter in a parameter group. |
| “Name” | Name of the parameter |
| “Description” | Description of the parameter |
**Table 717: “Conversion”**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Conversion”</td>
<td>TRUE: The value is calculated with scaling and offset.</td>
</tr>
<tr>
<td>“RAW data type”</td>
<td>Format of the raw data: Unsigned / Signed / Float / Double.</td>
</tr>
<tr>
<td>“Byte order”</td>
<td>Little endian or big endian of the raw signal.</td>
</tr>
<tr>
<td>“Scaling”</td>
<td>The factor (for “Conversion”=TRUE).</td>
</tr>
<tr>
<td>“Offset”</td>
<td>The offset (for “Conversion”=TRUE).</td>
</tr>
<tr>
<td>“Minimum value”</td>
<td>Expected minimum value of the converted signal (for informational purposes</td>
</tr>
<tr>
<td></td>
<td>only).</td>
</tr>
<tr>
<td>“Maximum value”</td>
<td>Expected maximum value of the converted signal (for informational purposes</td>
</tr>
<tr>
<td></td>
<td>only).</td>
</tr>
<tr>
<td>“Unit”</td>
<td>Unit of the converted signal.</td>
</tr>
<tr>
<td>“ICE data type”</td>
<td>Resulting data type of the I/O channels.</td>
</tr>
</tbody>
</table>

**Dialog 'Add Parameter Group’**

**Table 718: “Database”**

CODESYS shows all parameters of the database in this tab.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Search”</td>
<td>Character string for filtering the PGN list.</td>
</tr>
<tr>
<td>“Reset”</td>
<td>Reset the filter for showing the full list.</td>
</tr>
<tr>
<td>“Filter”</td>
<td>Only PGs of the selected type are shown.</td>
</tr>
<tr>
<td>“Add PG”</td>
<td>Adds the selected parameter group to the TX signal list. Multiple selection</td>
</tr>
<tr>
<td></td>
<td>of parameter groups is possible.</td>
</tr>
</tbody>
</table>

**Table 719: “Custom”**

In this tab, you can created user-defined parameter groups.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
</table>
| “PGN (18 bit)”                     | Unique number of the PG (0...262143) that is calculated from the four sub-
|                                    | sequent parameters. You can type in this value directly into the field.    |
| “PDU specific (bit 0-7)”           | Evaluation of the value depends on the value in “PDU format”.               |
|                                    | ● If “PDU format” is between 0 and 239, then the message is addressable and |
|                                    | the field contains the target address.                                     |
|                                    | ● If “PDU format” is between 240 and 255, then the message is a broadcast   |
|                                    | message and the field contains the group expansion.                         |
| “PDU format (bit 8-15)”            | Protocol Data Unit                                                          |
|                                    | The format determines whether the message can be transmitted with a target  |
|                                    | address or always as a broadcast message.                                   |
| “Data page (bit 16)”               | Selection of the page for the PDU format (currently always 0).              |
| “Expanded data page (bit 17)”      | Not used: for future expansions.                                            |
| “Name”                             | Name of the new parameter group.                                            |
| “Length (bytes)”                   | Length of the parameter group (in bytes)                                   |
Table 721: “Custom”

<table>
<thead>
<tr>
<th><strong>Name</strong></th>
<th>Name of the new signal.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Length (Bits)</strong></td>
<td>Length of the signal.</td>
</tr>
<tr>
<td><strong>Description</strong></td>
<td>Description of the new signal.</td>
</tr>
<tr>
<td><strong>Conversion</strong></td>
<td>The value is scaled.</td>
</tr>
<tr>
<td><strong>Unit</strong></td>
<td>Unit of the signal value (for informational purposes only).</td>
</tr>
<tr>
<td><strong>RAW data type</strong></td>
<td>Unsigned / Signed / Float / Double.</td>
</tr>
<tr>
<td><strong>Byte sequence</strong></td>
<td>Little endian or big endian.</td>
</tr>
<tr>
<td><strong>Scaling</strong></td>
<td>Factor is “Conversion” is activated.</td>
</tr>
<tr>
<td><strong>Offset</strong></td>
<td>Offset if “Conversion” is activated.</td>
</tr>
</tbody>
</table>

Tab 'J1939-ECU - P2P RX Signals'

This dialog is available for local ECUs only. It shows all PGs (parameter groups) that should be received by other ECUs. In this dialog, individual groups can be activated or deactivated, and their parameters can be modified. In addition, new groups or signals can be added to the list.

The commands and parameters of this dialog are the same as those in the “TX Signals” tab.

See also

- Chapter 1.5.6.2.12.2.3.2 “Tab 'J1939 ECU - TX Signals’” on page 3343

Command 'Scan for Devices'

**Function:** This command establishes a brief connection to the hardware and determines the devices in the network. You can then adopt the devices found into the device tree of your project.

**Call:** Main menu “Project”, context menu of a device object in the device tree.

**Requirement:** The settings for communication with the control device are correct. The gateway and the PLC are running. The device supports the scan function.

To use the scan functionality, you must log in once in order to load the library to the controller.

The command refers to the master controller selected in the device tree. For example, an already inserted PROFINET IO controller can be selected and the command used to determine the I/O devices and I/O modules assigned to it.

After scanning, the dialog box “Scan for Devices” appears with the results.
Dialog 'Scan Devices'

Table 722: “Scanned Devices”

<table>
<thead>
<tr>
<th>Description</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Device name, Device type, Address, Station name, etc.”</td>
<td>Data about the scanned device depending on network type.</td>
</tr>
<tr>
<td></td>
<td>When you change a value in the list of scanned devices, the value is shown</td>
</tr>
<tr>
<td></td>
<td>in italics. This indicates that the new value has been changed in the editor</td>
</tr>
<tr>
<td></td>
<td>in CODESYS, but not in the device. When you download the value to the device,</td>
</tr>
<tr>
<td></td>
<td>it is shown normally.</td>
</tr>
<tr>
<td></td>
<td>Value that indicate differences between the project and the scanned device</td>
</tr>
<tr>
<td></td>
<td>are shown in orange.</td>
</tr>
<tr>
<td></td>
<td>If several device descriptions are available for the scanned device, then</td>
</tr>
<tr>
<td></td>
<td>the name is displayed in bold. The selection of the matching device description</td>
</tr>
<tr>
<td></td>
<td>is resolved differently for different fieldbuses. Details can be found in</td>
</tr>
<tr>
<td></td>
<td>the corresponding fieldbus sections.</td>
</tr>
<tr>
<td></td>
<td>If a device description cannot be found, then the following message is shown:</td>
</tr>
<tr>
<td></td>
<td>“Attention! The device was not found in the repository.” Depending on the</td>
</tr>
<tr>
<td></td>
<td>bus system, additional information is displayed, such as manufacturer number</td>
</tr>
<tr>
<td></td>
<td>and product number. The device cannot be inserted into the project without</td>
</tr>
<tr>
<td></td>
<td>the installed device description.</td>
</tr>
</tbody>
</table>

| “Show differences to project”                                               | The table in the dialog also shows additional configured devices (in the    |
|                                                                             | device tree of the project).                                               |
|                                                                             | The table shows all scanned devices. The configured devices are not shown.  |

| “Scan Devices”                                                              | Starts a new search.                                                       |
| “Copy all devices to project”                                               | The device that is selected in the table is inserted into the device tree in |
|                                                                             | the project. If nothing is selected, then all scanned devices are shown.    |

NOTICE!
If you insert devices, which are available in the device tree, to the device tree with “Copy All Devices to Project”, then the following should be noted. The data of the “Process Data” and “<...> I/O Mapping” tabs of the existing devices can be overwritten with the data of the recently inserted devices.

Table 723: “Configured Devices”

This part of the dialog is visible only when you select the “Show differences to project” option.

Differences between the scanned and configured devices are color-coded. Devices in green are identical on both sides. Device in red are available only in the view of the scanned or configured devices.

- ![Up Arrow] If you have selected a device in both views, then the scanned devices are inserted above the selected configured device.
- ![Down Arrow] If you have selected a device in both views, then the scanned devices are inserted below the selected configured device.
- ![Refresh] If you have selected a device in both views, then the configured devices are replaced by the selected scanned device.
- ![Copy Icon] All scanned devices are copied to the project.
- ![Delete Icon] Deletes the selected configure device.
Tab 'CANbus - General'

Table 724: "General"

<table>
<thead>
<tr>
<th>“Network”</th>
<th>Number of the CAN network to be linked via the CANbus interface. Permitted values: 0 to 100.</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Baud rate”</td>
<td>Baud rate (in bits per second) for transmitting data on the bus. The default value is used from the device description file (*.devdesc) of the CANbus device. You can select the baud rate from the drop-down list or type it directly into the input field.</td>
</tr>
</tbody>
</table>

1.5.6.2.13 EtherCAT configurator

Refer to the general description for information about the following tabs of the device editor.

- Tab '<device name> I/O Mapping' \(\%\) Chapter 1.3.1.25.1.19.11 "Tab '<device name> I/O Mapping'" on page 755
- Tab '<device name> IEC Objects' \(\%\) Chapter 1.3.1.25.1.19.12 "Tab '<device name> IEC Objects'" on page 759
- Tab '<device name> Parameters' \(\%\) Chapter 1.3.1.25.1.19.3 "Tab 'Parameters'" on page 746
- Tab '<device name> Status' \(\%\) Chapter 1.3.1.25.1.19.17 "Tab 'Status'" on page 768
- Tab '<device name> Information' \(\%\) Chapter 1.3.1.25.1.19.18 "Tab 'Information'" on page 768

Only in the case of special features is there an additional help page for the specific device editor.

If the "<device name> Parameters" tab is not shown, then select the "Show generic device configuration editors" option in the CODESYS options ("Device Editor" category).

The configuration of EtherCAT modules is based on the device description files for the master and slave devices employed and can be adapted in the project in configuration dialogs. In order to ensure the simplest and most error-free use possible, we recommend for standard applications that you activate the option for the "Automatic Configuration" of the master, so that the majority of the configuration settings are performed automatically.

Requirements

The requirement for the combination of EtherCAT devices with a CODESYS Control Win V3 is the installation of the program library WinPCap (freely downloadable, e.g. from winpcap.org). Furthermore, add the following entries to the CODESYS configuration file (\..\GatewayPLC\CODESYS\CODESYSSP.cfg):

- component.<subsequent number>=CmpEt100Drv
  Required only with the RTE. The RTE requires special network drivers. Available for Realtek RTL81x9/RTL8169, Intel Pro 100 / 1000
- component.<subsequent number>=CmpRTL81x9Mpd
  Required only with the RTE. Available for RTL8139.
- component.<subsequent number>=8169Mpd
  Required only with the RTE. Available for Realtek RTL8169 or RTL8168 (PCIe version))
The bus cycle task is set in the general PLC settings.

Access to the EtherCAT configuration by the application takes place via instances of the EtherCAT master and EtherCAT slave. If the EtherCAT master or EtherCAT slaves are inserted as objects into a project, instances are automatically created for master and slaves that can be addressed in the application program. For example a restart, a stop or a status check of the EtherCAT device can be performed from the application.

Furthermore, the EtherCAT library offers function blocks for the reading and writing of individual parameters, even during bus operation.

See also
- Chapter 1.5.6.2.13.1.1 “Tab 'EtherCAT Master - General'” on page 3348
- Device Editor Options Chapter 1.3.1.25.3.13.9 “Dialog 'Options' - 'Device Editor’” on page 1081
- PLC Settings Chapter 1.3.1.25.1.19.9 “Tab 'PLC Settings'” on page 751

EtherCAT master

1.5.6.2.13.1.1 Tab 'EtherCAT Master - General'............................................. 3348
1.5.6.2.13.1.2 Tab 'EtherCAT Master - Sync Unit Assignment' ..................... 3349
1.5.6.2.13.1.3 Tab 'EtherCAT Master - Parameter'......................................... 3350

Tab 'EtherCAT Master - General'

Object: “EtherCAT Master”

This tab is for the configuration of the basic settings for the EtherCAT master. The preset basic settings originate from the device description file.

Settings of the configuration parameters

| “Autoconfig Master/Slaves” | The greater part of the master and slave configuration is accomplished automatically, based on the device description file and implicit calculations. The dialog for the FMMU/Sync settings is not available. Even if this option of the master is activated, an expert mode can be activated explicitly for each individual slave that permits the manual editing of the automatically generated process data configuration. |

**NOTICE!**
The auto-configuration mode ("Autoconfig Master/Slaves" option) is activated by default and is adequate for standard applications. If the mode is not activated, all configuration settings for master and slave(s) must be made manually, for which expert knowledge is required! The auto-configuration mode option must be switched off for the configuration of slave-to-slave communication.
### Table 725: “EtherCAT NIC setting”

| “Destination address” | MAC address of the device in the EtherCAT network that is to receive the telegrams. Options:  
| “Broadcast:” | no “destination address (MAC)” need be specified. |

| “Source address” | MAC address of the controller (target system) or network name (name of the card, i.e. PLC (target system)) |

| “Network name” | Name or MAC of the network, depending on which of the following options is activated:  
| “Select network by MAC” | □: Network is specified by the MAC-ID. The project then cannot be used on another device, since each network adaptor has a unique MAC-ID.  
| “Select network by Name” | □: Network is identified by the network name and the project is device-independent.  
| “Browse” | Scans the network for the MAC-IDs or names of the target devices that are presently available. |

### Table 726: “Distributed Clocks”

| “Cycle time (μs)” | Time interval after which a new data telegram is dispatched on the bus. If the “Distributed clocks” function is activated in the slave, the master cycle time specified here is transferred to the slave clocks. In this way a precise synchronization of the data exchange can be achieved, which is important in particular if spatially distributed processes require simultaneous actions. Simultaneous actions are, for example, applications in which several axes must execute coordinated movements at the same time. A very precise, network-wide timebase with a jitter of substantially less than 1 microsecond can be achieved in this way. |

### Table 727: “Options”

| “Use LRW instead of LWR/LRD” | □: Direct communication from slave to slave is possible. In place of separate read (LRD) and write commands (LWR), combined read/write commands (LRW) are used. |

| “Send/Receive per task” | □: Read and write commands, i.e. the handling of the input and output messages, can be controlled with various tasks. |

| “Automatically restart slaves” | □: In the event of a communication breakdown the master immediately attempts to restart the slaves. |

### Table 728: “Master setting”

These settings can be edited only if the “Automatic Configuration” option is deactivated, otherwise this takes place automatically and they are not visible here.

| “Image in address” | First logical address of the first slave for input data |

| “Image out address” | First logical address of the first slave for output data |

- ☞ Chapter 1.5.6.2.13.2.1 “Tab ‘EtherCAT Slave - General’” on page 3350

### Tab 'EtherCAT Master - Sync Unit Assignment'

Object: “EtherCAT Master”

This tab shows all slaves that are inserted below a particular master with an assignment to the sync units.
With the EtherCAT sync units, several slaves are configured into groups and subdivided into smaller units. For each group, the working counter can be monitored for an improved and more precise error detection. As soon as a slave is missing in a sync unit group, the other slaves in the group are also shown as missing. Detection occurs immediately in the next bus cycle, as the working counter is continuously checked. With the device diagnostics, the missing group can be remedied as quickly as possible.

Unaffected groups remain operable without any interference.

Sync unit support is defined by the device description of the EtherCAT master and can be deactivated for OEM-specific device descriptions. By default, it is provided with a device description of version 3.5.8.0 and later.

<table>
<thead>
<tr>
<th>“Device Name”</th>
<th>Name of the slave</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Sync Unit”</td>
<td>Name of the selected sync unit. You can combine individual devices or whole groups (multiple selection) into one sync unit group.</td>
</tr>
<tr>
<td>“Add”</td>
<td>When you type a name in the text field, you can create a new sync unit.</td>
</tr>
<tr>
<td>“Delete”</td>
<td>Removes the selected sync unit. If slaves are assigned to the group to be deleted, then a warning is displayed. If you confirm the dialog prompt by clicking “Yes”, then these devices are reassigned to the default group.</td>
</tr>
</tbody>
</table>

Tab 'EtherCAT Master - Parameter'

Object: "EtherCAT master"

This tab contains the master parameters that are defined in the device description file.

If the auto-configuration mode is activated in the "Master" dialog, then the parameters are set automatically here in accordance with the specifications from the device description file and the network topology. Nothing should be changed in the generic editor, since an invalid configuration can be set here.

| “Value” | Editable, change is only effective if auto-configuration mode is deactivated. Whether or not the change becomes effective depends on the respective parameter. |

EtherCAT slave

1.5.6.2.13.2.1  Tab 'EtherCAT Slave - General' ......................................................... 3350
1.5.6.2.13.2.2  Tab 'EtherCAT Slave - FMMU/Sync' ............................................. 3353
1.5.6.2.13.2.3  Tab 'EtherCAT Slave - Expert Process Data’ .................................. 3354
1.5.6.2.13.2.4  Tab 'EtherCAT-Slave - Process data’ ........................................ 3355
1.5.6.2.13.2.5  Tab 'EtherCAT Slave - Startup Parameters’ ................................. 3356
1.5.6.2.13.2.6  Tab 'EtherCAT Slave - Parameters’ ........................................... 3357
1.5.6.2.13.2.7  Tab 'EtherCAT Slave - EoE Settings’ ........................................ 3358

Tab 'EtherCAT Slave - General’

Object: “EtherCAT Slave”

The basic settings for the EtherCAT slave are configured in this tab. The basic settings are preset from the device description file.
Table 729: “Address”

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>“AutoInc address”</td>
<td>Self-incrementing address (16-bit) that results from the position of the slave in the network. The address is used only during the system boot when the master assigns the EtherCAT addresses to its slaves. When the first message runs through the slaves for this purpose, each slave increments its “AutoInc address” by 1. The slave with address 0 receives the data. A possible input here is “-8”.</td>
</tr>
<tr>
<td>“EtherCAT address”</td>
<td>Final address of the slaves, assigned by the master during bootup. The address is independent of the position of the slave in the network.</td>
</tr>
</tbody>
</table>

Table 730: “Additional”

<table>
<thead>
<tr>
<th>Setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Expert settings”</td>
<td>Additional settings are possible for the start-up check and time monitoring (see below). The “Expert Process Data” tab is also available in the device editor. However, expert settings are not required for standard applications. The auto-configuration mode is recommended and sufficient for standard applications.</td>
</tr>
</tbody>
</table>

Table 731: “Distributed Clocks”

<table>
<thead>
<tr>
<th>Setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Select distributed clocks”</td>
<td>Drop-down list with all settings for distributed clocks in the device description file</td>
</tr>
<tr>
<td>“Activate”</td>
<td>Cycle time for the data exchange, displayed in the “Sync unit cycle (µs)” input field, is determined by the cycle time of the master. The master clock can thus synchronize the data exchange in the network.</td>
</tr>
</tbody>
</table>

The “Sync0” and “Sync1” settings described below are slave-dependent:

Table 732: “Sync0”

<table>
<thead>
<tr>
<th>Setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Activate sync 0”</td>
<td>Synchronization unit “Sync0” is used. A synchronization unit describes a set of process data that is exchanged synchronously.</td>
</tr>
<tr>
<td>“Sync unit cycle”</td>
<td>The master cycle time (multiplied by the factor selected from the drop-down list) is used as the synchronization cycle time for the slave. “Cycle time (µs)” displays the cycle time currently set.</td>
</tr>
<tr>
<td>“User defined”</td>
<td>A user-defined cycle time (in microseconds) can be specified in the “Cycle time (µs)” field.</td>
</tr>
</tbody>
</table>

Table 733: “Sync1”

<table>
<thead>
<tr>
<th>Setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Activate sync 1”</td>
<td>Synchronization unit “Sync1” is used. A synchronization unit describes a set of process data that is exchanged synchronously.</td>
</tr>
<tr>
<td>“Sync unit cycle”</td>
<td>The master cycle time (multiplied by the factor selected from the drop-down list) is used as the synchronization cycle time for the slave. The “Cycle time (µs)” field displays the cycle time currently set.</td>
</tr>
<tr>
<td>“User defined”</td>
<td>A user-defined cycle time (in microseconds) can be specified in the “Cycle time (µs)” field.</td>
</tr>
</tbody>
</table>

Table 734: “Startup Checking”

<table>
<thead>
<tr>
<th>Setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Check vendor ID”</td>
<td>By default the vendor ID and product ID of the device are checked against the current configuration settings when the system boots up. If they do not agree, then the bus is stopped and no further actions are executed. This is done to prevent an incorrect configuration from being loaded onto the bus system. Options to deactivate the corresponding check.</td>
</tr>
<tr>
<td><strong>Check revision number</strong></td>
<td>☑: The revision number is checked during the system bootup according to your selection in the drop-down list.</td>
</tr>
<tr>
<td><strong>Download of the expected module configuration</strong></td>
<td>☑: For the online check of the configured and actual module configuration. If the configurations conflict with each other, then the device still switches to “Run” and an entry is recorded in the device log.</td>
</tr>
</tbody>
</table>

**Table 735: “Timeouts”**

By default, time monitoring is not defined for the following actions. If necessary, an appropriate “timeout” time can be specified here (in milliseconds):

| **SDO access** | Transmits the SDO list at system start. Specified in milliseconds. |
| **I -> P** | Switch from “Init” mode to “Preoperational” mode. Specified in milliseconds. |
| **P -> S / S -> O** | Switch from “Preoperational” mode to “Safe Operational” mode or from “Safe Operational” mode to “Operational” mode. Specified in milliseconds. |

**Table 736: “DC Cyclic Unit Control: Assign to Local \( \mu \)C”**

One or more options for the “Distributed clocks” function can be activated here that should be used on the local microprocessor. The check is performed in the registry at 0x980 in the EtherCAT slave. Possible settings:

| **Cycle unit** |
| **Latch unit 0** |
| **Latch unit 1** |

**Table 737: “Watchdog”**

| **Set multiplier** | Watchdogs PDI and SM receive their signals from the local terminal clock, divided by the watchdog multiplier |
| **Set PDI watchdog** | This watchdog triggers if there is no PDI communication with the EtherCAT slave controller for longer than the PDI (Process Data Interface) watchdog time that has been set and activated. |
| **Set SM watchdog** | This watchdog triggers if there is no EtherCAT process data communication with the terminal for longer than the SM (SyncManager) watchdog time that has been set and activated |

**Table 738: “Identification”**

In this section, you set the device identification of the slave. In this way, you can make the address of the slave independent of its position in the bus.

The following options are visible only if the “Enable expert settings” option or “Optional” option is activated.

| **Deactivated** | The identification of the slave is not checked. |
| **Configured station alias (ADO 0x0012)** | Address that is stored in the EEPROM of the device. You can change this value in online mode. |
| **Write EEPROM** | Visible in online mode only for “Configured station alias”. Writes the defined address for “Value” to the EEPROM of the slave. |
| **Explicit device identification (ADO 0x0134)** | The device identification is hard set on the hardware (for example, by DIP switches). It is displayed in “Actual address”. |
| **Data word (2 bytes)** | A 2-byte value for the identification is saved in the slave. |
| **Value** | Expected value for the check. If the actual value does not match this setting, then an error is issued. |
**Tab 'EtherCAT Slave - FMMU/Sync'**

Object: "EtherCAT Slave"

The tab shows the FMMUs and Sync Manager of the EtherCAT slave as they are defined in the device description file. There is a possibility to edit the FMMUs and Sync Manager, e.g. for the configuration of slave-to-slave communication.

Requirement: The auto-configuration mode in the EtherCAT master is deactivated.

Table 739: "FMMU"

Table shows the Fieldbus Memory Management Units of the slave, which are used for handling the process data. In each case the allocation of the logical address ("GlobalStartAdr") to a physical address ("Ph. Start Address") is defined. Mapping bit by bit is possible.

<table>
<thead>
<tr>
<th>&quot;Edit&quot;</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Add&quot;</td>
</tr>
<tr>
<td>&quot;Delete&quot;</td>
</tr>
</tbody>
</table>

Table 740: "Edit FMMU"

| "Global Start Address" |
| "Length/Byte" |
| "Start Bit" |
| "End Bit" |
| "Phys. Start Address" |
| "Phys. Start Bit" |
| "Access" |
| "Read" |
| "Write" |
| "Flags" |
| "Activate" |

Table 741: "Sync Manager"

Display and editing of the synchronization manager of the slave. Amongst other things the physical start address, the type of access, the buffer and the physical address to which the interrupts are to be sent are defined for each available Sync Manager type (mailbox in, mailbox out, inputs, outputs).
Table 742: “Edit Syncman”

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>“Phys. start address”</td>
<td></td>
</tr>
<tr>
<td>“Length”</td>
<td></td>
</tr>
<tr>
<td>“Buffer”</td>
<td>“1”</td>
</tr>
<tr>
<td></td>
<td>“3”</td>
</tr>
<tr>
<td>“Access”</td>
<td>“Read”</td>
</tr>
<tr>
<td></td>
<td>“Write”</td>
</tr>
<tr>
<td>“Interrupts”</td>
<td>“to EtherCAT”</td>
</tr>
<tr>
<td></td>
<td>“to PDI”</td>
</tr>
<tr>
<td>“Flag control”</td>
<td>“Activate”</td>
</tr>
<tr>
<td>“Watchdog”</td>
<td>“Trigger”</td>
</tr>
<tr>
<td>“SyncMan type”</td>
<td>“”</td>
</tr>
</tbody>
</table>

See also
- ☞ Chapter 1.5.6.2.13.2.1 “Tab 'EtherCAT Slave - General' ” on page 3350
- ☞ Chapter 1.5.6.2.13.1.1 “Tab 'EtherCAT Master - General'” on page 3348

Tab 'EtherCAT Slave - Expert Process Data'

Object: “EtherCAT Slave”

This tab provides a different and more detailed view of the process data that is also displayed in the “Process Data” dialog box. Downloads are also selected here for PDO assignment and PDO configuration.

Requirement: The check box for the expert settings of the slave is selected.

See also:
- ☞ Chapter 1.5.6.2.13.2.1 “Tab 'EtherCAT Slave - General' ” on page 3350

Table 743: “Sync Manager”

List of sync managers with data size and PDO type

Table 744: “PDO Assignment (16#1C12)”

List of PDOs that are assigned to the selected “Sync Manager”

If a check box is selected, then the PDOs are activated and I/O channels are created. This is similar to the simple PDO configuration window.

Table 745: “PDO List”

List of PDOs that are assigned to the selected “Sync Manager”

You can add new PDOs or edit or delete existing PDOs by executing the respective commands (“Add”, “Delete”, “Edit”) in the command bar or shortcut menu.

Table 746: “Edit PDO List”

<table>
<thead>
<tr>
<th>Name</th>
<th>Index</th>
</tr>
</thead>
</table>

See also
- ☞ Chapter 1.5.6.2.13.2.1 “Tab 'EtherCAT Slave - General' ” on page 3350
“Direction”
- “TxPDO (Input)”: The PDO is transmitted from the master to the slave.
- “RxPDO (Output)”: The PDO is transmitted from the slave to the master.

“Flags”
- “Mandatory”: The PDO is required and cannot be deactivated in the “PDO Assignment”.
- “Fixed Content”: The PDO contents are fixed and cannot be changed. Therefore, it is also not possible to add entries to the “PDO Content”.
- “Virtual PDO”: Reserved for future use.

“Exclude PDOs”
It is possible to define an exclusion list. When one PDO is active in the “PDO Assignment”, other PDOs are inactive and cannot be activated.

“Sync unit”
ID of the sync manager where the PDO should be assigned.

Table 747: “PDO Content”
Displays the contents of the PDOs that are selected in the “PDO List”. You can add new entries or edit or delete existing entries by executing the respective commands (“Add”, “Delete”, “Edit”) in the command bar or shortcut menu. You can change the PDO order by clicking “Move Up” and “Move Down”.

Table 748: “Download”

| “PDO assignment”              | Specific CoE commands for initializing the 0x1cxx objects are generated and written to the slave. |
| “PDO configuration”           | The CoE commands for 0x16xx or 0x1axx are generated, loading the PDO mapping to the slave. As a rule, the default values originate from the ESI file and the device must support this functionality. For example, if a device has a fixed configuration, then these commands are seen as flawed. |

See also
- Chapter 1.5.6.2.13.2.4 “Tab ‘EtherCAT-Slave - Process data’ ” on page 3355
- Chapter 1.5.6.2.13.2.1 “Tab ‘EtherCAT Slave - General’ ” on page 3350

Tab 'EtherCAT-Slave - Process data’
Object: “EtherCAT slave”
This tab of the EtherCAT configurator displays the process data for the inputs and outputs of the slave. The data originate from the device description file.

Table 749: “Select outputs”
The table shows the outputs of the slave defined by “Start address”, “Type”, and “Index”.
If outputs of the device are activated here (for writing), these outputs can be assigned to project variables in the “EtherCAT I/O Mapping” dialog.

Table 750: “Select inputs”
The table shows the inputs of the slave defined by “Name”, “Type”, and “Index”.
If inputs of the device are activated here (for reading), these inputs can be assigned to project variables in the “EtherCAT I/O Mapping” dialog.

See also
- Chapter 1.5.6.2.13.2.1 “Tab ‘EtherCAT Slave - General’ ” on page 3350
Tab 'EtherCAT Slave - Startup Parameters'

Object: "EtherCAT Slave"

The SDOs (Service Data Objects) or IDNs that transmit specified parameters to the device at the system start are defined for the current slave in this tab.

The object directory with the required data objects is described in the EtherCAT-ML description file or in an EDS file which is referenced in the XML file.

Requirement: The device supports CAN over EtherCAT or Servodrive over EtherCAT.

Some modules that are inserted below a slave have their own start parameters. These parameters are then also shown in this list but cannot be edited here. The parameters are changed in the editor of the corresponding module.

Table 751: SDO table

<table>
<thead>
<tr>
<th>Column</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Line”</td>
<td>Line number</td>
</tr>
<tr>
<td>“Idn”</td>
<td>Bit length of the SDO</td>
</tr>
<tr>
<td>“Abort on Error”</td>
<td>The transfer is interrupted in the case of an error with error status.</td>
</tr>
<tr>
<td>“Jump to Line on Error”</td>
<td>In case of error the transfer is resumed with the SDO at the specified “Line”.</td>
</tr>
<tr>
<td>“Next Line”</td>
<td>The transfer is resumed with the SDO at the next line.</td>
</tr>
<tr>
<td>“Comment”</td>
<td>Input field for comment</td>
</tr>
<tr>
<td>“Move Up”</td>
<td>Moves the selected line upwards by one line</td>
</tr>
<tr>
<td>“Move Down”</td>
<td>Moves the selected line downwards by one line</td>
</tr>
<tr>
<td>“Add”</td>
<td>Opens the dialog “Select an entry from the object directory”. In this dialog you can change the parameters of the SDO before the SDO is added to the configuration. By specifying new index/subindex entries, a new object can be added to the SDO that is not yet described in the EDS file. This is useful if only an incomplete object directory or none at all is present.</td>
</tr>
<tr>
<td>“Delete”</td>
<td>Removes the selected entry.</td>
</tr>
<tr>
<td>“Change”</td>
<td>Opens the dialog “Select an Entry from the Object Directory” in order to change the parameters of the selected SDOs or IDNs in the table</td>
</tr>
</tbody>
</table>

Servodrive over EtherCAT

Table 752: “Select an Entry from the Object Directory”

<table>
<thead>
<tr>
<th>Column “Idn”</th>
</tr>
</thead>
<tbody>
<tr>
<td>Basic value of the IDN, Editable, (open with double click)</td>
</tr>
<tr>
<td>Column “Base value”</td>
</tr>
</tbody>
</table>
By specifying new "PSet"/"Offset" entries, a new object can be added to the IDN that is not yet described in the XML file. This is useful if only an incomplete object directory or none at all is present.

By specifying new PSet/Offset entries, a new object can be added to the IDN that is not yet described in the XML file. This is useful if only an incomplete object directory or none at all is present.

"Offset"

"Bit length"

"Value"

"Channel"

"As list"

The length is calculated automatically.

---

### Table 753: “Select an Entry from the Object Directory”

<table>
<thead>
<tr>
<th>Column “Flags”</th>
<th>Display of access flags: RW (read/write), RO (read only), WO (write only)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Column “Base value”</td>
<td>Editable (open with double click)</td>
</tr>
</tbody>
</table>

**Input fields**

- **"IDN"**
  - "S"
  - "P"

- **"PSet"**
  - By specifying new "PSet"/"Offset" entries, a new object can be added to the IDN that is not yet described in the XML file. This is useful if only an incomplete object directory or none at all is present.

- **"Offset"**
  - By specifying new PSet/Offset entries, a new object can be added to the IDN that is not yet described in the XML file. This is useful if only an incomplete object directory or none at all is present.

- **"Bit length"**
  - Drop-down list for the selection of the bit length

- **"Value"**
  - Drop-down list for the selection of the value

- **"Channel"**
  - This drop-down list is automatically displayed if the object has several sub-objects.

- **"As list"**
  - Parameters are loaded as a list. The first four bytes indicate the length.

---

**See also**

- Chapter 1.5.6.2.13.2.1 “Tab ‘EtherCAT Slave - General’ ” on page 3350

---

### Tab 'EtherCAT Slave - Parameters'

Object: "EtherCAT slave"

This tab contains the slave parameters that are defined in the device description file.

If the auto-configuration mode of the master is activated, then the parameters are set automatically here in accordance with the specifications from the device description file and the network topology. For standard applications it is also normally not required to edit them.
"Value"  Only a few parameters are editable. Change is only effective if auto-configuration mode is deactivated.
In principle the user should not change anything here, since in doing so an invalid configuration could be created, thus preventing the slave from entering the operational state.

- Chapter 1.5.6.2.13.2.1 “Tab 'EtherCAT Slave - General' ” on page 3350
- Chapter 1.5.6.2.13.1.3 “Tab 'EtherCAT Master - Parameter'” on page 3350

Tab 'EtherCAT Slave - EoE Settings'

Object: “EtherCAT slave”

This Tab is for the configuration of the communication settings of the slaves, which support the Ethernet over EtherCAT (EoE).

Requirement:
- When using the CODESYS Control Win V3 the installation of the Microsoft Loopback-Adapter as virtual Ethernet adapter is required. A description which explains the installation of this adapter is available in the internet.

Table 754: “Settings”

<table>
<thead>
<tr>
<th>“Virtual Ethernet Port”</th>
<th>☑: Activates the EOE functionality of the slave. A “Virtual MAC Id” must be defined.</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Virtual MAC Id:”</td>
<td>Input field for the “Virtual MAC Id”</td>
</tr>
<tr>
<td>“Switch Port”</td>
<td>☑: Device serves a switch; no further network settings are required.</td>
</tr>
<tr>
<td>“IP Port”</td>
<td>☑: The device serves as a IP port - the “IP Settings” have to be defined.</td>
</tr>
</tbody>
</table>

Table 755: “IP Settings”

The Ethernet communication parameters must be set according to the parameters of the virtual Ethernet adapter

<table>
<thead>
<tr>
<th>“IP Address”</th>
<th>IP Address of the slave in the network, length: 4 Byte</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>The IP Port must be in the same range as the virtual Ethernet adapter. For example if address of network adapter is 192.168.1.1 and subnet mask is 255.255.255.0 then the IP port must be in the range of 192.168.1.2 to 192.168.1.254</td>
</tr>
<tr>
<td>“Subnet mask”</td>
<td>Subnet mask, length: 4 Byte</td>
</tr>
<tr>
<td>“Default Gateway”</td>
<td>Default Gateway, length: 4 Byte</td>
</tr>
<tr>
<td>“DNS Server”</td>
<td>IP Address of the DNS server</td>
</tr>
<tr>
<td>“DNS Name”</td>
<td>Name of the DNS Server</td>
</tr>
</tbody>
</table>

See also
- Chapter 1.5.6.2.13.2.1 “Tab 'EtherCAT Slave - General' ” on page 3350

EtherCAT module

1.5.6.2.13.3.1  Tab 'EtherCAT Module - Startup Parameters'  3358

Tab 'EtherCAT Module - Startup Parameters'

Object: “EtherCAT Module”
The SDOs (Service Data Objects) or IDNs that transmit specified parameters to the device at
the system start are defined for the current module in this tab.

The object directory with the required data objects is described in the EtherCAT-ML description
file or in an EDS file which is referenced in the XML file.

Requirement: The device supports CAN over EtherCAT or Servodrive over EtherCAT.

Some modules have their own start parameters which are displayed in the tab. The parameters can be modified there. Likewise, the parameters are also displayed in the slave, but they are blocked there.

Table 756: SDO table

<table>
<thead>
<tr>
<th>Column</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Line”</td>
<td>Line number</td>
</tr>
<tr>
<td>“Idn”</td>
<td></td>
</tr>
<tr>
<td>“Bit Length”</td>
<td>Bit length of the SDO</td>
</tr>
<tr>
<td>“Abort on error”</td>
<td>The transfer is interrupted in the case of an error with error status.</td>
</tr>
<tr>
<td>“Jump to Line on Error ”</td>
<td>In case of error the transfer is resumed with the SDO at the specified “Line”.</td>
</tr>
<tr>
<td>“Next Line “</td>
<td>The transfer is resumed with the SDO at the next line.</td>
</tr>
<tr>
<td>“Comment”</td>
<td>Input field for comment</td>
</tr>
<tr>
<td>“Move Up”</td>
<td>Moves the selected line upwards by one line</td>
</tr>
<tr>
<td>“Move Down”</td>
<td>Moves the selected line downwards by one line</td>
</tr>
<tr>
<td>“Add”</td>
<td>Opens the dialog “Select an entry from the object directory”. In this dialog you can change the parameters of the SDO before the SDO is added to the configuration. By specifying new index/subindex entries, a new object can be added to the SDO that is not yet described in the EDS file. This is useful if only an incomplete object directory or none at all is present.</td>
</tr>
<tr>
<td>“Delete”</td>
<td>Removes the selected entry.</td>
</tr>
<tr>
<td>“Change”</td>
<td>Opens the dialog “Select an entry from the object directory” in order to change the parameters of the selected SDOs or IDNs in the table</td>
</tr>
</tbody>
</table>

Servodrive over EtherCAT

Table 757: “Select an Entry from the Object Directory”

<table>
<thead>
<tr>
<th>Column</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Column “Idn”</td>
<td></td>
</tr>
<tr>
<td>Column “Base value”</td>
<td>Basic value of the IDN, Editable, (open with double click)</td>
</tr>
</tbody>
</table>

Input fields
"IDN"  ● “S”
     ● “P”

"PSet"  By specifying new “PSet/Offset” entries, a new object can be added to the IDN that is not yet described in the XML file. This is useful if only an incomplete object directory or none at all is present.

"Offset"  By specifying new PSet/Offset entries, a new object can be added to the IDN that is not yet described in the XML file. This is useful if only an incomplete object directory or none at all is present.

"Bit Length"  Drop-down list for the selection of the bit length

"Value"  Drop-down list for the selection of the value

"Channel"  This drop-down list is automatically displayed if the object has several sub-objects.

"As List"  Parameters are loaded as a list. The first four bytes indicate the length.  
   : The length is calculated automatically.

---

**“CAN over EtherCAT”**

Table 758: “Select an Entry from the Object Directory”

| Column “Flags” | Display of access flags: RW (read/write), RO (read only), WO (write only) |
| Column “Base value” | Editable (open with double click) |

| “Name” | Input field for displaying and changing the name |
| “Index: 16#” | By specifying new index/subindex entries, a new object can be added to the SDO that is not yet described in the EDS file. |
| “Subindex: 16#” | Value range of the object |
| “Bit length” | The size of each value can be maximum one byte (0-255). It can also be a hexadecimal in IEC syntax (for example, 16#ad). If the “Byte array” option is enabled, then the values have to be specified as a comma-separated list (for example, 1,2,3,4). |
| “Value” | The complete object is written with one access and all subindexes are set at the same time. The time needed for the transfer is reduced because not every sub-index has to be transferred individually. |
| “Full access” | Values can be specified as a comma-separated byte array. |

See also

- Chapter 1.5.6.2.13.2.1 “Tab ’EtherCAT Slave - General’ ” on page 3350

**Bus cycle task - EtherCAT**

**General information**  As a rule, for each IEC task the used input data is read at the start of each task (1) and the written output data is transferred to the I/O driver at the end of the task (3). The implementation in the I/O driver is decisive for further transfer of the I/O data. The implementation is therefore responsible for the timeframe and the specific time when the actual transmission occurs on the respective bus system.
The bus cycle task of the PLC can be defined globally for all fieldbuses in the PLC settings. For some fieldbuses, however, you can change this independent of the global setting. The task with the shortest cycle time is used as the bus cycle task (setting: "unspecified" in the PLC settings). In this task, the messages are normally transferred on the bus.

Other tasks copy only the I/O data from an internal buffer that is exchanged only with the physical hardware in the bus cycle task.

(1) Read inputs from input buffer       (2) IEC task
(3) Write outputs to output buffer      (4) Bus cycle
(5) Input buffer                        (6) Output buffer
(7) Copy data to/from bus              (8) Bus cycle task, priority 1, 1 ms
(9) Bus cycle task, priority 5         (10) Bus cycle task, priority 10, interrupted by task 5
(11) Bus cycle task, priority 10, interrupted by task 5

Using tasks
The "Task Deployment" provides an overview of used I/O channels, the set bus cycle task, and the usage of channels.

**WARNING!**
If an output is written in various tasks, then the status is undefined, as this can be overwritten in each case.

When the same inputs are used in various tasks, the input could change when a task is processed. This happens if the task is interrupted by a task with a higher priority and causes the process map to be read again. Solution: At the beginning of the IEC task, copy the input variables to variables and then work only with the local variables in the rest of the code.

Conclusion: Using the same inputs and outputs in several tasks does not make any sense and can lead to unexpected reactions in some cases.

**Behavior of the bus cycle with EtherCAT**
The pending network messages of the last cycle are read before the IEC inputs are copied.
If the “Send/Receive per Task” option is activated in the settings of the EtherCAT master, additional telegrams are transmitted to the devices employed per task and input or output employed. Channels that are used in a slow task are also transmitted less frequently. The bus load can thus be reduced.

See also

- Chapter 1.5.6.2.13.1.1 “Tab 'EtherCAT Master - General'” on page 3348
PROFINET IO configurator

Refer to the general description for information about the following tabs of the device editor.

- Tab '<device name> I/O Mapping' % Chapter 1.3.1.25.1.19.11 “Tab '<device name> I/O Mapping'” on page 755
- Tab '<device name> IEC Objects' % Chapter 1.3.1.25.1.19.12 “Tab '<device name> IEC Objects'” on page 759
- Tab '<device name> Parameters' % Chapter 1.3.1.25.1.19.3 “Tab 'Parameters'' on page 746
- Tab '<device name> Status' % Chapter 1.3.1.25.1.19.17 “Tab 'Status'' on page 768
- Tab '<device name> Information' % Chapter 1.3.1.25.1.19.18 “Tab 'Information'' on page 768

Only in the case of special features is there an additional help page for the specific device editor.

If the "<device name> Parameters" tab is not shown, then select the “Show generic device configuration editors” option in the CODESYS options (“Device Editor” category).

PROFINET IO is based on the working model of PROFIBUS DP, but it uses Ethernet TCP/IP for cyclic communication between the central PLC and the distributed field devices. It is appropriate for fast transmission of I/O data, as well as required data, IT functions, and parameters.

Even though the PROFIBUS master-slave method is replaced in principle by a provider-consumer model for PROFINET IO, the PROFINET IO real-time communication for cyclic transmission of user data requires a defined master-slave configuration. Therefore, there is a master (PROFINET IO controller), slaves (PROFINET IO devices), and I/O modules below the slaves.

In order for you to insert and configure PROFINET IO devices as objects in the device tree, you must first install the I/O controller, I/O devices, and I/O modules on the local system. If this does not happen automatically with the installation of the current development system, then you have to do it explicitly by means of the device repository dialog. The GSDML device description files that are supplied with the hardware are used as a basis for this.

You insert a PROFINET IO controller into the device tree below a device that has a PLC logic node. The device description file of the PROFINET IO controller defines which PROFINET IO devices you can insert below it. Similarly, the description of an I/O device defines which PROFINET IO modules you can insert below the I/O devices. For I/O modules that have a fixed position, CODESYS automatically provides slots in the device tree. The "Insert Device" command is used for inserting devices. Please note that it is also possible to deactivate a device temporarily for network operation.

See also
- Command 'Insert device' % Chapter 1.3.1.25.2.4.3 “Command 'Insert Device'” on page 898

PROFINET IO - Controller

The configuration of the PROFINET IO controller consists of the controller and the PROFINET IO devices (slaves) inserted below it. The devices are configured by the controller and exchange data with it.

CODESYS provides two different PROFINET IO controllers: A variant especially for Hilscher CIFX adapters and a variant that runs on any Ethernet adapter.
When inserting the Ethernet-based controller, two tasks are created implicitly that are required by the PROFINET IO communication stack.

- **“Profinet_CommunicationTask”**: This task includes the acyclic communication services, such as establishing connections and diagnostics. These services are not time-critical due to very weak real-time demands. Therefore the task is low priority.
- **“Profinet_IOTask”**: This is where the actual PROFINET IO real-time data exchange takes place. Pending I/O data packets are processed in each cycle (see Slave Configurator: “Send Clock”). Therefore, a cycle time of 1ms is required (for 1ms send clock).

If the smallest selected sender interval for the slaves of this controller is 2ms, 4ms, or greater, then also a cycle time of 2ms or 4 ms can be selected. However, the priority of the task should be selected as high as possible.

For maximum I/O performance with minimum delay when reading/writing, I/O data can be updated in this task (→ insert own POU that updates I/Os in this task). No blocking or persisting operations should be executed in the I/O task, such as visualization or file access. If the task is blocked too long, then the watchdog cancels the connection for communicating with the slave (see Slave Configurator: “Watchdog”).

**NOTICE!**

We recommend that you activate the “Refresh I/Os in Stop” option in the PLC settings. Otherwise the communication is canceled when the application stops at a breakpoint.

---

**With the scan functionality, you can easily ascertain the present hardware structure and transmit to the device tree.**

See also

-]% Chapter 1.5.6.2.14.3 “Command ‘Scan For Devices’” on page 3375
- ]% Chapter 1.5.6.2.14.1.1 “PROFINET IO Controller - General” on page 3364

**PROFINET IO Controller - General**

Object: PROFINET IO Controller

The PROFINET IO controller, like the slaves, is identified by the station name. In the case of CIFX variants, you can also configure the IP settings here. Otherwise the settings are applied from the superordinate Ethernet node.

<table>
<thead>
<tr>
<th>“Station name”</th>
<th>The station name of the device. It is used for unique identification of the device in the network.</th>
</tr>
</thead>
<tbody>
<tr>
<td>“IP Parameter”</td>
<td></td>
</tr>
<tr>
<td>“IP address”</td>
<td>Note: Available for CIFX PROFINET only.</td>
</tr>
<tr>
<td>“Subnet mask”</td>
<td>If you insert the controller below an Ethernet adapter, then you have to define the IP parameters in the dialog of the Ethernet adapter.</td>
</tr>
<tr>
<td>“Default gateway”</td>
<td></td>
</tr>
<tr>
<td>“Default Slave IP Parameter”</td>
<td></td>
</tr>
<tr>
<td>“First IP address”</td>
<td>Range of IP addresses that CODESYS uses by default when inserting PROFINET IO devices into the device tree. If you use the “Auto IP” function in the scan dialog, then IP addresses are also used from this range. The next free IP address is selected here.</td>
</tr>
<tr>
<td>“Last IP address”</td>
<td></td>
</tr>
<tr>
<td>“Subnet mask”</td>
<td></td>
</tr>
<tr>
<td>“Default gateway”</td>
<td></td>
</tr>
</tbody>
</table>
### PROFINET IO Provider / Consumer Status

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Application stop --&gt; Substitute values</strong></td>
<td>When the user stops the application, the provider state is set to &quot;BAD&quot;. Then the slaves set the inputs and outputs to predefined substitute values. For more information, see &quot;CODESYS default values - PROFINET IO substitute values&quot; at the end of this chapter.</td>
</tr>
<tr>
<td><strong>Add to I/O mapping</strong></td>
<td>☑️: The incoming status information is added to the I/O mapping for all modules; provider state for the input data and consumer state for the output data.</td>
</tr>
<tr>
<td><strong>Port Data</strong></td>
<td>Neighboring device with port that is connected to this port. You can accept this setting in the &quot;PROFINET IO Controller Topology&quot; tab.</td>
</tr>
<tr>
<td><strong>Cable length</strong></td>
<td>Length of the network cable (in meters)</td>
</tr>
<tr>
<td>- &lt; 10</td>
<td></td>
</tr>
<tr>
<td>- &lt; 25</td>
<td></td>
</tr>
<tr>
<td>- &lt; 50</td>
<td></td>
</tr>
<tr>
<td>- &lt; 100</td>
<td>When a cable length is specified, it is checked when the controller is powered up. An incorrect cable length causes an error message.</td>
</tr>
<tr>
<td><strong>MAU type</strong></td>
<td>Type of network cable</td>
</tr>
<tr>
<td><strong>Watchdog</strong></td>
<td>Note: Available for CIFX PROFINET only.</td>
</tr>
<tr>
<td><strong>Enable</strong></td>
<td>☑️: If the CIFX firmware is not set within the given time (for example, in the case of an exception error in the application), then it is reset. The connection is terminated and the slaves switch to their substitute values. The defaults for the watchdog originate from the device description.</td>
</tr>
</tbody>
</table>

### CODESYS default values - PROFINET IO substitute values

PROFINET IO devices set their inputs and outputs to predefined substitute values when there is an interruption. These values are defined in the field device in contrast with default values. These values are usually zero, but specific substitute values can also be configured depending on the device.

The substitute values are set in the following cases:

- The connection is interrupted.
- The controller sets the provider state for the incoming data to "BAD".
- Other interruptions occur (for example, exception in host application, incorrect parameterization)

If the "Application Stop --> Substitute Values" option is enabled, then the controller sets the provider states to "BAD" at application stop. In this case, the slaves set their substitute values. All incoming data from the controller is ignored (including default values).

If the default values defined in the application should be set for an application stop, then you have to disable this option. Moreover, you should enabled the "Update IO while stop" option (in the "PLC Settings"). Otherwise, the CODESYS PROFINET IO controller is stopped.

### PROFINET IO Controller - Overview

Object: PROFINET IO controller

This tab shows an overview of all devices with their communication settings. The station name, the IP configuration, and the transmission speed can all be modified directly in the list. For the other parameters, the "Communication Settings" dialog opens.

These settings are overwritten by the settings of the device.
Table 759

<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Default IP&quot;</td>
<td>Sets the IP address of the selected device to a value within the standard address range. This range is defined in the settings of the controller. The command is also available in the context menu of the selected line.</td>
</tr>
<tr>
<td>&quot;I/O Performance&quot;</td>
<td>Opens the &quot;Communication Settings&quot; dialog. Here is where the transmission speed for I/O data is configured for all selected devices. For individual parameters (RT class, SendClock, reduction), a value range can be selected that is valid for all selected devices. The command is also available in the context menu of the selected line.</td>
</tr>
</tbody>
</table>

Table 760: Dialog 'Communication Settings'

<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Send Clock (ms)&quot;</td>
<td>Send clock time (in milliseconds).</td>
</tr>
<tr>
<td>&quot;Reduction Ratio&quot;</td>
<td>Scaling factor. The transmission interval results from the &quot;Send Clock&quot; multiplied by the &quot;Reduction Ratio&quot;. Therefore, a &quot;Send Clock&quot; of 1ms and a Reduction Ratio of 4 means that I/O data is sent every 4ms.</td>
</tr>
<tr>
<td>&quot;Phase&quot;</td>
<td>With a &quot;Reduction Ratio&quot; of n, the transmission interval is subdivided into phases 1 to n (where transmission is in one phase only). You can determine the phase for transmission for the purpose of load distribution. If &quot;Send Clock&quot; = 1 and &quot;Reduction Ratio&quot; = 4 (as in the example above), then you could configure phases 1 - 4. For four slaves with this send clock and reduction ratio settings, you could assign one of the four phases to each of the four slaves. In this way, only one data package is sent in each of the four phases of the clock time and the load is distributed equally.</td>
</tr>
<tr>
<td>&quot;Watchdog (ms)&quot;</td>
<td>Monitoring time. Possible values: 3 ms to 1920 ms. (In steps, depends on &quot;Send Clock&quot; * &quot;Reduction Ratio&quot;) A connection is canceled when the controller or the PROFINET IO device does not receive I/O data from the communication peer within this time period. The device enters error mode and switches the outputs to default values.</td>
</tr>
<tr>
<td>&quot;VLAN ID&quot;</td>
<td>VLAN identifier: Number between 0 and 4095 for VLAN type 802.1Q. Note: For newer devices compliant with PROFINET IO specification V2.3, only &quot;0&quot; is still permitted.</td>
</tr>
<tr>
<td>&quot;RT Class&quot;</td>
<td>If available, you can select the required RT class from the list (real-time communication).</td>
</tr>
</tbody>
</table>

See also

- § Chapter 1.5.6.2.14.1.1 “PROFINET IO Controller - General” on page 3364
- § Chapter 1.5.6.2.14.1.4.1 “PROFINET IO-Device - General” on page 3367

PROFINET IO Controller - Bus Cycle Task

General information

As a rule, for each IEC task the used input data is read at the start of each task (1) and the written output data is transferred to the I/O driver at the end of the task (3). The implementation in the I/O driver is decisive for further transfer of the I/O data. The implementation is therefore responsible for the timeframe and the specific time when the actual transmission occurs on the respective bus system.

The bus cycle task of the PLC can be defined globally for all fieldbuses in the PLC settings. For some fieldbuses, however, you can change this independent of the global setting. The task with the shortest cycle time is used as the bus cycle task (setting: "unspecified" in the PLC settings). In this task, the messages are normally transferred on the bus.

Other tasks copy only the I/O data from an internal buffer that is exchanged only with the physical hardware in the bus cycle task.
Using tasks

The "Task Deployment" provides an overview of used I/O channels, the set bus cycle task, and the usage of channels.

WARNING!

If an output is written in various tasks, then the status is undefined, as this can be overwritten in each case.

When the same inputs are used in various tasks, the input could change when a task is processed. This happens if the task is interrupted by a task with a higher priority and causes the process map to be read again. Solution: At the beginning of the IEC task, copy the input variables to variables and then work only with the local variables in the rest of the code.

Conclusion: Using the same inputs and outputs in several tasks does not make any sense and can lead to unexpected reactions in some cases.

PROFINET IO - Device

PROFINET IO does not provide any additional settings. Its functionality corresponds to the general description.

PROFINET IO - Device

1.5.6.2.14.1.4.1  PROFINET IO-Device - General............................ 3367
1.5.6.2.14.1.4.2  PROFINET IO Device - Options........................ 3369
1.5.6.2.14.1.4.3  PROFINET IO Device - IOxS............................ 3369
1.5.6.2.14.1.4.4  PROFINET IO Device - Status............................ 3370

PROFINET IO-Device - General

Object: PROFINET IO Device
In this dialog, you configure a communication link (PROFINET IO: application relation) to a PROFINET IO field device.

For all settings in the present dialog, the device description determines if the values here are editable and the values that are predefined or possible.

<table>
<thead>
<tr>
<th><strong>“Station name”</strong></th>
<th>The station name of the device. It is used for unique identification of the device in the network.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>“Station status”</strong></td>
<td>32-bit error code compliant with the PROFINET IO specification. In case of error, the status is provided here, for example, when establishing a connection fails or a link is interrupted. For some errors, an additional description is provided.</td>
</tr>
</tbody>
</table>

**“IP Parameters”**

- **“IP address”**
- **“Subnet mask”**
- **“Default gateway”**

**“Communication”**

- **“Send clock (ms)”** Send clock time (in milliseconds).
- **“Reduction ratio”** Scaling factor
  
The transmission interval results from the “Send clock” multiplied by the “Reduction ratio”. Therefore, a “Send clock” of 1ms and a “Reduction Ratio” of 4 means that I/O data is sent every 4ms.

- **“Phase”** With a “Reduction ratio” of n, the transmission interval is subdivided into phases 1 to n (where transmission is in one phase only). You can determine the phase for transmission for the purpose of load distribution.
  
  If “Send clock” = 1 and “Reduction ratio” = 4 (as in the example above), then you could configure phases 1 - 4. For four slaves with this send clock and reduction ratio settings, you could assign one of the four phases to each of the four slaves. In this way, only one data package is sent in each of the four phases of the clock time and the load is distributed equally.

- **“Watchdog (ms)”** Monitoring time. Possible values: 3 ms to 1920 ms. (in steps, depends on “Send clock” * “Reduction ratio”)
  
  A connection is canceled when the controller or the PROFINET IO device does not receive I/O data from the communication peer within this time period. The device enters error mode and switches the outputs to default values.

- **“VLAN ID”** VLAN identifier: Number between 0 and 4095 for VLAN type 802.1Q.
  
  Please note: For newer devices compliant with PROFINET IO specification V2.3, only "0" is still permitted.

- **“RT class”** if available, you can select the required RT class from the list (real-time communication).

**“User Parameters”**

- **“Set All Default Values”** CODESYS resets all settings to default values (see default value column) from the GSDML file.
- **“Read All Values”** CODESYS reads the current values from the device and updates them in the editor.
- **“Write All Values”** CODESYS writes the current values from the editor to the I/O device. Not all I/O devices support parameter updates in run mode. If not, then an error message is displayed.
PROFINET IO Device - Options

Object: PROFINET IO Device

On this tab, you define whether the device is configured by another controller and therefore is operated as a shared device.

In addition, you can define the neighboring devices and connection cable.

| “Shared device” | ✔️: This device is configured by another controller. Modules that are used by the other controller must be deactivated by setting this option. |
| “Peer station/port” | Neighboring device with port that is connected to this port. You can accept this setting in the “PROFINET IO Controller Topology” tab. |
| “Cable length” | Length of the network cable (in meters)  
- < 10  
- < 25  
- < 50  
- < 100  
When a cable length is specified, it is checked when the controller is powered up. An incorrect cable length causes an error message. |
| “MAU type” | Type of network cable  
- Copper  
- Optical fiber  
When a cable type is specified, it is checked when the controller is powered up. An incorrect cable type causes an error message. |

See also

- PROFINET IO Device - IOxS

Object: PROFINET IO device

In this dialog, you can access I/O provider data and I/O consumer data.

In addition to the actual input and output data, PROFINET sends status information (provider states). These indicate whether or not the values are valid (PS == 0x80). For example, the input values of a slave could be invalid if a hardware failure is present. Likewise, you could explicitly declare the outputs sent by the controller as invalid if an error occurs somewhere in the application. Then the slaves activate the default values if they are configured.

Submodules without their own I/Os also provide a provider state for inputs.

“Show I/O channels” | Note: The channels are deactivated when the option “Add I/O mapping” is activated in the “PROFINET IO Controller - General” tab.  
- ✔️: The associated module provides additional I/Os with the provider state in the I/O mapping dialog. The corresponding module can be opened by double-clicking the address field. |
"Submodule"  
"Variable"  
"Address"

The I/O variable that is mapped to the input/output.

IEC address where this variable is mapped (I/O mapping dialog).

The value "hex 80" (bit 7) from the provider state bytes display the status "OK". For detailed information about the definition of this byte, refer to the technical specification for PROFINET IO.

In the case of an I/O provider state that is mapped for output data, the output PS must be set to "GOOD" (=0x80) explicitly. Otherwise, the device will not accept the data. Then the slaves activate the default values if they are configured.

If the application is stopped, then the unmapped I/O provider states for outputs are set to "bad". Mapped I/O provider states are set to their configured values.

PROFINET IO Device - Status
Object: PROFINET IO Device
This tab shows the diagnostics data and the alarms of the device.

"Diagnostics Data"  
"Alarms"  
"Acknowledge"

The diagnostics currently located in the device. The current diagnostics data is reloaded from the status display as long as a link can be established to the device.

All alarms of this device that have been reported by this device since the start of the application. The alarms are stored in a ring buffer on the controller.

Deletes the alarms from the ring buffer.

PROFINET IO - Module

1.5.6.2.14.1.5.1  PROFINET IO Module - General.......................................... 3370
1.5.6.2.14.1.5.2  PROFINET IO Module - Status............................................ 3371

PROFINET IO Module - General
Object: PROFINET IO Module

Table 761: “Module Information”

| “ID number” | Identification of the module (from the device description). |
| "Slot number" | Position of the I/O module below the I/O device, starting at “1” for the first module and incremented for each additional module. This results automatically from the current structure in the device tree. |
Table 762: “User Parameters”

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Set All Default Values”</td>
<td>CODESYS resets all settings to default values (see default value column) from the GSDML file.</td>
</tr>
<tr>
<td>“Read All Values”</td>
<td>CODESYS reads the current values from the device and updates them in the editor.</td>
</tr>
<tr>
<td>“Write All Values”</td>
<td>CODESYS writes the current values from the editor to the I/O module. Not all I/O module support parameter updates in run mode. If not, then an error message is displayed.</td>
</tr>
</tbody>
</table>

PROFINET IO Module - Status

Object: PROFINET IO Module

This tab shows the diagnostics data and the alarms of the module.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Diagnostics Data”</td>
<td>The diagnostics currently located in the device. The current diagnostics data is reloaded from the status display as long as a link can be established to the device.</td>
</tr>
<tr>
<td>“Alarms”</td>
<td>All alarms of this module that have been reported by this device since the start of the application. The alarms are stored in a ring buffer on the controller.</td>
</tr>
<tr>
<td>“Acknowledge”</td>
<td>Deletes the alarms from the ring buffer.</td>
</tr>
</tbody>
</table>

PROFINET IO - Slave device

The configuration of the PROFINET IO slave device consists of the device itself as well as the modules inserted below.

CODESYS provides two different PROFINET IO slave devices: A variant especially for Hilscher CIFX adapters and a variant that runs on any Ethernet adapter.

When inserting the Ethernet-based slave devices, two tasks are created implicitly that are required by the PROFINET IO communication stack.

- “Profinet_CommunicationTask”: This task includes the acyclic communication services, such as establishing connections and diagnostics. These services are not time-critical due to very weak real-time demands. Therefore the task is low priority.
- “Profinet_IOTask”: This is where the actual PROFINET IO real-time data exchange takes place. Pending I/O data packets are processed in each cycle (see Slave Configurator: “Send clock”). Therefore, a cycle time of 1ms is required (for 1ms send clock).

For maximum I/O performance with minimum delay when reading/writing, I/O data can be updated in this task (insert own POU that updates I/Os in this task). No blocking or persisting operations should be executed in the I/O task, such as visualization or file access. If the task is blocked too long, then the watchdog cancels the connection for communicating with the slave (see Slave Configurator: “Watchdog”).

NOTICE!

We recommend that you activate the “Refresh I/Os in Stop” option in the PLC settings. Otherwise the communication is canceled when the application stops at a breakpoint.

See also

- Chapter 1.5.6.2.14.2.1 “PROFINET IO Slave device - General” on page 3372
- Chapter 1.5.6.2.14.2.3 “PROFINET IO Slave device - General” on page 3374
PROFINET IO Slave device - General

Object: PROFINET IO Slave Device

The tab shows the basic communication preferences.

According to the Profinet standard, the PROFINET IO slave device is responsible for the IP settings of the used adapter. It must save remanent IP settings and be able to reset or modify at the request of the controller (IP=0.0.0.0). Initial state (Reset to factory) is with deactivated IP suite (IP=0.0.0.0).

However, if the PROFINET IO device is a programmable logic controller and connected with the (CODESYS) programming environment via one and the same Ethernet adapter, then modifying and resetting the IP address is interruptive (connection interruption between IDE and PLC). Therefore, one of the modes is provided that deviates from the standard ("Use project parameters"). You must also use this option for Windows and VxWorks systems.

Table 763: “IP and Name Assignment”

<table>
<thead>
<tr>
<th>“Use remanent data”</th>
<th>The IP settings and the station name of the file are used. The file is stored in the file system. The data is set by the controller and saved to a file by the device.</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Use project parameters”</td>
<td>The IP settings and the station name of the project are used (settings of the Ethernet adapter). This option must be selected for Windows, VxWorks, and WinCE, because changing the IP address is not possible for these systems.</td>
</tr>
<tr>
<td>“Station name”</td>
<td>Station name of the PROFINET IO device</td>
</tr>
</tbody>
</table>

Table 764: “IO Provider / Consumer Status”

<table>
<thead>
<tr>
<th>“Use incoming”</th>
<th>✓: The I/O data for the provider and consumer states is generated which is received by the controller.</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Use outgoing”</td>
<td>✓: The I/O data for the provider and consumer states is generated which is sent to the controller.</td>
</tr>
</tbody>
</table>
| “Substitute values”          | The substitute values for the output data become active when the corresponding provider status (Output Data PS) is set to Bad. The output data is sent by the controller and copied to the %I area of the Profinet modules.  
  The following options are available for the substitute values:
  - “Inactive”: The outputs are set to "inactive" (example: 0).
  - “Last value”: The output data retains the last valid value (provider status = Good). The value is retained even if the connection to the controller has been interrupted. |

In online mode, the station name and the IP settings are displayed “Status” tab.
PROFINET IO Slave device – Configuration

Linux configuration

The following configuration steps are necessary in Linux for a standard-compliant behavior (example here for Debian/Raspi):

- Select the option “Use operating system settings” for the Ethernet adapter.
- Deactivate the protocol filters and set the IP address in the config file of the CODESYS runtime:
  - [SysEthernet]
    Linux.ProtocolFilter=3
  - [SysSocket]
    Adapter.0.Name="eth0"
    Adapter.0.EnableSetIpAndMask=1
- Configure the operating system so that the Ethernet adapter starts without the IP address in "promiscuous mode". In /etc/network/interfaces:
  - auto eth0
    iface eth0 inet manual
    pre-up ifconfig $IFACE promisc up
    post-down ifconfig $IFACE promisc down
    If necessary, deactivate the “Zero Conf” services, such as the avahi daemon:
    update-rc.d -f avahi-daemon remove

See also

- § Chapter 1.5.6.2.14.2.1 “PROFINET IO Slave device - General” on page 3372

VxWorks configuration

The following configuration steps are necessary for standard-compliant behavior in VxWorks:

- Select the option “Use operating system settings” for the Ethernet adapter.
- Setting the IP address in the config file of the CODESYS runtime is permitted. At first, do not set any IP address for this adapter:
  - [SysSocket]
    Adapter.0.Name="eth0"
    Adapter.0.EnableSetIpAndMask=1
    Adapter.0.ipaddress=
    Adapter.0.subnetmask=
- Configure the operating system so that the Ethernet adapter starts without the IP address in "promiscuous mode". To do this, you have to add these entries to the startup script (for the desired adapter):
  - ifconfig ("gei0 promisc")
    ipAttach 1,"gei0"

See also

- § Chapter 1.5.6.2.14.2.1 “PROFINET IO Slave device - General” on page 3372

Windows configuration

The following configuration steps are necessary for standard-compliant behavior in Windows:

1. For CODESYS Control Win V3: Install WinPCap (for example, included in the setup of Wireshark).
   When using CODESYS Control RTE V3, WinPCap is not necessary.
2. Set a fixed IP address for the Ethernet adapter to be used in the Windows control panel.
   Caution: Any IP address from the auto-config (169.254.x.y) range must not be used. Instead, use an address for the local subnet (for example, 192.168.0.x / 255.255.255.0).
3. Setting the IP address in the config file of the CODESYS runtime is permitted:
   - [SysSocket]
   - Adapter.0.Name="Profinet Adapter"
   - Adapter.0.EnableSetIpAndMask=1
   - Adapter.0.ipaddress=
   - Adapter.0.subnetmask=

4. Start WinPLC.
   At the initial start, some settings for the adapter specified in step 3 are applied in the Windows Registry. These are effective after the system has restarted.

5. After the system is restarted, check the control panel whether an auto-config address has been used for the adapter (--> remove).


7. In the CODESYS project, set the option "Use operating system settings" on the Ethernet adapter device.

See also
- Chapter 1.5.6.2.14.2.1 “PROFINET IO Slave device - General” on page 3372
- Chapter 1.5.6.2.14.5.1 “Runtime system configuration” on page 3378

Windows CE configuration
When using Windows CE, the following should be taken into account:
- Select the option “Use operating system settings” for the Ethernet adapter.
- Setting the IP address in the *.cfg file is not permitted.
- Select the option “Use project parameters” from the “General” tab.

See also
- Chapter 1.5.6.2.14.2.1 “PROFINET IO Slave device - General” on page 3372

PROFINET IO Slave device - General
Object: PROFINET IO slave device
This tab shows the basic communication preferences.

<table>
<thead>
<tr>
<th>&quot;Use remanent data&quot;</th>
<th>The IP settings and the station name of the file are used. Initially, the IP address is 0.0.0.0 according to the standard and the station name is blank. When a controller sets these values with the &quot;store remanent&quot; option, then they are stored here.</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Use project parameters&quot;</td>
<td>When starting the device, the values defined in the project for IP configuration and station name are always used initially. This option must be selected for Windows, VxWorks, and WinCE, because changing the IP address is not possible for these systems.</td>
</tr>
<tr>
<td>&quot;Station name&quot;</td>
<td>The name of the device in the network. Note: The station name and the IP settings can deviate from the (default) settings configured in the project. They can be set by the controller in online mode and in some cases stored persistently (this means when this is specified for the controller in the Set IP or Set station name commands). After a restart, the device is configured with these values as long as the &quot;Use remanent data&quot; option is set.</td>
</tr>
</tbody>
</table>
Initial settings for identification in the network.

Caution: This data can be modified by the PROFINET IO controller. The data is stored remanent on the file system of the controller. After the controller is restarted, this stored data goes into effect. The settings here are then ignored.

In online mode, the station name and the IP settings are displayed “Status” tab.

Command 'Scan For Devices'

Function: This command establishes a brief connection to the hardware and determines the devices in the network. You can then adopt the devices found into the device tree of your project.

Call: Main menu “Project”, context menu of a device object in the device tree.

Requirement: The settings for communication with the control device are correct. The gateway and the PLC are running. The device supports the scan function.

To use the scan functionality, you must log in once in order to load the library to the controller.

The command refers to the master controller selected in the device tree. For example, an already inserted PROFINET IO controller can be selected and the command used to determine the I/O devices and I/O modules assigned to it.

After scanning, the dialog box “Scan for Devices” appears with the results.

Dialog 'Scan Devices'

Table 765: “Scanned Devices”

Data about the scanned device depending on network type.

When you change a value in the list of scanned devices, the value is shown in italics. This indicates that the new value has been changed in the editor in CODESYS, but not in the device. When you download the value to the device, it is shown normally.

Value that indicate differences between the project and the scanned device are shown in orange.

If several device descriptions are available for the scanned device, then the name is displayed in bold. The selection of the matching device description is resolved differently for different fieldbuses. Details can be found in the corresponding fieldbus sections.

If a device description cannot be found, then the following message is shown: "Attention! The device was not found in the repository." Depending on the bus system, additional information is displayed, such as manufacturer number and product number. The device cannot be inserted into the project without the installed device description.

“Show differences to project”

☑: The table in the dialog also shows additional configured devices (in the device tree of the project).

☐: The table shows all scanned devices. The configured devices are not shown.
Table 766: "Configured Devices"

This part of the dialog is visible only when you select the “Show differences to project” option.

Differences between the scanned and configured devices are color-coded. Devices in green are identical on both sides. Device in red are available only in the view of the scanned or configured devices.

<table>
<thead>
<tr>
<th>Difference Description</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>If you have selected a device in both views, then the scanned devices are</td>
<td>inserted above the selected configured device.</td>
</tr>
<tr>
<td>inserted below the selected configured device.</td>
<td></td>
</tr>
<tr>
<td>If you have selected a device in both views, then the configured devices are</td>
<td>replaced by the selected scanned device.</td>
</tr>
<tr>
<td>All scanned devices are copied to the project.</td>
<td></td>
</tr>
<tr>
<td>Deletes the selected configured device.</td>
<td></td>
</tr>
</tbody>
</table>

**NOTICE!**

If you insert devices, which are available in the device tree, to the device tree with “Copy All Devices to Project”, then the following should be noted. The data of the “Process Data” and “<..> I/O Mapping” tabs of the existing devices can be overwritten with the data of the recently inserted devices.

---

**Special settings for PROFINET**

**“ID No.”**

Display only when
- I/O device and controller are part of the same logical network
- IP address and network mask of the I/O device are set correctly

**“Station name”**

You can modify these entries or create them again before you accept the device and its settings into the configuration of the project. If you do not define an address, then the presets are used. The presets are defined in “PNIO Controller Parameter”.

**“IP address”**

To determine the module structure of the station, it must possess a valid IP address (see setting the IP address with “Set IP” or “Denominate”).

**NOTICE:** Each device must have received a station name because the name is required in network operation for certain functionalities.

To declare the new name in the bus system, you must click the “Denominate” button for the selected device entry.
"I&M" Use this command for retrieving the I&M data (identification and maintenance) of the device. The data is shown in the dialog “I&M data”.

"Auto IP" With this command, you automatically set a valid IP address within the network (network settings of the master) if the address scanned by the device is invalid.

"<--> " Use this command for assigning the settings (“Station name”, “IP address”) from the configured device to the scanned device.

"--> " Use this command for assigning the settings (“Station name”, “IP address”) from the scanned device to the configured device.

"Reset" Resets the device settings to default values.

"Blink LED" In the “Found Devices” view, you can identify a device by its hardware by selecting the entry in the dialog and clicking “Blink LED”. The device should respond with a flashing signal.

"Set Name+IP" CODESYS attempts to set the station name and IP address that were specified for the selected slave.

“Show only unnamed stations” Only devices are listed that do not have an assigned station name.

Please note the option of aligning the configuration settings of a PROFINET IO device with the settings of the corresponding hardware device © Chapter 1.3.1.8.2 “Mapping a hardware structure in the device tree” on page 159.

Dialog I&M data

| “I&M version” | Implemented version of I&M functions |
| "Vendor ID" | ID of the manufacturer of the submodule |
| "Order ID" | Order number |
| "Serial Number" | Serial number of the submodule (unique production number) |
| "Hardware Revision" | Hardware version of the submodule |
| "Software Revision" | Software version of the submodule |
| "Location" | Location where the device is installed (for example, in a plant or production hall) |
| "Function" | Function or task of the submodule |
| "Date" | Day of installation or commissioning of the device or module |
| "Description" | Any individual information |

For some devices, the parameters “Function”, “Location”, “Date”, and “Description” can be modified.

Command 'Profinet Topology'

Function: The command displays the network topology of the PROFINET IO network as a tree structure. For each port of the PROFINET IO device, the neighboring device connected here and its port are displayed (if available).

Call: Context menu of a PROFINET IO controller.

Requirement: The communication settings to the controller are correct. The gateway and the PLC are running. The device supports the scan function.
refreshes the view

“I&M“ Gets the I&M data (identification and maintenance) of the device and view in a dialog.

“Auto IP“ Automatic setting of a valid IP address for the selected device in case the address scanned by the device is invalid.

“Port State“ Possible states:
- ● : Unknown or not connected
- ● : OK
- ● : Error

“Neighbour“ Name of the neighboring device in the network.
In the case that the device does not support the function or does not have a valid IP configuration, "No data!" is displayed. Details about the error cause are shown as a tooltip.

“Neighbour-Port“ Name of the neighboring port in the network.

Troubleshooting

This section describes the troubleshooting for CODESYS PROFINET IO drivers (PLC and slave device). In particular, it covers problems that can occur in connection with the different runtime variants. This section does not provide an introduction to PROFINET diagnosis.

NOTICE!
Please note the configuration steps for each platform.

Runtime system configuration

When problems of any kind occur, first check the configuration of the operating system and the CODESYS runtime system, as described below.

Firewall and package filter

The PROFINET stacks do not use the configured Ethernet adapters of the PLC exclusively. The PROFINET protocols run parallel with the network protocols (for example, UDP), as well as building on these. In this way, these Ethernet adapters can also be used for CODESYS communication. However, the firewall configuration (if active) must be modified to the PROFINET requirements.

Network protocols used
- ● PROFINET RT (Ethtype = 0x8892)
- ● UDP, Port 0x8894, as well as free ports in the range of 0xC000 – 0xFFFF ( – 0xC0FF is usually sufficient).
  Also for incoming packages.
- ● LLDP (Ethtype 0x88CC)
- ● ARP (Ethtype 0x0806)

It is usually sufficient to unblock the UDP ports at the firewall.

The PROFINET-RT protocol is not an IP protocol. It cannot be routed and is restricted to the local network.

If you suspect that the firewall is the cause of the communication problems, then deactivate it temporarily in an isolated environment.
Runtime system configuration

For communication with the PROFINET RT protocol, the PROFINET IEC stacks use a special runtime system interface (SysEthernet) that allows the transmission of RAW Ethernet frames. This runtime system component must be activated in the configuration file of PLC.

```
[ComponentManager]
Component.1=CmpSysEthernet
```

For Linux systems, the protocol filter must also be deactivated.

```
[SysEthernet]
Linux.ProtocolFilter=3
[SysSocket]
```

For some platforms, a package filter is also active on the operating system level, and the Ethernet adapter must be switched explicitly to promiscuous mode (for example for Linux).

For example, in `/etc/network/interfaces`:

```
auto eth0
pre-up ifconfig $IFACE promisc up
post-down ifconfig $IFACE promisc down
```

CODESYS Control Win V3

For the operation of PROFINET, WinPCap must be installed on the system (for example, included in the setup of Wireshark).

If an Ethernet adapter is added again to the system, then the system has to be restarted. Otherwise, the adapter may not be detected. In this case, the driver displays the ErrorFindEthernet error in the "Status" tab.

Diagnosis methods

For more information about troubleshooting, refer to the following views in CODESYS or external software, each depending on the case:

Status dialog in the device configurator

If the object of the PROFINET controller or sPROFINET device has a red symbol in the device tree, or if no communication is taking place at all, then take a look at the "Status" tab in the device configuration dialog.

![Status dialog](image)

Here we are looking at the controller only. The same basically applies to the device.

The most important fields in the diagnosis structure displayed here are as follows:

"Controller status":

Current status of the controller; the individual values correspond to the enumeration `Profinet.ControllerState`:
Specific conclusions about problem sources can often be drawn from these values, for example for license issues or doubled station names.

The status value is also available in the PLC application:

**“Received frames” / “Received RT frames”:**

Under specific conditions, the controller status switches to “Run”, but communication is still not possible. The values “Received frames” / “Received RT frames” can provide more information:

“Received frames” shows the number of all Ethernet frames that were received over the Ethernet interface. “Received RT frames” shows the subset of the PROFINET RT frames.

If both counters remain at 0 after starting the controller although the device is in a network, then this indicates basic problems with the runtime system component `SysEthernet`.

If only “Received frames” increments but not “Received RT frames” although PROFINET devices are in the network, then this indicates a problem with the runtime system configuration (firewall, packet filter, etc.).

**“Recv time” / “Send time”:**

The maximum value and the average value are displayed here for both sending and receiving of the Ethernet frame. A performance indicator for the link to the Ethernet adapter (runtime system interface `SysEthernet`).

**PROFINET IO slave configurator**

In the case of a failed connection (slave symbol is red), the cause is displayed in the “General” tab of the device configuration dialog. (PNIOStatus according to PI Specification: Application Layer protocol for decentralized periphery Technical Specification for PROFINET IO, Version 2.3Ed2MU2, Date: February 2015, Order No.: 2.722)
On weak systems or systems with a poor link to the Ethernet component (SysEthernet), connection disruptions can be caused by jitter or high cycle times. The controller is affected above all, the device is not critical in this case. The "Monitoring" tab in the "Task configuration" helps to detect these problems.

The "Profinet_IOTask" is important because the PROFINET RT data packets are transmitted at a constant transmission speed. If this transmission speed cannot be maintained, then the connection is terminated by the communication watchdog "DataHoldTimer" of the PROFINET devices.

As recommended values, the average cycle time should not exceed 250 µs and the maximum cycle time should not exceed 1 ms.

The "Profinet_IOTask" should therefore run as high priority, and they should not block other IEC tasks (same or higher priority). In the context of this task, no blocking or long-lasting calls should occur.

In case of error in the configuration or an interruption of the connection, the PROFINET driver may provide more information in the log view of the device configurator of the PLC:

An established tool for network diagnostics in general is the free software Wireshark (www.wireshark.org). This can be used for recording all network traffic on the port of the controller/device. Wireshark can also be installed on another device (for example, a Windows PC) as long as it is in the same network as the PROFINET device. In this case, however, a switch must be used with a mirror port.

Performance

The PROFINET RT data packets are transmitted to both the controller and device at a constant transmission speed. If this transmission speed cannot be maintained, then the connection is terminated by the communication watchdog "DataHoldTimer" of the PROFINET devices, even during the phase of establishing a connection.

On weak systems or systems with a poor link to the Ethernet component (SysEthernet), unstable data transmission can occur.

Then either the transmission speed has to be reduced in the Reduction ratio setting, or the communication watchdog has to be intensified. The performance of the CPU is less important than the performance of the PROFINET driver. The link of the Ethernet interface is decisive. In addition to the hardware, the software-specific link (operating system and Ethernet drivers) also plays an important role.

Examples of a possible number of connections and transmission speed:
<table>
<thead>
<tr>
<th>OS</th>
<th>CPU</th>
<th>Connections * transmission speed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Windows (RTE)</td>
<td>Atom</td>
<td>50 * 1 ms</td>
</tr>
<tr>
<td>VxWorks</td>
<td>Power-PC</td>
<td>30 * 4 ms</td>
</tr>
<tr>
<td>Linux</td>
<td>Raspi V3</td>
<td>6 * 1 ms</td>
</tr>
</tbody>
</table>

PROFINET and other drivers

The PROFINET controller and device can run on one and the same Ethernet adapter with other drivers, such as Modbus/TCP or Ethernet-IP. However, they require exclusive access to the low-level Ethernet component (SysEthernet). Therefore, for example, they cannot run parallel with EtherCAT on one adapter. If multiple Ethernet adapters are available, then one PROFINET driver each (controller or device) can be operated there.

PROFINET IO and older runtime system versions

PROFINET drivers require at least one runtime system V3.5.7.0.

Important: PROFINET drivers in version 3.5.10.0 and later also function on older runtime systems, but they require the UDP library also in version 3.5.10.0 and later.

For RTS Windows RTE, at least V3.5.10.0 is recommended (problems regarding reserved UDP ports from the PROFINET area).

Hardware requirements

- PROFINET IO requires an Ethernet adapter that supports at least 100 mbps with full duplex.
- All applied network components (switches) also have to support this transmission mode.
- MAC addresses
  - For some controllers with multiple Ethernet adapters, unique MAC addresses are not assigned. This can lead to serious malfunctions. In this case, the user can usually set a MAC address.

Effects and causes

PLC - Device object displays red symbol

It is imperative that you check the “Controller status” in the “Status” tab of the device.

Causes

- IP or station name of the controller are doubled
- Driver does not have a license
- Internal error

See also

- “Status dialog in the device configurator” on page 3379
In order for the scan dialog to show the PROFINET devices currently located in the PROFINET network, the data is read in two stages:

- Then all devices are recorded in the PROFINET RT protocol.
- In UDP/IP, the module configuration is read for each device.

Possible problems:

<table>
<thead>
<tr>
<th>Effect</th>
<th>Cause</th>
</tr>
</thead>
<tbody>
<tr>
<td>PROFINET devices are displayed in the scan dialog, but not the modules of the devices.</td>
<td>The PROFINET device does not have an IP address, or it is located in another subnet than the controller and therefore it is unreachable via UDP. The firewall is configured incorrectly. Another clue for this are error messages in the “Log” of the PLC.</td>
</tr>
<tr>
<td>Scan dialog displays nothing.</td>
<td>The driver is not started, and the application is stopped. No PROFINET RT packets can be received.</td>
</tr>
</tbody>
</table>

Possible problems:

See also

- “Firewall and package filter” on page 3378
- “Status dialog in the device configurator” on page 3379

The controller is started and it is green (after about 3 sec.), but...

<table>
<thead>
<tr>
<th>Effect</th>
<th>Cause / Background</th>
</tr>
</thead>
<tbody>
<tr>
<td>...all configured PROFINET slaves remain red.</td>
<td>No connection to the network. Issues with firewall. Check that the device scan functions correctly.</td>
</tr>
<tr>
<td>...a PROFINET slave remains red.</td>
<td>Device not in network. Check that the station name is correct. Slave terminates the connection with errors. Performance problems; the communication watchdog starts when establishing the connection.</td>
</tr>
<tr>
<td>...PROFINET slaves switch between green, red, and orange.</td>
<td>When establishing the connection, the symbol of the slave is an oranges triangle. The station status is Unknown. If the Unknown state lasts for a long time before the symbol switches to red (error), then this means that the response from the slave is blocked by a firewall. In the red/error state, the station status shows the causes. Typical errors are the causes are:</td>
</tr>
<tr>
<td>Stat</td>
<td>Cause</td>
</tr>
<tr>
<td>us:</td>
<td></td>
</tr>
<tr>
<td>818</td>
<td>Incorrect station name device cannot be reached over the network (see 1.3.1.2)</td>
</tr>
<tr>
<td>13F</td>
<td>DNS unknown RealStationName</td>
</tr>
<tr>
<td>02</td>
<td></td>
</tr>
<tr>
<td>CF8</td>
<td>Transmit clock cannot be maintained (see 1.4, 1.3.3)</td>
</tr>
<tr>
<td>1FD</td>
<td></td>
</tr>
<tr>
<td>05</td>
<td></td>
</tr>
</tbody>
</table>
Effect | Cause / Background
---|---
CF81FD06 | AR cmi timeout | Transmit clock cannot be maintained (see 1.4, 1.3.3)

... PROFINET slave is green, but it has a red exclamation mark, or individual modules are red.
The controller has an active connection, and therefore everything is okay.
The PROFINET device reports PROFINET diagnostic messages only for individual modules.

See also
- $“PROFINET IO slave configurator” on page 3380$
- $“Status dialog in the device configurator” on page 3379$
- $“Task monitor (monitoring)” on page 3381$
- $Chapter 1.5.6.2.14.5.3 “Performance” on page 3381$

1.5.6.3 Protocols and special servers
1.5.6.3.1 IEC60870-5-104 (Telecontrol)

General information IEC60870

Introduction

The implemented IEC60870-5-104 protocol allows link-ups between AC500 CPUs with onboard Ethernet and external systems. The link-up takes place via the onboard Ethernet interface of the CPU. The telecontrol protocol according to IEC60870-5 is used.

The CPU can work as both control station and substation. In control direction, setpoints and commands can be set; in monitoring direction the control station sends status values, real values and discrete values to the substation. Via general inquiry, the substation requests the control station to send all status values, real values and discrete values. Otherwise, these values are sent by the control station on a change-driven basis, cyclically or when triggered by an application. Status values, real values and discrete values may contain timestamps. These are filled in with the time of the process station when sent. The CPU can time-synchronize the telecontrol link.

A module accepts the configuration of the physical interface (link layer) and the general protocol parts (application layer).

Send and receive blocks are available for data exchange. These blocks exist for the IEC60870-5 data types setpoint value, command value, double command value, status value, double status value, real value and discrete value. The inputs/outputs of the send and receive blocks are combined with the signals to be communicated. See documentation of IEC60870 library for more information.

Limits of supported devices

AC500 V3 (Standard):
- PM5630: Support of 5 control stations and/or substations with 1.000 information objects overall on ETH1 and ETH2.
- PM5650: Support of 10 control stations and/or substations with 5.000 information objects overall on ETH1 and ETH2.
- PM5670: Support of 20 control stations and/or substations with 10.000 information objects overall on ETH1 and ETH2.

Data flow control

Each send or receive block can only process one data message. Ideally, new data are available at each user task run-through or new data can be sent.
If the output OV (send block only) indicates TRUE, the Function Block computes more quickly than the data can be sent. This can happen if the receive block is not computed quickly enough and has thus not collected all the data.

Alternatively, this block sends either cyclically or if the input value is changed. Ideally, the topical data can be sent via the telecontrol link in connection with every user task run-through.

Data integrity

With IEC60870-5 protocol, a distinction is made between data transmission in the monitoring direction (status values, real values, discrete values) and in the control direction (commands and setpoints).

All data transmissions are acknowledged from the link communication level by the receiver. This acknowledgement is not sent to the sender of the data in every telecontrol link.

For data transmission in control direction, additional acknowledgement (e.g. ACTTERM) is possible. These acknowledgements are not sent by every telecontrol link either. For safe data transmission, it is necessary, in such cases, to configure data readback. The receiver then sends the data received back to the sender via the corresponding send blocks.

Information in the monitoring direction is acknowledged by the receiver on the lowest communication level (link level) when received. This acknowledgement is generated by the telecontrol head itself with some telecontrol heads. In the event of overload/overrun, a data message may be lost. For data in the control direction, so-called ACTTERM acknowledgement can be used. This additional acknowledgement is sent back to the sender when the data have been executed in the process. If data are to be sent in the monitoring direction with guaranteed transmission, it is necessary to read back the sent value via another variable and, after observing a monitoring time, resend in the event of an error.

Data transmission

Send blocks

On the basis of the communication protocol, it is sensible to restrict the data types at one send block to one type. Therefore, there are 5 types of send blocks: send of status values, commands, real values, setpoints and discrete values. These types are mapped to the IEC1131 data types BOOL, REAL and DINT. See documentation of IEC60870 Library for more information.

Operating modes of the send blocks

The send blocks know three operating modes to send their data:

● Caused by request pin (SEND)
● Send in connection with a change of data (AUTO)
● Cyclic send of data (CYCLE)

Send via request pin

The SEND signal is evaluated on the rising edge, the RDY signal remains applied for one computation cycle. If a rising edge is generated again at the SEND signal although no acknowledgement has yet been received from the receiver, the OV pin is set in order to indicate that an overrun has happened. The evaluation of the receive acknowledgement is carried out before the evaluation of whether transmission is to take place. This means, assuming that there is an appropriately fast telecontrol link, that in connection with change-driven and cyclic transmission, a transmission job can be sent in connection with every computation of the block. In connection with send via the request pin it is possible to send only in connection with every second computation (send takes place only with a rising edge).
Change-driven send of data

Data are always sent when the value of the input variables changes. When changes take place, there is an internal simulation that the SEND pin changed from 0 to 1.

In order to prevent unnecessarily frequent send in the event of mild fluctuations in the input value, a threshold value can be configured for real values and setpoints. The input value is not sent until it differs positively or negatively from the value last sent by more than the threshold value.

If the input value changes again although no acknowledgement has yet been received from the receiver, the OV pin is set in exactly the same way as in connection with send via the request pin. If an error occurs during send, the job is automatically retried until the value has been sent without error.

Cyclic send

The data are automatically sent after expiration of a configurable cycle time (SCANDOWN). This cycle time is indicated in multiples of the task cycle time in which the block is computed. In this operating mode, an overrun error can occur if the transmission is faster than the response time of the receiver. For setpoints, it is necessary to ensure that an acknowledgement is generated by the receiver which is not sent until the setpoint is accepted. The send block is not ready for transmission again until after this acknowledgement has been received.

Receive blocks

In receive direction, the jobs enter the device module via the interface. The device module selects the correct receive block using the telecontrol address. To this end, during installation the receive blocks pass their parameterized telecontrol addresses to the device module. The device module stores the data received and the receive blocks make the data available at their output pins in connection with the next computation of the user task.

Configuration changes >= Automation Builder 1.1/CBP 2.4

The IEC 60870 protocol allows link-ups between AC500 CPUs with onboard Ethernet (e.g. PM595-4ETH and PM591-2ETH) and external systems.

The link-up takes place via the onboard Ethernet interface of the CPU. As of Automation Builder Version 1.1 telecontrol is also supported for CPUs that provide more than one Ethernet interface (e.g. PM595-4ETH and PM591-2ETH). This allows to use different Ethernet interfaces for IEC 60870 connections, hence, telecontrol configuration is changed. Further, as of this version terminology is aligned with IEC 60870 standard and provides additional features that are described in this chapter. For a description on principle telecontrol configuration.
Control and Substations ≥ CBP 2.4 > Chapter 1.5.6.3.1.2.2 “Control station and substation configuration” on page 3387

Import Export ≥ CBP 2.4 > Chapter 1.5.6.3.1.2.1 “Configuration changes >= Automation Builder 1.1/CBP 2.4” on page 3386

Validity Check of Configuration ≥ CBP 2.4 > Chapter 1.5.6.3.1.2.4 “Validity check of configuration” on page 3407

Control station and substation configuration

The CPU can work as both, control station and substation.

<table>
<thead>
<tr>
<th>Control station</th>
<th>Client, master, controlling station: Synonyms for a higher-level station (central station, monitors other stations)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Control direction</td>
<td>Data transfer direction from the control station to the substation</td>
</tr>
<tr>
<td>Substation</td>
<td>Server, slave, controlled station: synonyms for a subordinate IEC 60870-5-104 telecontrol station (which is monitored)</td>
</tr>
<tr>
<td>Monitoring direction</td>
<td>Data transfer direction from the substation to the controlling station</td>
</tr>
</tbody>
</table>

Configure a control station in the device tree PLC -> Interfaces -> Ethernet -> ETHx:

1. Right-click “ETHx ➔ Add objects”.
2. Select the control station from the list and click “Add object”. Configure substations and further control stations in the same way. As of Automation Builder 1.1 any combination of control stations and substations can be configured, in due consideration of a total number of 10 stations.
3. Double-click the new control station node to open parameter configuration. In the Link Layer tab access to the Ethernet interface is configured.
As of Automation Builder 2.2.1 the V3 PLC telecontrol station objects from GVL IEC60870_5_104_Connection_GVL can only be used in structured text by adding the namespace of the GVL as prefix.

Example:

```
byteValue := IEC60870_5_104_Connection_GVL.IEC_60870_5_104_Control-station.Con;
```

Tab link layer

In order to provide flexible usage of control stations and substations as of Automation Builder 1.1 configuration of substations has been changed. As several substations can be operated with several Ethernet interfaces, select the Ethernet interface to be used from the pull-down menu. Enter the IP address to the control station and if required to another control station (redundant connection). If no IP address is defined, the substation accepts connection to any control station.

This field is not available in the Link Layer tab of control stations. Selection of ETH interface is only possible for substations. The control station is always configured on both interfaces by default.

Timeout settings

T1, T2, T3: The values for the connection control and message replication; timeout1/2/3.

Buffer settings

This parameter gives the maximum number of outstanding messages and acknowledgement behaviour.

Send buffer (k): Maximum difference receive sequence number to send state variable.

Rec buffer (w): Latest acknowledge after receiving w I format APDUs.
Network settings

Network settings are available for control stations and for substations. The IP address of the control station and if available the IP address of another control station (redundant IP address) can be selected by the user.

For an overview on the configured Ethernet interfaces for the control stations and substations, double-click the “Protocols” node.

Tab application layer

Settings

The application layer is the communication layer with which the send and receive blocks work.

![Application layer settings](image)

**Use ACTTERM**

This parameter concerns only setpoints and commands. If this parameter is checked, an acknowledgement with set ‘actterm’ is generated as reason for transmission at the time at which the receive block is computed and outputs its telecontrol data at its output pins. On transmission side, the data block awaits the reception of this ACTTERM acknowledgement and reacts with its corresponding output (see IEC60870 Library) to the reception of this acknowledgement. For commands with execution time, the acknowledgement is generated when the command is terminated, for commands with continuous execution time and for setpoints, the acknowledgement is generated when the data are output to the output pins.

**ForeignAcknowledgement**

If this option is not enabled (default), a message that was sent is considered as ok as soon as transmission was successful. If you enable this option, a message that was sent is not considered as ok until a success message (foreign acknowledge) is returned from the receiver.

**Application timeout**

This time indicates how long an acknowledgement will be awaited on the application level. An acknowledgement is generated only for commands and setpoints on the application level.

**Station address**

The station address defines which station will be subject to a count query. The values define the 2 bytes for the common telecontrol address (Common addr.). The values concerned are as follows:
0: The station address is not used.
1...254: The count is queried on the station defined by the station address.
255: The count is queried on all accessible stations.

**Timesync**

After each new establishment of a link and once per hour, a ‘coarse time synchronisation’ message is generated. This time synchronisation is only supported from AC500 to external systems. Time synchronisation from an external system to AC500 is not provided! Incoming time synchronisation messages are confirmed by the process station but not executed. Greenwich Mean Time (GMT) is used as the time for the synchronisation.

**Send 'Init end' after reconnection**

After each establishment of a link or only in connection with the first establishment of a link and after reconfiguration, an init end message is generated. After the init end message, there is a general inquiry, if configured.

**General inquiry**

**Activated**

This parameter concerns only real values, discrete values and status values. The device module generates a general inquiry message after each new establishment of a link. The other side then generates a message with the reason for transmission ‘general inquiry’ for every data point and subsequently an init end message. This procedure ensures that, in the event of a new establishment of a link, all data are available on the reception side in topical form.

**With parameters**

If general inquiry is activated the parameter values are sent.

**Without integrated totals**

With a general inquiry no integrated total values are sent.

**Counter interrogation**

**Group**

General, 1 .. 4: The count inquiry is executed for a specific group of counters (1 .. 4). The count inquiry is executed for all groups of counters.

**With reset**

The reset quality bit is sent along with the count inquiry.

**With relocate**

The relocate quality bit is sent along with the count inquiry.

**Tab information objects**

Open the ‘Information object’ tab to configure so called Information objects and a Common address (known as 'data points' and 'Global address' in former Automation Builder versions). In this tab different information objects and their services for transmission are defined. A data point or information object is identified via a system-wide unambiguous address containing a maximum 5 bytes.

1. Right-click in the empty view and select ‘Add Information Object with ASDU’ to add a data group. Select the desired object from the list (e.g. M_SP_NA_1).

⇒ An information object with a corresponding ASDU (Application Service Data Unit) is created.
2. Configure the settings in the “Information Object” tab to your convenience.

3. Double-click a table cell to modify pre-set values. For some ASDUs additional sub information objects can be configured. For this, right-click the already existing ASDU and select “Add Information Object” to selected ASDU option. This allows configuration of 16 data points at the most (depending on the ASDU type). With “Remove Information Object” the selected ASDU is deleted.

**Description of the columns**
- **ASDU name**: node name of the information object (name of the ASDU).
- **Data type**: Data type of the ASDU.
- **ASDU type**: Type of ASDU.
- **Common addr**: Common address of the ASDU (known as ‘Global Address’ in former AB Versions). Byte 1/2 of the common telecontrol address of the block (range: 0...255).
- **Info obj addr**: Together with common address Info obj addr defines the endpoint (range: 0...255).
- **Norm start**: Low limit (0 %) of the normalized range for real values and setpoints.
- **Norm end**: High limit (100 %) of the normalized range for real values and setpoints.
- **Threshold**: Threshold limit beyond which a change of the input value referred to.
- **Description**: Table cell for free text. Use this field to describe your configuration settings e.g. differences between configuration variants.

**Format of common addr and info obj addr**

The following address formats of your entries in the columns **Common addr** and **Info obj addr** of the Tab **Information Objects** are possible:
- 1.2 and 3.4.5 (Default format)
- 1-2 and 3-4-5
- 258 or hex 0x102 and 197637 or hex 0x30405
- 513 or hex 0x201 and 328707 or hex 0x50403

Previously you have to choose your preferred address format:
1. Click “Tools” and then “Options...”
   ⇒ The Window Options appears

2. Select IEC 60870-5-104, make your choice and click OK.

Import options of information objects

The User can accept the imported IEC60870 Information Objects as single change or change as block.
IEC60870-5-104 Multiple connections

An AC500 with more than one substation connection must be able to identify the corresponding control station clearly. This identification takes place exclusively via the control station’s IP address. In order to make it possible for a non-redundant control station to have redundant access to a substation with 2 Ethernet connections. The local substation address is ignored during connection establishment.

In the following descriptions, the term station must not be confused with the individual connection. One station can have several connections. An IEC60870-5-104 communication always takes place between a control station and a substation. A control station can manage several substations and also simultaneously be a substation for one or several control stations. However, these must then be realized using different stations.

A PLC may not be configured for another PLC repeatedly as a substation or a control station unless a disjunctive Ethernet infrastructure is used for this.

Redundant connections must be specified as such in the configuration.

An AC500 can be used only once as control station for another AC500, it makes no sense to use the same AC500 repeatedly as a control station for the same substation. Such a structure is configured as a redundant control station as long as only one AC500 exists as a control station per substation. However, this control station may have 2 IP addresses. Therefore, this configuration must either have the IP address 0.0.0.0 entered on the substation for the control station, meaning that all IP addresses are accepted and no other control station can access this AC500 or alternatively the possible control station addresses must be specified (ETH1 and ETH2).
Tree constellation

PM57x/- PM58x/- PM59x-ETH, PM5650-2ETH:

If you plan to control several substations with the AC500, they can be cascaded. This results in a tree structure.

In the following, the notation 192.168.1.0/24 is used for TCP/IP networks. Here, the figure /24 specifies the network mask with 255.255.255.0 and 192.168.1.0 describes the network. The valid addresses for this Class C network are 192.168.1.1 to 192.168.1.254! Only the last byte of the address is provided on the respective devices, with e.g. .10. This means that the respective device has the address 192.168.1.10.
Minimal structure

A control station with an Ethernet interface is connected to a substation with an Ethernet interface.

Configuration at control station

PM57x-/ PM58x-/ PM59x-ETH, PM5650-2ETH:
The respective substation IP address must be specified at the control station. For this, in the network settings of the control station (1) enter the IP address of the substation (in the example: 192.168.1.25). Option "Enable redundant connection" must be disabled.

Configuration at substation

PM57x-/ PM58x-/ PM59x-ETH, PM5650-2ETH:
Either the control station IP address or the general address 0.0.0.0 must be specified at the substation (2). For this, in the network settings of the substation enter the IP address of the control station (in the example: 192.168.1.10). Option "Enable redundant connection" must be disabled.

If the general address 0.0.0.0 is used at the substation, no further control station can be configured on this controller for a further substation.
**Minimal redundancy structure**

The most simple redundant structure with an AC500 consists of a redundant control station (not AC500) which is connected to the AC500 substation with 2 different IP addresses. These redundant control stations must synchronize which control station is active.

Only one control station can be active at any given time.

![Diagram of redundant structure](image)

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>(1)</td>
<td>Control station 1A (Not AC500)</td>
</tr>
<tr>
<td>(2)</td>
<td>Control station 1B (Not AC500)</td>
</tr>
<tr>
<td>(3)</td>
<td>Substation</td>
</tr>
<tr>
<td>(4)</td>
<td>Redundancy link</td>
</tr>
</tbody>
</table>

**Configuration at control stations**  The respective substation IP address must be specified at the control stations 1 and 2 (not AC500). For this, in the network settings of both control stations enter the IP address of the substation (in the example: 192.168.1.25).

**Configuration at substation**  PM57x-/ PM58x-/ PM59x-ETH, PM5650-2ETH:

Either the control station IP addresses or the general address 0.0.0.0 must be specified at the substation (3). For this, in the network settings of the substation enter the IP addresses of the control station (in the example: 192.168.1.10 and 192.168.1.11). Option "Enable redundant connection" must be enabled.
If the general address 0.0.0.0 is used at the substation, no further control station can be configured on this controller for a further substation.

Network redundancy
For network redundancy a control station can reach a substation via 2 paths.
Both the control station and the substation can have 2 different IP addresses. Without special network routing, 2 separate networks should exist, within which both the substation and the control station each have 2 interfaces.
Possible variants of network redundancy are described in the following.

Network redundancy with 2 separate networks

| (1) | Control station with 2 redundant paths |
| (2) | 1 Substation with 2 Ethernet interfaces |

Configuration at control stations
PM591-2ETH, PM595-4ETH, PM5650-2ETH:
The substation’s IP addresses must be specified at the control stations (1). For this, in the network settings of the control station enter the IP addresses of the substation (in the example: 192.168.1.25 and 192.168.2.26). Option “Enable redundant connection” must be enabled.
PM591-2ETH, PM595-4ETH, PM5650-2ETH:
Either the control station's IP addresses or the general address 0.0.0.0 must be specified at the substation (2). For this, in the network settings of the substation enter the IP addresses of the control station (in the example: 192.168.1.10 and 192.168.2.11). Option “Enable redundant connection” must be enabled.

If the general address 0.0.0.0 is used at the substation, no further control station on another substation can be configured on this controller. Equally, the substation connection must be activated for both interfaces.

Network redundancy with 1 network and 2 Ethernet ports in substation

(1) Control station with 2 paths to reach substation
(2) 1 Substation with 2 Ethernet interfaces

PM591-2ETH, PM595-4ETH, PM5650-2ETH:
The substation’s IP addresses must be specified at the control stations (1). For this, in the network settings of the control station enter the IP addresses of the substation (in the example: 192.168.1.25 and 192.168.1.26). Option “Enable redundant connection” must be enabled.
Configuration at substation

PM591-2ETH, PM595-4ETH, PM5650-2ETH:

Either the control station's IP addresses or the general address 0.0.0.0 must be specified at the substation (2). For this, in the network settings of the substation enter the IP addresses of the control station (in the example: 192.168.1.10 and 192.168.2.11). Option “Enable redundant connection” must be enabled.

If the general address 0.0.0.0 is used at the substation, no further control station on another substation can be configured on this controller. Equally, the substation connection must be activated for both interfaces.

Network redundancy with 1 network and 1 Ethernet port in substation

No online redundancy.

Only one connection will be established.

Configuration at control stations

PM591-2ETH, PM595-4ETH, PM5650-2ETH:

The substation's IP addresses must be specified at the control stations (1). For this, in the network settings of the control station enter the IP addresses of the substation (in the example: 192.168.1.25 and 0.0.0.0). Option “Enable redundant connection” must be disabled.
Configuration at substation

PM591-2ETH, PM595-4ETH, PM5650-2ETH:

Either the control station's IP addresses or the general address 0.0.0.0 must be specified at the substation (2). For this, in the network settings of the substation enter the IP addresses of the control station (in the example: 192.168.1.10 and 192.168.2.11). Option "Enable redundant connection" must be enabled.

If the general address 0.0.0.0 is used at the substation, no further control station on another substation can be configured on this controller. Equally, the substation connection must be activated for both interfaces.

Network redundancy with 2 Ethernet ports in substation

![Network redundancy diagram]

(1) Control station with 2 paths to reach substation
(2) 1 Substation with 2 Ethernet interfaces

Configuration at control stations

PM591-2ETH, PM595-4ETH, PM5650-2ETH:

The substation's IP addresses must be specified at the control stations (1). For this, in the network settings of the control station enter the IP addresses of the substation (in the example: 192.168.1.25 and 192.168.1.26). Option "Enable redundant connection" must be enabled.

Configuration at substation

PM591-2ETH, PM595-4ETH, PM5650-2ETH:
Either the control station's IP addresses or the general address 0.0.0.0 must be specified at the substation (2). For this, in the network settings of the substation enter the IP addresses of the control station (in the example: 192.168.1.11 and 0.0.0.0). Option “Enable redundant connection” must be disabled.

If the general address 0.0.0.0 is used at the substation, no further control station on another substation can be configured on this controller. Equally, the substation connection must be activated for both interfaces.

**Full control station redundancy**

A control station can consist of two fully redundant units (not AC500s), which are connected via a redundancy link. These control stations must ensure that only one of them at a time is actively connected to the substation and communicates with it. The inactive control station, however, can establish non-active connection with a substation and monitor it with keep alive packages.

Configuration at control stations

The substation’s IP address must be specified at the control stations (1) (not AC500). For this, in the network settings of the control station enter the IP addresses of the substation (in the example: 192.168.1.25 and 192.168.2.26).

**Configuration at substation**

PM591-2ETH, PM595-4ETH, PM5650-2ETH: Either the control station's IP addresses or the general address 0.0.0.0 must be specified at the substation (2). For this, in the network settings of the substation enter the IP addresses of the control station (in the example: 192.168.1.10 and 192.168.2.11). Option “Enable redundant connection” must be enabled.
If the general address 0.0.0.0 is used at the substation, no further control station on another substation can be configured on this controller. Equally, the substation connection must be activated for both interfaces.

Multiple control stations on the same network

As of firmware version 2.4, an AC500 can be used as a substation for several control stations. For this, the control stations must be distinguished by their IP addresses. Should a control station have more than one IP address (redundancy), both possible IP addresses should also be entered for the allocated substation connection. As a result, even despite being equipped with several Ethernet interfaces, a device can only be one allocated control station at a time for a determined substation. Thus, several substations can be configured for different control stations on a AC500.

PM57x/- PM58x/- PM59x-ETH, PM5650-2ETH:
The substation’s IP address must be specified at the control stations. For this, in the network settings of the control station (1 and 2) enter the IP addresses of the substation (in the example: 192.168.1.25). Option “Enable redundant connection” must be disabled.

PM57x/- PM58x/- PM59x-ETH, PM5650-2ETH:
Both control station's IP addresses must be specified at the substation (3). For this, in the network settings of the substation enter the IP addresses of the control stations (in the example: 192.168.1.10 and 192.168.1.11). Option “Enable redundant connection” must be disabled.

Multiple control stations on different networks

As of firmware version 2.4, an AC500 can have several local Ethernet interfaces which can be used for separate control station connections. For this, a control station must be identified via its IP address. The substation address used locally is not used to distinguish a connection in order to enable a network and therefore route redundancy. On AC500, the acceptance of IEC60870-5-104 connections on an interface can only be prevented.

<table>
<thead>
<tr>
<th>(1) Control station 1</th>
<th>(2) Control station 2</th>
<th>(3) 2 Substations with 2 Ethernet interfaces (2nd port and 2nd connection)</th>
</tr>
</thead>
</table>

Configuration at control stations

PM57x-/ PM58x-/ PM59x-ETH, PM5650-2ETH:
The substation's IP addresses must be specified at the control stations (1 and 2). For this, in the network settings of the control station enter the IP addresses of the substation (in the example: 192.168.1.25 and 192.168.2.26). Option “Enable redundant connection” must be disabled.

Configuration at substations

PM591-ETH, PM595-ETH, PM5650-2ETH:
Both control station's IP addresses must be specified at the substation (3) under both substation connections. For this, in the network settings of the substation enter the IP addresses of the control stations (in the example: 192.168.1.10 and 192.168.2.11). Option “Enable redundant connection” must be disabled.
Double connection

This configuration does work. But it is senseless!

It is possible to configure a double connection between 2 stations using 2 separate networks (at least logically separated sub-networks).

However, such a setup has no advantages via-à-vis the minimal structure right at the start of Chapter 1.5.6.3.1.2.6.1.1 “Minimal structure” on page 3395.

For this setup, connection data must be double configured and double resources are also required at the stations, not providing any advantages whatsoever.

Rather the opposite is true, because such configurations are highly prone to errors.

(1) 2 Control stations with 2 Ethernet interfaces
(2) 2 Substations with 2 Ethernet interfaces
Faulty configuration

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>This configuration does not work!</td>
</tr>
</tbody>
</table>

If an AC500 is configured as a control station, the interface which is used to reach the substation is not defined.

The decision as to which interface is used for this is taken by TCP/IP when running.

It is also dependent on the current network configuration.

Here, the current link status and the order of link recognition may be decisive for the interface to be used.

Such a scenario would not result in stable communication as both substations cannot clearly distinguish the control stations.

Instead, the connection management for a substation will assume that the control station has lost the connection and then establishes a connection.

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2 Control stations with 2 Ethernet interfaces</td>
</tr>
<tr>
<td>2</td>
<td>2 Substations with 2 Ethernet interfaces</td>
</tr>
</tbody>
</table>

Export a CSV file

As an alternative many values can be modified at a time by exporting the configuration to a CSV file. After modifying the file data, import the CSV file \( \Rightarrow \) Chapter 1.5.6.3.1.2.3 “Import/Export functionality” on page 3406.
Import/Export functionality

As of Automation Builder 1.1 (CBP >= 2.4) configuration of control stations and substations can be exported/imported via CSV file. Open the CSV file with a spreadsheet software (e.g. Microsoft Excel) and modify the values within the file to your convenience:

1. Export configuration data: right-click the node of the control station or substation to be exported.

2. Click “Export ➔ IEC 60870-5-104 information objects (CSV)” and store the CSV file to a desired directory.

3. Open the CSV file with a spreadsheet software (e.g. Microsoft Excel) and change the values to your convenience. Added table columns are only accepted after the last column.

4. Import configuration data: right-click the node of the control station or substation that has been exported previously.

5. Click “Import ➔ IEC 60870-5-104 information objects (CSV)” and select the CSV file from the file system. Configuration data is imported.

As of Automation Builder 1.1.1 during file import the project data is compared with the project data that is already available. In order to prevent data from being overwritten inadvertently, you can select the data that shall be imported in the “Project Compare - Differences” window:

Data on the left side of the window refers to already available project data. This data is displayed under “Control station ➔ Information objects” tab. Data on the right side of the window refers to new data that can be imported after your confirmation. Decide whether to import (and overwrite) the data or not.
Data in black color means the existing data and the data to be imported is identical.
Data in red color means the existing data and the data to be imported differ. Decide whether to import the new data (and to overwrite the existing data) or not.
Data in blue color means, the data to be imported is new and will be added to the existing data.
Data that has been confirmed for the import already is displayed in green color (after clicking the [Accept Single] button).

In order to move data from one side of the window to another, select the data and click the [Accept Single] button. Data is highlighted in yellow.

To confirm the import of all new data, click the top entry (here: All: ASDU name - ASDU type - Common addr - ...). Then, click the [Accept Single] button.

Close the "Project Compare - Differences" tab, save your project and confirm the message. The changes are displayed in the “Information objects" tab.

Validity check of configuration

We recommend you to verify the IEC configuration of control stations and substations: Right-click a control station or substation -> Check configuration.
The check will look for the following topics:

- Duplicate addresses.
- Stations without any Information objects.
- ASDU names, which are not unique.

When a check finds errors or incompatibilities this will be reported in a separate messages view at the bottom of the window:

With a double-click on the error line, the part of the configuration with the violation will be opened. Now, you can correct the error.

IEC60870 compatibility list
AC500 V2.4 IEC60870-5-104 Compatibility List

9 Interoperability

This companion standard presents sets of parameters and alternatives from which subsets must be selected to implement particular telecontrol systems. Certain parameter values, such as the choice of "structured" or "unstructured" fields of the INFORMATION OBJECT ADDRESS of ASDUs represent mutually exclusive alternatives. This means that only one value of the defined parameters is admitted per system. Other parameters, such as the listed set of different process information in command and in monitor direction allow the specification of the complete set or subsets, as appropriate for given applications. This clause summarizes the parameters of the previous clauses to facilitate a suitable selection for a specific application. If a system is composed of equipment stemming from different manufacturers, it is necessary that all partners agree on the selected parameters.

The interoperability list is defined as in IEC 60870-5-101 and extended with parameters used in this standard. The text descriptions of parameters which are not applicable to this companion standard are strike-through (corresponding check box is marked black).

NOTE In addition, the full specification of a system may require individual selection of certain parameters for certain parts of the system, such as the individual selection of scaling factors for individually addressable measured values.

The selected parameters should be marked in the white boxes as follows:

- Function or ASDU is not used
- Function or ASDU is used as standardized (default)
- Function or ASDU is used in reverse mode
- Function or ASDU is used in standard and reverse mode

The possible selection (blank, X, R, or B) is specified for each specific clause or parameter.

A black check box indicates that the option cannot be selected in this companion standard.

9.1 System or device
(system-specific parameter, indicate definition of a system or a device by marking one of the following with "X")

- System definition
- Controlling station definition (Master)
- Controlled station definition (Slave)

9.2 Network configuration
(network-specific parameter, all configurations that are used are to be marked "X")

- Point-to-point
- Multiple-point-to-point
- Multipoint
- Multipoint-star
AC500 V2.4 IEC60870-5-104 Compatibility List

9.3 Physical layer
(network-specific parameter, all interfaces and data rates that are used are to be marked "X")

Transmission speed (control direction)

<table>
<thead>
<tr>
<th>Unbalanced interchange Circuit V.24/V.28 Standard</th>
<th>Unbalanced interchange Circuit V.24/V.28 Recommended if &gt;1 200 bit/s</th>
<th>Balanced interchange Circuit X.24/X.27</th>
</tr>
</thead>
<tbody>
<tr>
<td>100 bit/s</td>
<td>2 400 bit/s</td>
<td>2 400 bit/s</td>
</tr>
<tr>
<td>200 bit/s</td>
<td>4 800 bit/s</td>
<td>4 800 bit/s</td>
</tr>
<tr>
<td>300 bit/s</td>
<td>9 600 bit/s</td>
<td>9 600 bit/s</td>
</tr>
<tr>
<td>600 bit/s</td>
<td></td>
<td>19 200 bit/s</td>
</tr>
<tr>
<td>1 200 bit/s</td>
<td></td>
<td>38 400 bit/s</td>
</tr>
</tbody>
</table>

Transmission speed (monitor direction)

<table>
<thead>
<tr>
<th>Unbalanced interchange Circuit V.24/V.28 Standard</th>
<th>Unbalanced interchange Circuit V.24/V.28 Recommended if &gt;1 200 bit/s</th>
<th>Balanced interchange Circuit X.24/X.27</th>
</tr>
</thead>
<tbody>
<tr>
<td>100 bit/s</td>
<td>2 400 bit/s</td>
<td>2 400 bit/s</td>
</tr>
<tr>
<td>200 bit/s</td>
<td>4 800 bit/s</td>
<td>4 800 bit/s</td>
</tr>
<tr>
<td>300 bit/s</td>
<td>9 600 bit/s</td>
<td>9 600 bit/s</td>
</tr>
<tr>
<td>600 bit/s</td>
<td></td>
<td>19 200 bit/s</td>
</tr>
<tr>
<td>1 200 bit/s</td>
<td></td>
<td>38 400 bit/s</td>
</tr>
</tbody>
</table>

9.4 Link layer
(network-specific parameter, all options that are used are to be marked "X". Specify the maximum frame length. If a non-standard assignment of class 2 messages is implemented for unbalanced transmission, indicate the Type ID and COT of all messages assigned to class 2.)

Frame format FT 1.2, single character 1 and the fixed time out interval are used exclusively in this companion standard.

Link transmission

- Balanced transmission
- Unbalanced transmission

Frame length

- Maximum length L (number of octets)

Address field of the link

- not present (balanced transmission only)
- One octet
- Two-octets
- Structured
- Unstructured

- 2 -
AC500 V2.4 IEC60870-5-104 Compatibility List

When using an unbalanced link layer, the following ASDU types are returned in class 2 messages (low priority) with the indicated causes of transmission:

- The standard assignment of ASDUs to class 2 messages is used as follows:

<table>
<thead>
<tr>
<th>Type identification</th>
<th>Cause of transmission</th>
</tr>
</thead>
<tbody>
<tr>
<td>9, 11, 13, 21</td>
<td>&lt;1&gt;</td>
</tr>
</tbody>
</table>

- A special assignment of ASDUs to class 2 messages is used as follows:

<table>
<thead>
<tr>
<th>Type identification</th>
<th>Cause of transmission</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Note: (In response to a class 2 poll, a controlled station may respond with class 1 data when there is no class 2 data available).

9.5 Application layer

Transmission mode for application data

Mode 1 (Least significant octet first), as defined in 4.10 of IEC 60870-5-4, is used exclusively in this companion standard.

Common address of ASDU

(system-specific parameter, all configurations that are used are to be marked "X")

- One octet  
  X Two octets

Information object address

(system-specific parameter, all configurations that are used are to be marked "X")

- One octet  
  Structured

- Two octets  
  Unstructured

X Three octets

Cause of transmission

(system-specific parameter, all configurations that are used are to be marked "X")

- One octet  
  Two octets (with originator address). Originator address is set to zero if not used

X Two octets

Length of APDU

(system-specific parameter, specify the maximum length of the APDU per system)

The maximum length of APDU for both directions is 253. It is a fixed system parameter.

X Maximum length of APDU per system in control direction
AC500 V2.4 IEC60870-5-104 Compatibility List

Selection of standard ASDUs

Process information in monitor direction
(station-specific parameter, mark each Type ID "X" if it is only used in the standard direction, "R" if only used in the reverse direction, and "B" if used in both directions).

<table>
<thead>
<tr>
<th>Type ID</th>
<th>Description</th>
<th>ASDU ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;1&gt;</td>
<td>Single-point information</td>
<td>M_SP_NA_1</td>
</tr>
<tr>
<td>&lt;2&gt;</td>
<td>Single-point information with time tag</td>
<td>M_SP_TA_1</td>
</tr>
<tr>
<td>&lt;3&gt;</td>
<td>Double-point information</td>
<td>M_DP_NA_1</td>
</tr>
<tr>
<td>&lt;4&gt;</td>
<td>Double-point information with time tag</td>
<td>M_DP_TA_1</td>
</tr>
<tr>
<td>&lt;5&gt;</td>
<td>Step position information</td>
<td>M_ST_NA_1</td>
</tr>
<tr>
<td>&lt;6&gt;</td>
<td>Step position information with time tag</td>
<td>M_ST_TA_1</td>
</tr>
<tr>
<td>&lt;7&gt;</td>
<td>Bitstring of 32 bit</td>
<td>M_BO_NA_1</td>
</tr>
<tr>
<td>&lt;8&gt;</td>
<td>Bitstring of 32 bit with time tag</td>
<td>M_BO_TA_1</td>
</tr>
<tr>
<td>&lt;9&gt;</td>
<td>Measured value, normalized value</td>
<td>M_ME_NA_1</td>
</tr>
<tr>
<td>&lt;10&gt;</td>
<td>Measured value, normalized value with time tag</td>
<td>M_ME_TA_1</td>
</tr>
<tr>
<td>&lt;11&gt;</td>
<td>Measured value, scaled value</td>
<td>M_ME_NB_1</td>
</tr>
<tr>
<td>&lt;12&gt;</td>
<td>Measured value, scaled value with time tag</td>
<td>M_ME_TB_1</td>
</tr>
<tr>
<td>&lt;13&gt;</td>
<td>Measured value, short floating point value</td>
<td>M_ME_NC_1</td>
</tr>
<tr>
<td>&lt;14&gt;</td>
<td>Measured value, short floating point value with time tag</td>
<td>M_ME_TC_1</td>
</tr>
<tr>
<td>&lt;15&gt;</td>
<td>Integrated totals</td>
<td>M_IT_NA_1</td>
</tr>
<tr>
<td>&lt;16&gt;</td>
<td>Integrated totals with time tag</td>
<td>M_IT_TA_1</td>
</tr>
<tr>
<td>&lt;17&gt;</td>
<td>Event of protection equipment with time tag</td>
<td>M_EP_NA_1</td>
</tr>
<tr>
<td>&lt;18&gt;</td>
<td>Packed start events of protection equipment with time tag</td>
<td>M_EP_TB_1</td>
</tr>
<tr>
<td>&lt;19&gt;</td>
<td>Packed output circuit information of protection equipment with time tag</td>
<td>M_EP_TC_1</td>
</tr>
<tr>
<td>&lt;20&gt;</td>
<td>Packed single-point information with status change detection</td>
<td>M_SP_TA_1</td>
</tr>
<tr>
<td>&lt;21&gt;</td>
<td>Measured value, normalized value without quality descriptor</td>
<td>M_ME_ND_1</td>
</tr>
<tr>
<td>&lt;30&gt;</td>
<td>Single-point information with time tag CP56Time2a</td>
<td>M_SP_TB_1</td>
</tr>
<tr>
<td>&lt;31&gt;</td>
<td>Double-point information with time tag CP56Time2a</td>
<td>M_DP_TB_1</td>
</tr>
<tr>
<td>&lt;32&gt;</td>
<td>Step position information with time tag CP56Time2a</td>
<td>M_ST_TB_1</td>
</tr>
<tr>
<td>&lt;33&gt;</td>
<td>Bitstring of 32 bit with time tag CP56Time2a</td>
<td>M_BO_TB_1</td>
</tr>
<tr>
<td>&lt;34&gt;</td>
<td>Measured value, normalized value with time tag CP56Time2a</td>
<td>M_ME_TD_1</td>
</tr>
<tr>
<td>&lt;35&gt;</td>
<td>Measured value, scaled value with time tag CP56Time2a</td>
<td>M_ME_TE_1</td>
</tr>
<tr>
<td>&lt;36&gt;</td>
<td>Measured value, short floating point value with time tag CP56Time2a</td>
<td>M_ME_TF_1</td>
</tr>
<tr>
<td>&lt;37&gt;</td>
<td>Integrated totals with time tag CP56Time2a</td>
<td>M_IT_TB_1</td>
</tr>
<tr>
<td>&lt;38&gt;</td>
<td>Event of protection equipment with time tag CP56Time2a</td>
<td>M_EP_TD_1</td>
</tr>
<tr>
<td>&lt;39&gt;</td>
<td>Packed start events of protection equipment with time tag CP56Time2a</td>
<td>M_EP_TE_1</td>
</tr>
<tr>
<td>&lt;40&gt;</td>
<td>Packed output circuit information of protection equipment with time tag CP56Time2a</td>
<td>M_EP_TF_1</td>
</tr>
</tbody>
</table>

In this companion standard only the use of the set <30> – <40> for ASDUs with time tag is permitted.

- 4 -
### AC500 V2.4 IEC60870-5-104 Compatibility List

#### Process information in control direction
(station-specific parameter, mark each Type ID "X" if it is only used in the standard direction, "R" if only used in the reverse direction, and "B" if used in both directions).

<table>
<thead>
<tr>
<th>Type ID</th>
<th>Description</th>
<th>ASDU Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>☒ &lt;45&gt;</td>
<td>Single command</td>
<td>C_SC_NA_1</td>
</tr>
<tr>
<td>☒ &lt;46&gt;</td>
<td>Double command</td>
<td>C_DC_NA_1</td>
</tr>
<tr>
<td>☒ &lt;47&gt;</td>
<td>Regulating step command</td>
<td>C_RC_NA_1</td>
</tr>
<tr>
<td>☒ &lt;48&gt;</td>
<td>Set point command, normalized value</td>
<td>C_SE_NA_1</td>
</tr>
<tr>
<td>☒ &lt;49&gt;</td>
<td>Set point command, scaled value</td>
<td>C_SE_NB_1</td>
</tr>
<tr>
<td>☒ &lt;50&gt;</td>
<td>Set point command, short floating point value</td>
<td>C_SE_NC_1</td>
</tr>
<tr>
<td>☒ &lt;51&gt;</td>
<td>Bitstring of 32 bit</td>
<td>C_BO_NA_1</td>
</tr>
<tr>
<td>☒ &lt;58&gt;</td>
<td>Single command with time tag CP56Time2a</td>
<td>C_SC_TA_1</td>
</tr>
<tr>
<td>☒ &lt;59&gt;</td>
<td>Double command with time tag CP56Time2a</td>
<td>C_DC_TA_1</td>
</tr>
<tr>
<td>☒ &lt;60&gt;</td>
<td>Regulating step command with time tag CP56Time2a</td>
<td>C_RC_TA_1</td>
</tr>
<tr>
<td>☒ &lt;61&gt;</td>
<td>Set point command, normalized value with time tag CP56Time2a</td>
<td>C_SE_TA_1</td>
</tr>
<tr>
<td>☒ &lt;62&gt;</td>
<td>Set point command, scaled value with time tag CP56Time2a</td>
<td>C_SE_TB_1</td>
</tr>
<tr>
<td>☒ &lt;63&gt;</td>
<td>Set point command, short floating point value with time tag CP56Time2a</td>
<td>C_SE_TC_1</td>
</tr>
<tr>
<td>☒ &lt;64&gt;</td>
<td>Bitstring of 32 bit with time tag CP56Time2a</td>
<td>C_BO_TA_1</td>
</tr>
</tbody>
</table>

Either the ASDUs of the set <45> – <51> or of the set <58> – <64> are used.

#### System information in monitor direction
(station-specific parameter, mark with an "X" if it is only used in the standard direction, "R" if only used in the reverse direction, and "B" if used in both directions).

<table>
<thead>
<tr>
<th>Type ID</th>
<th>Description</th>
<th>ASDU Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>☒ &lt;70&gt;</td>
<td>End of initialization</td>
<td>M_EI_NA_1</td>
</tr>
</tbody>
</table>

#### System information in control direction
(station-specific parameter, mark each Type ID "X" if it is only used in the standard direction, "R" if only used in the reverse direction, and "B" if used in both directions).

<table>
<thead>
<tr>
<th>Type ID</th>
<th>Description</th>
<th>ASDU Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>☒ &lt;100&gt;</td>
<td>Interrogation command</td>
<td>C_IC_NA_1</td>
</tr>
<tr>
<td>☒ &lt;101&gt;</td>
<td>Counter interrogation command</td>
<td>C_CI_NA_1</td>
</tr>
<tr>
<td>☒ &lt;102&gt;</td>
<td>Read command</td>
<td>C_RD_NA_1</td>
</tr>
<tr>
<td>☒ &lt;103&gt;</td>
<td>Clock synchronization command (option see 7.6)</td>
<td>C_CS_NA_1</td>
</tr>
<tr>
<td>☒ &lt;104&gt;</td>
<td>Test command</td>
<td>C_TS_NA_1</td>
</tr>
<tr>
<td>☒ &lt;105&gt;</td>
<td>Reset process command</td>
<td>C_RP_NA_1</td>
</tr>
<tr>
<td>☒ &lt;106&gt;</td>
<td>Delay acquisition command</td>
<td>C_CD_NA_1</td>
</tr>
<tr>
<td>☒ &lt;107&gt;</td>
<td>Test command with time tag CP56Time2a</td>
<td>C_TS_TA_1</td>
</tr>
</tbody>
</table>
AC500 V2.4 IEC60870-5-104 Compatibility List

Parameter in control direction
(station-specific parameter, mark each Type ID "X" if it is only used in the standard direction, "R" if only used in the reverse direction, and "B" if used in both directions).

[X] <110>: Parameter of measured value, normalized value P_ME_NA_1
[X] <111>: Parameter of measured value, scaled value P_ME_NB_1
[X] <112>: Parameter of measured value, short floating point value P_ME_NC_1
[X] <113>: Parameter activation P_AC_NA_1

File transfer
(station-specific parameter, mark each Type ID "X" if it is only used in the standard direction, "R" if only used in the reverse direction, and "B" if used in both directions).

[X] <120>: File ready F_FR_NA_1
[X] <121>: Section ready F_SR_NA_1
[X] <122>: Call directory, select file, call file, call section F_SC_NA_1
[X] <123>: Last section, last segment F_LS_NA_1
[X] <124>: Ack file, ack section F_AF_NA_1
[X] <125>: Segment F_SG_NA_1
[X] <126>: Directory (blank or X, only available in monitor (standard) direction) F_DR_TA_1
[X] <127>: Query Log – Request archive file F_SC_NB_1

Type identifier and cause of transmission assignments
(station-specific parameters)

Shaded boxes: option not required.
Black boxes: option not permitted in this companion standard
Blank: functions or ASDU not used.
Mark Type Identification/Cause of transmission combinations:
"X" if only used in the standard direction;
"R" if only used in the reverse direction;
"B" if used in both directions.

<table>
<thead>
<tr>
<th>Type identification</th>
<th>Cause of transmission</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1 2 3 4 5 6 7 8 9 10 11 12 13 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47</td>
</tr>
<tr>
<td>&lt;1&gt; M_SP_NA_1</td>
<td>X X X X</td>
</tr>
<tr>
<td>&lt;2&gt; M_SP_TA_1</td>
<td>X X X X</td>
</tr>
<tr>
<td>&lt;3&gt; M_DP_NA_1</td>
<td>X X X X</td>
</tr>
<tr>
<td>&lt;4&gt; M_DP_TA_1</td>
<td>X X X X</td>
</tr>
<tr>
<td>&lt;5&gt; M_ST_NA_1</td>
<td>X X X X</td>
</tr>
<tr>
<td>&lt;6&gt; M_ST_TA_1</td>
<td>X X X X</td>
</tr>
<tr>
<td>&lt;7&gt; M_BO_NA_1</td>
<td>X X X X</td>
</tr>
<tr>
<td>&lt;8&gt; M_BO_TA_1</td>
<td>X X X X</td>
</tr>
<tr>
<td>&lt;9&gt; M_ME_NA_1</td>
<td>X X X X</td>
</tr>
<tr>
<td>&lt;10&gt; M_ME_TA_1</td>
<td>X X X X</td>
</tr>
<tr>
<td>&lt;11&gt; M_ME_NB_1</td>
<td>X X X X</td>
</tr>
<tr>
<td>&lt;12&gt; M_ME_TB_1</td>
<td>X X X X</td>
</tr>
</tbody>
</table>
### PLC Automation with V3 CPUs

PLC integration > Configuration in Automation Builder for AC500 V3 products

<table>
<thead>
<tr>
<th>Type identification</th>
<th>Cause of transmission</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 2 3 4 5 6 7 8 9 10 11 12 13 20 to 36 37 to 41 44 45 46 47</td>
<td></td>
</tr>
<tr>
<td>&lt;13&gt; M_ME_NC_1</td>
<td>x x x x</td>
</tr>
<tr>
<td>&lt;14&gt; M_ME_TC_1</td>
<td></td>
</tr>
<tr>
<td>&lt;15&gt; M_IT NA_1</td>
<td>x x</td>
</tr>
<tr>
<td>&lt;16&gt; M_IT TA_1</td>
<td></td>
</tr>
<tr>
<td>&lt;17&gt; M_EP_TA_1*</td>
<td></td>
</tr>
<tr>
<td>&lt;18&gt; M_EP_TB_1</td>
<td></td>
</tr>
<tr>
<td>&lt;19&gt; M_PS_NA_1</td>
<td></td>
</tr>
<tr>
<td>&lt;20&gt; M_ME_ND_1</td>
<td></td>
</tr>
<tr>
<td>&lt;21&gt; M_ME_TC_1</td>
<td></td>
</tr>
<tr>
<td>&lt;22&gt; M_IT_NA_1</td>
<td>x</td>
</tr>
<tr>
<td>&lt;23&gt; M_IT_TA_1</td>
<td></td>
</tr>
<tr>
<td>&lt;24&gt; M_EP_TA_1</td>
<td></td>
</tr>
<tr>
<td>&lt;25&gt; M_EP_TB_1</td>
<td></td>
</tr>
<tr>
<td>&lt;26&gt; M_IT_NA_1</td>
<td>x</td>
</tr>
<tr>
<td>&lt;27&gt; M_IT_TA_1</td>
<td></td>
</tr>
<tr>
<td>&lt;28&gt; M_EP_TA_1</td>
<td></td>
</tr>
<tr>
<td>&lt;29&gt; M_EP_TB_1</td>
<td></td>
</tr>
<tr>
<td>&lt;30&gt; M_ME_NC_1</td>
<td>x</td>
</tr>
<tr>
<td>&lt;31&gt; M_ME_TC_1</td>
<td></td>
</tr>
<tr>
<td>&lt;32&gt; M_IT_NA_1</td>
<td>x</td>
</tr>
<tr>
<td>&lt;33&gt; M_IT_TA_1</td>
<td></td>
</tr>
<tr>
<td>&lt;34&gt; M_EP_TA_1</td>
<td></td>
</tr>
<tr>
<td>&lt;35&gt; M_EP_TB_1</td>
<td></td>
</tr>
<tr>
<td>&lt;36&gt; M_ME_NC_1</td>
<td>x</td>
</tr>
<tr>
<td>&lt;37&gt; M_ME_TC_1</td>
<td></td>
</tr>
<tr>
<td>&lt;38&gt; M_IT_NA_1</td>
<td>x</td>
</tr>
<tr>
<td>&lt;39&gt; M_IT_TA_1</td>
<td></td>
</tr>
<tr>
<td>&lt;40&gt; M_EP_TA_1</td>
<td></td>
</tr>
<tr>
<td>&lt;41&gt; M_EP_TB_1</td>
<td></td>
</tr>
<tr>
<td>&lt;42&gt; M_ME_NC_1</td>
<td>x</td>
</tr>
<tr>
<td>&lt;43&gt; M_ME_TC_1</td>
<td></td>
</tr>
<tr>
<td>&lt;44&gt; M_IT_NA_1</td>
<td>x</td>
</tr>
<tr>
<td>&lt;45&gt; M_IT_TA_1</td>
<td></td>
</tr>
<tr>
<td>&lt;46&gt; M_EP_TA_1</td>
<td></td>
</tr>
<tr>
<td>&lt;47&gt; M_EP_TB_1</td>
<td></td>
</tr>
<tr>
<td>&lt;48&gt; M_ME_NC_1</td>
<td>x</td>
</tr>
<tr>
<td>&lt;49&gt; M_ME_TC_1</td>
<td></td>
</tr>
<tr>
<td>&lt;50&gt; M_IT_NA_1</td>
<td>x</td>
</tr>
<tr>
<td>&lt;51&gt; M_IT_TA_1</td>
<td></td>
</tr>
<tr>
<td>&lt;52&gt; M_EP_TA_1</td>
<td></td>
</tr>
<tr>
<td>&lt;53&gt; M_EP_TB_1</td>
<td></td>
</tr>
<tr>
<td>&lt;54&gt; M_ME_NC_1</td>
<td>x</td>
</tr>
<tr>
<td>&lt;55&gt; M_ME_TC_1</td>
<td></td>
</tr>
<tr>
<td>&lt;56&gt; M_IT_NA_1</td>
<td>x</td>
</tr>
<tr>
<td>&lt;57&gt; M_IT_TA_1</td>
<td></td>
</tr>
<tr>
<td>&lt;58&gt; M_EP_TA_1</td>
<td></td>
</tr>
<tr>
<td>&lt;59&gt; M_EP_TB_1</td>
<td></td>
</tr>
<tr>
<td>&lt;60&gt; M_ME_NC_1</td>
<td>x</td>
</tr>
<tr>
<td>&lt;61&gt; M_ME_TC_1</td>
<td></td>
</tr>
<tr>
<td>&lt;62&gt; M_IT_NA_1</td>
<td>x</td>
</tr>
<tr>
<td>&lt;63&gt; M_IT_TA_1</td>
<td></td>
</tr>
<tr>
<td>&lt;64&gt; M_EP_TA_1</td>
<td></td>
</tr>
<tr>
<td>&lt;65&gt; M_EP_TB_1</td>
<td></td>
</tr>
<tr>
<td>&lt;66&gt; M_ME_NC_1</td>
<td>x</td>
</tr>
<tr>
<td>&lt;67&gt; M_ME_TC_1</td>
<td></td>
</tr>
<tr>
<td>&lt;68&gt; M_IT_NA_1</td>
<td>x</td>
</tr>
<tr>
<td>&lt;69&gt; M_IT_TA_1</td>
<td></td>
</tr>
<tr>
<td>&lt;70&gt; M_EP_TA_1</td>
<td></td>
</tr>
<tr>
<td>&lt;71&gt; M_EP_TB_1</td>
<td></td>
</tr>
<tr>
<td>&lt;72&gt; M_ME_NC_1</td>
<td>x</td>
</tr>
<tr>
<td>&lt;73&gt; M_ME_TC_1</td>
<td></td>
</tr>
<tr>
<td>&lt;74&gt; M_IT_NA_1</td>
<td>x</td>
</tr>
<tr>
<td>&lt;75&gt; M_IT_TA_1</td>
<td></td>
</tr>
<tr>
<td>&lt;76&gt; M_EP_TA_1</td>
<td></td>
</tr>
<tr>
<td>&lt;77&gt; M_EP_TB_1</td>
<td></td>
</tr>
<tr>
<td>&lt;78&gt; M_ME_NC_1</td>
<td>x</td>
</tr>
<tr>
<td>&lt;79&gt; M_ME_TC_1</td>
<td></td>
</tr>
<tr>
<td>&lt;80&gt; M_IT_NA_1</td>
<td>x</td>
</tr>
<tr>
<td>&lt;81&gt; M_IT_TA_1</td>
<td></td>
</tr>
<tr>
<td>&lt;82&gt; M_EP_TA_1</td>
<td></td>
</tr>
<tr>
<td>&lt;83&gt; M_EP_TB_1</td>
<td></td>
</tr>
<tr>
<td>&lt;84&gt; M_ME_NC_1</td>
<td>x</td>
</tr>
<tr>
<td>&lt;85&gt; M_ME_TC_1</td>
<td></td>
</tr>
<tr>
<td>&lt;86&gt; M_IT_NA_1</td>
<td>x</td>
</tr>
<tr>
<td>&lt;87&gt; M_IT_TA_1</td>
<td></td>
</tr>
<tr>
<td>&lt;88&gt; M_EP_TA_1</td>
<td></td>
</tr>
<tr>
<td>&lt;89&gt; M_EP_TB_1</td>
<td></td>
</tr>
<tr>
<td>&lt;90&gt; M_ME_NC_1</td>
<td>x</td>
</tr>
<tr>
<td>&lt;91&gt; M_ME_TC_1</td>
<td></td>
</tr>
<tr>
<td>&lt;92&gt; M_IT_NA_1</td>
<td>x</td>
</tr>
<tr>
<td>&lt;93&gt; M_IT_TA_1</td>
<td></td>
</tr>
<tr>
<td>&lt;94&gt; M_EP_TA_1</td>
<td></td>
</tr>
<tr>
<td>&lt;95&gt; M_EP_TB_1</td>
<td></td>
</tr>
<tr>
<td>&lt;96&gt; M_ME_NC_1</td>
<td>x</td>
</tr>
<tr>
<td>&lt;97&gt; M_ME_TC_1</td>
<td></td>
</tr>
<tr>
<td>&lt;98&gt; M_IT_NA_1</td>
<td>x</td>
</tr>
<tr>
<td>&lt;99&gt; M_IT_TA_1</td>
<td></td>
</tr>
<tr>
<td>&lt;100&gt; M_EP_TA_1</td>
<td></td>
</tr>
<tr>
<td>&lt;101&gt; M_EP_TB_1</td>
<td></td>
</tr>
<tr>
<td>&lt;102&gt; M_ME_NC_1</td>
<td>x</td>
</tr>
<tr>
<td>&lt;103&gt; M_ME_TC_1</td>
<td></td>
</tr>
<tr>
<td>&lt;104&gt; M_IT_NA_1</td>
<td>x</td>
</tr>
<tr>
<td>&lt;105&gt; M_IT_TA_1</td>
<td></td>
</tr>
<tr>
<td>&lt;106&gt; M_EP_TA_1</td>
<td></td>
</tr>
<tr>
<td>&lt;107&gt; M_EP_TB_1</td>
<td></td>
</tr>
<tr>
<td>&lt;108&gt; M_ME_NC_1</td>
<td>x</td>
</tr>
<tr>
<td>&lt;109&gt; M_ME_TC_1</td>
<td></td>
</tr>
<tr>
<td>&lt;110&gt; M_IT_NA_1</td>
<td>x</td>
</tr>
<tr>
<td>&lt;111&gt; M_IT_TA_1</td>
<td></td>
</tr>
<tr>
<td>&lt;112&gt; M_EP_TA_1</td>
<td></td>
</tr>
<tr>
<td>&lt;113&gt; M_EP_TB_1</td>
<td></td>
</tr>
<tr>
<td>&lt;114&gt; M_ME_NC_1</td>
<td>x</td>
</tr>
<tr>
<td>&lt;115&gt; M_ME_TC_1</td>
<td></td>
</tr>
<tr>
<td>&lt;116&gt; M_IT_NA_1</td>
<td>x</td>
</tr>
<tr>
<td>&lt;117&gt; M_IT_TA_1</td>
<td></td>
</tr>
<tr>
<td>&lt;118&gt; M_EP_TA_1</td>
<td></td>
</tr>
<tr>
<td>&lt;119&gt; M_EP_TB_1</td>
<td></td>
</tr>
<tr>
<td>&lt;120&gt; M_ME_NC_1</td>
<td>x</td>
</tr>
<tr>
<td>&lt;121&gt; M_ME_TC_1</td>
<td></td>
</tr>
<tr>
<td>&lt;122&gt; M_IT_NA_1</td>
<td>x</td>
</tr>
<tr>
<td>&lt;123&gt; M_IT_TA_1</td>
<td></td>
</tr>
<tr>
<td>&lt;124&gt; M_EP_TA_1</td>
<td></td>
</tr>
<tr>
<td>&lt;125&gt; M_EP_TB_1</td>
<td></td>
</tr>
<tr>
<td>&lt;126&gt; M_ME_NC_1</td>
<td>x</td>
</tr>
<tr>
<td>&lt;127&gt; M_ME_TC_1</td>
<td></td>
</tr>
<tr>
<td>&lt;128&gt; M_IT_NA_1</td>
<td>x</td>
</tr>
<tr>
<td>&lt;129&gt; M_IT_TA_1</td>
<td></td>
</tr>
<tr>
<td>&lt;130&gt; M_EP_TA_1</td>
<td></td>
</tr>
<tr>
<td>&lt;131&gt; M_EP_TB_1</td>
<td></td>
</tr>
<tr>
<td>&lt;132&gt; M_ME_NC_1</td>
<td>x</td>
</tr>
<tr>
<td>&lt;133&gt; M_ME_TC_1</td>
<td></td>
</tr>
<tr>
<td>&lt;134&gt; M_IT_NA_1</td>
<td>x</td>
</tr>
<tr>
<td>&lt;135&gt; M_IT_TA_1</td>
<td></td>
</tr>
<tr>
<td>&lt;136&gt; M_EP_TA_1</td>
<td></td>
</tr>
<tr>
<td>&lt;137&gt; M_EP_TB_1</td>
<td></td>
</tr>
<tr>
<td>&lt;138&gt; M_ME_NC_1</td>
<td>x</td>
</tr>
<tr>
<td>&lt;139&gt; M_ME_TC_1</td>
<td></td>
</tr>
<tr>
<td>&lt;140&gt; M_IT_NA_1</td>
<td>x</td>
</tr>
<tr>
<td>&lt;141&gt; M_IT_TA_1</td>
<td></td>
</tr>
<tr>
<td>&lt;142&gt; M_EP_TA_1</td>
<td></td>
</tr>
<tr>
<td>&lt;143&gt; M_EP_TB_1</td>
<td></td>
</tr>
<tr>
<td>&lt;144&gt; M_ME_NC_1</td>
<td>x</td>
</tr>
<tr>
<td>&lt;145&gt; M_ME_TC_1</td>
<td></td>
</tr>
<tr>
<td>&lt;146&gt; M_IT_NA_1</td>
<td>x</td>
</tr>
<tr>
<td>&lt;147&gt; M_IT_TA_1</td>
<td></td>
</tr>
<tr>
<td>&lt;148&gt; M_EP_TA_1</td>
<td></td>
</tr>
<tr>
<td>&lt;149&gt; M_EP_TB_1</td>
<td></td>
</tr>
</tbody>
</table>
AC500 V2.4 IEC60870-5-104 Compatibility List

<table>
<thead>
<tr>
<th>Type identification</th>
<th>Cause of transmission</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1 2 3 4 5 6 7 8 9 10 11 12 13 20 to 36 37 to 41</td>
</tr>
<tr>
<td>&lt;127&gt; F_SC_NB_1*</td>
<td></td>
</tr>
</tbody>
</table>

* Blank or X only
9.6 Basic application functions

Station initialization
(station-specific parameter, mark “X” if function is used)

Remote initialization

Cyclic data transmission
(station-specific parameter, mark “X” if function is only used in the standard direction, “R” if only used in the reverse direction, and “B” if used in both directions)

Cyclic data transmission

Read procedure
(station-specific parameter, mark “X” if function is only used in the standard direction, “R” if only used in the reverse direction, and “B” if used in both directions)

Read procedure

Spontaneous transmission
(station-specific parameter, mark “X” if function is only used in the standard direction, “R” if only used in the reverse direction, and “B” if used in both directions)

Spontaneous transmission

Double transmission of information objects with cause of transmission spontaneous
(station-specific parameter, mark each information type “X” where both a Type ID without time and corresponding Type ID with time are issued in response to a single spontaneous change of a monitored object)

The following type identifications may be transmitted in succession caused by a single status change of an information object. The particular information object addresses for which double transmission is enabled are defined in a project-specific list.

- Single-point information M_SP_NA_1, M_SP_TA_1, M_SP_TB_1 and M_PS_NA_1
- Double-point information M_DP_NA_1, M_DP_TA_1 and M_DP_TB_1
- Step position information M_ST_NA_1, M_ST_TA_1 and M_ST_TB_1
- Bitstring of 32 bit M_BO_NA_1, M_BO_TA_1 and M_BO_TB_1 (if defined for a specific project)
- Measured value, normalized value M_ME_NA_1, M_ME_TA_1, M_ME_ND_1 and M_ME_TD_1
- Measured value, scaled value M_ME_NB_1, M_ME_TB_1 and M_ME_TE_1
- Measured value, short floating point number M_ME_NC_1, M_ME_TC_1 and M_ME_TF_1
AC500 V2.4 IEC60870-5-104 Compatibility List

**Station interrogation**
(station-specific parameter, mark "X" if function is only used in the standard direction, "R" if only used in the reverse direction, and "B" if used in both directions).

- X global
- group 1
- group 2
- group 3
- group 4
- group 5
- group 6
- group 7
- group 8
- group 9
- group 10
- group 11
- group 12
- group 13
- group 14
- group 15
- group 16

Information object addresses assigned to each group must be shown in a separate table.

**Clock synchronization**
(station-specific parameter, mark "X" if function is only used in the standard direction, "R" if only used in the reverse direction, and "B" if used in both directions).

- X Clock synchronization
- Day of week used
- RES1, GEN (time tag substituted/ not substituted) used
- SU-bit (summertime) used

optional, see 7.6

**Command transmission**
(object-specific parameter, mark "X" if function is only used in the standard direction, "R" if only used in the reverse direction, and "B" if used in both directions).

- X Direct command transmission
- X Direct set point command transmission
- Select and execute command
- Select and execute set point command
- C_SE ACTTERM used

- No additional definition

- X Short-pulse duration (duration determined by a system parameter in the outstation)

- X Long-pulse duration (duration determined by a system parameter in the outstation)

- X Persistent output

- Supervision of maximum delay in command direction of commands and set point commands

- Maximum allowable delay of commands and set point commands
Transmission of integrated totals
(station- or object-specific parameter, mark "X" if function is only used in the standard direction, "R" if only used in the reverse direction, and "B" if used in both directions).

- Mode A: Local freeze with spontaneous transmission
- Mode B: Local freeze with counter interrogation
- Mode C: Freeze and transmit by counter-interrogation commands
- Mode D: Freeze by counter-interrogation command, frozen values reported

- Counter read
- Counter freeze without reset
- Counter freeze with reset
- Counter reset

- General request
- Request counter group 1
- Request counter group
- Request counter group 3
- Request counter group 4

Parameter loading
(object-specific parameter, mark "X" if function is only used in the standard direction, "R" if only used in the reverse direction, and "B" if used in both directions).

- Threshold value
- Smoothing factor
- Low limit for transmission of measured values
- High limit for transmission of measured values

Parameter activation
(object-specific parameter, mark "X" if function is only used in the standard direction, "R" if only used in the reverse direction, and "B" if used in both directions).

- Act/deact of persistent cyclic or periodic transmission of the addressed object

Test procedure
(station-specific parameter, mark "X" if function is only used in the standard direction, "R" if only used in the reverse direction, and "B" if used in both directions).

- Test procedure
AC500 V2.4 IEC60870-5-104 Compatibility List

File transfer
(station-specific parameter, mark "X" if function is used).
File transfer in monitor direction

☐ Transparent file
☐ Transmission of disturbance data of protection equipment
☐ Transmission of sequences of events
☐ Transmission of sequences of recorded analogue values

File transfer in control direction

☐ Transparent file

Background scan
(station-specific parameter, mark "X" if function is only used in the standard direction, "R" if only used in the reverse direction, and "B" if used in both directions).

X Background scan

Acquisition of transmission delay
(station-specific parameter, mark "X" if function is only used in the standard direction, "R" if only used in the reverse direction, and "B" if used in both directions).

☐ Acquisition of transmission delay

Definition of time outs

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Default value</th>
<th>Remarks</th>
<th>Selected value</th>
</tr>
</thead>
<tbody>
<tr>
<td>t₀</td>
<td>30 s</td>
<td>Time-out of connection establishment</td>
<td></td>
</tr>
<tr>
<td>t₁</td>
<td>15 s</td>
<td>Time-out of send or test APDUs</td>
<td></td>
</tr>
<tr>
<td>t₂</td>
<td>10 s</td>
<td>Time-out for acknowledges in case of no data messages t₂ &lt; t₁</td>
<td></td>
</tr>
<tr>
<td>t₃</td>
<td>20 s</td>
<td>Time-out for sending test frames in case of a long idle state</td>
<td></td>
</tr>
</tbody>
</table>

Maximum range for timeouts t₀ to t₂: 1 s to 255 s, accuracy 1 s.
Recommended range for timeout t₃: 1 s to 48 h, resolution 1 s.
Long timeouts for t₃ may be needed in special cases where satellite links or dialup connections are used (for instance to establish connection and collect values only once per day or week).

Maximum number of outstanding I format APDUs k and latest acknowledge APDUs (w)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Default value</th>
<th>Remarks</th>
<th>Selected value</th>
</tr>
</thead>
<tbody>
<tr>
<td>k</td>
<td>12 APDUs</td>
<td>Maximum difference receive sequence number to send state variable</td>
<td></td>
</tr>
<tr>
<td>w</td>
<td>8 APDUs</td>
<td>Latest acknowledge after receiving w I format APDUs</td>
<td></td>
</tr>
</tbody>
</table>

Maximum range of values k: 1 to 32767 \(2^{15}-1\) APDUs, accuracy 1 APDU
AC500 V2.4 IEC60870-5-104 Compatibility List

Maximum range of values $w$: 1 to 32767 APDUs, accuracy 1 APDU (Recommendation: $w$ should not exceed two-thirds of $k$).

<table>
<thead>
<tr>
<th>Portnumber</th>
<th>Parameter</th>
<th>Value</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Portnumber</td>
<td></td>
<td>2404</td>
<td>In all cases</td>
</tr>
</tbody>
</table>

Redundant connections

- 2 Number N of redundancy group connections used

RFC 2200 suite

RFC 2200 is an official Internet Standard which describes the state of standardization of protocols used in the Internet as determined by the Internet Architecture Board (IAB). It offers a broad spectrum of actual standards used in the Internet. The suitable selection of documents from RFC 2200 defined in this standard for given projects has to be chosen by the user of this standard.

- Ethernet 802.3
- Serial X.21 interface
- Other selection from RFC 2200:

  List of valid documents from RFC 2200
  1. ..........................................................
  2. ..........................................................
  3. ..........................................................
  4. ..........................................................
  5. ..........................................................
  6. ..........................................................
  7. etc.
1.5.6.3.2 Modbus protocol

Modbus on TCP/IP protocol

Configuration of Modbus TCP/IP server

Adding a Modbus TCP/IP server to device tree

A Modbus TCP/IP Server instance can be added to any specific Ethernet interface / IP address. Each interface supports max. one instance of “Modbus TCP/IP Server”. Other protocols can be added in parallel.

1. Right click on ETH interface and click “Add object”.
   ⇒ The window “Add object below: ETH” appears.
2. Select “Modbus TCP/IP Server” and click “Add object”.
   ⇒ The node “Modbus_TCP_IP_Server” is added.

Setting the parameters of Modbus_TCP_IP_Server

Byte Order
Format/Endianness for the transmission of WORD values (register) within the request/response telegram (default: “Big Endian”).

Port
TCP Port on which the Server listens.

Startup Behaviour
This parameter specifies how the Server behaves when configuration data is loaded (e.g. on
download). It's default value is “Active”. This means the Server is immediately addressable after
configuration has been performed. In case the Server should be activated later on during run-
time by means of Function Block ModTcpServOnOff this parameter value has to be set to “No
activity”. Parameter Behaviour in state inactive then specifies the Server's behaviour during the
inactive phase.

**Behaviour in state inactive**

This parameter specifies how the Server behaves in inactive state. This state may be set at the
very beginning (parameter Startup Behaviour = “No activity”) and/or requested during runtime
calling Function Block ModTcpServOnOff. It's default value is “No activity”. This means the
Server is not addressable at all (no listening socket on TCP/IP) when it is inactive. Using this
setting, any requests by Modbus TCP Clients lead to the result *Failed to connect to Server or
Timeout*. All other parameter values make the Server respond with an exception code to any
requests by Modbus TCP Clients.

The presentation of the icon next to the Modbus TCP Server in the device tree depends on the
state of the Server:

<table>
<thead>
<tr>
<th>Active</th>
<th>Inactive</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1" alt="Active Icon" /></td>
<td><img src="image2" alt="Inactive Icon" /></td>
</tr>
</tbody>
</table>

**Attention:**

*Exception code 9 is actually not defined by Modbus specification. This may
cause problems using a different Modbus TCP Client than AC500 V3.*

**Disable**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Default</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Disable write to %MB from</td>
<td>0</td>
<td>0 ... 65535</td>
<td>Disable write access starting at %MBx</td>
</tr>
<tr>
<td>Disable write to %MB to</td>
<td>0</td>
<td>0 ... 65535</td>
<td>Disable write access up to %MBx</td>
</tr>
<tr>
<td>Disable read from %MB from</td>
<td>0</td>
<td>0 ... 65535</td>
<td>Disable read access starting at %MBx</td>
</tr>
<tr>
<td>Disable read from %MBx to</td>
<td>0</td>
<td>0 ... 65535</td>
<td>Disable read access up at %MBx</td>
</tr>
</tbody>
</table>

It is possible to disable read and/or write access to individual segments. Reading/writing is disa-
bled beginning at the set start address and is valid up to the set end address (inclusive).
Configuration of Modbus TCP/IP client

Adding a Modbus TCP/IP client to the device tree

The “Modbus_TCP_IP_Client” instance has to be added to the common Ethernet Client protocols’ node. This node supports max. one instance of Modbus TCP/IP Client. Other protocols can be added in parallel.

1. Right click on the node “Protocols” and click “Add object”.
   ≫ The window “Add object below: Protocols” appears.
2. Select “Modbus TCP/IP Client” and click “Add object”.
   ≫ Node “Modbus_TCP_IP_Client” is added.

Depending on a Server’s IP-Address the Client sends its requests via the Ethernet interfaces available.

Setting the parameters of Modbus_TCP_IP_Client

Modbus TCP/IP Client does not have any parameters.

Modbus on RTU protocol

Protocol description can be found in the chapter for Serial interfaces "Configuring Modbus RTU on serial interface" on page 3325.

1.5.6.3 SNTP protocol

Introduction of the SNTP protocol

The Protocol SNTP (Simple Network Time Protocol) provides the functionality to synchronize the clock of a PLC to an external time source. For further information and specification of the protocol please refer to the document RFC4330.

The following modes are supported by the implementation of the AC500 PLC:

- SNTP Client
- SNTP Server
- SNTP Client and Server

The Function Block PmSntpInfo can be used to read diagnosis information of the protocol.

Refer to the documentation of the library ABB_Pm_AC500.lib for further information.
If a high precision of system time is wanted, use a fully functional NTP server or at least a SNTP server with a high-precision time-source (e.g. DCF-77 receiver). Avoid cascading several levels of SNTP server / SNTP clients.

Client requests are normally sent at intervals depending on the frequency tolerance of the client clock and the required accuracy. However, under no conditions requests should be sent at less than one minute intervals (see RFC 4330). Keep that in mind when setting polling-interval of the SNTP client, especially if a huge amount of clients use one single server.

Be sure not to use broadcast or multicast addresses as server or backup-server since current SNTP implementation does not support manycast mode.

Configuration of the SNTP protocol

SNTP client configuration

Implementation of SNTP client and SNTP server is based on protocol version 4.

For SNTP client configuration add a new object “SNTP Client” under “Protocols (Client Protocols)”. Only one instance of the SNTP Client is possible per PLC.

The following parameters are available:

Not all parameters are shown in the user interface.

It should not be necessary to change the default values of the other parameters for the most applications.

But there is the possibility to edit them in the generic parameter editor.
### Server 1

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Default</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IP address</td>
<td>0.0.0.0</td>
<td>Valid IP address</td>
<td>IP address of a server which is used as external time source.</td>
</tr>
<tr>
<td>Minimum polling rate</td>
<td>6</td>
<td>2^6 = 64 s</td>
<td>Specifies the lower limit of the polling rate. It is calculated as power of 2 and has the unit [s]. The actual polling rate is determined by the protocol itself but it will not be lower than this limit.</td>
</tr>
<tr>
<td>Maximum polling rate</td>
<td>10</td>
<td>2^10 = 1024 s</td>
<td>Specifies the upper limit of the polling rate. It is calculated as power of 2 and has the unit [s]. The actual polling rate is determined by the protocol itself but it will not be higher than this limit.</td>
</tr>
</tbody>
</table>

### Server 2

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Default</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enable</td>
<td>FALSE</td>
<td>TRUE or FALSE</td>
<td>Enable server</td>
</tr>
<tr>
<td>IP address</td>
<td>0.0.0.0</td>
<td>Valid IP address</td>
<td>IP address of a server which is used as external time source.</td>
</tr>
<tr>
<td>Minimum polling rate</td>
<td>6</td>
<td>2^6 = 64 s</td>
<td>Specifies the lower limit of the polling rate. It is calculated as power of 2 and has the unit [s]. The actual polling rate is determined by the protocol itself but it will not be lower than this limit.</td>
</tr>
<tr>
<td>Maximum polling rate</td>
<td>10</td>
<td>2^10 = 1024 s</td>
<td>Specifies the upper limit of the polling rate. It is calculated as power of 2 and has the unit [s]. The actual polling rate is determined by the protocol itself but it will not be higher than this limit.</td>
</tr>
</tbody>
</table>

### Time Jumps

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Default</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enable</td>
<td>TRUE</td>
<td>TRUE or FALSE</td>
<td>Enables the option “Time jumps”</td>
</tr>
<tr>
<td>Time jump threshold</td>
<td>1.0 s</td>
<td>0 ... 3.403e+38</td>
<td>Specifies the threshold value for time steps in seconds</td>
</tr>
<tr>
<td>Limit</td>
<td>-1</td>
<td>-1 ... 2147483647</td>
<td>Number of first clock updates after that this option is deactivated, a negative value activates this option permanently.</td>
</tr>
</tbody>
</table>
Max change

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Default</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enable</td>
<td>TRUE</td>
<td>TRUE or FALSE</td>
<td>Enables the option “Max change”</td>
</tr>
<tr>
<td>Maximum change offset</td>
<td>3600 s</td>
<td>0 … 4294967295</td>
<td>Maximum allowed clock offset in seconds</td>
</tr>
<tr>
<td>Start after</td>
<td>3</td>
<td>0 … 4294967295</td>
<td>Specifies the number of first clock updates after that this option is activated</td>
</tr>
<tr>
<td>Ignore after</td>
<td>-1</td>
<td>-1 … 2147483647</td>
<td>Specifies the number of ignored clock updates which exceed the maximum offset. The protocol will be stopped when this value will be exceed. It is never stopped when a negative value is set.</td>
</tr>
</tbody>
</table>

SNTP server configuration

Implementation of SNTP client and SNTP server is based on protocol version 4.

For SNTP server configuration add a new object “SNTP Server” under of the available “Ethernet interfaces (ETH1-ETHn)”. Only one instance of the SNTP Server is possible per PLC.

The following parameters are available:

Not all parameters are shown in the user interface.
It should not be necessary to change the default values of the other parameters for the most applications.
But there is the possibility to edit them in the generic parameter editor.
### Access control

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Default</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enable</td>
<td>FALSE</td>
<td>TRUE or FALSE</td>
<td>Enables Access Control</td>
</tr>
<tr>
<td>Network address</td>
<td>0.0.0.0</td>
<td>Valid IP address</td>
<td>Network address of allowed clients</td>
</tr>
<tr>
<td>Subnet mask</td>
<td>24</td>
<td>8 ... 32</td>
<td>Subnet mask of the network address</td>
</tr>
</tbody>
</table>

### Local server

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Default</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enable</td>
<td>FALSE</td>
<td>TRUE or FALSE</td>
<td>This option enables the protocol to run as local server. That means without synchronization to an external time source.</td>
</tr>
<tr>
<td>Stratum</td>
<td>10</td>
<td>1 ... 15</td>
<td>Stratum of the server when it is used as local server</td>
</tr>
<tr>
<td>Distance</td>
<td>1 s</td>
<td>0 ... 3.403e+38</td>
<td>Distances in seconds of the server when it is used as local server</td>
</tr>
<tr>
<td>Orphan</td>
<td>FALSE</td>
<td>TRUE or FALSE</td>
<td>Enables or disables the orphan mode</td>
</tr>
</tbody>
</table>

### 1.5.6.3.4 FTP server

#### Configuration of FTP server

As of SystemFW 3.1.0 the FTP server is listening only on the Ethernet interface, which the protocol is configured on. It is not possible to have an FTP server on both Ethernet interfaces.

1. Under “Ethernet -> ETH [1,2,...]” add a new object and select “FTP Server” from the list.
2. Double-click the “FTP_Server” item to open FTP server configuration and change the default settings of the parameters, if required.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Default</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FTP Server</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Port</td>
<td>21</td>
<td>21</td>
<td>Do not change the default setting. The parameter specifies the port which is used to connect to the FTP server on the PLC.</td>
</tr>
<tr>
<td>Sessions</td>
<td>1</td>
<td>1...4</td>
<td>Enter the max. number of allowed simultaneous and parallel connections to the FTP server. Each session uses one socket. Note: Some FTP clients require several connections to work.</td>
</tr>
<tr>
<td>Passwords</td>
<td></td>
<td></td>
<td>Set each user's passwords for login. No entry = no password.</td>
</tr>
<tr>
<td>system</td>
<td></td>
<td></td>
<td>System RAM disk</td>
</tr>
<tr>
<td>sdcard</td>
<td></td>
<td></td>
<td>Inserted SD card</td>
</tr>
<tr>
<td>userdisk</td>
<td></td>
<td></td>
<td>User section of the flashdisk.</td>
</tr>
<tr>
<td>flashdisk</td>
<td></td>
<td></td>
<td>Only available with PM5675-2ETH</td>
</tr>
</tbody>
</table>
1.5.6.3.5 MQTT client protocol

System technology

The MQTT protocol is a lightweight communication protocol which is widely used on the internet to connect embedded devices to the cloud.

The MQTT (Message Queuing Telemetry Transport) Client library allows to integrate an AC500 Processor Module to act as a client in the MQTT protocol. Thus, it is possible to exchange data between the AC500 and other devices connected to the MQTT network.

In the figure below, there is an MQTT network with one broker (MQTT broker in the middle) and five clients. The figure shows the main functions of MQTT to send and receive data: publish and subscribe. The clients can publish messages with a specific topic to send data (e.g., the temperature of a connected sensor with a timestamp) to the MQTT broker. For example, the client “AC500_1” publishes a message to topic “topic/2”. On the other hand, sides clients can also subscribe to topics to receive data. For example, the client “Laptop” has subscribed topic “topic/2”. So all messages with the topic “topic/2” which has been published to the MQTT broker will be sent immediately to the client “Laptop”. This creates a message flow from the client “AC500_1” to the laptop.

To realize the MQTT behavior, there are several function blocks implemented in the MQTT Client Library & Chapter 1.4.2.7 “MQTT client library” on page 2196.

Table 767: Function blocks overview

<table>
<thead>
<tr>
<th>Function Block</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MqttConnectWithCertBuffer</td>
<td>Every MQTT use case starts with establishing a connection to an MQTT broker. Therefore, a connection structure needs to be created. The connection structure is used to identify the connection for subsequent operations like publish or subscribe. It is possible to establish an SSL connection. Using an SSL connection, at least a certificate for the server is needed. Certificates can be loaded from a buffer (program variable) or a file which is stored on the PLC.</td>
</tr>
<tr>
<td>MqttConnectWithCertFile</td>
<td>These function blocks can be used on an established MQTT connection to realize the desired use case.</td>
</tr>
<tr>
<td>MqttGetReceivedPacket</td>
<td></td>
</tr>
<tr>
<td>MqttPing</td>
<td></td>
</tr>
<tr>
<td>MqttPublish</td>
<td></td>
</tr>
<tr>
<td>MqttSubscribe</td>
<td></td>
</tr>
<tr>
<td>MqttUnsubscribe</td>
<td></td>
</tr>
<tr>
<td>MqttDisconnect</td>
<td>This function block is the end of each use case.</td>
</tr>
</tbody>
</table>

One MQTT send use case could look like this:
It makes sense for several publish messages in a row (e.g., one message per second) not always open a new connection.

One MQTT receive use case could look like this:

---

### TLS version

The MQTT client uses the TLS version 1.2.

### Configuration in Automation Builder

For the MQTT client no configuration is needed.

### Configuration in CODESYS

All function blocks have to be called in tasks with cyclically processing.

You can use the function blocks with:

- PLC_PRG with automatic task configuration or manual task configuration.
- One single program or different programs.
- One single task or different tasks.

With different programs assigned to different tasks you can define different cycle times and priorities.

### Limitations

- No persistent session. After an interrupted connection, the client needs to subscribe on topics again in case of reconnect.
- One connection (MQTT_CONNECTION) cannot be shared between multiple tasks. Different connections can be used by different tasks or even within the same task.
- Only one FB can operate on a single connection at the same time. Always wait for the FB to complete before calling the next FB. To use two different FB's in parallel (like publish and receive) it is necessary to have two different connections, otherwise they must be called one after the other.

### Hardware

The MQTT protocol requires AC500 devices with integrated Ethernet.

### Examples

Example projects for the libraries can be found in the folder: `\Users\Public\Documents\AutomationBuilder\Examples`.
1.5.6.3.6 AC500 V3 secure protocols

Introduction

The following protocols can be secured using certificates:

- Communication between Automation Builder and the PLC (e.g. Programming, Monitoring)
- Communication between the PLC’s webserver and visualization clients (browsers)
- Communication between the PLC’s FTP server and FTP clients
- Communication between the OPC UA server and OPC UA clients

As a prerequisite to enable secure communication on one or more protocols, the required certificates need to be present on the PLC.

For security reasons ABB does not encourage the use of self-signed certificates. ABB shall not be held liable for any damage or loss that arises due to the use of self-signed certificates on AC500 PLCs.

Self-signed certificates protect against eavesdropping if used correctly. They do not offer any secure means of authentication.

Certificate handling

Automation Builder offers a convenient “Security Screen” to manage certificates on connected PLCs.

It can be accessed through the shield icon on the lower right corner of the main window:

Use the tab “Devices” to manage certificates on the PLC.

It offers to:

- show certificates available on the PLC
- import and export certificates
- create new (self-signed) certificates
- trust or untrust certificates
Configuring secure protocols

Encrypted communication between Automation Builder and the PLC

Via tab “Communication Settings”

Via “Security Screen” in Automation Builder
Install a trusted certificate

Ensure the PLC clock is set to the current time and date when using certificates on the PLC. Otherwise the certificate cannot be used to secure a protocol (see also AC500_Battery Chapter 1.5.5.1.3.2 “AC500 battery” on page 3116 and Chapter 1.5.6.3.3.2.1 “SNTP client configuration” on page 3425).

When trying to log in or when you set the PLC as active path, there will be a one-time pop-up asking you to add the PLC’s certificate to the trusted certificates:

After trusting the PLC’s certificate, the communication between the Automation Builder and the PLC is now encrypted. This is shown by additional yellow lines around the communication path on the “Communication Settings” page.

Secure web server

1. Generate or import a certificate for the web server

2. Attach a web server node to either ETH1 or ETH2 or both and configure security mode.

   This will automatically insert a visualization into the project.

   The available modes of operation are:
   - http only
   - https only
   - Both (http and https)
   - Redirect http to https
3. Download and set the PLC to RUN.
4. Connect to the web server using the configured method: https://<your PLC’s IP address>/webvisu.htm.

In case you are using a self-signed certificate, your browser will show some warnings.
If you are aware of the risks of self-signed certificates, this can be ignored.
☞ Further information on page 3431

Secure FTP

1. Import a certificate to the PLC for FTP or create a self-signed certificate.

Ensure the PLC clock is set to the current time and date when using certificates on the PLC. Otherwise the certificate cannot be used to secure a protocol (see also AC500_Battery Chapter 1.5.5.1.3.2 “AC500 battery” on page 3116 and Chapter 1.5.6.3.3.2.1 “SNTP client configuration” on page 3425).

2. Add an FTP server to either ETH1 or ETH2
3. Set the parameter “Security Mode” to either “BOTH” or “FTPS only”.
☞ You can use any FTP client that supports FTPS explicit mode (FTPES).

In case you are using a self-signed certificate, the FTP client will show some warnings or notice that it does not know the certificate and wants you to check it.
☞ Further information on page 3431

OPC UA secure

OPC UA uses mutual authentication, which means that both partners must have their own certificate and know the other’s certificate, before being able to establish a connection!
1. Create a new certificate in your OPC UA client.

   **Ensure the PLC clock is set to the current time and date when using certificates on the PLC. Otherwise the certificate cannot be used to secure a protocol (see also AC500_Battery Chapter 1.5.5.1.3.2 “AC500 battery” on page 3116 and Chapter 1.5.6.3.3.2.1 “SNTP client configuration” on page 3425).**

2. Import that certificate to the “Trusted Certificates” in your PLC using the “Security Screen”.

3. Import a certificate for the OPC UA server on the PLC or create a self-signed certificate.

4. Export that cert to the PC and provide it as a trusted certificate to your OPC UA client.

5. Reboot the PLC and check that it is in RUN and both certificates are on the PLC (via the “Security Screen”).

6. Add the PLC as OPC UA server in your OPC UA client.

7. Connect to the OPC UA Server.
   - You can interact normal with the UA server.

   **In case you are using a self-signed certificate, you will see some warning message (depending on the OPC UA client).**

   **If you are aware of the risks of self-signed certificates, this can be ignored.**

   ☞ Further information on page 3431

The certificate warnings will only go away when using a certificate from a trusted certification authority or a certificate derived from this by an intermediate certification authority (e.g. a company CA).

That process is done via PLCShell command “cert-createcsr”, then getting the file from the PLC via the filebrowser tab in “cert/export” and getting that signing request turned into a real certificate by a certification authority.

Import the certificate generated by your certification authority using the security screen.
1.5.6.3.7 KNX configurator

Refer to the general description for information about the following tabs of the device editor.

- **Tab `<device name> I/O Mapping`** % Chapter 1.3.1.25.1.19.11 “Tab `<device name> I/O Mapping” on page 755
- **Tab `<device name> IEC Objects`** % Chapter 1.3.1.25.1.19.12 “Tab `<device name> IEC Objects” on page 759
- **Tab `<device name> Parameters`** % Chapter 1.3.1.25.1.19.3 “Tab 'Parameters’” on page 746
- **Tab `<device name> Status`** % Chapter 1.3.1.25.1.19.17 “Tab 'Status” on page 768
- **Tab `<device name> Information`** % Chapter 1.3.1.25.1.19.18 “Tab 'Information’” on page 768

Only in the case of special features is there an additional help page for the specific device editor.

*If the "<device name> Parameters" tab is not shown, then select the “Show generic device configuration editors” option in the CODESYS options (“Device Editor” category).*

With the KNX editor from CODESYS, you define the communication objects of your building automation. The communication objects are exported and made available to the ETS5 program. Linking the communication objects to the different KNX devices is performed exclusively in the ETS5 program. Therefore, only the objects are generated in CODESYS. The objects are linked to variables from the PLC program by means of “I/O mapping”.

You add an Ethernet adapter below the controller. Then you add the KNX device below the adapter. You can insert only one KNX device per controller.

See also

- **Device Editor Options** % Chapter 1.3.1.25.3.13.9 “Dialog ‘Options’ - ‘Device Editor’” on page 1081
ETS5 Software - 'DCA' Plug-In

Linking the communication objects of the different KNX devices is performed exclusively in the ETS5 program. To do this, you need the ETS5 software (light or professional version). You also need the KNX product file available from KNX.

**Programming steps**

1. Create a project in CODESYS.
2. Download the CODESYS project to the controller.
   - The CRC is also downloaded to the controller.
3. Create an export file in CODESYS.
   - The CRC is also saved in the export file.
4. Read the export file into the configuration of ETS5.
5. Parameterize the objects in ETS5.
6. Start the program on the controller.
7. Transfer the KNX configuration to the controller.
   - The CRC is also transferred. The runtime system checks whether or not both CRCs match. When they match, the KNX device is identified as functional by the green arrow. If not, then an error is issued in the logger. In case of error, the process data (inputs/outputs) is not updated.

**Tab 'KNX - General'**

Object: KNX

The tab in the configurator of the KNX editor shows an overview of all communication objects. The I/Os of the communication objects are applied automatically to the I/O mapping.

Entries can be edited directly in the table or in the “Communication object” dialog. Existing entries can be copied via copy&paste. The next free channel number is used automatically in this case.

<table>
<thead>
<tr>
<th>Table 768: “Address settings”</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>“Add”</strong></td>
</tr>
<tr>
<td><strong>“Edit”</strong></td>
</tr>
<tr>
<td><strong>“Delete”</strong></td>
</tr>
<tr>
<td><strong>“Export to ETS”</strong></td>
</tr>
<tr>
<td><strong>“Export to ETS”</strong></td>
</tr>
<tr>
<td><strong>“Import CSV”</strong></td>
</tr>
<tr>
<td><strong>“Identification”</strong></td>
</tr>
</tbody>
</table>

**Dialog 'Communication object'**

| **“Number of group object”** | Unique channel number. Gaps in the numbering is permitted. If the channel number is already assigned, then an error text is displayed and the “OK” button is disabled. |
| **“Type”** | Determines whether or not the object in CODESYS is used as “Input” or “Output”. |
**Data point type**
The data types (DPT = Data Point Types) are specified in the KNX standard. In CODESYS, a selection of the most common data types is available. Only the basic data type can be selected, without units (for example, DPT9.*).

**Name of group object**
Any object name. Depending on the data type, a predefined text is automatically added.

**Function of group object**
Any function name. Depending on the data type, a predefined text is automatically added.

**Watchdog Timeout**
If no new message has been received after this time has elapsed, then the status bit Timeout is set.

---

Tab 'I/O Mapping'
Object: KNX
The I/O channels are generated for each communication object:

<table>
<thead>
<tr>
<th>Table 769: General I/Os</th>
</tr>
</thead>
<tbody>
<tr>
<td>Signal</td>
</tr>
<tr>
<td>&quot;Program LED Status&quot;</td>
</tr>
<tr>
<td>&quot;Program Button&quot;</td>
</tr>
</tbody>
</table>

Table 770: I/O channels of the communication object

<table>
<thead>
<tr>
<th>Signal</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Status byte&quot;</td>
<td>Status byte as defined in the KNX stack. This allows you to determine in the application whether or not data has been received. The status can be reset by means of the ResetStatusFlags method.</td>
</tr>
</tbody>
</table>
| "Trigger/Disable Cyclic, send on change" | Depending on the configuration in the ETS program, this output has the following function:  
  • If at least one of the options “send on difference”, “send on change”, or “Cyclic sending” is enabled, then the output is defined as deactivation. If it is set to TRUE, then cyclic sending or send on change is stopped.  
  • If none of the options “send on difference”, “send on change”, or “Cyclic sending” is enabled, then sending is triggered by a rising edge. |
| "Trigger Output"              | Value for the input or outputs – depending on the corresponding communication object. |

ETS5 - Tab 'Parameter'
The parameter page of the ETS5 configuration software is available only after you have imported the CODESYS configuration file. The parameter page is where you define the sending behavior of the values.
Table 771: “General Information”

<table>
<thead>
<tr>
<th>parameter</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default gateway</td>
<td>Default gateway for sending</td>
</tr>
<tr>
<td>Telegram rate</td>
<td>Sending rate of telegrams</td>
</tr>
<tr>
<td>Project title</td>
<td>In CODESYS, these parameters can be defined in the project information. They are imported to ETS5 in the XML file and displayed here.</td>
</tr>
<tr>
<td>Application date</td>
<td>“Identifier”: CRC of the configuration. The CRC is also displayed in CODESYS and must be identical to the CRC displayed here so that communication can be started.</td>
</tr>
<tr>
<td>Identifier</td>
<td>“Application state”</td>
</tr>
<tr>
<td>Version</td>
<td>“Description”</td>
</tr>
</tbody>
</table>

The objects are subdivided into groups of ten (1 .. 10, 11 .. 20, 21 .. 30, etc.). A maximum of 1000 communication objects is possible.

Table 772: “Object 1 .. 10”

<table>
<thead>
<tr>
<th>parameter</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;type&gt;</td>
<td>Type of the object</td>
</tr>
<tr>
<td>Communication direction</td>
<td>“Output (PLC to KNX)”: The value is sent from the CODESYS controller to the KNX object. For this communication direction, more settings are possible (“Send condition”, etc.). “Input (KNX to PLC)”: The value is sent to the CODESYS controller.</td>
</tr>
<tr>
<td>Send condition</td>
<td>“No automatic sending”: No send when value is changed “Send on change”: Send each time value is changed “Send on difference”: Send when the change in value corresponds to at least the value for “Sending difference”.</td>
</tr>
<tr>
<td>Sending difference</td>
<td>Requirement: “Send condition” is “send on difference”. The value is passed when its change is at least this value.</td>
</tr>
<tr>
<td>Cyclic sending</td>
<td>“Disable”: No cyclic sending “Enable (seconds)”, “Enable (minutes)”: Cyclic sending – regardless of the “Send condition”.</td>
</tr>
<tr>
<td>Cycle time [hh:mm:ss]</td>
<td>Rate for cyclic sending (in hours/minutes/seconds) Requirement: “Cyclic sending” is set to “Enable (seconds)”.</td>
</tr>
<tr>
<td>Cycle time [hh:mm]</td>
<td>Rate for cyclic sending (in hours/minutes) Requirement: “Cyclic sending” is set to “Enable (minutes)”</td>
</tr>
</tbody>
</table>

1.5.6.3.8 CODESYS BACnet

Overview

BACnet is a standardized data communication protocol for Building Automation and Control networks as defined in the ANSI/ASHRAE Standard 135 and ISO 16484-5.

The BACnet protocol defines "Services" to allow communication between devices. Examples include Who-Is, I-Am, Who-Has, and I-Have for device and object search and identification, as well as Read-Property and Write-Property for the exchange of data.
The BACnet protocol defines a number of "Object" types on which the services operate. Each object is characterized by its "Properties".

The BACnet objects are combined in a BACnet "Device". A BACnet device represents the functionality of a physical device.

BACnet integration into CODESYS implements the ANSI/ASHRAE Standard 135-2012 "protocol revision 14" and is based on the AMEV AS-A and AS-B standards. Integration allows access to the properties of BACnet objects and the configuration parameters of a BACnet device by means of an IEC application. You can program a dynamic BACnet configuration and have access to the BACnet functions in the BACnet network by reading and writing BACnet object properties.

The installation of the packages contains the following parts:

- **Package: CODESYS BACnet**
  - BACnet plug-in component
  - Device descriptions for BACnet servers, BACnet objects, and BACnet clients
  - Libraries: CmpBACnet, BACnet, and BACnetDefaultImpl
- **BACnet runtime component**

See also

- Chapter 1.5.6.3.8.6 “BACnet Configuration in the CODESYS Project” on page 3445
- Chapter 1.5.6.3.8.2 “Architecture, Components, and their Tasks” on page 3440
- Chapter 1.5.6.3.8.7 “Access to BACnet Objects and Their Properties” on page 3447
- Chapter 1.5.6.3.8.8 “Use of BACnet client function blocks” on page 3451
- Chapter 1.5.6.3.8.4 “Getting Started” on page 3442
- Chapter 1.5.6.3.8.10 “Implementation Requirements when Using the Current BACnet API/Stack” on page 3453

Documentation for the BACnet library and the CmpBACnet library is provided in the Library Manager.

**Architecture, Components, and their Tasks**

CODESYS BACnet includes three IEC libraries:

- CmpBACnet
- BACnet
- BACnetDefaultImpl

It also includes the BACnet plugin for the CODESYS development environment, which performs consistency checks on library versions and BACnet object IDs, as well as providing EDE export/import.

The BACnet stack (BACstack) is an optional component of the CODESYS runtime (CmpBACnet component).

The IEC library CmpBACnet represents the integration of the BACnet stack into a CODESYS IEC environment and provides the BACnet data types as well as the BACstack methods. The sole use of the IEC library CmpBACnet (without the BACnet and BACnetDefaultImpl libraries) would result in complex and lengthy IEC application code.

The BACnet library simplifies BACnet application development considerably as compared to the sole use of CmpBACnet, especially in the following areas:

- Starting and stopping the BACnet stack
- Using BACnet server objects and their properties
- Triggering asynchronous requests (mainly client service requests) and processing the request transaction
- Processing of callbacks from the BACnet stack (see IBACnetEventConsumer) and distributing the callbacks to multiple receivers in the application
Furthermore, the BACnet library provides a plug-in mechanism (BACnetServerPlugin) for extending certain aspects of the BACnet library. BACnetServerPlugin is the basis for the BACnetDefaultImpl library.

The BACnetDefaultImpl library is used for the additional simplification of BACnet application development. The BACnet standard ASHRAE 135 leaves some aspects of the practical use of BACnet open. The most notable examples include the following:

- Persistence of server objects
- Storage and persistence of Trend Log, Trend Log Multiple, and Event Log entries
- Update of the date/time information of the device object

CODESYS implementations for these aspects are necessarily based on some assumptions about the PLC and the application, for example:

- The PLC has a file system
- The PLC file system does not have excessively strict limitations regarding data volume, speed, or number of write cycles

Due to these assumptions, such implementations cannot be fixed in the BACnet library, because this would prevent the actualization of other use cases and the compliance with other constraints.

The BACnetDefaultImpl library is the place for these kinds of default implementations. In the best case, when the assumptions are correct, these standard implementations can be used easily. If not, then a standard implementation of a particular aspect can be used very well as an example and a basis for a PLC/application-specific implementation.

See also

- % Chapter 1.5.6.3.8.5 “BACnet Configuration in the Runtime” on page 3444
- % Chapter 1.5.6.3.8.3 “When to Use the BACnet Library or the CmpBACnet Library” on page 3441

When to Use the BACnet Library or the CmpBACnet Library

The IEC library BACnet is intended as a layer over the IEC library CmpBACnet. However, the layer does not hide the library because this would require the BACnet library to have “facade” functions for CmpBACnet functions. These facade functions would result in larger application code and increased runtime requirements. This is difficult for the PLC to accept. For this reason, it is necessary to know when elements from the BACnet library or CmpBACnet library are to be used.

General rules:

- Starting and stopping the BACnet stack
  Always use BACnetServer.StartBACnetStack and BACnetServer.StopBACnetStack or AutoStart. Never directly use the corresponding functions of the CmpBACnet library, such as CmpBACnet.BACnetServerInit.

- Using BACnet server objects and their properties
  Always use the specified function blocks in IEC-lib-BACnet, such as BACnetAnalogValue. Never directly use the corresponding functions of the BACnet library, such as CmpBACnet.BACnetStorePropertyInstance.
● Triggering of asynchronous requests
  Always use the specified client function blocks of the BACnet library, such as 
  BACnetClientReadProperty. Never directly use the corresponding functions of the 
  CmpBACnet library, such as CmpBACnet.BACnetReadProperty. All functions of the 
  CmpBACnet library that require a BACnetAsyncTransactionToken belong to this cate-
  gory and should never be used directly.

● Processing of callbacks from the BACnet stack and distributing the callbacks to multiple 
  receivers in the application
  Always use IBACnetEventConsumer and BACnetServer.RegisterHook/
  UnregisterHook/RegisterCallback/UnregisterCallback. Never directly use the 
  corresponding functions of the CmpBACnet library, such as CmpBACnet.BACnetSetHook 
  or CmpBACnet.BACnetSetCallback.

When is it appropriate and safe to directly call the functions of the CmpBACnet library?

Basically, it is only necessary to call functions of CmpBACnet directly when a corresponding 
functionality is not provided in the BACnet library. Check the BACnet library first before trying to 
use CmpBACnet directly. It is possible to use blocking functions in CmpBACnet, such as 
BACnet*CbCompletion, BACnetIam(Ex), or BACnetIHave(Ex), BACnetUnconf*.

Most often, you will use BACnet*CbCompletion to implement your specific 
IBACnetEventConsumer.BACnetEventCallbacks. But first check whether or not the 
BACnetDefaultImpl library already contains an appropriate standard implementation.

See also

- § Chapter 1.5.6.3.8.2 “Architecture, Components, and their Tasks” on page 3440

Getting Started

The following instruction guide you to a simple application that uses two BACnet objects in 
order to 1) write an analog BACnet value at runtime and 2) sound an alarm according to a 
BACnet notification class when the current BACnet value is outside of a specific range.

Requirements:

- The CODESYS BACnet package is installed.
- The runtime supports BACnet. For this to work, the component CmpBACnet has to be 
  entered and configured in the configuration file of the runtime.

1. Create a standard project in CODESYS with a PLC device that supports BACnet.
2. Select the PLC object in the device tree and click “Insert Device” to insert a “BACnet 
   Server”.
3. Select the BACnet server object and click “Insert Device” to insert a “BACnet_Analog_Value” object and a “BACnet_Notification_Class” object below it.
   The inserted BACnet objects are displayed in the “Devices” view.

4. Double-click the “BACnet_Analog_Value” object to open its configuration editor.
   On the “BACnet Parameter” tab, you see the initial configuration of the BACnet AnalogValue object.

5. Modify the object properties in the “Value” column as follows:
   - AckedTransitions: '111'
   - NotificationClass: 1
   - HighLimit: 10
   - LimitEnable: '11'
   - LowLimit: 0
   Now your application provides a BACnet AnalogValue and a BACnet NotificationClass. Now the configuration is set so that the BACnet AnalogValue will generate alarms as soon as its Present_Value property reaches a value outside of [0; 10].

6. In the PLC_PRG program of your application, define a variable lrVal of the type LREAL and insert the following code: Hint: You can also press the [F2] key to get the function block instances of the BACnet objects from the Input assistant ("Instance Calls" category).
   
   ```
   BACnet_Analog_Value.PresentValue :=
   BACnet_Analog_Value.PresentValue + 0.0001;
   lrVal := BACnet_Analog_Value.PresentValue;
   ```

7. Build ([F11]), download the error-free application to the controller, and start it.
   The BACnet property Present_Value of the BACnet Analog Value object is incremented.
8. Start an external BACnet client (for example, Innea BACnet Explorer Free) that runs at a different network address than the runtime (for example, using a virtual machine). Use it to monitor the values of BACnet property `Present_Value` and variable `lrVal` in the BACnet network.

   ⇒ You can see how the BACnet property `Present_Value` is incremented and as a consequence the value of the variable `lrVal` increases. As soon as `Present_Value` exceeds the value of 10, an alarm of the type `BACnet-NotificationClass` inserted in the application is generated on the BACnet server device.

This is an example of how you can access the BACnet properties of a BACnet object at runtime. If no explicit property is provided by the BACnet object function block, then at least it provides auxiliary functions for accessing certain or all BACnet properties.

See also
- Chapter 1.5.6.3.8.1 “Overview” on page 3439
- Chapter 1.5.6.3.8.5 “BACnet Configuration in the Runtime” on page 3444
- Chapter 1.5.6.3.8.6 “BACnet Configuration in the CODESYS Project” on page 3445
- Chapter 1.5.6.3.8.7 “Access to BACnet Objects and Their Properties” on page 3447

**BACnet Configuration in the Runtime**

CODESYS BACnet uses the component `CmpBACnet` for communication. This component has to be entered and configured in the configuration file of the runtime as follows.

```plaintext
[ComponentManager]
Component.1=CmpBACnet
[CmpBACnet]
; configuration file of the BACnet stack
IniFile=bacstac.ini
; task priority of the BACstack process task, default: TASKPRIO_NORMAL_END (159)
TaskPrio=128
; name of the named pipe for the communication with the BACstack
AppName=/tmp/BACnetServer
```

In general, CODESYS separates access to the file system within the IEC application and access from the program code of the runtime components. Different root directories can be specified in the configuration file of the runtime.

```plaintext
[SysFile]
; root file path for runtime components
FilePath=D:/Temp/Default
; root file path for IEC applications
IecFilePath=D:/Temp/IEC
```

An IEC application with CODESYS BACnet writes the configuration file of the BACnet stack (using information from `BACnet.BACnetIPdatalink`) when starting the BACnet server and then reads the file within the runtime component `CmpBACnet`. Therefore, shared read and write access of the IEC application and the runtime component have to be enabled for this configuration file. This is done either by using an absolute path or by using placeholders (for CODESYS Control V3.5.16.0 and later).

**Absolute path:**

```plaintext
...
[CmpBACnet]
IniFile=D:/Temp/bacstac.ini
...
```

**Placeholder (example: $BACnet$:)**
Furthermore, CODESYS allows for the "sandboxing" of various aspects of the runtime, including access to the file system. In this case, access to the file system within the IEC application and access from the program code of the runtime components are restricted to the configured paths. For more information, see the configuration of [SysFile]:

...  
[SysFile]  
; ForceFilePath=1  
; ForceIecFilePath=1  
...  

For more information about this, see the help for CODESYS Control V3 Runtime System, "CODESYS Control FilePath & Placeholders". This help is available in the 3S-Smart Software Solutions GmbH Customer Portal.

When configuring the "sandboxing" of the file system, you should note that it is possible to access the configuration file of the BACnet stack as described above. Therefore, the alternative using placeholders is mandatory in this case when the path to `bacstac.ini` is not already configured elsewhere.

See also

- Chapter 1.5.6.3.8.2 "Architecture, Components, and their Tasks" on page 3440

### BACnet Configuration in the CODESYS Project

#### BACnet configuration in the device tree and device editor

In CODESYS, you can insert a BACnet server and BACnet objects as devices below a PLC device in the project tree. The BACnet server represents the BACnet device and handles the communication with the runtime system.

One server device is possible per application, including the object devices as children. You can also insert BACnet clients below the server.

Requirement: The CODESYS BACnet package is installed. A project with a BACnet-capable PLC device has been created.

1. Select the PLC device in the project tree and add a "BACnet Server" by clicking "Project ➔ Add Device".
2. Select the BACnet server in the tree and insert the required BACnet objects below it.
   - Servers and objects are provided with an initial configuration that conforms with the AMEV AS-A and AS-B standard. This is illustrated in the generic device editor.
3. Double-click the BACnet server or a BACnet object to open its device editor.
   - On the "BACnet Interface Parameters" tab, you can modify the initial configuration in the "Value" column.

A complete configuration of server and object properties by means of this device configuration editor is not possible. This is possible by means of IEC program code. See below.
You can import and export the current BACnet configuration (located in the device tree of the project) to and from CSV files in EDE format (Engineering Data Exchange).

The functionality is based on the "Template for the Exchange of Engineering Data for BACnet Systems" as an Excel file. The export writes the required as well as optional information about the BACnet configuration to the base export file *_EDE.csv. In doing so, other CSV files with information about the "State Texts", "Unit Texts", and "Object Types" are created automatically: *_ObjTypes.csv, *_StateTexts.csv, and *_Units.csv.

**Export:**

The command **"BACnet ➔ Export to EDE File"** opens the “EDE Export” dialog for the following input:

<table>
<thead>
<tr>
<th>“Project name”</th>
<th>Name of the project that is currently being exported from. Found in the &lt;project name&gt;.csv file at PROJECT_NAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Version of the reference file”</td>
<td>Version of the export file. Example: 1.0.0.0 Found in the CSV file at VERSION_OF_REFERENCE_FILE</td>
</tr>
<tr>
<td>“Author of last change”</td>
<td>Found in the CSV file at AUTHOR_OF_LAST_CHANGE</td>
</tr>
<tr>
<td>“EDE file to export”</td>
<td>Path of the location of the export file in the local file system. Note: The name of the CSV file has to correspond to the following syntax: &lt;name&gt;_EDE.csv. Example: proj1_EDE.csv</td>
</tr>
<tr>
<td>“Separator”</td>
<td>Character that should be used to separate the text in the export file. Selection: comma, semicolon, space, or tab</td>
</tr>
<tr>
<td>“Quoting”</td>
<td>Specifies whether the exported texts in the EDE file should be placed in single quotation marks, double quotation marks, or no quotation marks.</td>
</tr>
</tbody>
</table>

**Import:**

The **“BACnet ➔ Import from EDE File”** opens the “EDE Import” dialog. Here you specify the following information for the import:

| “EDE file to import”             | In the file system, select an EDE file (*_EDE.csv) that contains the desired BACnet configuration. Any other CSV files (*_ObjTypes.csv, *_StateTexts.csv, *_Units.csv) that may be found are also read automatically. Note: Only CSV files that correspond to the usual EDE file naming convention are included. See above: “EDE file to export” |
| “Separator”                      | The separator used in the selected CSV file is identified and set here automatically.                          |
| “Quoting”                        | The quotation mark used in the selected CSV file is identified and set here automatically.                    |
| “EDE file to export”             | Path of the export file in the local file system.                                                            |
| “Separator”                      | Character that should be used to separate the text in the export file. Selection: comma, semicolon, space, or tab |
| “Quoting”                        | Specifies whether the exported texts in the EDE file should be placed in single quotation marks, double quotation marks, or no quotation marks. |
**Import EDE Information**

- **Project name**
  Not editable. `PROJECT_NAME` from the EDE file.
- **Version of the reference file**
  Not editable. `VERSION_OF_REFERENCEFILE` from the EDE file.
- **Author of last change**
  Not editable. `AUTHOR_OF_LAST_CHANGE` from the EDE file.
- **Timestamp of last change**
  Not editable. `TIMESTAMP_OF_LAST_CHANGE` from the EDE file.
- **Version of layout**
  Not editable. `VERSION_OF_LAYOUT` from the EDE file.

**Import Selection**

- **BACnet device to import**
  The BACnet devices defined in the EDE file are available for selection. The import operation inserts the selected device into the device tree below the “PLC to import to” set below.

- **PLC to import to**
  The devices available in the project are available for selection. The imported BACnet configuration is inserted below the selected device.

---

**Note the following during import:**

The EDE import ignores the following information which may be included in the EDE file:

- Non-standardized BACnet object types (entry in the "Object Types" sheet of the data exchange template, and in the EDE file *_ObjTypes.csv)
- Non-standardized BACnetEngineeringUnits, meaning unit codes (entry in the "Unit Texts" sheet of the data exchange template, and in the EDE file *_Units.csv)
- The optional information "settable" (entry in the "EDE" sheet of the data exchange template, and in the EDE file *_EDE.csv)
- The optional information "supports COV" (entry in the "EDE" sheet of the data exchange template, and in the EDE file *_EDE.csv)

Acceptance of EDE file entries into the BACnet device object:

- The entry "keyname" is stored in the parameter "EDEkey".
- The entry "vendor-specific-address" is stored in the parameter "EDEVendorSpecificAddress".
- The "Unit Text" that belongs to the unit code is stored in the parameter "EDEunitText".

**BACnet configuration using application code**

So that you can access all configuration parameters and BACnet object properties with IEC program code, the library `BACnet` provides a function block for BACnet servers and another function block for BACnet objects. This is mapped to an instance variable on the "I/O Mapping" tab of the respective device configuration editor. Example: The function block `BACnetAnalogValue` is provided for the BACnet object `BACnet Analog Value`. This is automatically instantiated in the project by means of the variable `BACnet_Analog_Value`.

In this way, you can dynamically configure all BACnet properties of an object with your application.

See also

- Chapter 1.5.6.3.8.7 "Access to BACnet Objects and Their Properties" on page 3447

**Access to BACnet Objects and Their Properties**

Each function block of an BACnet object inserted as a device in the project tree is derived from the `BACnetStaticObjectBase` function block (`BACnet` library) and therefore from `BACnetObjectBase`. 
BACnetObjectBase provides the capability of changing any BACnet property for any BACnet object function block by means of the general function SetPropertyInstance.

The data for the BACnet properties to be set are transferred to SetPropertyInstance in the form of the IEC_BACNET_PROPERTY_CONTENTS structure. Besides an enumeration that describes the data type, this structure also contains a buffer that has a POINTER TO BYTE to the actual BACnet property data. This makes it possible to transfer any BACnet property data structures within the IEC_BACNET_PROPERTY_CONTENTS structure. The specified data type defines the structures or data to which the buffer points.

The data and the data type must correspond to each other in accordance with the following table.

<table>
<thead>
<tr>
<th>IEC_BACNET_DATA_TYPE</th>
<th>CODESYS data type (struct)</th>
</tr>
</thead>
<tbody>
<tr>
<td>DATA_TYPE_BOOLEAN</td>
<td>IEC_BACNET_BOOLEAN</td>
</tr>
<tr>
<td>DATA_TYPE_UNSIGNED</td>
<td>IEC_BACNET_UNSIGNED</td>
</tr>
<tr>
<td>DATA_TYPE_SIGNED</td>
<td>IEC_BACNET_SIGNED</td>
</tr>
<tr>
<td>DATA_TYPE_REAL</td>
<td>IEC_BACNET_REAL</td>
</tr>
<tr>
<td>DATA_TYPE_DOUBLE</td>
<td>IEC_BACNET_DOUBLE</td>
</tr>
<tr>
<td>DATA_TYPE_OCTET_STRING</td>
<td>IEC_BACNET_OCTET_STRING</td>
</tr>
<tr>
<td>DATA_TYPE_CHAR_STRING</td>
<td>IEC_BACNET_STRING</td>
</tr>
<tr>
<td>DATA_TYPE_BIT_STRING</td>
<td>IEC_BACNET_BIT_STRING</td>
</tr>
<tr>
<td>DATA_TYPE_ENUMERATED</td>
<td>IEC_BACNET_ENUMERATED</td>
</tr>
<tr>
<td>DATA_TYPE_DATE</td>
<td>IEC_BACNET_DATE</td>
</tr>
<tr>
<td>DATA_TYPE_TIME</td>
<td>IEC_BACNET_TIME</td>
</tr>
<tr>
<td>DATA_TYPE_OBJECT_IDENTIFIER</td>
<td>IEC_BACNET_OBJECT_ID</td>
</tr>
<tr>
<td>DATA_TYPE_PRI_ARRAY_ITEM</td>
<td>IEC_BACNET_PRIORITY_ARRAY_ITEM</td>
</tr>
<tr>
<td>DATA_TYPE_ADDRESS</td>
<td>IEC_BACNET_ADDRESS</td>
</tr>
<tr>
<td>DATA_TYPE_DATE_TIME</td>
<td>IEC_BACNET_DATE_TIME</td>
</tr>
<tr>
<td>DATA_TYPE_OBJ_PROP_REFERENCE</td>
<td>IEC_BACNET_OBJ_PROP_REFERENCE</td>
</tr>
<tr>
<td>DATA_TYPE_PROPERTY_STATES</td>
<td>IEC_BACNET_PROPERTY_STATES</td>
</tr>
<tr>
<td>DATA_TYPE_EVENT_PARAMETER</td>
<td>IEC_BACNET_EVENT_PARAMETER</td>
</tr>
<tr>
<td>DATA_TYPE_DATE_RANGE</td>
<td>IEC_BACNET_DATE_RANGE</td>
</tr>
<tr>
<td>DATA_TYPE_WEEK_N_DAY</td>
<td>IEC_BACNET_WEEK_AND_DAY</td>
</tr>
<tr>
<td>DATA_TYPECALENDAR_ENTRY</td>
<td>IEC_BACNETCALENDAR_ENTRY</td>
</tr>
<tr>
<td>DATA_TYPE_TIME_VALUE</td>
<td>IEC_BACNET_TIME_VALUE</td>
</tr>
<tr>
<td>DATA_TYPE_SPECIAL_EVENT</td>
<td>IEC_BACNET_SPECIAL_EVENT</td>
</tr>
<tr>
<td>DATA_TYPE_RECIPIENT</td>
<td>IEC_BACNET_RECIPIENT</td>
</tr>
<tr>
<td>DATA_TYPE_SETPOINT_REFERENCE</td>
<td>IEC_BACNET_SETPOINT_REFERENCE</td>
</tr>
<tr>
<td>DATA_TYPE_DESTINATION</td>
<td>IEC_BACNET_DESTINATION</td>
</tr>
<tr>
<td>IEC_BACNET_DATA_TYPE</td>
<td>CODESYS data type (struct)</td>
</tr>
<tr>
<td>--------------------------------------</td>
<td>--------------------------------------------------</td>
</tr>
<tr>
<td>DATA_TYPE_DAILY_SCHEDULE</td>
<td>IEC_BACNET_DAILY_SCHEDULE</td>
</tr>
<tr>
<td>DATA_TYPE_TIME_STAMP</td>
<td>IEC_BACNET_TIME_STAMP</td>
</tr>
<tr>
<td>DATE_TYPE_PROPERTY_REFERENCE</td>
<td>IEC_BACNET_PROPERTY_REFERENCE</td>
</tr>
<tr>
<td>DATA_TYPE_ADDRESS_BINDING</td>
<td>IEC_BACNET_ADDRESS_BINDING</td>
</tr>
<tr>
<td>DATA_TYPE_DEV_OBJ_PROP_REFERENCE</td>
<td>IEC_BACNET_DEV_OBJ_PROP_REFERENCE</td>
</tr>
<tr>
<td>DATA_TYPE_CLIENT_COV</td>
<td>IEC_BACNET_CLIENT_COV</td>
</tr>
<tr>
<td>DATA_TYPE_LOG_RECORD</td>
<td>IEC_BACNET_LOG_RECORD</td>
</tr>
<tr>
<td>DATA_TYPE_ACTION_LIST</td>
<td>IEC_BACNET_ACTION_LIST</td>
</tr>
<tr>
<td>DATA_TYPE_READ_ACCESS_SPEC</td>
<td>IEC_BACNET_READ_ACCESS_SPEC</td>
</tr>
<tr>
<td>DATA_TYPE_READ_ACCESS_RESULT</td>
<td>IEC_BACNET_READ_ACCESS_RESULT</td>
</tr>
<tr>
<td>DATA_TYPE_COV_SUBSCRIPTION</td>
<td>IEC_BACNET_COV_SUBSCRIPTION</td>
</tr>
<tr>
<td>DATA_TYPE_DEV_OBJ_REFERENCE</td>
<td>IEC_BACNET_DEV_OBJ_REFERENCE</td>
</tr>
<tr>
<td>DATA_TYPE_DEV_OBJ_PROP_VALUE</td>
<td>IEC_BACNET_DEV_OBJ_PROP_VALUE</td>
</tr>
<tr>
<td>DATA_TYPE_ACCUMULATOR_RECORD</td>
<td>IEC_BACNET_ACCUMULATOR_RECORD</td>
</tr>
<tr>
<td>DATA_TYPE_PRESCALE</td>
<td>IEC_BACNET_PRESCALE</td>
</tr>
<tr>
<td>DATA_TYPE_SCALE</td>
<td>IEC_BACNET_SCALE</td>
</tr>
<tr>
<td>DATA_TYPE_EVENT_LOG_RECORD</td>
<td>IEC_BACNET_EVENT_LOG_RECORD</td>
</tr>
<tr>
<td>DATA_TYPE_LOG_RECORD_MULTIPLE</td>
<td>IEC_BACNET_LOG_RECORD_MULTIPLE</td>
</tr>
<tr>
<td>DATA_TYPE_SHED_LEVEL</td>
<td>IEC_BACNET_SHED_LEVEL</td>
</tr>
<tr>
<td>DATA_TYPE_AUTHENTICATION_POLICY</td>
<td>IEC_BACNET_AUTHENTICATION_POLICY</td>
</tr>
<tr>
<td>DATA_TYPE_AUTHENTICATION_FACTOR</td>
<td>IEC_BACNET_AUTHENTICATION_FACTOR</td>
</tr>
<tr>
<td>DATA_TYPE_CREDENTIAL_AUTHENTICATION_FACTOR</td>
<td>IEC_BACNET_CREDENTIAL_AUTHENTICATION_FACTOR</td>
</tr>
<tr>
<td>DATA_TYPE_ACCESS_RULE</td>
<td>IEC_BACNET_ACCESS_RULE</td>
</tr>
<tr>
<td>DATA_TYPE_ASSIGNED_ACCESS_RIGHTS</td>
<td>IEC_BACNET_ASSIGNED_ACCESS_RIGHTS</td>
</tr>
<tr>
<td>DATA_TYPE_AUTHENTICATION_FACTOR_FORMAT</td>
<td>IEC_BACNET_AUTHENTICATION_FACTOR_FORMAT</td>
</tr>
<tr>
<td>DATA_TYPE_OPTIONAL_STRING</td>
<td>IEC_BACNET_OPTIONAL_STRING</td>
</tr>
<tr>
<td>DATA_TYPE_PROPERTY_ACCESS_RESULT</td>
<td>IEC_BACNET_PROPERTY_ACCESS_RESULT</td>
</tr>
</tbody>
</table>
### IEC_BACNET_DATA_TYPE

<table>
<thead>
<tr>
<th>Codesys data type (struct)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DATA_TYPE_VT_SESSION</td>
<td>IEC_BACNET_VT_SESSION</td>
</tr>
<tr>
<td>DATA_TYPE_SESSION_KEY</td>
<td>IEC_BACNET_SESSION_KEY</td>
</tr>
<tr>
<td>DATA_TYPE_NETWORK_SECURITY_POLICY</td>
<td>IEC_BACNET_NETWORK_SECURITY_POLICY</td>
</tr>
<tr>
<td>DATA_TYPE_KEY_IDENTIFIER</td>
<td>IEC_BACNET_KEY_IDENTIFIER</td>
</tr>
<tr>
<td>DATA_TYPE_SECURITY_KEY_SET</td>
<td>IEC_BACNET_SECURITY_KEY_SET</td>
</tr>
<tr>
<td>DATA_TYPE_NEW_STANDARD</td>
<td>IEC_BACNET_RAW_ASN1_VALUE</td>
</tr>
<tr>
<td>DATA_TYPE_PROPRIETARY</td>
<td>IEC_BACNET_RAW_ASN1_VALUE</td>
</tr>
<tr>
<td>DATA_TYPE_EXTENDED</td>
<td>IEC_BACNET_RAW_ASN1_VALUE</td>
</tr>
<tr>
<td>DATA_TYPE_RAW_ASN1_VALUE</td>
<td>IEC_BACNET_RAW_ASN1_VALUE</td>
</tr>
</tbody>
</table>

The basic function block BACnetObjectBase provides auxiliary functions (for example, SetBACnetRealProperty, SetBACnetBooleanProperty) for some of the frequently used data types.

#### Example of the correct usage of IEC_BACNET_PROPERTY_CONTENTS

```plaintext
METHOD SetBACnetRealProperty
VAR_INPUT
    propID: CmpBACnet.IEC_BACNET_PROPERTY_ID;
    rVal : REAL;
END_VAR
VAR
    BACreal : CmpBACnet.IEC_BACNET_REAL := 0.0;
    conts : CmpBACnet.IEC_BACNET_PROPERTY_CONTENTS := ( tag :=
        CmbBACnet.IEC_BACNET_DATA_TYPE.DATA_TYPE_REAL,
        nElements := 1,
        buffer :=
            ( nBufferSize := SIZEOF(LREAL), pBuffer := ADR(BACreal) ),
        rawBuffer := ( nBufferSize := 0, pBuffer :=0 ) );
END_VAR
BACreal := rVal;
SetPropertyInstance(propID, -1, conts);
```

This code sets a BACnet property of the type REAL to the value 0.0.

Accordingly, in order to read BACnet properties in the most general case, the BACnetObjectBase function block provides the GetPropertyInstance function. Within the IEC_BACNET_PROPERTY_CONTENTS structure, you can transfer either 1) a buffer pre-initialized with the correct data type, which is then filled, or 2) a buffer of size 0, which is initialized automatically, but which you release again by means of the FreeStackAllocatedMemory function.

Regardless of this general case, auxiliary functions are provided for the most frequent cases. For the most frequently used BACnet properties, there is also a selection of function block properties for each BACnet object function block that allow direct access.
Use of BACnet client function blocks

In addition to BACnet objects, BACnet clients can also be inserted as “devices” under a BACnet server.

Like the objects and the server, each client device also brings along a function block that is instantiated under the same name as the client device.

Unlike BACnet objects, a BACnet client does not require a complex (static) configuration, thus a client function block can be used without creating a BACnet client as "device". In this case however it must be "connected" to its BACnetServer in IEC-code via RegisterToServer(), and thus get activated. This "activation" takes place automatically, when a BACnet client is inserted as “device”.

The BACnet client function blocks implement the "Common Behaviour Model", and so they provide inputs and outputs for flow control.

- Inputs xExecute and xAbort
- Outputs xBusy, xDone, xError and xAborted
- Inputs and outputs for the client request parameters and results
The following code uses the function module BACnetClientReadProperty from the BACnet library to send a client read property request to a specific BACnet client device in the BACnet network. The received response is read out into a local variable:

```plaintext
PROGRAM PLC_PRG
VAR
  initDone : BOOL := FALSE;
  readProp : BACnet.BACnetClientReadProperty;
  readPropCnt : INT := 0;
  readPropErrorCnt : INT := 0;
  readPropCancelCnt : INT := 0;
  readPropVal : REAL;
END_VAR

IF NOT initDone THEN
  readProp:RegisterToServer(BACnet_Server);
  readProp(
    dwTargetDeviceNumber:= 43,
    objType:=
      BACnet.CmpBACnet.IEC_BACNET_OBJECT_TYPE.OBJ_ANALOG_VALUE,
    objInst:= 1,
    propID:=
      BACnet.CmpBACnet.IEC_BACNET_PROPERTY_ID.PROP_PRESENT_VALUE,
    nIndex:= -1);
END_IF

IF readProp.xExecute AND readProp.xBusy AND readProp.iState = 3 THEN
  IF readPropCnt MOD 2 = 0 THEN // cancel every second request
    readProp.xAbort := TRUE;
    readPropCancelCnt := readPropCancelCnt + 1;
  END_IF
END_IF

IF NOT readProp.xExecute AND NOT readProp.xAborted AND NOT readProp.xDone THEN
  readProp.xExecute := TRUE;
  readPropCnt := readPropCnt + 1;
END_IF

IF readProp.xExecute THEN
  IF readProp.xDone THEN
    END_IF
    readProp.xExecute := FALSE;
  END_IF
  IF readProp.xError THEN
    END_IF
    readProp.xExecute := FALSE;
    readPropErrorCnt := readPropErrorCnt + 1;
  END_IF
  IF readProp.xAborted THEN
    END_IF
    readProp.xExecute := FALSE;
    readProp.xAbort := FALSE;
  END_IF
END_IF

readProp();

IF readProp.xDone OR readProp.xError THEN
  IF readProp.xError THEN
    // handle error
  ;
```

Example of IEC code for a client action
If a client request provides a response, the response content is provided in the form of one or
more output variables, as in the above example. However, this may vary depending on the
client action. The example additionally uses the auxiliary function `GetRealFromContents()`
to read the output variable `readProp.result` of type `IEC_BACNET_PROPERTY_CONTENTS`.

See also
- % Chapter 1.5.6.3.8.1 “Overview” on page 3439

**BACnet data types**

The BACnet library mostly uses BACnet data types - defined in CmpBACnet library - at their
interface (method arguments, method return values, function block properties)

If other data types are used on the interface - in this case mostly IEC basic data types - the cor-
responding methods and properties perform an implicit conversion to / from the corresponding
internally used BACnet data types.

Examples of this:

<table>
<thead>
<tr>
<th>Conversion</th>
<th>Using BOOL (TRUE/FALSE) instead of CmpBACnet.IEC_BACNET_BOOLEAN improves the readability of the IEC code.</th>
</tr>
</thead>
<tbody>
<tr>
<td>IEC BOOL &lt;-&gt; CmpBACnet.IEC_BACNET_BOOLEAN</td>
<td>Using BOOL (TRUE/FALSE) instead of CmpBACnet.IEC_BACNET_BOOLEAN improves the readability of the IEC code.</td>
</tr>
<tr>
<td>IEC WSTRING &lt;-&gt; CmpBACnet.IEC_BACNET_STRING</td>
<td>The use of WSTRING is necessary, since a conversion to/from the internally used BAC data types CmpBACnet.IEC_BACNET_STRING_TYPE.BACNET_STRING_UTF_8 or CmpBACnet.IEC_BACNET_STRING_TYPE.BACNET_STRING_ISO_8859_1 takes place.</td>
</tr>
<tr>
<td>IEC DATE/TIME / BACnetDateTime &lt;-&gt; CmpBACnet.IEC_BACNET_DATE/ CmpBACnet.IEC_BACNET_TIME</td>
<td>The use of IEC DATE/TIME is more common for the IEC programmer and is therefore preferred on the interface.</td>
</tr>
</tbody>
</table>

At the BACnet library interface, BACnet data types other than the BACnet data types are only
used in well-founded exceptions to simplify the application IEC code. The usually minimal over-
head coming with a conversion is negligible.

Siehe auch
- % Chapter 1.5.6.3.8.1 “Overview” on page 3439

**Implementation Requirements when Using the Current BACnet API/Stack**

In the implementation of a BACnet API/stack described here, a large part of the object behavior
specified in the BACnet standard is already pre-implemented. However, there are specific object
actions that have to be implemented by each specific application (the user application, or in
other words your application) in order to fully comply with the BACnet standard (for example, to
obtain BTL certification).
This description lists all object actions and object behavior that are already implemented or not implemented in the current BACnet API/stack (see § Chapter 1.5.6.3.8.10.1 “Implemented Standard Object Actions and Behavior in the Current BACnet API/Stack” on page 3454 and § Chapter 1.5.6.3.8.10.2 “Objects with Implementation of Standard Actions” on page 3460). Moreover, you will find descriptions and an example of how missing object behavior is implemented when using the current BACnet API/stack (see § Chapter 1.5.6.3.8.10.3 “Example and Explanations for the Implementation of Service Hooks” on page 3462).

See also
● § Chapter 1.5.6.3.8.1 “Overview” on page 3439

### Implemented Standard Object Actions and Behavior in the Current BACnet API/Stack

<table>
<thead>
<tr>
<th>Subsection</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.5.6.3.8.10.1.1</td>
<td>Service indication hooks with default action implementation</td>
</tr>
<tr>
<td>1.5.6.3.8.10.1.2</td>
<td>Service indication hooks with internal used implementation</td>
</tr>
<tr>
<td>1.5.6.3.8.10.1.3</td>
<td>Service indication hooks without default action implementation</td>
</tr>
</tbody>
</table>

Many service requests are already handled in this API so that an application does not have to register all service hooks to run as a BACnet device. Service hooks describe the BACnet network requests that have to be resolved in the manner defined in the BACnet standard.

### Service indication hooks with default action implementation

Here is a list of hooks that already have some default action implemented and normally do not need any implementation in the user application:

- **AcknowledgeAlarm service hook**
  After verifying the validity of the request, the API will attempt to locate the specified object. If the object exists and if the Time Stamp parameter matches the most recent time for the event being acknowledged, then the following will happen: The bit in the Acked_Transitions property of the object that corresponds to the value of the Event State Acknowledged parameter will be set to 1, a result(+) primitive will be issued, and an event notification with a Notify Type parameter equal to ACK_NOTIFICATION will be issued. Otherwise, a result(-) primitive will be issued. An acknowledgment notification will use the same type of service (confirmed or unconfirmed) directed to the same recipients to which the original confirmed or unconfirmed event notification was sent. The Time Stamp conveyed in the acknowledgment notification will not be derived from the Time Stamp of the original event notification, but rather the time at which the acknowledgment notification is generated.

- **GetAlarmSummary service hook**
  After verifying the validity of the request, the API will search all event-initiating objects that have an Event State property not equal to NORMAL and a Notify Type property whose value is ALARM. A positive response containing the alarm summaries for objects found in this search will be constructed. If no objects are found that meet these criteria, then a list of length zero will be returned.

- **GetEnrollmentSummary service hook**
  After verifying the validity of the request, the API will search for all event-initiating objects that meet the search criteria specified in the request primitive. The search criteria are the logical conjunctions of all of the explicitly stated filters and the default values for any filters that were omitted in the request primitive. A positive response containing the enrollment summaries for objects found in this search will be constructed. If no objects are found that meet these criteria, then a list of length zero will be returned.
SubscribeCOV service hook

If neither Lifetime nor Issue Confirmed Notifications are present, then the request will be considered to be a cancellation. Any COV context that already exists for the same BACnet address contained in the PDU that carries the SubscribeCOV request and has the same Subscriber Process Identifier and Monitored Object Identifier will be disabled and a SimpleAck is returned. Cancellations that are issued for which no matching COV context can be found will succeed as if a context had existed, returning SimpleAck. If the Lifetime parameter is not present but the Issue Confirmed Notifications parameter is present, then a value of zero (indefinite lifetime) will be assumed for the lifetime. If the Issue Confirmed Notifications parameter is present but the object to be monitored does not support COV reporting, then an error will be returned. If the object to be monitored does support COV reporting, then a check will be made to locate an existing COV context for the same BACnet address contained in the PDU that carries the SubscribeCOV request and has the same Subscriber Process Identifier and Monitored Object Identifier. If an existing COV context is found, then the request will be considered a re-subscription and will succeed as if the subscription had been newly created. If no COV context can be found that matches the request, then a new COV context will be established that contains the BACnet address from the PDU that carries the SubscribeCOV request and the same Subscriber Process Identifier and Monitored Object Identifier. If no context can be created, then an error will be returned. If a new context is created, or a re-subscription is received, then the COV context will be initialized and given a lifetime as specified by the Lifetime parameter if present, or zero if the Lifetime parameter is not present. The subscription will be automatically cancelled after that many seconds have elapsed unless a re-subscription is received. A lifetime of zero will indicate that the subscription is indefinite and no automatic cancellation will occur. In either case, a SimpleAck will be returned. A ConfirmedCOVNotification or UnconfirmedCOVNotification will be issued as soon as possible after the successful completion of a subscription or re-subscription request, as specified by the Issue Confirmed Notifications parameter.

AddListElement service hook

After verifying the validity of the request, the API will attempt to modify the object identified in the Object Identifier parameter. If the identified object exists and has the property specified in the Property Identifier parameter, an attempt will be made to add all of the elements specified in the List of Elements parameter to the specified property. If this attempt is successful, a SimpleAck primitive will be issued. If one or more of the elements is already present in the list, it will be ignored, that is, will not be added to the list. Ignoring an element that already exists will not cause the service to fail. If the specified object does not exist, the specified property does not exist, or the specified property is not a list, then the service will fail and an error response primitive will be issued. If one or more elements cannot be added to the list and they are not already members, an error response primitive will be issued and no elements will be added to the list. The effect of this service will be to add to the list all of the specified elements that are not already present or to add no elements to the list at all.

RemoveListElement service hook

After verifying the validity of the request, the API will attempt to modify the object identified in the Object Identifier parameter. If the identified object exists and it has the property specified in the Property Identifier parameter, an attempt will be made to remove the elements in the List of Elements from the property of the object. If one or more of the elements does not exist or cannot be removed because of insufficient authority or other problems, none of the elements will be removed and an error response primitive will be issued.

ReadProperty service hook

After verifying the validity of the request, the API will attempt to access the specified property of the specified object. If the access is successful, a ComplexAck primitive, which returns the accessed value, will be generated. If the access fails, an error primitive will be generated, indicating the reason for the failure. When the object-type in the Object Identifier parameter contains the value Device Object and the instance in the Object Identifier parameter contains the value 4194303, the API will treat the Object Identifier as if it correctly matched the local Device object. This allows the device instance of a device that does not generate I-Am messages to be determined.
**ReadPropertyMultiple service hook**

After verifying the validity of the request, the API will attempt to access the specified properties of the specified objects and will construct a List of Read Access Results in the order specified in the request. If the List of Property References portion of the List of Read Access Specifications parameter contains the property identifier ALL, REQUIRED, or OPTIONAL, then the List of Read Access Results will be constructed as if each property being returned had been explicitly referenced. While there is no requirement that the request be carried out “automatically” nonetheless the API will ensure that all readings are taken in the shortest possible time subject only to higher priority processing. The request will continue to be executed until an attempt has been made to access all specified properties. If none of the specified objects is found or if none of the specified properties of the specified objects can be accessed, either an error primitive or a ComplexAck primitive that returns error codes for all properties that could not be accessed. When the object-type in the Object Identifier portion of the Read Access Specification parameter contains the value Device Object and the instance of that Object Identifier parameter contains the value 4194303, the API will treat the Object Identifier as if it correctly matched the local Device object. This allows the device instance of a device that does not generate I-Am messages to be determined.

For BACnet server user applications, which can't use the internal database and thus have to implement support for the ReadPropertyMultiple service, they only need to implement the ReadProperty service hook because ReadPropertyMultiple requests are split into single ReadProperty requests internally and completions are collected to build the response.

**WriteProperty service hook**

After verifying the validity of the request, the API will attempt to modify the specified property of the specified object using the value provided in the Property Value parameter. If the modification attempt is successful, a SimpleAck primitive will be issued. If the modification attempt fails, an error primitive will be issued indicating the reason for the failure.

**WriteProperty Multiple service hook**

For each Write Access Specification contained in the List of Write Access Specifications, the value of each specified property will be replaced by the property value provided in the Write Access Specification and a SimpleAck primitive will be issued, indicating that the service request was carried out in its entirety. If, in the process of carrying out the modification of the indicated properties in the order specified in the List of Write Access Specifications, a property is encountered that cannot be modified, the PI will issue an error response primitive indicating the reason for the failure. The result of this service will be either that all of the specified properties or only the properties up to, but not including, the property specified in the First Failed Write Attempt parameter were successfully modified. A BACnet-Reject-PDU will be issued only if no write operations have been successfully executed, indicating that the service request was rejected in its entirety. If any of the write operations contained in the List of Write Access Specifications could not be successfully executed, an error response indicating the reason for the failure will be issued as described above.

**ReadRange service hook**

The API will first verify the validity of the Object Identifier, Property Identifier and Property Array Index parameters and will return an error response with the appropriate error class and code if the object or property is unknown, if the referenced data is not a list or array, or if it is currently inaccessible for another reason. If the Range parameter is not present, then the API will read and attempt to return all of the available items in the list or array. If the Range parameter is present and specifies the By Position parameters, then the API will read and attempt to return all of the items specified. The items specified include the item at the index specified by Reference Index plus up to Count - 1 items following if Count is positive, or up to -1 - Count items preceding if Count is negative. The first element of a list will be associated with index 1. If the Range parameter is present and specifies the By Time parameter or the By Sequence Number parameters, then the API will reject the request by returning an error response with the appropriate error class and code. The returned response will convey the number of items read and returned using the Item Count parameter. The actual items will be returned in the Item Data parameter. If the returned response includes the first positional
index, then the Result Flags parameter will contain the FIRST_ITEM flag set to TRUE; otherwise it will be FALSE. If the returned response includes the last positional index, then the Result Flags will contain the LAST_ITEM flag set to TRUE; otherwise it will be FALSE. If there are no items in the list that match the Range parameter criteria, then a ComplexAck will be returned with an Item Count of 0 and no First Sequence Number parameter.

The default implementation of this service will NOT support the access to the log-buffer property of the trend- and event-logging objects. If a user application needs to support logging objects, the storage and the access to those data has to be implemented by the user application.

**SubscribeCOVProperty service hook**

If neither Lifetime nor Issue Confirmed Notifications are present, then the request will be considered to be a cancellation. Any COV context that already exists for the same BACnet address contained in the PDU that carries the SubscribeCOVProperty request and has the same Subscriber Process Identifier, Monitored Object Identifier and Monitored Property Identifier will be disabled and a SimpleAck is returned. Cancellations that are issued for which no matching COV context can be found will succeed as if a context had existed, returning SimpleAck. If an existing COV context is found, it will be removed from the Active_COV_Subscriptions property in the Device object. If the Issue Confirmed Notifications parameter is present but the property to be monitored does not support COV reporting, then an error will be returned. If the property to be monitored does support COV reporting, then a check will be made to locate an existing COV context for the same BACnet address contained in the PDU that carries the SubscribeCOVProperty request and has the same Subscriber Process Identifier, Monitored Object Identifier and Monitored Property Identifier. If an existing COV context is found, then the request will be considered a re-subscription and will succeed as if the subscription had been newly created. If no COV context can be found that matches the request, then a new COV context will be established that contains the BACnet address from the PDU that carries the SubscribeCOVProperty request and the same Subscriber Process Identifier, Monitored Object Identifier and Monitored Property Identifier. The new context will be included in the Active_COV_Subscriptions property of the Device object. If no context can be created, then an error will be returned. If a new context is created, or a re-subscription is received, then the COV context will be initialized and given a lifetime as specified by the Lifetime parameter. The subscription will be automatically cancelled after that many seconds have elapsed unless a re-subscription is received. A SimpleAck will be returned and a ConfirmedCOVNotification or UnconfirmedCOVNotification will be issued as soon as possible after the successful completion of a subscription or resubscription request, as specified by the Issue Confirmed Notifications parameter.

**GetEventInformation service hook**

After verifying the validity of the request, the API will search for all event-initiating objects that meet the following conditions, beginning with the object following (in ascending internal ordering of object instances) the object specified by the Last Received Object Identifier parameter, if present have an Event_State property whose value is not equal to NORMAL, or have an Acked_Transitions property that has at least one of the following bits (TO-OFFNORMAL, TO-FAULT, TO-NORMAL) set to FALSE. A positive response containing the event summaries for objects found in this search shall be constructed. If no objects are found that meet these criteria, then a list of length zero will be returned. As many of the included objects as can be returned within the APDU will be returned. If more objects exist that meet the criteria but cannot be returned in the APDU, the More Events parameter will be set to TRUE, otherwise it will be set to FALSE.

**I-Am service hook**

After verifying the validity of the request, the API will add the received information to the device object property Address_Binding which is used to resolve device instance numbers into BACnet MAC addresses which are needed for communication with that device.

**I-Have service hook**

After verifying the validity of the request, the API will add the received information to an internal sorted list which is used to resolve object names into object identifiers which are needed for accessing those objects.
Who-Has service hook

The API will transmit the Who-Has unconfirmed request, normally using a broadcast address. If the Device Instance Range Low Limit and Device Instance Range High Limit parameters are present, then the API whose Device Object-Identifier instance number falls in the range Device Instance Range Low Limit <= Device Object-Identifier Instance Number <= Device Instance Range High Limit will be qualified to respond. If the Object Name parameter is present, then the API receiving that contain an object with an Object Name property value matching the Object Name parameter will respond with an I-Have service request. If the Object Identifier parameter is present, then the API receiving that contain an object with an Object Identifier property value matching the Object Identifier parameter will respond with an I-Have service request.

Who-Is service hook

The API will transmit the Who-Is unconfirmed request, normally using a broadcast address. If the Device Instance Range Low Limit and Device Instance Range High Limit parameters are omitted, then the API will return their Device Object-Identifier in individual responses using the I-Am service. If the Device Instance Range Low Limit and Device Instance Range High Limit parameters are present, then the API whose Device Object-Identifier instance number falls within the range Device Instance Range Low Limit <= Device Object-Identifier Instance Number <= Device Instance Range High Limit will return their Device Object-Identifier using the I-Am service.

AtomicReadFile service hook

The API will first verify the validity of the Object Identifier and will return an error response result(-) with the appropriate error class and code if the object is unknown. This service hook only supports STREAM accessed files. After that the validity of the local file access on the local hard drive is checked and an error response result(-) with the appropriate error class and code will be returned if the file start position is invalid or if the file is not accessible for some reason. If the validity check is successful the specified number Request Record Count of octets is read from the local file. If the actual number of read octets is less than the specified one, the Returned Record Count of the returned result(+) indicates the number of read octets and if the end of the file was reached the End Of File Parameter of the returned result(+) is set to TRUE.

This service hook is implemented using the SysFile-Interface of CODESYS only supporting STREAM access. Consequently the user application may want to change especially this service hook for the file access.

AtomicWriteFile service hook

The API will first verify the validity of the Object Identifier and will return an error response result(-) with the appropriate error class and code if the object is unknown. This service hook only supports STREAM accessed files. After that the validity of the local file access on the local hard drive is checked and an error response result(-) with the appropriate error class and code will be returned if the file start position is invalid or if the file is not accessible for some reason. If the File Start Position parameter of the service request exceeds the local file size, the file will be extended for writing the incoming octets. If the File Start Position parameter has the value -1, the local file will get appended by the incoming octets. If the writing of the local file failed, an appropriate result(-) response will be issued. If the writing of the local file succeeded, a result(+) will be issued containing the appropriate File Start Position.

This service hook is implemented using the SysFile-Interface of CODESYS only supporting STREAM access. Consequently the user application may want to change especially this service hook for the file access.

CreateObject service hook

The API will first verify the validity of the request by checking if the Object Specifier of the request specifies a creatable object type (according to the intern rule $\text{IsBACnetObjectAMVECreatable}$) and will return an appropriate error result(-) if that is not the case. If specified by the Object Specifier the validity of the object instance number is checked and an appropriate error result(-) is returned if the instance number is not available anymore. If no List of Initial Values is given, all properties of the created object
are created using default values as it is done for properties not specified in a given List of Initial Values. If the creation of the object fails an appropriate error result(-) is issued, whereas the First Failed Element Number of the response is set to 0. If the creation succeeded an result(+) response is issued, which carries the object identifier of the created object.

This implementation of this service hook has to be accounted for to be only a comprehensive proposal for its implementation. As it is a purely local matter of the user application how this service request is treated it is very likely, that the user application wants to supply its own service hook for another treatment of the request.

DeleteObject service hook

The API will first verify the validity of the Object Identifier and will return an error response result(-) with the appropriate error class and code if the object is unknown. If the specified object can be deleted a result(+) response is issued, if the deletion of the objects fails, i.e. the object can not be dynamically deleted (according to the intern rule BACnet.IsBACnetObjectAMEVCreatable), or for any other reason a result(-) is issued.

This implementation of this service hook has to be accounted for to be only a comprehensive proposal for its implementation. As it is a purely local matter of the user application how this service request is treated it is very likely, that the user application wants to supply its own service hook for another treatment of the request.

See also

- § Chapter 1.5.6.3.8.10 “Implementation Requirements when Using the Current BACnet API/Stack” on page 3453
- § Chapter 1.5.6.3.8.1 “Overview” on page 3439

Service indication hooks with internal used implementation

The following listed hooks do not have any default action implemented, but the advanced client API uses this set of functions to acquire the data from remote devices:

- ConfirmedCOVNotification service hook
- ConfirmedEventNotification service hook
- UnconfirmedCOVNotification service hook
- UnconfirmedEventNotification service hook

These hooks for example can be used, to receive the answered information after a SubscribeCOVProperty request, i.e. after a subscription of a certain Property of an Object of a Device the ConfirmedCOVNotification hook will be called whenever the latter Device send a COVNotification to inform about a change of the Property.

See also

- § Chapter 1.5.6.3.8.10 “Implementation Requirements when Using the Current BACnet API/Stack” on page 3453
- § Chapter 1.5.6.3.8.1 “Overview” on page 3439

Service indication hooks without default action implementation

The following listed hooks do not have any default action implemented. The application needs to implement its own request handling routine if the service has to be supported:

- DeviceCommunicationControl service hook
- ReinitializeDevice service hook
- TimeSynchronization service hook
- UtcTimeSynchronization service hook
The hooks listed here may need to be implemented by the user application (depending on the user application and its specifications concerning the BACnet standard), since no pre-implementation can be done due to the hardware correlation of this hooks.

See also
- Chapter 1.5.6.3.8.10 “Implementation Requirements when Using the Current BACnet API/Stack” on page 3453
- Chapter 1.5.6.3.8.1 “Overview” on page 3439

Objects with Implementation of Standard Actions

1.5.6.3.8.10.2.1 Common object property actions.................................................. 3460
1.5.6.3.8.10.2.2 Objects without default action implementation.............................. 3462

Common object property actions

Following common object property actions are implemented for all objects which have those listed properties.

Property Status_Flags

The Status_Flags property has a bit-string value which has 4 bits. The values of 3 of those bits are related to the values of other properties in the same object. This following relationship is defined in the BACnet standard and is implemented as actions in this API: Bit IN_ALARM is logical FALSE if the Event_State property has a value of NORMAL, otherwise logical TRUE. Bit FAULT is logical TRUE if the Reliability property is present and does not have a value of NO_FAULT_DETECTED, otherwise logical FALSE. Bit OUT_OF_SERVICE is logical TRUE if the Out_Of_Service property is present and has a value of TRUE, otherwise logical FALSE.

Property Out_Of_Service

The Out_Of_Service property, of type BOOL, is an indication whether (TRUE) or not (FALSE) the physical point that the object represents is not in service. In case that it is not in service, different other properties need to be writable. For e.g. the Present_value and Reliability property need to be writable to simulate different events and states an object supports which will not occur during normal operation. So if the Out_Of_Service property is set to TRUE those properties will be writable, even if the user application had specified NO access rights to them.

Properties related to intrinsic reporting

Objects which have defined intrinsic reporting procedures in the standard and which are also supported by this API, will have updated their properties related to intrinsic reporting automatically by internal actions in the API. For e.g. the property Event_State will be set to the appropriate enumerated value for the intrinsic state that the object is in. The property Event_Time_Stamp is updated when the event notification is sent. Bits in the property Acked_Transitions are modified accordingly too.

Properties related to change of value reporting

Objects which have defined COV reporting procedures in the standard and which are also supported by this API, will have updated their properties related to COV reporting automatically by internal actions in the API. Also sending change of value notifications to registered subscribers to objects or object properties is done by the API.

Properties related to command prioritization

For BACnet objects, commands are prioritized based upon a fixed number of priorities that are assigned to command-issuing entities. A prioritized command (one that is directed at a commandable property of an object) is performed via a WriteProperty service request or a WritePropertyMultiple service request. The request primitive includes a conditional Priority parameter that ranges from 1 to 16. Each commandable property of an object has an associated priority table that is represented by the Priority_Array property. The Priority_Array consists of an array of commanded values in order of decreasing priority. The first value in the array corresponds to priority 1 (highest), the second value corresponds to
priority 2, and so on, to the sixteenth value that corresponds to priority 16 (lowest). If such a prioritized command is received by the API all necessary actions described in clause 19 in the BACnet standard will take place by internal actions to handle the prioritized command correctly. If the user application has registered callbacks for the properties Present_value and Priority_Array it will get informed about the current command state in the object.

**Accumulator object**

The Accumulator object type defines a standardized object whose properties represent the externally visible characteristics of a device that indicates measurements made by counting pulses. The pulse injection to this object can be done either by adding the proprietary property Prop_Pulse_Value_Source and filling it with a reference to a BINARY or BOOLEAN value source or by calling the BACnet.BACnetServer.UpdateAccumulatorDataSourceValue function, which is available for every inserted BACnet server in the CODESYS device tree. In any case toggling values will be counted by the object and all property actions defined in the BACnet standard will be done by the API. The intrinsic reporting procedure is also handled by internal actions in this API for this object.

**Analog-Input object**

The Analog-Input object type defines a standardized object whose properties represent the externally visible characteristics of a physical analog input. The properties Min_Pres_Val and Max_preset_Val which indicate the lowest and highest number in engineering units that can be reliably obtained for the Present_Value property of this object, will generate value out of range errors, if the Present_Value is set out of that range. Also the values of properties High_Limit, Low_Limit and Deadband are observed if they get outside this range. The optional present property Resolution, which specifies the smallest recognizable change in Present_Value in engineering unit, influences the setting of the Present_Value property directly and will adjust its value automatically. The Change-Of-value reporting procedure and the intrinsic reporting procedure are fully implemented by internal actions of this API for this object type.

**Analog-Output object**

The Analog-Output object type defines a standardized object whose properties represent the externally visible characteristics of a physical analog output. All actions implemented for the Analog-Input object are also implemented for this object type.

**Analog-Value object**

The Analog-Value object type defines a standardized object whose properties represent the externally visible characteristics of a analog value. An "analog value" is a control system parameter residing in the memory of the BACnet Device and has no physical device. All actions implemented for the Analog-Input object are also implemented for this object type.

**Averaging object**

The Averaging object type defines a standardized object whose properties represent the externally visible characteristics of a value that is sampled periodically over a specified time interval. The Averaging object records the minimum, maximum and average value over the interval, and makes these values visible as properties of the Averaging object. The sampled value may be the value of any BOOLEAN, INTEGER, UNSIGNED, ENUMERATED or REAL property value. The value of the over Object_Property_Reference referenced object is acquired and all depending property values such as Minimum_Value, Maximum_Value, Average_Value and others will be updated by internal actions in the API.

The common property actions implemented by this API also apply for further objects.

See also

- % Chapter 1.5.6.3.8.10 “Implementation Requirements when Using the Current BACnet API/Stack” on page 3453
- % Chapter 1.5.6.3.8.1 “Overview” on page 3439
Objects without default action implementation

In the present version of this API there is no internal action implemented to any to this object type related properties: Notification-Class object, Program object, Structured-View object.

The object types listed here correspond to the object types available in the CODESYS device tree as devices. Other object types may be creatable using the CmpBACnet library. For those object the general property actions described beforehand hold also, but no additional property actions of not listed objects and not listed properties are pre-implemented by this API.

See also

- % Chapter 1.5.6.3.8.10 “Implementation Requirements when Using the Current BACnet API/Stack” on page 3453
- % Chapter 1.5.6.3.8.1 “Overview” on page 3439

Example and Explanations for the Implementation of Service Hooks

1.5.6.3.8.10.3.1 CreateObject service hook implementation example........... 3462
1.5.6.3.8.10.3.2 Implementation of logging objects (Event-Log, Trend-Log, Trend-Log-Multiple).............................................................. 3467

This part provides a detailed description of the implementation of a service hook using the CreateObject hook as an example. Furthermore, the special behavior of the logging objects (event log object, trend log object, and trend log multiple object) is described and explained how their missing object actions can be implemented.

CreateObject service hook implementation example

The following example code shows how to implement a CreateObject service hook using the BACnet Stack/API.

First thing to do is to register the corresponding hook within any place in your program code. This is done by registering an FB CallbackFB implementing the Interface IBACnetEventConsumer using the function RegisterHook of the BACnetServer-FB meaning the FB inserted for the local BACnetServer-Device in the CODESYS device tree:

BACnetServer.RegisterHook(CmpBACnet.IEC_BACNET_CB_TYPE.CB_CREATE_OBJEC T, CallbackFB);

Having done so, this FBs function BACnetEventCallback (defined by the interface IBACnetEventConsumer) will be called whenever a CreateObject service request is collected by the BACnetServer (in other words the BACnet device). The following sample code demonstrates an implementation of the BACnetEventCallback function of the CallbackFB. It has the following declaration:

METHOD BACnetEventCallback : BOOL
VAR_INPUT
  /*If only one event is possibly called, the following Ids can be ignored. If there are
   * more at least the EventId should be checked, to get the
   * right type of pParameter. ID of the occured Event. Contains the
   * class and the event.*)
  EventId : DWORD;
  /*Id of the parameter structure (see pParameter)*/
usParamId : WORD;
  /*Version of the parameter structure (see pParameter)*/
usVersion : WORD;
  /*Pointer to the event specific parameter, that is specified by
  * Id (see GVL EVENTID in
  * CmpBACnet)*/
Id (see GVL EVENTID in
  CmpBACnet)*/
Notice the `pCreateObj` variable which will carry the essential information of the `CreateObject` request. The code the `BACnetEventCallback` function of the `CallbackFB` could be implemented like this:

```pascal
BACnetEventCallback := FALSE;

//Check if the issued service event is the one we are interested in
//meaning the create object event
IF EventId = CmpBACnet.EVT_CreateObject AND
   usParamId = CmpBACnet.EVTPARAMID_CmpBACnet_CreateObject AND
   usVersion = CmpBACnet.EVTVERSION_CmpBACnet_CreateObject THEN

   //Get the service information using the proper
   //event structure CmpBACnet.EVT_BACNET_CREATEOBJECT
   pCreateObj := pParameter;

   //The pCBStatus property of the event structure allows to define
   //the result of this service hook. Choosing the default action
   //will advice the API to use its default implementations if available
   //see also documentation of CB_STATUS_DEFAULT)
   pCreateObj^.pCBStatus^ := CmpBACnet.IEC_BACNET_CB_STATUS.CB_STATUS_DEFAULT;

   //Get the device instance number for the service request and
   //check if it is
   //our device that the request is for (only important for the case
   //of multiple
   //devices in one PLC)
   bacResult := CmpBACnet.BACnetDeviceAddressToInstNumber(pCreateObj^.destinationAddress);
   ADR(destInstNumber);
   IF bacResult = CmpBACnet.IEC_BACNET_STATUS.BACNET_STATUS_OK AND
      destInstNumber = m_Device.DeviceInstNumber THEN

      //Get the object specifier defining what to create
      objSpec := pCreateObj^.pServiceInfo^.objectSpecifier;
      //Check if the
      object type specified is creatable
      IF IsBACnetObjectAMEVCreatable(objSpec.objectSpec.typ) THEN
```
no object ID is given find a free one
objSpec.tag =

    CmpBACnet.IEC_BACNET_CREATE_OBJECT_TYPE.CREATE_OBJECT_TYPE
THEN

    objID.typ := objSpec.objectSpec.typ;
    objID.instNumber := 1;
    objCount := GetObjectCount();
    i := 1;

    WHILE i <= objCount DO
        GetObject(i, bacObj);
        runObjID := bacObj.ObjectID;
        IF runObjID.typ = objID.typ AND
            runObjID.instNumber = objID.instNumber THEN
            objID.instNumber := objID.instNumber + 1;
            i := 1;
        END_IF
        i := i + 1;
    END WHILE

    BACnetEventCallback := TRUE;

//If the object ID is given check if it is a valid/free object ID
ELSIF objSpec.tag =

    CmpBACnet.IEC_BACNET_CREATE_OBJECT_TYPE.CREATE_OBJECT_ID THEN

    objID := objSpec.objectSpec.objectID;
    objCount := GetObjectCount();
    i := 1;

    BACnetEventCallback := TRUE;

    WHILE i <= objCount DO
        GetObject(i, bacObj);
        runObjID := bacObj.ObjectID;
        IF runObjID.typ = objID.typ AND
            runObjID.instNumber = objID.instNumber THEN
            errorCode := ERR_CODE_OBJ_ID_ALREADY_EXISTS;
        END_IF
    END WHILE

2020/12/10 3ADR010583, 1, en_US 3464
BACnetEventCallback := FALSE;
EXIT;
END_IF
i := i + 1;
END_WHILE
ELSE
// No object specification given which can be interpreted, so
issue a standard error
BACnetEventCallback := FALSE;
END_IF
ELSE
// If the object type is not supported to be created issue the
// appropriate error
errCode := ERR_CODE_DYN_CREATION_NOT_SUPP;
BACnetEventCallback := FALSE;
END_IF

// If everything is all right till now (BACnetEventCallback is TRUE) try to
// create the object
IF BACnetEventCallback
THEN
IF pCreateObj^.pServiceInfo^.fValuesPresent > 0 THEN
// If property values are given by the service request use them
bacResult := CmpBACnet.BACnetConstructObject(destInstNumber,
ADR(objID),
pCreateObj^.pServiceInfo^.pValues,
pCreateObj^.pServiceInfo^.nValuesCount,
m_SupportUnsolicitedCOV, 1,
m_HideToOutside);
ELSE
// If no property values are given by the service request just
// create the object using the default implementation
bacResult := ConstructDefaultObject(objID.typ, objID.instNumber,
STRING_TO_WSTRING(CONCAT('CreatedObject',
DWORD_TO_STRING(objID.instNumber))),
bacObj, strBuffer);
END_IF
END_IF

// Check if everything went well
IF BACnetEventCallback AND bacResult = BACNET_STATUS_OK THEN
everything succeeded issue an result(+) response containing the
//object ID
CmpBACnet.BACnetCreateObjectResponseCbCompletion(pCreateObj^.phTransaction, ADR(objID));
//We treated
the service request, so no default API actions are
//necessary anymore
pCreateObj^.pCBStatus^ := CmpBACnet.IEC_BACNET_CB_STATUS.CB_STATUS_OK;
ELSE
//If something went wrong issue an error
BACnetEventCallback := FALSE;
//If no explicit error code was given till now, issue an internal error
IF errCode = 0 THEN
errCode := ERR_CODE_INTERNAL_ERROR;
END_IF
//Issue the result(-) response with the appropriate error code
CmpBACnet.BACnetCreateObjectErrorCbCompletion(pCreateObj^.phTransaction,
ERR_CLASS_OBJECT, errCode, 0);
//We treated
the service request, so no default API actions are
//necessary anymore
pCreateObj^.pCBStatus^ := CmpBACnet.IEC_BACNET_CB_STATUS.CB_STATUS_OK;
END_IF
END_IF
END_IF
END_IF

The main feature of this code example is to get the service request information by casting pCreateObj (EVT_BACNET_CREATEOBJECT) to the pParameter, to get access to the information in pParameter. pCreateObj is a typical hook request structure (found and declared in the library CmpBACnet). Most of them contain a variable pCBStatus which is the return value to the API. The latter return value decides whether the API further treats the service request (Default) or not (OK). The rest of the code mainly check the properties of the service request and then tries to create the object, whereas occurring errors are saved within the variable errCode. Finally result(+) or result(-) responses are issued using the functions CmpBACnet.BACnetCreateObjectResponseCbCompletion or CmpBACnet.BACnetCreateObjectErrorCbCompletion.

This is a typical example of an implemented service hook using the event mechanisms of the BACnet Stack/API: there is an event/callback you can register to in order to get a call whenever a service request was received. The implementation of the service hook then has to follow the specifications of the BACnet standard in order to provide a compliant service.

See also

- Chapter 1.5.6.3.8.10 “Implementation Requirements when Using the Current BACnet API/Stack” on page 3453
- Chapter 1.5.6.3.8.1 “Overview” on page 3439
Implementation of logging objects (Event-Log, Trend-Log, Trend-Log-Multiple)

For the Event-Log, Trend-Log and Trend-Log-Multiple object types no complete implementation of their functionality is provided by the BACnet Stack/API, because the saving of logged data is possibly very hardware specific and therefore very likely to be implemented by the specific user application. Nevertheless all property behaviour of the logging objects is pre-implemented in this API, so that basically only the recording and reading of the trended data has to be implemented by the user application.

A. Implementation of data recording for logging objects

Every logging object shall use its Log_Buffer property for storing the logged data. Since this property is only accessible (and therefore testable for BACnet standard compliance) via the ReadRange service, it is not necessary to care for its (user application) internal representation. Therefore the BACnet Stack/API handles the Log_Buffer as being only one record long and writes the incoming records to this single record whenever needed (i.e. the logged object produces new data). Therefore, for implementing a proper recording of the logged data, it is sufficient to just catch the write events to the Log_Buffer property and consequently saving the incoming single records.

For doing so, one can use the Write-Callback of the BACnet Stack/API. The latter callback is always called whenever a property of an object is written (assuming that the callback of the object was activated using the function BACnet.BACnetObjectBase.SetPropertyCallbackAttachment, which is part of any BACnet object FB in CODESYS). Registering such a Write-Callback is done very similar to registering a service hook with the only difference of the used register function. Again a CallbackFB (or two) implementing the Interface IBACnetEventConsumer can be registered using the function RegisterCallback function of the BACnetServer-FB:

```c
BACnetServer.RegisterCallback(CmpBACnet.IEC_BACNET_CALLBACK_TYPE.BACNET_CALLBACK_WRITE, CallbackFB, 0);
```

The corresponding function BACnetEventCallback in the CallbackFB now can be implemented to save the record coming along with the callback data (EVT_BACNET_WRITEPROPERTYCALLBACK), whenever called. For implementation be sure to filter for the correct object ID and the correct property ID for the callback data, so that you can be sure to only record data written to the Log_Buffer of a logging object.

B. Implementation of data reading for logging objects

As mentioned beforehand the Log_Buffer property of a logging object can only be read out using the ReadRange service. Consequently it is sufficient to implement the ReadRange service hook for implementing the data reading for the logging objects.

The ReadRange service hook may be implemented the same way as described for the CreateObject, see Chapter 1.5.6.3.8.10.3.1 "CreateObject service hook implementation example" on page 3462. That means registering a CallbackFB to the corresponding hook (RegisterHook). Within the BACnetEventCallback function of this CallbackFB you then have to filter for the correct object ID and property ID and get the correct service data (EVT_BACNET_READRANGE).

Within this ReadRange hook exactly that data has to be retuned (depending on the specified object ID), which was recorded beforehand in the Write-Callback (see A.). Thereby the ReadRange hook has to take care of the sorting out of data depending on the requested record numbers or record time stamps.

C. Comments

The procedure described in A. and B. holds for all logging objects meaning the Event-Log object, the Trend-Log object and the Trend-Log-Multiple object. Thereby it is not necessary to care about any object actions concerning properties, but only to implement the Write-Callback hook for saving the data and the ReadRange hook for reading this data out again.

See also
1.5.6.4 Data transfer and programming

1.5.6.4.1 Source download/upload

**Prerequisites**

- Communication settings are correct
- Project is saved on PC
- PLC is connected

1. Click “Online ➔ Source download to connected device”.
   - Project archive will be downloaded to PLC.

2. To verify download double-click node “PLC_AC500_V3”, select view “Files” and double-click folder “PlcLogic” of the Runtime view (if necessary click refresh button of Runtime view).
   - File Archive.prj will appear if download was successful.
Source upload

Prerequisite

● Project archive on PLC available (from previous source download)
● PLC is connected

1. Open Automation Builder.
2. Click “File ➔ Source upload...”.
   ⇒ Window Pick the device from where you want to upload the source archive... appears.
   
   ![Pick the device from where you want to upload the source archive...]

3. Select your PLC with the archive and click [OK].
   ⇒ Dialog Extract Project Archive appears.
4. Select your preferred folder and click [Extract].
   ⇒ Then you are prompted to open the project archive.
5. Click [Yes].
   ⇒ The project opens.
   
   Upload was successful.

1.5.6.4.2 Programming and testing

For information on programming see

● Programming Applications ➤ Chapter 1.3.1.9 “Programming of applications” on page 171
● Programming via Ethernet (TCP/IP) ➤ Chapter 1.3.1.11.1 “Configuring the connection to the PLC” on page 295
● Programming via Ethernet (Enter PLC IP_address) ➤ Chapter 1.5.6.2.2.4.2.2.1.1 “Enter a known PLC IP address” on page 3270
● Transferring Applications to the PLC ➤ Chapter 1.3.1.11 “Transferring an application to the PLC” on page 295

For information on testing/debugging see Testing and Debugging ➤ Chapter 1.3.1.12 “Testing and debugging” on page 309
1.5.6.4.3 Configuration of communication via Ethernet (TCP/IP)

Programming via Ethernet is only possible on a PC with Ethernet board and installed network. Programming can be done via the internal Ethernet communication module.

Programming via internal (onboard) Ethernet communication module:

Enter a known PLC IP address

1. Right-click the top node “PLC_AC500 <...>” and select “Communication Settings” from the context menu.
   ⇒ Dialog box Communication Settings <...> appears.

2. Enter your PLC IP Address and click [OK].
Enter PLC IP address by scanning devices

1. Right-click the top node “PLC_AC500 <...>” and select “Communication Settings” from the context menu.

   ⇒ Dialog box Communication Settings <...> appears.

2. Click ....

   ⇒ Dialog box Pick IP Address <...> appears.

3. Click [Scan], select your desired PLC and click [OK].

   ⇒ Entry is transferred to the dialog box Communication Settings <...>.

   Click [OK].

4. Click to log in the “PLC_AC500_V3” project.

Enter PLC IP address by [Advanced Settings…]

If a remote gateway instead of a local one has to be used it can be configured in the [Advanced Settings…].
1. Right-click the top node “PLC_AC500 <...>” and select “Communication Settings” from the context menu.
   ⇒ Dialog box Communication Settings <...> appears.

2. Enable checkbox Use advanced settings and click [Advanced Settings...].
   ⇒ Tab “Communication Settings” opens.

3. Check gateway or change if required.
   ⇒ Successful connection is indicated by green dot on the gateway icon.

4. **Manual entry of the IP address.**

   Check IP address or change if required.

5. Press ENTER to confirm changed IP address.
   ⇒ Successful communication is indicated by green dot on the PLC icon.
6. Or instead of the last two steps:

   Set the IP address via a scan.

Click [Scan Network], select your desired PLC and click [OK].

   ⇤ Successful connection is indicated by green dot on the gateway icon.

7. Click to log in the “PLC_AC500_V3” project.

1.5.6.4.4 PLC shell commands

   The PLC shell is used for requesting specific information from the controller. By entering a device-specific command the response is returned in a result window. The PLC shell can be issued without login.

   Proceed as follows:

1. Ensure the gateway is configured properly and a connection to the controller can be established.
2. In Automation Builder double-click the PLC node and open the tab “PLC Shell”.
3. Enter “?” in the command line of the tab window. All available PLC commands are listed.

If the gateway is able to establish a connection to the controller, an online connection to the PLC is opened automatically.

   The commands listed in online mode can differ from the commands shown when pressing the button […] as Automation Builder version and firmware version can differ ⇤ Chapter 1.1.5 “Information on the installed version” on page 27 ⇤ Chapter 1.5.6.1.4 “Firmware identification and update” on page 3245.

1.5.6.4.5 Watchlists

   Using Watchlists ⇤ Chapter 1.3.1.13.1.2 “Using watch lists” on page 330
   Changing values with Recipes ⇤ Chapter 1.3.1.13.2 “Changing values with recipes” on page 331
1.5.6.4.6 Reference to libraries

Library configuration is described in the Libraries and Solutions chapter of Chapter 1.4 “Libraries and solutions” on page 1966.

1.5.6.4.7 Programming in C-code

With the C code integration plugin from CODESYS, externally implemented C code files can be included in Automation Builder projects. For further information see CODESYS description Integrating C Modules of Chapter 1.3.1.9.11 “Integrating C Modules” on page 222.

1.5.6.5 Server installation

1.5.6.5.1 OPC server for AC500 V3 products

Introduction

Architecture of the CODESYS OPC server
Essential documents

For further information see Chapter 1.5.6.5.1.2 "Hints" on page 3479.

<table>
<thead>
<tr>
<th>File name</th>
<th>Comment</th>
<th>Where to find</th>
</tr>
</thead>
<tbody>
<tr>
<td>OPC_V3_how_to_use_E.pdf</td>
<td>OPC V3</td>
<td>C:\Program Files\ABB\CoDeSys OPC Server 3 AE</td>
</tr>
<tr>
<td>OPC_V3_how_to_use_D.pdf</td>
<td></td>
<td></td>
</tr>
<tr>
<td>AeConfigurator_User-Guide.pdf</td>
<td>OPC V3</td>
<td>C:\Program Files (x86)\3S CODESYS \CODESYS OPC Server 3</td>
</tr>
<tr>
<td>ReadMe.rtf</td>
<td>OPC V3</td>
<td>Installation ABB DM Suit 1.0.:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>\PLC - AC500\OPC Server\OPC-ServerV3.xAE\</td>
</tr>
<tr>
<td>ReleaseNotesOPCV3 AE for HA</td>
<td>OPC V3</td>
<td>Installation ABB DM Suit 1.0.:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>\PLC - AC500\OPC Server\OPC-ServerV3.xAE\</td>
</tr>
</tbody>
</table>

Work flow

Consideration and preparation

*) Chapter 1.5.6.5.1.2.2 “Installation of OPC server” on page 3482
Commission OPC server

*) Chapter 1.5.6.1.2.3.1 “Define symbols” on page 3485
1) See Chapter 1.5.6.5.1.2.5 “Configure AlarmEvents” on page 3491

2) See Chapter 1.5.6.5.1.2.5.1 “Check AlarmEvents” on page 3491
Adjustment to target OPC client

1) ☞ Chapter 1.5.6.5.1.1.2 “Essential documents” on page 3475 REF4.
2) ☞ Chapter 1.5.6.5.1.2.6 “Configure user account for OPC server” on page 3491
Hints

Default folder and contents

Windows 7 64-bit, windows server 2008 64-bit

<table>
<thead>
<tr>
<th>OPC Server V3</th>
<th>Windows 7 64-bit, Windows Server 2008 64-bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>WinCoDeSysOPC.exe</td>
<td>C:\Program Files (x86)\3S CoDeSys\CoDeSys OPC Server 3\</td>
</tr>
<tr>
<td>OPCConfig.exe</td>
<td></td>
</tr>
<tr>
<td>AEConfiguration.exe</td>
<td></td>
</tr>
<tr>
<td>CoDeSys_OP_C_Server_V3_User_Guide.pdf</td>
<td></td>
</tr>
<tr>
<td>CoDeSys_OP_C_Server_V3_Benutzerhandbuch.pdf</td>
<td></td>
</tr>
<tr>
<td>AeConfigurator_UserGuide.pdf</td>
<td></td>
</tr>
<tr>
<td>OPCServer.ini</td>
<td>C:\ProgramData\CoDeSysOPC\</td>
</tr>
<tr>
<td>OPCServerA.ini</td>
<td></td>
</tr>
<tr>
<td>OPCServer.log</td>
<td></td>
</tr>
<tr>
<td>Symbol file *.SDB, *.SYM</td>
<td>CBP open, after project build or rebuild all: in the project folder</td>
</tr>
<tr>
<td>Symbol file *.SDB</td>
<td>After login in AC500:</td>
</tr>
<tr>
<td></td>
<td>C:\ProgramData\Gateway Files\</td>
</tr>
<tr>
<td></td>
<td>After starting the OPC server:</td>
</tr>
<tr>
<td></td>
<td>C:\ProgramData\Gateway Files\Upload\</td>
</tr>
<tr>
<td>Gateway.exe</td>
<td>C:\Windows\SysWOW64\</td>
</tr>
</tbody>
</table>
### Win7 32 bit, windows server 2008 32 bit

<table>
<thead>
<tr>
<th>OPC Server V3</th>
<th>Win 7 32 bit, Windows Server 2008 32 bit</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>WinCoDeSysOPC.exe</code></td>
<td>C:\Program Files\3S CoDeSys\CoDeSys OPC Server 3\</td>
</tr>
<tr>
<td><code>OPCConfig.exe</code></td>
<td></td>
</tr>
<tr>
<td><code>AEConfiguration.exe</code></td>
<td></td>
</tr>
<tr>
<td><code>CoDeSys_OPC_Server_V3_User_Guide.pdf</code></td>
<td></td>
</tr>
<tr>
<td><code>CoDeSys_OPC_Server_V3_Benutzerhandbuch.pdf</code></td>
<td></td>
</tr>
<tr>
<td><code>AeConfigurator_UserGuide.pdf</code></td>
<td></td>
</tr>
<tr>
<td><code>OPCServer.ini</code></td>
<td>C:\ProgramData\CoDeSysOPC\</td>
</tr>
<tr>
<td><code>OPCServerA.ini</code></td>
<td></td>
</tr>
<tr>
<td><code>OPCServer.log</code></td>
<td></td>
</tr>
<tr>
<td>Symbol file <code>*.SDB, *.SYM</code></td>
<td>CBP open, after project build or rebuild all: in the project folder</td>
</tr>
<tr>
<td>Symbol file <code>*.SDB</code></td>
<td></td>
</tr>
<tr>
<td>After login in AC500:</td>
<td>C:\ProgramData\Gateway Files\</td>
</tr>
<tr>
<td>After starting the OPC server:</td>
<td>C:\ProgramData\Gateway Files\Upload\</td>
</tr>
<tr>
<td><code>Gateway.exe</code></td>
<td>C:\Windows\System32\</td>
</tr>
</tbody>
</table>
## Win XP 32-bit, Windows Server 2003 32-bit

<table>
<thead>
<tr>
<th>OPC Server V3</th>
<th>Win XP 32-bit, Windows Server 2008 32-bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>WinCoDeSysOPC.exe</td>
<td>C:\Program Files\3S CoDeSys\CoDeSys OPC Server 3\</td>
</tr>
<tr>
<td>OPCConfig.exe</td>
<td></td>
</tr>
<tr>
<td>AEConfiguration.exe</td>
<td></td>
</tr>
<tr>
<td>CoDeSys_OPC_Server_V3_User_Guide.pdf</td>
<td></td>
</tr>
<tr>
<td>CoDeSys_OPC_Server_V3_Benutzerhandbuch.pdf</td>
<td></td>
</tr>
<tr>
<td>AeConfigurator_UserGuide.pdf</td>
<td></td>
</tr>
<tr>
<td>OPCServer.ini</td>
<td></td>
</tr>
<tr>
<td>OPCServerA.ini</td>
<td></td>
</tr>
<tr>
<td>OPCServer.log</td>
<td></td>
</tr>
<tr>
<td>Symbol file *.SDB, *.SYM</td>
<td>CBP open, after project build or rebuild all: in the project folder</td>
</tr>
<tr>
<td>Symbol file *.SDB</td>
<td>After login in AC500:</td>
</tr>
<tr>
<td></td>
<td>C:\WINDOWS\Gateway Files\</td>
</tr>
<tr>
<td></td>
<td>After start CODESYS OPC server:</td>
</tr>
<tr>
<td></td>
<td>C:\WINDOWS\Gateway Files\Upload\</td>
</tr>
<tr>
<td>Gateway.exe</td>
<td>C:\Windows\System32\</td>
</tr>
</tbody>
</table>

If folder C:\ProgramData\ is missing, select “Show hidden files, folders and drives” at “Control Panel ➔ All Control Panel Items ➔ Folder Options ➔ View ➔ Hidden files and folders”.

---

2020/12/10 3ADR010583, 1, en_US 3481
Installation of OPC server

**Prerequisites**

*The following applications are closed:*
- All OPC clients
- ABB OPC tunnel
- CODESYS gateway server

*Ensure termination of the following processes:*
- Gateway.exe
- CoDeSysOPC.exe
- WinCoDeSysOPC.exe
- OCTsvc.exe

**Installing with Automation Builder**

2. Click button of *Latest Automation Builder version (recommended)* and run the installer.

3. Open “Installer Options and Additional Tools” and click [Install Additional Tools].
4. Agree to the “License Terms”.

![Automation Builder installation interface](image-url)
5. Select “Version 2 and/or 3” and install.
   ⇒ All required files are installed for OPC and the OPC server is registered automatically as user application.
Manual registration and unregistration

It is possible to register or to uninstall the OPC server manually either as COM server (user application) or as a service.

Register the OPC server as interactive software in the Windows registry:

**Command for OPC 3:** `WinCoDeSysOPC/RegServer`

Register the OPC server as system service:

**Command for OPC 3:** `WinCoDeSysOPC/Service`

Unregister the OPC server from the Windows registry and from the service entry:

**Command for OPC 3:** `WinCoDeSysOPC/UnRegServer`

Please see REF1 chapter 3 (OPC 3) Table on page 3475 for details.
Register OPC server V3 as a system service

Prerequisites
- All programs, processes and services which connect to the OPC server are closed.
  1. Start the “Command Prompt” with command “cmd” in the “Start ➔ Run...” window.
  2. Go to the CoDeSysOPC V2 installation folder.
  3. Unregister the OPC server with WinCoDeSysOPC/UnRegServer.
  4. Register the OPC server as system service with WinCoDeSysOPC/Service.

OPC clients for tests

Free of charge test clients can be found in the web:


Symbol file

Define symbols

1. Right-click on “Application” in CODESYS V3 project and click “Add Object”.

2. Choose “Symbol Configuration” and click [Add object].
3. Select your programs and/or single symbols and click [Build].
   ⇒ A symbol file will be automatically downloaded to the PLC with Project Download.

With double-click in the device tree to “Symbol Configuration” you can change the “Symbol Configuration” settings.

   To restrict traffic and load, choose only symbols you need.

Configure OPC server

Configure OPC Server V3

2. If the configuration is needed furthermore, save the configuration.

The actual configuration at start of OPC server will always be read from `OPCServer.ini`.

![OPC Server Configuration](image)

**Update rate**

- The Update Rate may not be 0 (ms)!
- The default value of 200 ms is a suitable value of many applications.
- The adjustment for the Update Rate depends on the number of symbols (variables).
- For a big number of symbols it can be better to increase the Update Rate.

The checkboxes **Sync Init** and **Enable logging (Default events)** must be enabled.
3. Select “PLC_FW3”.

- If the *.sdb files should be loaded from the “Gateway Files” directory on PC, the project name must be identical with project name in CODESYS. The extension is not necessary.
- If the symbol information should be loaded from AC500 V2.x, the project name is not required and can also be empty.
- The parameters displayed in the screenshot above are recommended default settings.
- The checkbox Active must be enabled.
- Enabled checkbox “Enable logging” allows a later diagnosis.
4. Select “Connection” and click [Edit].

5. Enter the TCP/IP address of the target PLC at *PLC name or address* and enable *Use Tcp/Ip blockdriver*.

6. Enter the TCP/IP address of the target PLC at *IP Address of PLC* and click [OK].

7. Click “File ➔ Save” OPCserver.ini and “File ➔ Exit” OPCConfig.

**Check OPC function with AC500**

It is urgently recommended to check the function of the previous configuration steps.

In order to check the OPC function without AC500, see .
Check OPC server V3

1. Start OPCExplorer.exe and connect “CoDeSys.OPC.DA”.

2. Add Group, add Items, select available Items in Server “CoDeSys.OPC.DA”. Add to Tag List, close the Item browser.

   ⇒ If anything is right, then “CoDeSys.OPC.DA” is connected, is running and the “Quality” of the items is good.

Check processes with windows task manager

Correct configuration: All processes run with the same “User Name” and with the same “Session ID”.
Configure AlarmEvents

Refer to REF2 AeConfigurator_UserGuide.pdf for details. Table on page 3475.

Check AlarmEvents

The function of the “AlarmEvents” can be checked with “MatrikonOPC Explorer”.

The “AlarmEvents” can be simulated by writing the value of the Items.

Configure user account for OPC server

Please refer to REF3 ReadMe.rtf and REF4 ReleaseNotes OPCV3 AE for HA. Table on page 3475.

OPC server V3 on windows server 2003 / 2008 / 2012

When running the OPC Server V3 on Windows Server 2003 / 2008 / 2012 multiple sessions need to be supported. Therefore the installation of the OPC server as service running with a dedicated user account is recommended.
Configuration steps

- Create specific user, no administrator account is required
- Register V3 OPC Server as service
- Configure V3 OPC Server as service

Create specific user

Register the OPC Server executable as service from the command line.

With command “WinCoDeSysOPC /Service” WinCoDeSysOPC.exe gets installed as system service.

Started once, the service will stay “started” until the system gets terminated.
The communication to the configured PLCs survives.

Also here the service gets installed in the current position of WinCoDeSysOPC.exe.
At “Computer Management ➔ Services and Applications ➔ Services” open the “Properties” of the “CoDeSysOPCDAService”.

Complete the Service Configuration
Testing

Check Users and Session during Test Cases

Check the “Session ID” and “User Name” of
● Gateway.exe,
● WinCoDeSysOPC.exe and
● OPC Client

on different test cases like multi session with terminal service sessions.

Potential issues

Session isolation

Situation
With Windows Server 2003, Windows Server 2008, Windows 7 services are alone in session 0. User applications run in session 1 (2 and so on).

Services:
A Windows service is a computer program that operates in the background.

Windows services can be configured to start when the operating system is started or can be started manually and run in the background as long as Windows is running. They can operate when a user is not logged on.

Services are:
Windows operating systems include numerous services. OPC client like S+ OPC scanner PGIM, Aspen CIM-IO Manager, ICONICS, .. can also installed as service.

User applications are:
Microsoft Word, Notepad, MatrikonExplorer, ControlBuilderPlus.exe and Codesys.exe

Problem
Service and user application are isolated in their session. They can not communicate with each other directly.

OPC Server uses, like the CBP and CODESYS, the gateway server from CODESYS (gateway.exe) for the communication with the AC500 and starts the gateway in their session. That creates undefined behavior, if the OPC Server runs as a service. The gateway server is not able to run in multi sessions.

Resolutions
● Install all OPC clients and OPC Server, which use the gateway server, in the same session.
● The OPC Server as a service (session 0) may not be connected at the same time (in parallel) with an OPC server as a user application or CBP or CODESYS (all in session 1) with the AC500. If this function is necessary, different PC or virtual machines must be used.
● Use tools like OPC tunnel. In a DigiVis 500 setup context the OPC server must not be registered as service. The OPC tunnel itself starts the OPC server within its service.

See also

1.5.6.5.2 OPC UA server for AC500 V3 products

General
OPC UA server can be added as an Object below the Ethernet Interfaces ETH1 or ETH2.

The user can access the variable interface of the PLC via a client. At the same time, communication can be protected by means of encryption.

The CODESYS OPC UA server supports the following features:
● Browsing of data types and variables
● Standard read/write services
Creating a project for OPC UA access

1. Click “File ➔ New Project ➔ AC500 project” in Automation Builder 2.1 or newer.
2. Choose a PLC - AC500 V3 and click [Add object].
3. Right-click on node ETH1 or ETH2 and “Add object”.
4. Choose OPC UA Server in the dialog and click [Add object].
5. Declare some variables of different types in the program.
6. Right-click “Application ➔ Add object”. Choose Symbol configuration and click [Add object].
7. Enable checkbox Support OPC UA Features in the dialog Add symbol configuration.
8. Double-click “Symbol configuration” in the Devices tree to open the editor Symbol configuration.
9. Click [Build].

The variables are displayed in a tree structure.
10. Activate the variables that you want to publish to an OPC UA client. Specify the access rights.
11. Download the project to the PLC.

Use node name

1. Double-click node “OPC_UA_Server”.
2. Set parameter Use node name to TRUE.
3. Double-click node “PLC_AC500_V3 <...>”.
4. Click “Device” and “Rename active device...”
5. Enter new device name in the following dialog and click [OK].

Use UaExpert client

The OPC UA client UaExpert is available for download from the Unified Automation website and can be used free of charge (freeware license).

Using this client, you can connect to the AC500 OPC UA server.

The following description refers to this program. Other OPC UA clients work in a similar way.
1. Start the *UaExpert* program.

2. Click on the “blue cross symbol”.

3. Double-click on the “blue cross symbol” in the *Add Server* dialog.

4. Enter URL and click [OK].
   - The URL appears in the *Add Server* dialog.

5. Select “Advanced” tab and click [OK].

6. Click [Connect] button.
7. Expand the project tree in the Address Space window.

![Address Space window](image)

8. Drag and drop the needed symbols to Data Access View.

Working with encryption

Creating a certificate

2. Select the “Devices” tab.
   - The certificate information opens.
3. Select the PLC in the left Information view.
   - All services of the PLC that require a certificate are displayed in the right Information view.
4. Select the service “OPC UA Server”.
5. Click the icon ![Certificate Settings](image) to create a new certificate for the device.
   - Certificate Settings dialog appears.
6. Define the certificate parameters according to the image above and click “[OK].
   ⇒ The certificate is created on the PLC.

7. Upload the certificate to your PC.
8. Restart the runtime system.

For further information see Chapter 1.5.6.3.6.3.4 “OPC UA secure” on page 3434.

Encrypted connection with UaExpert client

1. Start the UaExpert program.

2. Click on the “blue cross symbol”.
4. Enter URL and click [OK].
   ⇒ The URL appears in the Add Server dialog.
5. Select “Advanced” tab.

6. Choose option “Basic256sha256” of drop-down list Security Policy and “Sign & Encrypt” of drop-down list Message Security Mode and click [OK].
7. Click menu “Settings” and “Manage Certificates”

8. Click [Create new Application Certificate...].
   ⇒ Dialog New Application Instance Certificate opens.
9. Enter the required informations and click [OK].
   
   Dialog “Manage Certificates” opens

10. Click [Copy Application Certificate To...] your PC.

11. Download the certificate to AC500 via the Security Screen view.

12. Click [Connect] button in the UaExpert client.
Dialog Certificate Validation opens.

Working with a trusted certificate will avoid this error message.

14. Enable checkbox Accept the server certificate temporarily for this session and click [Continue].

Dialog Connect Error opens
15. Click [Ignore]

16. Check settings in dialog *Manage Certificates.*

**Changing variables via UaExpert client**

1. Expand in view *Address Space “Objects → DeviceSet → PM5670 → Resources → Application → PLC_PRG”.*
   
   The variables of the global variable list are visible.
2. Drag and drop the variables to the Data Access View.
3. Change values in the column Value.

Configuring OPC UA client

Operating modes

Polling
- Objects will be continuously updated in a defined interval
- Create higher load then Subscription
- Is recommended only for a few Symbols

Pub/Sub Not yet supported

Subscription (recommended mode)
- Updated objects depending on the publishing interval and filters
- Method to reduce load
- Different intervals
- Filter possible (coming in AC500)

<table>
<thead>
<tr>
<th>Client defines a group of symbols with</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Publishing interval</td>
<td>Interval, in which server publish data to client</td>
</tr>
<tr>
<td>Sampling interval</td>
<td>Interval for sampling and storing data at server and send in each publishing interval</td>
</tr>
<tr>
<td>Queue size</td>
<td>Array of data to save data if sampling Interval is faster than publishing Interval (At AC500 in the moment only 1)</td>
</tr>
<tr>
<td>Data change filter</td>
<td>Can be used to reduce traffic from server to client. Criteria:</td>
</tr>
<tr>
<td></td>
<td>● Change of data,</td>
</tr>
<tr>
<td></td>
<td>● Change of status</td>
</tr>
<tr>
<td></td>
<td>● Change of time stamp</td>
</tr>
<tr>
<td></td>
<td>AC500 is fix configured for change of data and change of status.</td>
</tr>
</tbody>
</table>
Using OPC UA with subscription mode

Recommendations:

- Define only variables you need as symbols
- Do not configure publishing intervals to short (increase load)
- Use different subscriptions with different publishing intervals in order to decrease load
- Do not use sampling intervals faster then publishing intervals as long as AC500 OPC UA Server don't support Queue Size different from 1
- Be careful: Setting „0“ at sampling Interval at client will be interpreted in server as „as fast as possible“, which is 100ms at AC500 and create a high load.

Publishing and sampling intervals in UaExpert
1. Right-Click on an Item in Data Access View and click “Subscription Settings”.

![Subscription Settings window](image)

2. Set the recommended values.

   **Life Time Count:** Number of publishing intervals in which client has to send publish requests to the server. After this period without request from client, subscription in server will be deleted.

   **Max Keep Alive Count:** If there are no new data to send, Server can skip a publishing interval. After the alive count, server has to send, even if there are no new data.

   Click **[OK]**.

3. Right-Click on an Item in Data Access View and click “Monitored Item Settings”.

![Monitored Item Settings window](image)

4. Set the recommended values.

1.5.6.6 **Convert an AC500 V2 project to an AC500 V3 project**

*Instructions on how to convert a V2 project to a V3 project and differences between V2 and V3.*

Keywords: Migration, conversion, V2, V3, differences between V2 and V3, application example, checkbox “Change to AC500 V3 PLC”
1.5.7 Storage devices for AC500 V3 products
1.5.7.1 Introduction of AC500 storage devices for AC500 Products
1.5.7.1.1 Overview

AC500 PLCs offer a variety of storage devices. The following table gives a short overview and a description on these storage devices:

IEC access means that the storage device can be accessed by Function Blocks of an IEC program.

FTP access means that the device can be accessed via FTP server on the PLC (if available).

<table>
<thead>
<tr>
<th>Component</th>
<th>Description</th>
<th>IEC Access</th>
<th>FTP Access</th>
<th>CPUs AC500 V3</th>
</tr>
</thead>
<tbody>
<tr>
<td>userdisk</td>
<td>User disk for custom data (flash)</td>
<td>Yes</td>
<td>Yes</td>
<td>All</td>
</tr>
<tr>
<td>home/userdisk</td>
<td>Internal persistent mass storage placed in the internal flash device</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(customer data)</td>
<td>Can be used for any application purpose</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PlcLogic</td>
<td>Internal persistent mass storage placed in the internal flash device</td>
<td>Yes</td>
<td>Yes</td>
<td>All</td>
</tr>
<tr>
<td>home/PlcLogic</td>
<td>Used for configuration data, user application (boot project), WEB visu files, etc.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(customer data)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SRAM</td>
<td>Battery-buffered device, non-volatile RAM</td>
<td>Yes</td>
<td>No</td>
<td>All</td>
</tr>
<tr>
<td></td>
<td>Used for retain/persistent and ProzM variables</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>system</td>
<td>System RAM disk (Temp directory) for storing the firmware</td>
<td>Yes</td>
<td>No</td>
<td>All</td>
</tr>
<tr>
<td></td>
<td>For internal firmware use only!</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## Component Description

<table>
<thead>
<tr>
<th>Component</th>
<th>Description</th>
<th>IEC Access</th>
<th>FTP Access</th>
<th>CPUs</th>
</tr>
</thead>
<tbody>
<tr>
<td>flashdisk</td>
<td>Internal persistent mass storage device</td>
<td>Yes</td>
<td>Yes</td>
<td>PM5675-2ETH</td>
</tr>
<tr>
<td></td>
<td>Can be used for any application purpose</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>sdcard</td>
<td>SD memory card (removable)</td>
<td>Yes</td>
<td>Yes</td>
<td>All</td>
</tr>
<tr>
<td></td>
<td>Removable persistent mass storage device</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Can be used for any application purpose</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

1.5.7.1.2 Functionalities

<table>
<thead>
<tr>
<th>Filesystem Name</th>
<th>As of CPU firmware</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>userdisk</td>
<td>V3.0.0</td>
<td>Boot project (size depends on PLC type)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>WEB visu files for web server</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Symbol file for OPC server and CP600 panels</td>
</tr>
<tr>
<td></td>
<td></td>
<td>User data via CAA_File_xxx.lib *)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Files via Automation Builder file download</td>
</tr>
<tr>
<td></td>
<td>V3.1.0</td>
<td>Save persistent data</td>
</tr>
<tr>
<td>SRAM</td>
<td>V3.1.0</td>
<td>Save retain and persistent data</td>
</tr>
<tr>
<td>system</td>
<td>V3.0.0</td>
<td>Load / save boot project</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Firmware update</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Internal system files</td>
</tr>
<tr>
<td>flashdisk</td>
<td>V3.1.0</td>
<td>User data via CAA_File_xxx.lib *)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Files via Automation Builder file download</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Files via FTP server</td>
</tr>
<tr>
<td>sdcard</td>
<td>V3.0.0</td>
<td>Firmware update,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>User data via CAA_File_xxx.lib *)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Files via Automation Builder file download</td>
</tr>
<tr>
<td></td>
<td>V3.1.0</td>
<td>Save persistent data</td>
</tr>
</tbody>
</table>

*) Examples for the filename with path (sFileName for FILE.Open) specified by the user ('mydir' is optional, but must be an existing directory):

- 'userdisk/myfile.txt'
- 'sdcard/mydir/myfile.txt'
- 'flashdisk/myfile.txt'
The maximum number of files opened at the same time is limited to 1007.

The max. length of the user string (path and filename) is 241 characters.

Unlike the PLC's memory areas like %M or Retain, where 1 byte actually consumes 1 byte, all storage device utilize a file system.

That means there is a difference between a files size and its size on the disk.

On disks the files are stored in so-called clusters which are a group of disk sectors. "Size on disk" refers to the amount of cluster(s) a file is taking up, while "file size" is an actual byte count of the file data. So you will usually find that the size on disk is larger than the file size. This is not an error, but a result of the disk organization via a file system. Since sector and cluster sizes vary depending on a disk's size and the used file system, the ratios between the size on disk and the file size also vary between the various storage devices.

### 1.5.7.1.3 Memory sizes

<table>
<thead>
<tr>
<th>PLC type</th>
<th>system RAM disk</th>
<th>userdisk</th>
<th>SRAM disk (Retain, ProzM area)</th>
<th>flashdisk</th>
<th>SD memory card</th>
</tr>
</thead>
<tbody>
<tr>
<td>PM5630-2ET H</td>
<td>Dynamically /max. 7,6 MB</td>
<td>40 MB</td>
<td>256 kB Retain and persistent 128 kB ProzM 128 kB</td>
<td>None</td>
<td>see MC502 -&gt; Technical Data \ Chapter 1.5.3.8.1.1 “MC502 - SD memory card” on page 3028</td>
</tr>
<tr>
<td>PM5650-2ET H</td>
<td>Dynamically /max. 16 MB</td>
<td>246 MB (as of V3.0.x) 381 MB (as of V3.1)</td>
<td></td>
<td>None</td>
<td>see MC502 -&gt; Technical Data \ Chapter 1.5.3.8.1.1 “MC502 - SD memory card” on page 3028</td>
</tr>
<tr>
<td>PLC type</td>
<td>system RAM disk</td>
<td>userdisk PlcLogic</td>
<td>SRAM disk (Retain, ProzM area)</td>
<td>flashdisk</td>
<td>SD memory card</td>
</tr>
<tr>
<td>----------</td>
<td>-----------------</td>
<td>-------------------</td>
<td>------------------------------</td>
<td>----------</td>
<td>----------------</td>
</tr>
<tr>
<td>PM5670-2ETH</td>
<td>Dynamically /max. 69 MB</td>
<td>858 MB</td>
<td>2 MB 1 MB retain and persistent 512 kB ProzM</td>
<td>None</td>
<td>see MC502 -&gt; Technical Data § Chapter 1.5.3.8.1.1 “MC502 - SD memory card” on page 3028</td>
</tr>
<tr>
<td>PM5675-2ETH</td>
<td>Dynamically /max. 69 MB</td>
<td>858 MB</td>
<td>8 GB</td>
<td>see MC502 -&gt; Technical Data § Chapter 1.5.3.8.1.1 “MC502 - SD memory card” on page 3028</td>
<td></td>
</tr>
</tbody>
</table>

It is not possible to use 100% of a device's memory space. About 10% of the total available space must remain unused at any time to maintain normal device operation.

### 1.5.7.1.4 Storage device details

This section contains some details on each storage device. For further details on specific topics please also refer to the following chapters:

- **Storage device sizes**
  - see Chapter 1.5.7.1.3 “Memory sizes” on page 3510
- **IEC access**
  - see CAA File Library
  - see SysInt Library
- **FTP access**
  - see FTP server
- **PLC shell commands**
  - see PLC shell commands § Chapter 1.5.6.4.4 “PLC shell commands” on page 3473

### SRAM

The SRAM is a battery-buffered, nonvolatile RAM and is used for the retain/persistent and the ProzM variables. If a battery is inserted into the Processor Module, the data stored in the SRAM will not get lost during a power-down cycle.

During PLC startup, the SRAM will be deleted automatically if no or an empty battery is inserted into the Processor Module. In this case the information

```
ABBInitSram_SetupMemory : SRAM cleared
```
and the warning
Retain size in config changed, or retain area got corrupted
are written into the log file.
Further information see Handling of Remanent Variables_V3 © Chapter 1.5.5.1.1 “Handling of remanent variables for AC500 V3 products” on page 3105.

SD memory card

The SD memory card is a removable persistent mass storage device and can be used for any application purpose. Both firmware updates and boot project updates can be run from the SD memory card © Chapter 1.5.7.2 “SD memory card in AC500” on page 3513.

<table>
<thead>
<tr>
<th>Size</th>
<th>Product specific, see table Memory Sizes © Chapter 1.5.7.1.3 “Memory sizes” on page 3510</th>
</tr>
</thead>
<tbody>
<tr>
<td>IEC access</td>
<td>CAA File Library</td>
</tr>
<tr>
<td></td>
<td>SysInt_Library</td>
</tr>
</tbody>
</table>

Flash disk

The flash disk is an internal persistent mass storage device and can be used for any application purpose.
It has a memory capacity of 8 GB (preformatted).
The flash disk is capable of high data throughput, however, the actual values to be achieved depend on the use cases. If the performance seems to get insufficient, check the following:
- If the PLCs CPU load is high, reduce overall CPU load of the PLC to have more performance for file operations.
- If the device has low free space, cleanup the disk.
Please consider the cluster size of 4 kB in your application design to achieve optimal usage of the flash disks space and access performance. For example, 10 files with 10 byte each require 10*4 kB disk space, while 1 file with 100 byte requires only 4 kB.

Number of max. write cycles

Technically, the flash chip used in V3 flash disk has 20000 Erase-Cycles (Write cycles).
Due to the produced write overhead, the optimum achievable number of write cycles is 10000 (for typical payload sizes of 256 kB).

Example

The write overhead is indicated by the write amplification factor (WAF).

\[ WAF = \frac{\text{Flash Write (in Bytes)}}{\text{Host Write (in Bytes)}} \]

Table 773: Rule of thumb for assessing the flash lifetime for an application:

<table>
<thead>
<tr>
<th>Typical payload sizes</th>
<th>WAF</th>
<th>Max. write cycles</th>
</tr>
</thead>
<tbody>
<tr>
<td>256 kB</td>
<td>2</td>
<td>10000</td>
</tr>
<tr>
<td>128 kB</td>
<td>4</td>
<td>5000</td>
</tr>
<tr>
<td>64 kB</td>
<td>8</td>
<td>2500</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
## 1.5.7.2 SD memory card in AC500

The SD memory card is a removable persistent mass storage device and can be used for any application purpose. Both firmware updates and boot project updates can be run from the SD memory card.

### NOTICE!

**Removal of the SD memory card**

Do not remove the SD memory card during access. For MC activity the black square (■) is shown on display as long as a file is open on Memory card. Remove only when the black square for MC activity does not blink. Otherwise the SD memory card and/or files on it might get corrupted and/or normal PLC operation might be disturbed.

### 1.5.7.2.1 Firmware update with SD memory card

**File content:** Firmware version V3.x

**General update process**

Information on the firmware: *Firmware identification and update* § Chapter 1.1.17.2.6 “AC500 V3 firmware installation and update” on page 59

The main components of the V3 CPU firmware are:

- BootFW (boot firmware): responsible for the starting of the UpdateFW or the SystemFW
- UpdateFW (update firmware): responsible for the update of BootFW, UpdateFW, SystemFW, UpdateHook and boot project
- SystemFW (system firmware, CPUFW): Runtime system of the PLC, additionally responsible for the update of the DisplayFW (display firmware) and the firmware of the coupler
- DisplayFW (display firmware): firmware of the display

Additionally the update process includes the following parts:

- Coupler (coupler firmware): firmware of the different couplers
- UpdateHook: Specific patches for the PLC
- UserProgram: boot project of the application
- License features: Import and export of license files. The license file for the "ImportLicense" is a Wbb or a WibuCmRaU file. The license file of the "ExportLicense" is a WibuCmRaC file.
The firmware updates are triggered by the command file `SDCARD.INI`. In addition a result file of the firmware update is generated (`SDCARD.RDY`, identical path as `SDCARD.INI`). For the group [FirmwareUpdate] the parameters 0, 11, 12 and 13 are defined. For each firmware update two files are necessary. The firmware file and the corresponding signature file.

For example:

AC500_V3_SystemFirmware_V3.0.1.73.tar.bz2
AC500_V3_SystemFirmware_V3.0.1.73.tar.bz2.sig
AC500_V3_DisplayFirmware_V3.0.0.0.app
AC500_V3_DisplayFirmware_V3.0.0.0.app.sig

For the user program the application file and the application CRC are necessary. For example:

`Application.app`
`Application.crc`

If the signature file and the firmware file do not match, no update is performed and the correspondent error result is written to the file `SDCARD.RDY`.

If the update firmware is running the display shows the text `update`. The blinking of the RUN and the ERR LED's indicates the update process.

At the end of the update process a “reboot” is executed and the system firmware is started for the finishing of the update process. The positive result of the update process is signaled by the blinking of the RUN LED (ERR LED is off) and the display shows `done`. A negative result is signaled by the blinking of the ERR LED (RUN LED is off) and the display shows `FAIL`. The file “SDCARD.RDY” includes the results of the different updates. After an update of a coupler CODESYS Control is started in safe mode (no download or starting of the application is possible) and the PLC needs a reboot (power down/up; the display shows `please` and `reboot` alternately).

As of system firmware 3.2 the compatibility file "Version.txt" (with the corresponding signature file "Version.txt.sig", identical path as "SDCARD.INI") is necessary for the update process. The update firmware checked the compatibility of the following parts:

- CPUFW (system firmware)
- BootFW (boot firmware)
- UpdateFW (update firmware)
- DisplayFW (display firmware)

A missing "Version.txt" or a missing/corrupt "Version.txt.sig" file is signalled at the component "CPUFW" (file "SDCARD.RDY").

If the update process would result in incompatible parts of firmware no update is performed in the update firmware. After starting of the system firmware the compatibility of the coupler firmware is checked additionally. The check of the compatibility of the firmware is executed always (independent of the parameter for the component). Incompatibility is signalled at the corresponding component (file SDCARD.RDY).
<table>
<thead>
<tr>
<th>Configuration</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BootFW=x</td>
<td>See description CPUFW. The component's path key for the boot firmware in module's section [CPU] is “BootFW”.</td>
</tr>
<tr>
<td>UpdateFW=x</td>
<td>See description CPUFW. The component's path key for the update firmware in module’s section [CPU] is “UpdateFW”.</td>
</tr>
<tr>
<td>DisplayFW=x</td>
<td>See description CPUFW. The component's path key for the display firmware in module's section [CPU] is “DisplayFW”.</td>
</tr>
<tr>
<td>UpdateHook=x</td>
<td>11 = Execute UpdateHook always with the file specified in module's section [CPU] and component's path key “UpdateHook”.</td>
</tr>
<tr>
<td>ImportLicense=x</td>
<td>12 = Import the license always with the file specified in module's section [CPU] and component's path key “ImportLicense”. The license file is a Wbb or a WibuCmRaU file. The update process imports this file into the plc. Note: Do not use parameter 11 for license import.</td>
</tr>
<tr>
<td>ExportLicense=x</td>
<td>12 = Export the license always to the file specified in module's section [CPU] and component's path key “ExportLicense”. The exported license file is a WibuCmRaC file. Note: Do not use parameter 11 for license export.</td>
</tr>
</tbody>
</table>
| Coupler0=x | 0 = No update.  
11 = Update firmware always with the file specified in module’s section [Coupler0] and component's path key "Boot" and/or “Firmware”.  
12 = Update with different version, the update is only performed if the version of the file specified by the component path key "Boot" and/or “Firmware” in module’s section [Coupler0] differs from the current version of the Coupler.  
13 = Update with newer version, the update is only performed if the version of the file specified by the component key "Boot" and/or “Firmware” in module’s section [Coupler0] is newer than the current version of the Coupler. |
| Coupler1=x | Update module slot 1; see description Coupler0, module section is [Coupler1]*. |
| Coupler2=x | Update module slot 2; see description Coupler0, module section is [Coupler2]*. |
| Coupler3=x | Update module slot 3; see description Coupler0, module section is [Coupler3]*. |
| Coupler4=x | Update module slot 4; see description Coupler0, module section is [Coupler4]*. |
| Coupler5=x | Update module slot 5; see description Coupler0, module section is [Coupler5]*. |
| Coupler6=x | Update module slot 6; see description Coupler0, module section is [Coupler6]*. |
| [UserProg] | 0 = No update.  
11 = Update user program always with the file specified in module's section [CPU] and component's path key “UserProgram”. |
Example: SDCARD.INI as of CPU firmware V3.x

[Status]
; FunctionOfCard
; 0 = Perform no function when inserting the card or voltage ON
; 1 = Load user program according to entry in group [UserProg]
; 2 = Start firmware update according to entry in group [FirmwareUpdate]
; 3 = Update firmware according to entry in group [FirmwareUpdate] and load user program according to entry in [UserProg]
FunctionOfCard=0

[FirmwareUpdate]
; 0 = No update
; 11 = Update with file specified in module's section <modsec>, component's path key <pathkey>
; 12 = Like 11, but check version of file to be updated differs from current one.
; 13 = Like 11, but check version of file to be updated is newer than current one.
CPUPFW=0 ;<modsec>=[CPU], <pathkey>= CPUPFW
BootFW=0 ;<modsec>=[CPU], <pathkey>= BootFW
UpdateFW=0 ;<modsec>=[CPU], <pathkey>= UpdateFW
DisplayFW=0 ;<modsec>=[CPU], <pathkey>= DisplayFW
UpdateHook=0 ;<modsec>=[CPU], <pathkey>= UpdateHook
ImportLicense=0 ;<modsec>=[CPU], <pathkey>= ImportLicense
ExportLicense=0 ;<modsec>=[CPU], <pathkey>= ExportLicense
Coupler0=0 ;<modsec>=[Coupler0], <pathkey>= Firmware
Coupler1=0 ;<modsec>=[Coupler1], <pathkey>= Firmware
Coupler2=0 ;<modsec>=[Coupler2], <pathkey>= Firmware
Coupler3=0 ;<modsec>=[Coupler3], <pathkey>= Firmware
Coupler4=0 ;<modsec>=[Coupler4], <pathkey>= Firmware
Coupler5=0 ;<modsec>=[Coupler5], <pathkey>= Firmware
Coupler6=0 ;<modsec>=[Coupler6], <pathkey>= Firmware

[UserProg]
; 0 = No update
; 11 = Update with file specified in module's section <modsec>, component's path key <pathkey>
UserProgram=0 ;Update user program. <modsec>=[CPU], <pathkey>= UserProgram

[CPU]; Path/file of CPU's system firmware to update
CPUFW= ;Path/file of CPU's system firmware to update
BootFW= ;Path/file of CPU's boot firmware to update
UpdateFW= ;Path/file of CPU's update firmware to update
UpdateHook = ;Path/file of UpdateHook to update
DisplayFW= ;Path/file of Display's firmware to update
ImportLicense= ;Path/file of import license file
ExportLicense= ;Path/file for export license file
UserProgram= ;Path/file of user program to update

[Coupler0] Firmware= ;Path/file of internal coupler's firmware to update

[Coupler1] Firmware= ;Path/file of external coupler's firmware slot 1 to update

[Coupler2] Firmware= ;Path/file of external coupler's firmware slot 2 to update

[Coupler3]
Firmware= ;Path/file of external coupler's firmware slot 3 to update

[Coupler4] Firmware= ;Path/file of external coupler's firmware slot 4 to update

[Coupler5] Firmware= ;Path/file of external coupler's firmware slot 5 to update

[Coupler6] Firmware= ;Path/file of external coupler's firmware slot 6 to update

Example SDCARD.INI for SD Card PM5650-2ETH for update only the system firmware (SystemFW):
[FirmwareUpdate]
CPUPFW=11
[CPU]
CPUFW=/SystemFirmware/ AC500_V3_SystemFirmware_V3.1.3.zzz.tar.bz2

Initializing an SD memory card using a PC

It is possible to create the above described file structure on the hard disk of a PC with SD memory card interface. To do this, create the required directories and an ASCII file named SDCARD.INI with Notepad, for example.

1.5.7.3 Flash memory for AC500 V3 products

AC500 Processor Modules for V3 products (PM56xx) are equipped with non-removable and non-volatile onboard user flash memory for program and data storage. The integrated flash management, including a wear levelling algorithm and a power-fail protected file system, is designed for robustness and operation in industrial environments and applications. The user flash memory can be accessed from the user program using the CAA_File_Library.

NOTICE!
The User Flash Memory has a finite number of write cycles.

Important: Programmers should keep the amount of cyclic written data low to ensure long availability.

1.5.7.4 Health monitoring

AC500 V3 products are equipped with non-removable and non-volatile onboard user flash memory for program and data storage. The integrated flash management, including a wear levelling algorithm and a power-fail protected file system, is designed for robustness and operation in industrial environments and applications.
Keep the amount of cyclic written data low to assure long availability of the flash memory. The spent/remaining lifetime information of the flash memory can be acquired with the Function Blocks PmDiskStatus and PmDiskLifetimeUsed in the “Library Manager” of Automation Builder.

Further information is provided in the documentation of the AC500_Pm library. See the function blocks PmDiskLifetimeUsed and PmDiskStatus in the “Library Manager” of Automation Builder.

1.6 Diagnosis and debugging for AC500 V3 products

1.6.1 The diagnosis system

The diagnosis system enables uniform diagnosis of the CPU and its local interfaces, of the local I/O bus with the connected S500 I/O devices and of the fieldbuses connected via communication modules, considering the special features of the various fieldbuses. The safety CPU is also integrated into the diagnosis system.

Diagnosis data of the devices can be accessed by
- CPU display
- Automation Builder
- IEC application

To forward the information to notify them by, e.g., webserver or OPC UA server, the data retrieved in IEC application can be stored in variables.

Fig. 285: Overview of the diagnosis system

All diagnosis data is assigned to a device. System diagnosis (e.g., battery low) is coming from the CPU device or one of the child objects (e.g., watchdog diagnosis of a task object).

Diagnosis is available for devices with representation in the Automation Builder device tree.

Diagnosis messages include the severity of an error. Error severity can be used for defining system behavior, e.g., activating the error LED or stop the PLC.

CPU parameters

Chapter 1.6.1.5.1 “Error severity” on page 3550.
Types of diagnosis messages

- An event describes the current state of the device. It does not have to be acknowledged.
- An alarm describes that at a certain point of time, there was a diagnosis message. It does not say anything about the current state of the device. Alarms must be acknowledged by the user. After acknowledging, the alarm disappears for all consumers.

Every diagnosis message has a come time.

Device state

With reference to diagnosis, there are different device states:

- Device without events and without unacknowledged alarms.
- Device with events or unacknowledged alarms.
- Device does not respond and is not available for online connection.

Diagnosis descriptions

Diagnosis messages are always available for all consumers.

- ABB AC500 V3 devices:
  - Events and unacknowledged alarms.
  - Every diagnosis message with come time, location, error number and text.
- 3rd party devices:
  - Events and unacknowledged alarms.
  - Every diagnosis message with come time, location and error number.
  - Clear text information if available either from a standard or from the device description.
  - If available: Extended diagnosis: Additional data coming from the device for manual analysis.

Extended diagnosis

Some devices are able to provide extended diagnosis. This additional device-dependant diagnosis will only be collected on request and will be device type specific (e.g. bus scan request on PROFINET I/O controller). Main intention is to cover commissioning use cases, when very specific information is required that typically cannot be stored in error numbers in a reasonable way.

1.6.1.1 Access to diagnosis data

Access to device state

- Error LED on CPU © Chapter 1.6.1.2 “Diagnosis in CPU display” on page 3520
- Automation Builder device tree © Chapter 1.6.1.3 “Diagnosis in Automation Builder” on page 3524
- IEC application via device name © Chapter 1.6.1.4.3 “Device diagnosis” on page 3540
- IEC application via list of all available diagnosis © Chapter 1.6.1.4.2 “System diagnosis” on page 3531
- External access via global IEC variables Fig. 285

Access to diagnosis descriptions

- CPU display © Chapter 1.6.1.2.2 “Diagnosis descriptions” on page 3520 (for CPU, local I/O bus and connected S500 I/O modules, not for communication modules and field buses)
- Automation Builder via “All messages” window © Chapter 1.6.1.3.2 “Diagnosis descriptions” on page 3525:
  - Support for (bulk) acknowledgement of alarm
  - Access to extended diagnosis data of 3rd party devices devices (if available), without any interpretation
- IEC application via a list of all current diagnosis either of a device (device object from the Automation Builder tree) \( \Leftrightarrow \) Chapter 1.6.1.4.3 “Device diagnosis” on page 3540 or of the complete PLC \( \Leftrightarrow \) Chapter 1.6.1.4.2 “System diagnosis” on page 3531:
  - Navigation chronologically in both directions (starting either from the oldest or newest diagnosis)
  - Access to the diagnosis numerically (evaluation by IEC application), textual (use on HMI) or to extended diagnosis data of 3rd party devices (if available)
  - Acknowledgement of alarms
- External access via global IEC variables by using the IEC application features for getting all relevant information

**Access to extended diagnosis**

Extended diagnosis data will be displayed in Automation Builder via “All messages” window. The data is displayed as it is provided from the device without any interpretation. Refer to, e.g., the manual of the device to get information about the extended diagnosis data.

### 1.6.1.2 Diagnosis in CPU display

#### 1.6.1.2.1 Device state

If there is at least one active diagnosis message, the error LED ERR is on.

The behavior of the error LED depends on the setting of CPU parameter “Error LED” \( \Leftrightarrow \) Chapter 1.5.6.2.3.4 “Parameters of the processor module” on page 3276.

**Diagnosis of AC500-eCo CPUs can only be shown by LED ERR at CPU. No display is available.**

#### 1.6.1.2.2 Diagnosis descriptions

\( \Leftrightarrow \) Chapter 1.5.5.1.4.4.5 “Reading out diagnosis messages on the CPU” on page 3138.

<table>
<thead>
<tr>
<th>Entry</th>
<th>Length [byte]</th>
<th>Values</th>
<th>Description</th>
<th>Display</th>
</tr>
</thead>
<tbody>
<tr>
<td>Error severity</td>
<td>1</td>
<td>0 .. 255</td>
<td>Used values: 1, 2, 3, 4, 11 ( \Leftrightarrow ) Chapter 1.6.1.5.1 “Error severity” on page 3550</td>
<td>Ex abc</td>
</tr>
<tr>
<td>Hardware ID (HwId)</td>
<td>1</td>
<td>0 .. 255</td>
<td>Location of diagnosis, e.g., subdevice, as three-letter word ( \Leftrightarrow ) Further information on page 3521</td>
<td>Ex abc</td>
</tr>
<tr>
<td>Error code</td>
<td>2</td>
<td>1 .. 65535</td>
<td>Error number (low word)</td>
<td>12345</td>
</tr>
<tr>
<td>SubSysteminfo byte 1</td>
<td>1</td>
<td>0 .. 255</td>
<td>Depends on hardware ID</td>
<td>d1 123</td>
</tr>
<tr>
<td>SubSysteminfo byte 2</td>
<td>1</td>
<td>0 .. 255</td>
<td>Depends on hardware ID</td>
<td>d2 123</td>
</tr>
<tr>
<td>SubSysteminfo byte 3</td>
<td>1</td>
<td>0 .. 255</td>
<td>Depends on hardware ID</td>
<td>d3 123</td>
</tr>
<tr>
<td>SubSysteminfo byte 4</td>
<td>1</td>
<td>0 .. 255</td>
<td>Depends on hardware ID</td>
<td>d4 123</td>
</tr>
</tbody>
</table>
CPU display does not show any communication modules or fieldbus diagnosis. To view these diagnosis messages use Automation Builder > Chapter 1.6.1.3 “Diagnosis in Automation Builder” on page 3524 or IEC application > Chapter 1.6.1.4 “Diagnosis in IEC application” on page 3526.

This is valid for:
- all external communication modules incl. safety CPUs, CM574-RS, FM502-CMS
- CANopen on onboard CAN interface
- fieldbuses on Ethernet interfaces ETH1/ETH2 like PROFINET IO controller, EtherCAT master, etc.

For identification of the location of a diagnosis the hardware ID and in addition for CPU diagnosis the SubSysteminfo byte 1 is used.

The location is displayed with 3 characters.

<table>
<thead>
<tr>
<th>Hardware ID</th>
<th>Value</th>
<th>SubSysteminfo byte 1</th>
<th>Value</th>
<th>Display</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>0</td>
<td>CPU itself</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>CPU</td>
<td>0</td>
<td>RAM</td>
<td>17</td>
<td></td>
</tr>
<tr>
<td>CPU</td>
<td>0</td>
<td>Flash</td>
<td>18</td>
<td></td>
</tr>
<tr>
<td>CPU</td>
<td>0</td>
<td>Flashdisk</td>
<td>19</td>
<td></td>
</tr>
<tr>
<td>CPU</td>
<td>0</td>
<td>SD memory card</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>CPU</td>
<td>0</td>
<td>Display</td>
<td>21</td>
<td></td>
</tr>
<tr>
<td>CPU</td>
<td>0</td>
<td>Battery</td>
<td>22</td>
<td></td>
</tr>
<tr>
<td>CPU</td>
<td>0</td>
<td>RTC (real-time clock)</td>
<td>23</td>
<td></td>
</tr>
<tr>
<td>CPU</td>
<td>0</td>
<td>FPU (floating point unit)</td>
<td>24</td>
<td></td>
</tr>
<tr>
<td>CPU</td>
<td>0</td>
<td>Power supply</td>
<td>25</td>
<td></td>
</tr>
<tr>
<td>Communication module 1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Communication module 2</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Communication module 3</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Communication module 4</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Hardware ID</td>
<td>Value</td>
<td>SubSysteminfo byte 1</td>
<td>Value</td>
<td>Display</td>
</tr>
<tr>
<td>-------------</td>
<td>-------</td>
<td>----------------------</td>
<td>-------</td>
<td>---------</td>
</tr>
<tr>
<td>Communication module 5</td>
<td>5</td>
<td></td>
<td></td>
<td><img src="image" alt="Display" /></td>
</tr>
<tr>
<td>Communication module 6</td>
<td>6</td>
<td></td>
<td></td>
<td><img src="image" alt="Display" /></td>
</tr>
<tr>
<td>COM1 serial interface 1</td>
<td>7</td>
<td></td>
<td></td>
<td><img src="image" alt="Display" /></td>
</tr>
<tr>
<td>COM2 reserved for serial interface 2</td>
<td>8</td>
<td></td>
<td></td>
<td><img src="image" alt="Display" /></td>
</tr>
<tr>
<td>CAN interface</td>
<td>9</td>
<td></td>
<td></td>
<td><img src="image" alt="Display" /></td>
</tr>
<tr>
<td>Onboard I/O (eCo)</td>
<td>10</td>
<td></td>
<td>No display</td>
<td></td>
</tr>
<tr>
<td>Option board 1 (eCo)</td>
<td>11</td>
<td></td>
<td>No display</td>
<td></td>
</tr>
<tr>
<td>Option board 2 (eCo)</td>
<td>12</td>
<td></td>
<td>No display</td>
<td></td>
</tr>
<tr>
<td>Option board 3 (eCo)</td>
<td>13</td>
<td></td>
<td>No display</td>
<td></td>
</tr>
<tr>
<td>I/O bus</td>
<td>14</td>
<td></td>
<td></td>
<td><img src="image" alt="Display" /></td>
</tr>
<tr>
<td>Ethernet ETH1</td>
<td>15</td>
<td></td>
<td></td>
<td><img src="image" alt="Display" /></td>
</tr>
<tr>
<td>Ethernet ETH2</td>
<td>16</td>
<td></td>
<td></td>
<td><img src="image" alt="Display" /></td>
</tr>
</tbody>
</table>

1.6.1.2.3 Reading out diagnosis messages on the CPU

*Table 774: Example: no diagnosis message in status list*

<table>
<thead>
<tr>
<th>State</th>
<th>Display</th>
<th>Result on pressing one of the function keys</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>DIAG</td>
</tr>
<tr>
<td>0</td>
<td>The processor module is in RUN/STOP mode.</td>
<td>State 1 is displayed</td>
</tr>
<tr>
<td>1</td>
<td><img src="image" alt="Display" /></td>
<td>No action</td>
</tr>
</tbody>
</table>
Table 775: Example: diagnosis messages in status list

<table>
<thead>
<tr>
<th>State</th>
<th>Display</th>
<th>Result on pressing one of the function keys</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td><strong>DIAG</strong></td>
</tr>
<tr>
<td>0</td>
<td>The processor module is in RUN/STOP mode.</td>
<td>State 1 is displayed</td>
</tr>
<tr>
<td>1</td>
<td><img src="image.png" alt="Image" /></td>
<td>Number of diagnosis messages; here 4</td>
</tr>
</tbody>
</table>
| 2     | ![Image](image.png) | Diagnosis message example: *Error battery empty or missing*  
Toggling between state 2 and 3 | Selects displayed diagnosis message and shows details | Go to first/next diagnosis message in status list | Go to last/previous diagnosis message in status list | Return into RUN/STOP mode. | Acknowledge and return into RUN/STOP mode. |
| 3     | ![Image](image.png) | Error ID example  
Toggling between state 2 and 3 | State 2 is displayed | State 2 is displayed | State 6 is displayed | State 0 is displayed | State 0 is displayed |

Table 776: Example: error battery empty or missing

<table>
<thead>
<tr>
<th>State</th>
<th>Display</th>
<th>Result on pressing one of the function keys</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td><strong>DIAG</strong></td>
</tr>
</tbody>
</table>
| 0     | ![Image](image.png) | E4 = error severity 4  
bAt = subdevice battery  
Toggling between state 0 and 1 | State 2 is displayed | State 2 is displayed | State 6 is displayed | State 0 is displayed | State 0 is displayed |

2020/12/10
3ADR010583, 1, en_US 3523
<table>
<thead>
<tr>
<th>State</th>
<th>Display</th>
<th>Result on pressing one of the function keys</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>![Image]</td>
<td>State 3 is displayed</td>
</tr>
<tr>
<td></td>
<td>Error ID example</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Toggling between state 0 and 1</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>![Image]</td>
<td>State 4 is displayed</td>
</tr>
<tr>
<td></td>
<td>Error number 8 Battery is missing or empty</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>![Image]</td>
<td>State 5 is displayed</td>
</tr>
<tr>
<td></td>
<td>Detail 1 Subdevice 22: battery</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>![Image]</td>
<td>State 6 is displayed</td>
</tr>
<tr>
<td></td>
<td>Detail 2 Error type 0: device</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>![Image]</td>
<td>State 1 is displayed</td>
</tr>
<tr>
<td></td>
<td>Detail 3 Error type number 0: device itself</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>![Image]</td>
<td>State 1 is displayed</td>
</tr>
<tr>
<td></td>
<td>Detail 4 Additional information 0: none</td>
<td></td>
</tr>
</tbody>
</table>

### 1.6.1.3 Diagnosis in Automation Builder

#### 1.6.1.3.1 Device state

In Automation Builder, colored icons next to the devices’ nodes in the device tree indicate the device state of each single device. 

*Chapter 1.6.2.3 “Project tree in online mode” on page 3553.*
1.6.1.3.2 Diagnosis descriptions

Displayed text

For output of diagnosis messages in textual format Automation Builder and IEC application use text lists. Both application use the same text lists. The text lists are part of the device description. When inserting a new device in device tree of project, the corresponding text list is loaded. This text lists are part of PLC program and will be downloaded into the PLC.

It is necessary to include a visualization, even if visualization will not be used. Without visualization the text lists will not be included.

The text lists are generated automatically. We recommend that you do not change them manually because the changes can be overwritten automatically and without prompting.

The text lists for 3rd party devices are created during reading of the device description sheets, e.g., GSDML files for PROFINET I/O devices.

The name of a text list for a PROFINET I/O device is: Diag_PNIO_Vendor ID_Device ID

Example

CI501-PNIO: Diag_PNIO_26_22
26 = vendor ID ABB, 22 = device ID CI501-PNIO

The text list for the AC500 PROFINET I/O modules contains all text needed for PROFINET standard diagnosis and AC500 process alarm handling.

Which texts are used, depends on parameter "Selection of diagnosis method". Double-click on a PROFINET I/O module and open tab “General”.

2020/12/10 3ADR010583, 1, en_US 3525
1.6.1.3.3 System diagnosis

In Automation Builder the system diagnosis is activated by default and can be deactivated in:
“Tools ➔ Options ➔ Diagnosis ➔ Enable subtree diagnosis”

![System diagnosis](image)

1.6.1.3.4 Device diagnosis

Each node in the device tree has a diagnosis view, which displays the diagnosis messages for this device only.

1. Double-click on a device.
2. Select the tab “Diagnosis”.

Example

Battery empty or missing.

Example

Wrong module configured on I/O bus.

1.6.1.4 Diagnosis in IEC application

There are two possibilities for accessing the diagnosis messages in the IEC application:

- System diagnosis: Access to diagnosis messages of the whole PLC
- Device diagnosis: Access to the diagnosis messages of a device

For both possibilities common data types (structures and enumerations) are defined in the library AC500_DiagTypes ➔ Chapter 1.6.1.4.1 “Data types in library AC500_DiagTypes” on page 3527. The library is automatically included in PLC project.
For output of diagnosis messages in textual format Automation Builder and IEC application use text lists. Both application use the same text lists. The text lists are part of the device description. When inserting a new device in device tree of project, the corresponding text list is loaded. This text lists are part of PLC program and will be downloaded into the PLC.

It is necessary to include a visualization, even if visualization will not be used. Visualization

Without visualization the text lists will not be included.

The text lists are generated automatically. We recommend that you do not change them manually because the changes can be overwritten automatically and without prompting.

The text lists for 3rd party devices are created during reading of the device description sheets, e.g., GSDML files for PROFINET I/O devices.

The name of a text list for a PROFINET I/O device is: Diag_PNIO_Vendor ID_Device ID

Example

CI501-PNIO: Diag_PNIO_26_22
26 = vendor ID ABB, 22 = device ID CI501-PNIO

The text list for the AC500 PROFINET I/O modules contains all text needed for PROFINET standard diagnosis and AC500 process alarm handling.

Which texts are used, depends on parameter "Selection of diagnosis method": Double-click on a PROFINET I/O module and open tab “General”.

1.6.1.4.1 Data types in library AC500_DiagTypes

All data types regarding diagnosis are defined in the library AC500_DiagTypes.
Structure DIAG_VAL_TYPE

This data type specifies the format of all kinds of diagnosis messages in numeric format. It consists of one element for each detail of a diagnosis message.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Initial</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>diTimestamp</td>
<td>DT</td>
<td>DATE_AND_TIME#1970-1-1-0:0</td>
<td>RTC time of eDiagEvent_Occurred</td>
</tr>
<tr>
<td>uiMs</td>
<td>UINT</td>
<td>0</td>
<td>Milliseconds of event</td>
</tr>
<tr>
<td>eClass</td>
<td>teClass</td>
<td>teClass.eDiagClass_4_Warning</td>
<td>Error severity of diagnosis message</td>
</tr>
<tr>
<td>szDevice</td>
<td>STRING(80)</td>
<td>&quot;&quot;</td>
<td>Name of device, max. 80 characters</td>
</tr>
<tr>
<td>eHwInterfaceId</td>
<td>teHwId</td>
<td>teHwId_CPU</td>
<td>Identifier of hardware interface</td>
</tr>
<tr>
<td>dwSubSysteminfo</td>
<td>DWORD</td>
<td>0</td>
<td>Any number describing details or location within device, device-specific</td>
</tr>
<tr>
<td>dwAdditional</td>
<td>DWORD</td>
<td>0</td>
<td>Additional number describing details or location within device, optional, device-specific</td>
</tr>
<tr>
<td>dwErrorCode</td>
<td>DWORD</td>
<td>0</td>
<td>Error code</td>
</tr>
<tr>
<td>wSizeExtDiag</td>
<td>DWORD</td>
<td>0</td>
<td>Number of bytes of extended diagnosis data</td>
</tr>
</tbody>
</table>

Structure DIAG_TXT_TYPE

This data type specifies the format of all kinds of diagnosis messages in textual format. It consists of a single string containing all details of a diagnosis message.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Initial</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>szDiag</td>
<td>STRING(512)</td>
<td>&quot;&quot;</td>
<td>Diagnosis message as text, max. 512 characters</td>
</tr>
</tbody>
</table>

The text consists of the following data, separated by semicolon:
- Timestamp in Date_And_Time (DT) format of eDiagEvent_Occurred and added milliseconds
- Error severity
- Device name (max. 80 characters) as defined in Automation Builder device tree
- The error text itself, composed of the interpretation of dwSubSysteminfo and dwAdditional and the error text plus remedy (if available) from Automation Builder text list according dwErrorCode. Displayed as: error text -> remedy.

Example

Battery empty or missing.

Enumeration ERROR_ID

Type of the return values of all methods and functions to request information on diagnosis.
### Name

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Comment</th>
<th>Remedy</th>
</tr>
</thead>
<tbody>
<tr>
<td>NO_ERROR</td>
<td>16#0</td>
<td>Execution successfully completed</td>
<td></td>
</tr>
<tr>
<td>ERR_PARAMETER</td>
<td>16#1</td>
<td>Invalid parameter value in function call</td>
<td>Correct parameter</td>
</tr>
<tr>
<td>ERR_NO_SINK</td>
<td>16#2</td>
<td>Failed to register as sink</td>
<td></td>
</tr>
<tr>
<td>ERR_NO_TEXT_LIST</td>
<td>16#3</td>
<td>Failed to get a device text list</td>
<td>Check text lists</td>
</tr>
<tr>
<td>ERR_NO_TEXT_CONTENT</td>
<td>16#4</td>
<td>Failed to get at least one content from text list</td>
<td></td>
</tr>
<tr>
<td>ERR_COMPETING</td>
<td>16#5</td>
<td>Failed due to competing access of other method</td>
<td>Try again</td>
</tr>
<tr>
<td>ERR_ASYNC</td>
<td>16#6</td>
<td>Failed to create async process</td>
<td></td>
</tr>
<tr>
<td>ERR_INTERNAL</td>
<td>16#7</td>
<td>Any internal error during execution</td>
<td></td>
</tr>
<tr>
<td>BUSY</td>
<td>16#FFF</td>
<td>Busy</td>
<td></td>
</tr>
<tr>
<td>NO_ERROR_NO_DATA</td>
<td>16#FFFF</td>
<td>Execution successfully completed, no more</td>
<td></td>
</tr>
</tbody>
</table>

All values except "BUSY" are final results. In case "NO_ERROR" is returned, the requested action has been successfully performed. "NO_ERROR_NO_DATA" also indicates a successful completion. The only difference to "NO_ERROR" is the fact, that there is one (more) data to be provided. All other return values (except "BUSY") are final error states. In case a method or function returns "BUSY", it has to be called again in the following cycles until it returns a final result.

### Enumeration teClass

Specifies the error severity of diagnosis messages \(\&\) Chapter 1.6.1.5.1 “Error severity” on page 3550.

<table>
<thead>
<tr>
<th>Name</th>
<th>Error severity</th>
</tr>
</thead>
<tbody>
<tr>
<td>eDiagClass_2_SeriousError</td>
<td>2</td>
</tr>
<tr>
<td>eDiagClass_3_Error</td>
<td>3</td>
</tr>
<tr>
<td>eDiagClass_4_Warning</td>
<td>4</td>
</tr>
<tr>
<td>eDiagClass_Parameter</td>
<td>11</td>
</tr>
</tbody>
</table>

### Enumeration teEvent

The enumeration teEvent specifies the severity of diagnosis messages.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Initial</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>eDiagEvent_Occurred</td>
<td>DINT</td>
<td>1</td>
<td>Error occurred, remains &quot;active&quot; until eDiagEvent_Disappeared</td>
</tr>
<tr>
<td>eDiagEvent_Disappeared</td>
<td>DINT</td>
<td>2</td>
<td>Error disappeared, became &quot;active&quot; due to eDiagEvent_Occurred earlier on</td>
</tr>
<tr>
<td>eDiagEvent_Received</td>
<td>DINT</td>
<td>4</td>
<td>Received a diagnosis message which cannot be analyzed in detail, cannot disappear, needs to be acknowledged</td>
</tr>
<tr>
<td>eDiagEvent_Acknowledged</td>
<td>DINT</td>
<td>8</td>
<td>Acknowledge a diagnosis message which has been received by eDiagEvent_Received, removes diagnosis message from diagnosis system although error may still be present</td>
</tr>
</tbody>
</table>
The enumeration teHwId is only used for diagnosis output in CPU display to identify the location of a diagnosis message.

<table>
<thead>
<tr>
<th>Name</th>
<th>Initial</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>eDiagHwId_CPU</td>
<td>0</td>
<td>Any diagnosis message regarding CPU itself, like &quot;battery&quot;, &quot;sd card&quot;, etc.</td>
</tr>
<tr>
<td>eDiagHwId_Coupler1</td>
<td>1</td>
<td>Any diagnosis message regarding communication module at slot 1. May be indicated by any corresponding communication module driver (instance) or a protocol driver (instance)</td>
</tr>
<tr>
<td>eDiagHwId_Coupler2</td>
<td>2</td>
<td>Any diagnosis message regarding communication module at slot 2. May be indicated by any corresponding communication module driver (instance) or a protocol driver (instance)</td>
</tr>
<tr>
<td>eDiagHwId_Coupler3</td>
<td>3</td>
<td>Any diagnosis message regarding communication module at slot 3. May be indicated by any corresponding communication module driver (instance) or a protocol driver (instance)</td>
</tr>
<tr>
<td>eDiagHwId_Coupler4</td>
<td>4</td>
<td>Any diagnosis message regarding communication module at slot 4. May be indicated by any corresponding communication module driver (instance) or a protocol driver (instance)</td>
</tr>
<tr>
<td>eDiagHwId_Coupler5</td>
<td>5</td>
<td>Any diagnosis message regarding communication module at slot 5. May be indicated by any corresponding communication module driver (instance) or a protocol driver (instance)</td>
</tr>
<tr>
<td>eDiagHwId_Coupler6</td>
<td>6</td>
<td>Any diagnosis message regarding communication module at slot 6. May be indicated by any corresponding communication module driver (instance) or a protocol driver (instance)</td>
</tr>
<tr>
<td>eDiagHwId_COM1</td>
<td>7</td>
<td>Any diagnosis message regarding COM1. May be indicated by any corresponding interface driver (instance) or a protocol driver (instance)</td>
</tr>
<tr>
<td>eDiagHwId_COM2</td>
<td>8</td>
<td>Any diagnosis message regarding COM2. May be indicated by any corresponding interface driver (instance) or a protocol driver (instance)</td>
</tr>
<tr>
<td>eDiagHwId_CAN</td>
<td>9</td>
<td>Any diagnosis message regarding CAN interface. May be indicated by any corresponding interface driver (instance) or a protocol driver (instance)</td>
</tr>
<tr>
<td>eDiagHwId_OnboardIO</td>
<td>10</td>
<td>Onboard I/O, AC500-eCo only</td>
</tr>
<tr>
<td>eDiagHwId_OptionBoard1</td>
<td>11</td>
<td>Option board 1, AC500-eCo only</td>
</tr>
<tr>
<td>eDiagHwId_OptionBoard2</td>
<td>12</td>
<td>Option board 2, AC500-eCo only</td>
</tr>
<tr>
<td>eDiagHwId_OptionBoard3</td>
<td>13</td>
<td>Option board 3, AC500-eCo only</td>
</tr>
<tr>
<td>eDiagHwId_IOBus</td>
<td>14</td>
<td>I/O bus</td>
</tr>
<tr>
<td>eDiagHwId_ETH1</td>
<td>15</td>
<td>Any diagnosis message regarding ETH1. May be indicated by any corresponding interface driver (instance) or a protocol driver (instance)</td>
</tr>
<tr>
<td>eDiagHwId_ETH2</td>
<td>16</td>
<td>Any diagnosis message regarding ETH2. May be indicated by any corresponding interface driver (instance) or a protocol driver (instance)</td>
</tr>
</tbody>
</table>
1.6.1.4.2 System diagnosis

Library AC500_Diag provides several methods and functions to access the diagnosis messages on all devices in the PLC application. It contains also a function to convert numeric diagnosis into a textual format.

The variables and their assigned values can be referred to within the IEC application as well as they can be used to transfer diagnosis messages to any visualization client.

Device state

The library contains a single function block named “Diag”, providing several methods to process the device state.

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NumTotal</td>
<td>Provides the total number of currently active diagnosis messages</td>
</tr>
<tr>
<td>NumClass</td>
<td>Provides the number of currently active diagnosis messages related to the error severity</td>
</tr>
</tbody>
</table>

Method NumTotal

This method provides the total number of currently active diagnosis messages (including parameter errors, etc.).

<table>
<thead>
<tr>
<th>Scope</th>
<th>Name</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Return</td>
<td>NumTotal</td>
<td>DWORD</td>
<td>Number of diagnosis messages</td>
</tr>
</tbody>
</table>

Example

```plaintext
VAR
    fbDiag : AC500_Diag.Diag; (* Instance of FB Diag *)
    dwNumTotal : DWORD;
END_VAR

dwNumTotal := fbDiag.NumTotal();
```

Method NumClass

This method provides the number of currently active diagnosis messages related to the error severity.

<table>
<thead>
<tr>
<th>Scope</th>
<th>Name</th>
<th>Type</th>
<th>Initial</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Return</td>
<td>NumClass</td>
<td>DWORD</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Input</td>
<td>DataInVal</td>
<td>AC500_DiagTypes.teClass</td>
<td>eClass.eDiagClass_2_SeriousError</td>
<td>Error severity of diagnosis message</td>
</tr>
</tbody>
</table>
Example

VAR

fbDiag : AC500_Diag.Diag;
eDiagClass_4: AC500_DiagTypes.teClass := AC500_DiagTypes.teClass.eDiagClass_4.Warning; (* 4 *)
eDiagClass_3: AC500_DiagTypes.teClass := AC500_DiagTypes.teClass.eDiagClass_3.Error; (* 3 *)
eDiagClass_11: AC500_DiagTypes.teClass := AC500_DiagTypes.teClass.eDiagClass_Parameter; (* 11 *)
dwNumClass_4 : DWORD;
dwNumClass_3 : DWORD;
dwNumClass_11 : DWORD;
END_VAR

dwNumClass_4 := fbDiag.NumClass(eDiagClass := eDiagClass_4);
dwNumClass_3 := fbDiag.NumClass(eDiagClass := eDiagClass_3);
dwNumClass_11 := fbDiag.NumClass(eDiagClass := eDiagClass_11);
Diagnosis descriptions

The library contains a single function block "Diag", providing several methods to process diagnosis descriptions.

All methods for system diagnosis start with "Get...". For device diagnosis the prefix "Diag" is added: "DiagGet...". For better readability, only the method names for system diagnosis is used in the descriptions of the methods.

<table>
<thead>
<tr>
<th>Method for system diagnosis</th>
<th>Method for device diagnosis</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ack</td>
<td>DiagAck</td>
<td>Acknowledge a diagnosis alarm previously requested by using any Get... / DiagGet... method &quot;Chapter 1.6.1.4.2.2.1 &quot;Method Ack / DiagAck: acknowledgement&quot; on page 3535</td>
</tr>
<tr>
<td>GetFirstVal</td>
<td>DiagGetFirstVal</td>
<td>Get the first (oldest) diagnosis message, numeric values &quot;Chapter 1.6.1.4.2.2.2 &quot;Methods Get... / DiagGet...: get and sort diagnosis messages&quot; on page 3535</td>
</tr>
<tr>
<td>GetNextVal</td>
<td>DiagGetNextVal</td>
<td>Get the next diagnosis message, numeric values &quot;Chapter 1.6.1.4.2.2.3 &quot;Method Get-xxx-Val / DiagGet-xxx-Val: numeric values&quot; on page 3536</td>
</tr>
<tr>
<td>GetLastVal</td>
<td>DiagGetLastVal</td>
<td>Get the last (newest) diagnosis message, numeric values &quot;Chapter 1.6.1.4.2.2.3 &quot;Method Get-xxx-Val / DiagGet-xxx-Val: numeric values&quot; on page 3536</td>
</tr>
<tr>
<td>GetPrevVal</td>
<td>DiagGetPrevVal</td>
<td>Get the previous diagnosis message, numeric values &quot;Chapter 1.6.1.4.2.2.3 &quot;Method Get-xxx-Val / DiagGet-xxx-Val: numeric values&quot; on page 3536</td>
</tr>
<tr>
<td>GetFirstValExt</td>
<td>DiagGetFirstValExt</td>
<td>Get the first (oldest) diagnosis message, numeric and extended numeric values &quot;Chapter 1.6.1.4.2.2.4 &quot;Method Get-xxx-ValExt / DiagGet-xxx-ValExt: numeric values and extended numeric values&quot; on page 3536</td>
</tr>
<tr>
<td>GetNextValExt</td>
<td>DiagGetNextValExt</td>
<td>Get the next diagnosis message, numeric and extended numeric values &quot;Chapter 1.6.1.4.2.2.4 &quot;Method Get-xxx-ValExt / DiagGet-xxx-ValExt: numeric values and extended numeric values&quot; on page 3536</td>
</tr>
<tr>
<td>GetLastValExt</td>
<td>DiagGetLastValExt</td>
<td>Get the last (newest) diagnosis message, numeric and extended numeric values &quot;Chapter 1.6.1.4.2.2.4 &quot;Method Get-xxx-ValExt / DiagGet-xxx-ValExt: numeric values and extended numeric values&quot; on page 3536</td>
</tr>
<tr>
<td>GetPrevValExt</td>
<td>DiagGetPrevValExt</td>
<td>Get the previous diagnosis message, numeric and extended numeric values &quot;Chapter 1.6.1.4.2.2.4 &quot;Method Get-xxx-ValExt / DiagGet-xxx-ValExt: numeric values and extended numeric values&quot; on page 3536</td>
</tr>
<tr>
<td>GetFirstValAndTxt</td>
<td>DiagGetFirstValAndTxt</td>
<td>Get the first (oldest) diagnosis message, numeric values and text &quot;Chapter 1.6.1.4.2.2.5 &quot;Method Get-xxx-ValAndTxt / DiagGet-xxx-ValAndTxt: numeric values and text&quot; on page 3537</td>
</tr>
<tr>
<td>Method for system diagnosis</td>
<td>Method for device diagnosis</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>-----------------------------</td>
<td>-------------</td>
</tr>
<tr>
<td>GetNextValAndTxt</td>
<td>DiagGetNextValAndTxt</td>
<td>Get the next diagnosis message, numeric values and text ‡ Chapter 1.6.1.4.2.2.5 “Method Get-xxx-ValAndTxt / DiagGet-xxx-ValAndTxt: numeric values and text” on page 3537</td>
</tr>
<tr>
<td>GetLastValAndTxt</td>
<td>DiagGetLastValAndTxt</td>
<td>Get the last (newest) diagnosis message, numeric values and text ‡ Chapter 1.6.1.4.2.2.5 “Method Get-xxx-ValAndTxt / DiagGet-xxx-ValAndTxt: numeric values and text” on page 3537</td>
</tr>
<tr>
<td>GetPrevValAndTxt</td>
<td>DiagGetPrevValAndTxt</td>
<td>Get the previous diagnosis message, numeric values and text ‡ Chapter 1.6.1.4.2.2.5 “Method Get-xxx-ValAndTxt / DiagGet-xxx-ValAndTxt: numeric values and text” on page 3537</td>
</tr>
<tr>
<td>GetFirstValAndTxtExt</td>
<td>DiagGetFirstValAndTxtExt</td>
<td>Get the first (oldest) diagnosis message, numeric, extended numeric values and text ‡ Chapter 1.6.1.4.2.2.6 “Method Get-xxx-ValAndTxtExt / DiagGet-xxx-ValAndTxtExt: numeric values, extended numeric values and text” on page 3538</td>
</tr>
<tr>
<td>GetNextValAndTxtExt</td>
<td>DiagGetNextValAndTxtExt</td>
<td>Get the next diagnosis message, numeric, extended numeric values and text ‡ Chapter 1.6.1.4.2.2.6 “Method Get-xxx-ValAndTxtExt / DiagGet-xxx-ValAndTxtExt: numeric values, extended numeric values and text” on page 3538</td>
</tr>
<tr>
<td>GetLastValAndTxtExt</td>
<td>DiagGetLastValAndTxtExt</td>
<td>Get the last (newest) diagnosis message, extended numeric values and text ‡ Chapter 1.6.1.4.2.2.6 “Method Get-xxx-ValAndTxtExt / DiagGet-xxx-ValAndTxtExt: numeric values, extended numeric values and text” on page 3538</td>
</tr>
<tr>
<td>GetPrevValAndTxtExt</td>
<td>DiagGetPrevValAndTxtExt</td>
<td>Get the previous diagnosis message, extended numeric values and text ‡ Chapter 1.6.1.4.2.2.6 “Method Get-xxx-ValAndTxtExt / DiagGet-xxx-ValAndTxtExt: numeric values, extended numeric values and text” on page 3538</td>
</tr>
</tbody>
</table>
Method Ack / DiagAck: acknowledgement

This method can be used to acknowledge a diagnosis alarm previously requested by using any Get... / DiagGet... method. Alternatively, you can acknowledge an alarm in Automation Builder.

After acknowledgement, the alarm is deleted from the diagnosis system.

<table>
<thead>
<tr>
<th>Scope</th>
<th>Name for device diagnosis</th>
<th>Name for device diagnosis</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Return</td>
<td>Ack</td>
<td>DiagAck</td>
<td>AC500_DiagTypes.ERROR_ID</td>
<td>Variable containing details of diagnosis alarm to be acknowledged</td>
</tr>
<tr>
<td>Input</td>
<td>Data</td>
<td>Data</td>
<td>AC500_DiagTypes.DIAG_VAL_TYPE</td>
<td></td>
</tr>
</tbody>
</table>

Example

System diagnosis: acknowledge first diagnosis message

```var
 VAR
 fbDiag : AC500_Diag.Diag;
 eErrorId_First : AC500_DiagTypes.ERROR_ID; (* instance of FB Diag *)
 sFirstVal : AC500_DiagTypes.DIAG_VAL_TYPE; (* return value *)
 END_VAR

 eErrorId_First := fbDiag.Ack(Data := sFirstVal);
```

Methods Get... / DiagGet...: get and sort diagnosis messages

All these methods can be used to get the first (oldest), next, last (newest) or previous diagnosis message stored in diagnosis system. The only difference are the details the methods provide. While, e.g., Get-xxx-Val just provides the basic information in numeric format, Get-xxx-ValExt additionally provides this information by the extended diagnosis data of the entry.

The numeric format provided by these methods can be converted into textual format later on if required \(\%\) Chapter 1.6.1.4.2.2.7 “Function DiagValToTxt” on page 3539. Alternatively, the methods Get-xxx-ValAndTxt and Get-xxx-ValAndTxtExt can be used for numeric and textual format in parallel \(\%\) Chapter 1.6.1.4.2.2.5 “Method Get-xxx-ValAndTxt / DiagGet-xxx-ValAndTxt: numeric values and text” on page 3537 \(\%\) Chapter 1.6.1.4.2.2.6 “Method Get-xxx-ValAndTxtExt / DiagGet-xxx-ValAndTxtExt: numeric values, extended numeric values and text” on page 3538.

All methods may need multiple cycles to process the request. Therefore, they must be called in successive cycles until they return a final result \(\%\) Chapter 1.6.1.4.1.3 “Enumeration ERROR_ID” on page 3528.

**All diagnosis messages sorted by time, ascending**

1. Call any GetFirst... method until it indicates a final result.
2. If the result is not "NO_ERROR_NO_DATA": Call any GetNext... method as long as its final result is "NO_ERROR".

**All diagnosis messages sorted by time, descending**

1. Call any GetLast... method until it indicates a final result.
2. If the result is not "NO_ERROR_NO_DATA": Call any GetPrev... method as long as its final result is "NO_ERROR".

-xxx- = First, Next, Last, Prev. Example: GetFirstVal, DiagGetLastVal.

<table>
<thead>
<tr>
<th>Scope</th>
<th>Name for device diagnosis</th>
<th>Name for device diagnosis</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Return</td>
<td>Get-xxx-Val</td>
<td>DiagGet-xxx-Val</td>
<td>AC500_DiagTypes.ERROR_ID</td>
<td>Variable to write data to</td>
</tr>
<tr>
<td>Inout</td>
<td>Data</td>
<td>Data</td>
<td>AC500_DiagTypes.DIAG_VAL_TYPE</td>
<td></td>
</tr>
</tbody>
</table>

Example

System diagnosis: get values for first diagnosis message

```plaintext
VAR
    fbDiag : AC500_Diag.Diag;
    eErrorID_First : AC500_DiagTypes.ERROR_ID;
    sFirstVal : AC500_DiagTypes.DIAG_VAL_TYPE;
END_VAR

eErrorID_First := fbDiag.GetFirstVal(Data := sFirstVal);
```

Example

Online mode: battery empty or missing

<table>
<thead>
<tr>
<th>Expression</th>
<th>Type</th>
<th>Value</th>
<th>Prepared value</th>
<th>Address</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>fbDiag</td>
<td>AC500_Diag.Diag</td>
<td>NO_ERROR</td>
<td></td>
<td></td>
<td>Instance of FB Diag</td>
</tr>
<tr>
<td>eErrorID_First</td>
<td>AC500_DiagTypes.ERROR_ID</td>
<td>(* instance of FB Error *)</td>
<td></td>
<td>return value</td>
<td></td>
</tr>
<tr>
<td>sFirstVal</td>
<td>AC500_DiagAC500...</td>
<td></td>
<td></td>
<td></td>
<td>structure of returned values</td>
</tr>
<tr>
<td>dtTimestamp</td>
<td>DATE_AND_TIME</td>
<td>DTF:1970-1-1:00:00:00:15</td>
<td></td>
<td>RTC time of event</td>
<td></td>
</tr>
<tr>
<td>units</td>
<td>UNIT</td>
<td>355</td>
<td></td>
<td></td>
<td>Milliseconds of event</td>
</tr>
<tr>
<td>eClass</td>
<td>TCLASS</td>
<td>eDiagClass_4_Warning</td>
<td></td>
<td>Severity of event</td>
<td></td>
</tr>
<tr>
<td>sDevice</td>
<td>STRING(64)</td>
<td>'PLC_AC500_V3'</td>
<td></td>
<td>Name of device</td>
<td></td>
</tr>
<tr>
<td>aEvent</td>
<td>TEVENT</td>
<td>eDiagEvent_1_occurred</td>
<td></td>
<td>Type of event</td>
<td></td>
</tr>
<tr>
<td>hbInterf</td>
<td>TBWID</td>
<td>eDiagHwID_CPU</td>
<td></td>
<td>Identifier of hardware interface</td>
<td></td>
</tr>
<tr>
<td>dwSubSysID</td>
<td>DWORD</td>
<td>365058782</td>
<td></td>
<td>Any number describing its location</td>
<td></td>
</tr>
<tr>
<td>dwAdditional</td>
<td>DWORD</td>
<td>0</td>
<td></td>
<td>Additional number describing detail</td>
<td></td>
</tr>
<tr>
<td>uErrorCode</td>
<td>UINT</td>
<td>0</td>
<td></td>
<td>Error code</td>
<td></td>
</tr>
<tr>
<td>uSizeExtDiag</td>
<td>UINT</td>
<td>0</td>
<td></td>
<td>Number of bytes of extended data</td>
<td></td>
</tr>
<tr>
<td>hSource</td>
<td>POINTER TO BYTE</td>
<td>16#00703D00</td>
<td></td>
<td>Internal reference needed for test conversion</td>
<td></td>
</tr>
<tr>
<td>pConn</td>
<td>POINTER TO الطلاب</td>
<td>16#B14A1D4</td>
<td></td>
<td>Internal reference needed for test conversion</td>
<td></td>
</tr>
</tbody>
</table>

Method Get-xxx-ValExt / DiagGet-xxx-ValExt: numeric values and extended numeric values

### System diagnosis: get numeric values and extended numeric values for first diagnosis message

```plaintext
VAR
    fbDiag : AC500_Diag.Diag;
eErrorID_First : AC500_DiagTypes.ERROR_ID;
sFirstVal : AC500_DiagTypes.DIAG_VAL_TYPE;
abyExtData : ARRAY[0..1023] OF BYTE;
wSize : WORD := 1024;
wLength : WORD;
END_VAR

eErrorID_First := fbDiag.GetFirstValExt(Data := sFirstVal, pExt := ADR(abyExtData), Size := wSize, Length := wLength);
```

### Method Get-xxx-ValAndTxt / DiagGet-xxx-ValAndTxt: numeric values and text

- `xxx=` First, Next, Last, Prev. Example: GetFirstValAndTxt, DiagGetPrevValAndTxt

<table>
<thead>
<tr>
<th>Scope</th>
<th>Name for device diagnosis</th>
<th>Name for device diagnosis</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Return</td>
<td>Get-xxx-ValExt</td>
<td>DiagGet-xxx-ValExt</td>
<td>AC500_DiagTypes.ERROR_ID</td>
<td></td>
</tr>
<tr>
<td>Inout</td>
<td>Data</td>
<td>Data</td>
<td>AC500_DiagTypes.DIAG_VAL_TYPE</td>
<td>Variable to write data to</td>
</tr>
<tr>
<td>Input</td>
<td>pExt</td>
<td>pExt</td>
<td>POINTER TO BYTE</td>
<td>Address of buffer to copy extended data to</td>
</tr>
<tr>
<td>Input</td>
<td>Size</td>
<td>Size</td>
<td>WORD</td>
<td>Size of buffer to copy extended data to</td>
</tr>
<tr>
<td>Inout</td>
<td>Length</td>
<td>Length</td>
<td>WORD</td>
<td>Size of extended data copied to buffer</td>
</tr>
</tbody>
</table>
Example

System diagnosis: get numeric values and text for first diagnosis message

```vbnet
VAR
    fbDiag : AC500_Diag.Diag;
    eErrorID_First : AC500_DiagTypes.ERROR_ID;
    sFirstVal : AC500_DiagTypes.DIAG_VAL_TYPE;
    sFirstTxt : AC500_DiagTypes.DIAG_TXT_TYPE;
END_VAR

eErrorID_First := fbDiag.GetFirstValAndTxt(DataVal := sFirstVal, DataTxt := sFirstTxt);
```

Example

Online mode: battery empty or missing

```vbnet
3ADR010583, 1, en_US
```

Method `Get-xxx-ValAndTxtExt` / `DiagGet-xxx-ValAndTxtExt`: numeric values, extended numeric values and text

- `xxx-` = First, Next, Last, Prev. Example: `GetLastValAndTxtExt`, `DiagGetFirstValAndTxtExt`

<table>
<thead>
<tr>
<th>Scope</th>
<th>Name for device diagnosis</th>
<th>Name for device diagnosis</th>
<th>Type</th>
<th>Initial</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Return</td>
<td>Get-xxx-ValAndTxtExt</td>
<td>DiagGet-xxx-ValAndTxtExt</td>
<td>AC500_DiagTypes.ERROR_ID</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Inout</td>
<td>DataVal</td>
<td>DataVal</td>
<td>AC500_DiagTypes.DIAG_VAL_TYPE</td>
<td></td>
<td>Variable to write data to</td>
</tr>
</tbody>
</table>

PLC Automation with V3 CPUs
Diagnosis and debugging for AC500 V3 products > The diagnosis system

2020/12/10
System diagnosis: get numeric values, extended numeric values and text of first diagnosis message

Function DiagValToTxt

Call this function to convert a numeric diagnosis message into a textual one at any time, in case this has not yet been done using a method providing both types when requesting this information.

<table>
<thead>
<tr>
<th>Scope</th>
<th>Name</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Return</td>
<td>DiagValToTxt</td>
<td>AC500_DiagTypes.ERROR_ID</td>
<td></td>
</tr>
<tr>
<td>Inout</td>
<td>DataInVal</td>
<td>AC500_DiagTypes.DIAG_VAL_TYPE</td>
<td>Variable to convert</td>
</tr>
<tr>
<td>Inout</td>
<td>DataOutTxt</td>
<td>AC500_DiagTypes.DIAG_TXT_TYPE</td>
<td>Variable to write text to</td>
</tr>
</tbody>
</table>
### System diagnosis: convert first diagnosis message from numeric value to text

```plaintext
VAR
   eErrorID_ValToTxt : AC500_DiagTypes.ERROR_ID;
   sFirstVal : AC500_DiagTypes.DIAG_VAL_TYPE;
   szDataOutTxt : AC500_DiagTypes.DIAG_TXT_TYPE;
   (* return value *)
   (* structure of returned values *)
   (* string of converted data *)
END_VAR

eErrorID_ValToTxt := DiagValToTxt(DataInVal := sFirstVal, DataOutTxt := szDataOutTxt);
```

### Example: Battery empty or missing

- **Battery empty or missing**

#### PLC_PRG_2

<table>
<thead>
<tr>
<th>Expression</th>
<th>Type</th>
<th>Value</th>
<th>Prepared value</th>
<th>Address</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>eErrorID_Fire</td>
<td>ERROR_ID</td>
<td>NO_ERROR</td>
<td>return value</td>
<td></td>
<td></td>
</tr>
<tr>
<td>eErrorID_valToOut</td>
<td>ERROR_ID</td>
<td>NO_ERROR</td>
<td>return value</td>
<td></td>
<td></td>
</tr>
<tr>
<td>szDiagDate</td>
<td>DATE_AND_TIME</td>
<td>D745761-1-0-0-15-15-15</td>
<td>RTC time of event</td>
<td></td>
<td></td>
</tr>
<tr>
<td>vPLC</td>
<td>UINT</td>
<td>356</td>
<td>Milliseconds of event</td>
<td></td>
<td></td>
</tr>
<tr>
<td>szDevice</td>
<td>STRING(80)</td>
<td>PLC_AC500_V3</td>
<td>Severity of error event</td>
<td></td>
<td></td>
</tr>
<tr>
<td>eDevice</td>
<td>STRING(15)</td>
<td>DeviceName_Diag</td>
<td>Name of device</td>
<td></td>
<td></td>
</tr>
<tr>
<td>szDiagMessage</td>
<td>IDWORD</td>
<td>350098712</td>
<td>Any number description, location within device</td>
<td></td>
<td></td>
</tr>
<tr>
<td>szAdditional</td>
<td>IDWORD</td>
<td>1</td>
<td>Additional number of diagnostic messages</td>
<td></td>
<td></td>
</tr>
<tr>
<td>eErrorCode</td>
<td>UINT</td>
<td>1</td>
<td>Error code</td>
<td></td>
<td></td>
</tr>
<tr>
<td>vDateText</td>
<td>UINT</td>
<td>2</td>
<td>Number of bytes of extended diagnostic data</td>
<td></td>
<td></td>
</tr>
<tr>
<td>szSource</td>
<td>STRING(128)</td>
<td>1</td>
<td>Internal reference needed for text conversion</td>
<td></td>
<td></td>
</tr>
<tr>
<td>vErrorText</td>
<td>STRING(128)</td>
<td>1</td>
<td>Internal reference needed for text conversion</td>
<td></td>
<td></td>
</tr>
<tr>
<td>sErrorText</td>
<td>STRING(128)</td>
<td>1</td>
<td>String of converted text</td>
<td></td>
<td></td>
</tr>
<tr>
<td>szDiagText</td>
<td>STRING(128)</td>
<td>1</td>
<td>Diagnosis entry as text</td>
<td></td>
<td></td>
</tr>
<tr>
<td>szMessage</td>
<td>STRING(128)</td>
<td>1</td>
<td>Character to be inserted as separator</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### Library CAA Device Diagnosis (namespace DED)

Library CAA Device Diagnosis (namespace DED) is automatically included in the project. This library is needed for displaying and processing the device state.

In case the functionality of diagnosis is no longer needed in IEC application, we recommend to disable this setting.

---

### 1.6.1.4.3 Device diagnosis

#### Activate device diagnosis

While the notification of diagnosis messages at the display and the Automation Builder is enabled by default, the functionality to access diagnosis messages from within the IEC application needs to be explicitly enabled.

1. Double-click on the CPU in the device tree.
2. Select tab “PLC Settings”.
3. Under “Additional Settings” select “Enable Diagnosis for devices”.

Library CAA Device Diagnosis (namespace DED) is automatically included in the project. This library is needed for displaying and processing the device state.

In case the functionality of diagnosis is no longer needed in IEC application, we recommend to disable this setting.
Device state

1. Open one of the IEC application code editors.
2. Type the device's name as it is written in the device tree, followed by a dot (".").
3. Select the method `GetDeviceState` from the context menu or type the name of the method on yourself.

![Image of GetDeviceState method]

4. Assign the function's parameters.

```
eError : DED.ERROR; (* return value of call *)
bDiagAvailable : BOOL; (* at least one diagnosis available for this device *)
DeviceState_DC532 : DED.DEVICE_STATE; (* general device state *)
```

```DeviceState_DC532 := DC532.GetDeviceState(xDiagnosisInfoAvailable => bDiagAvailable, eError => eError);```

<table>
<thead>
<tr>
<th>Scope</th>
<th>Name</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Return</td>
<td>GetDeviceState</td>
<td>DEVICE_STATE</td>
<td>Current device state</td>
</tr>
<tr>
<td></td>
<td></td>
<td>% Further information on page 3541</td>
<td></td>
</tr>
<tr>
<td>Output</td>
<td>xDiagnosisInfoAvailable</td>
<td>BOOL</td>
<td>If TRUE, diagnosis messages are available regarding the concerning device (= node).</td>
</tr>
<tr>
<td>Output</td>
<td>eError</td>
<td>ERROR</td>
<td>Type of the return values of all methods and functions of library CAA Device Diagnosis</td>
</tr>
<tr>
<td></td>
<td></td>
<td>% Further information on page 3542</td>
<td></td>
</tr>
</tbody>
</table>

**Table 778: Enumeration DEVICE_STATE (part of the library CAA Device Diagnosis (DED))**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Initial</th>
<th>Icon in AB</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>UNKNOWN</td>
<td>INT</td>
<td>0</td>
<td></td>
<td>The device is in state unknown. Example: No supervision mechanism active</td>
</tr>
<tr>
<td>STOPPED</td>
<td>INT</td>
<td>1</td>
<td></td>
<td>The device is stopped.</td>
</tr>
<tr>
<td>RUNNING</td>
<td>INT</td>
<td>2</td>
<td>![Icon]</td>
<td>The device is running.</td>
</tr>
<tr>
<td>ERROR</td>
<td>INT</td>
<td>3</td>
<td></td>
<td>The device is in error state.</td>
</tr>
<tr>
<td>DISABLED</td>
<td>INT</td>
<td>4</td>
<td></td>
<td>The device is disabled in device tree.</td>
</tr>
<tr>
<td>Name</td>
<td>Type</td>
<td>Initial</td>
<td>Icon in AB</td>
<td>Comment</td>
</tr>
<tr>
<td>--------------------</td>
<td>------</td>
<td>---------</td>
<td>------------</td>
<td>-------------------------------------------------------------------------</td>
</tr>
<tr>
<td>NOT_CONFIGURED</td>
<td>INT</td>
<td>5</td>
<td></td>
<td>The device has not been yet configured by the stack. Example: Configuration phase not yet started</td>
</tr>
<tr>
<td>CONFIGURED</td>
<td>INT</td>
<td>6</td>
<td></td>
<td>The device has been configured by the stack. Example: Configuration phase finished but the device is not in running state</td>
</tr>
<tr>
<td>NOT_FOUND</td>
<td>INT</td>
<td>7</td>
<td></td>
<td>The device was not found on bus.</td>
</tr>
</tbody>
</table>

Table 779: Enumeration ERROR (part of the library CAA Device Diagnosis (DED))

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Initial</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>NO_ERROR</td>
<td>INT</td>
<td>0</td>
<td>No error</td>
</tr>
<tr>
<td>FIRST_ERROR</td>
<td>INT</td>
<td>1300</td>
<td>First library-specific error</td>
</tr>
<tr>
<td>TIME_OUT</td>
<td>INT</td>
<td>1301</td>
<td>Timeout occurred.</td>
</tr>
<tr>
<td>ABORT</td>
<td>INT</td>
<td>1302</td>
<td>Operation was aborted.</td>
</tr>
<tr>
<td>REF_INVALID</td>
<td>INT</td>
<td>1303</td>
<td>The interface reference was invalid.</td>
</tr>
<tr>
<td>NOT_SUPPORTED</td>
<td>INT</td>
<td>1304</td>
<td>The function is not supported.</td>
</tr>
<tr>
<td>ERROR_IO</td>
<td>INT</td>
<td>1305</td>
<td>A general I/O configuration error occurred.</td>
</tr>
<tr>
<td>PARAM_INVALID</td>
<td>INT</td>
<td>1306</td>
<td>Invalid parameter</td>
</tr>
<tr>
<td>NODE_NOTEXISTING</td>
<td>INT</td>
<td>1307</td>
<td>The specified node does not exist.</td>
</tr>
<tr>
<td>NO_MEMORY</td>
<td>INT</td>
<td>1308</td>
<td>Dynamic memory allocation is disabled, or system is out of memory.</td>
</tr>
<tr>
<td>ADR_NOT_FOUND</td>
<td>INT</td>
<td>1309</td>
<td>The specified I/O address is not valid.</td>
</tr>
<tr>
<td>INST_NOT_FOUND</td>
<td>INT</td>
<td>1310</td>
<td>There is no associated [Device] instance for the specific I/O address.</td>
</tr>
<tr>
<td>NO_DATA</td>
<td>INT</td>
<td>1311</td>
<td>There is no data available.</td>
</tr>
<tr>
<td>OPERATION_INVALID</td>
<td>INT</td>
<td>1312</td>
<td>Operation not possible due to the current state</td>
</tr>
<tr>
<td>FIRST_MF</td>
<td>INT</td>
<td>1350</td>
<td>First manufacturer-specific error</td>
</tr>
<tr>
<td>LAST_ERROR</td>
<td>INT</td>
<td>1399</td>
<td>Last error</td>
</tr>
</tbody>
</table>
Diagnosis descriptions

The library contains a single function block "Diag", providing several methods to process diagnosis descriptions.

All methods for system diagnosis start with "Get...". For device diagnosis the prefix "Diag" is added: "DiagGet...". For better readability, only the method names for system diagnosis is used in the descriptions of the methods.

<table>
<thead>
<tr>
<th>Method for system diagnosis</th>
<th>Method for device diagnosis</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ack</td>
<td>DiagAck</td>
<td>Acknowledge a diagnosis alarm previously requested by using any Get... / DiagGet... method (\overset{\text{Chapter 1.6.1.4.2.2.1}}{\Rightarrow}) &quot;Method Ack / DiagAck: acknowledgement&quot; on page 3535</td>
</tr>
<tr>
<td>GetFirstVal</td>
<td>DiagGetFirstVal</td>
<td>Get the first (oldest) diagnosis message, numeric values (\overset{\text{Chapter 1.6.1.4.2.2.2}}{\Rightarrow}) &quot;Methods Get... / DiagGet...: get and sort diagnosis messages&quot; on page 3535</td>
</tr>
<tr>
<td>GetNextVal</td>
<td>DiagGetNextVal</td>
<td>Get the next diagnosis message, numeric values (\overset{\text{Chapter 1.6.1.4.2.2.3}}{\Rightarrow}) &quot;Method Get-xxx-Val / DiagGet-xxx-Val: numeric values&quot; on page 3536</td>
</tr>
<tr>
<td>GetLastVal</td>
<td>DiagGetLastVal</td>
<td>Get the last (newest) diagnosis message, numeric values (\overset{\text{Chapter 1.6.1.4.2.2.3}}{\Rightarrow}) &quot;Method Get-xxx-Val / DiagGet-xxx-Val: numeric values&quot; on page 3536</td>
</tr>
<tr>
<td>GetPrevVal</td>
<td>DiagGetPrevVal</td>
<td>Get the previous diagnosis message, numeric values (\overset{\text{Chapter 1.6.1.4.2.2.3}}{\Rightarrow}) &quot;Method Get-xxx-Val / DiagGet-xxx-Val: numeric values&quot; on page 3536</td>
</tr>
<tr>
<td>GetFirstValExt</td>
<td>DiagGetFirstValExt</td>
<td>Get the first (oldest) diagnosis message, numeric and extended numeric values (\overset{\text{Chapter 1.6.1.4.2.2.4}}{\Rightarrow}) &quot;Method Get-xxx-ValExt / DiagGet-xxx-ValExt: numeric values and extended numeric values&quot; on page 3536</td>
</tr>
<tr>
<td>GetNextValExt</td>
<td>DiagGetNextValExt</td>
<td>Get the next diagnosis message, numeric and extended numeric values (\overset{\text{Chapter 1.6.1.4.2.2.4}}{\Rightarrow}) &quot;Method Get-xxx-ValExt / DiagGet-xxx-ValExt: numeric values and extended numeric values&quot; on page 3536</td>
</tr>
<tr>
<td>GetLastValExt</td>
<td>DiagGetLastValExt</td>
<td>Get the last (newest) diagnosis message, numeric and extended numeric values (\overset{\text{Chapter 1.6.1.4.2.2.4}}{\Rightarrow}) &quot;Method Get-xxx-ValExt / DiagGet-xxx-ValExt: numeric values and extended numeric values&quot; on page 3536</td>
</tr>
<tr>
<td>GetPrevValExt</td>
<td>DiagGetPrevValExt</td>
<td>Get the previous diagnosis message, numeric and extended numeric values (\overset{\text{Chapter 1.6.1.4.2.2.4}}{\Rightarrow}) &quot;Method Get-xxx-ValExt / DiagGet-xxx-ValExt: numeric values and extended numeric values&quot; on page 3536</td>
</tr>
<tr>
<td>GetFirstValAndTxt</td>
<td>DiagGetFirstValAndTxt</td>
<td>Get the first (oldest) diagnosis message, numeric values and text (\overset{\text{Chapter 1.6.1.4.2.2.5}}{\Rightarrow}) &quot;Method Get-xxx-ValAndTxt / DiagGet-xxx-ValAndTxt: numeric values and text&quot; on page 3537</td>
</tr>
<tr>
<td>Method for system diagnosis</td>
<td>Method for device diagnosis</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------------</td>
<td>----------------------------</td>
<td>-------------</td>
</tr>
<tr>
<td>GetNextValAndTxt</td>
<td>DiagGetNextValAndTxt</td>
<td>Get the next diagnosis message, numeric values and text © Chapter 1.6.1.4.2.2.5 “Method Get-xxx-ValAndTxt / DiagGet-xxx-ValAndTxt: numeric values and text” on page 3537</td>
</tr>
<tr>
<td>GetLastValAndTxt</td>
<td>DiagGetLastValAndTxt</td>
<td>Get the last (newest) diagnosis message, numeric values and text © Chapter 1.6.1.4.2.2.5 “Method Get-xxx-ValAndTxt / DiagGet-xxx-ValAndTxt: numeric values and text” on page 3537</td>
</tr>
<tr>
<td>GetPrevValAndTxt</td>
<td>DiagGetPrevValAndTxt</td>
<td>Get the previous diagnosis message, numeric values and text © Chapter 1.6.1.4.2.2.5 “Method Get-xxx-ValAndTxt / DiagGet-xxx-ValAndTxt: numeric values and text” on page 3537</td>
</tr>
<tr>
<td>GetFirstValAndTxtExt</td>
<td>DiagGetFirstValAndTxtExt</td>
<td>Get the first (oldest) diagnosis message, numeric, extended numeric values and text © Chapter 1.6.1.4.2.2.6 “Method Get-xxx-ValAndTxtExt / DiagGet-xxx-ValAndTxtExt: numeric values, extended numeric values and text” on page 3538</td>
</tr>
<tr>
<td>GetNextValAndTxtExt</td>
<td>DiagGetNextValAndTxtExt</td>
<td>Get the next diagnosis message, numeric, extended numeric values and text © Chapter 1.6.1.4.2.2.6 “Method Get-xxx-ValAndTxtExt / DiagGet-xxx-ValAndTxtExt: numeric values, extended numeric values and text” on page 3538</td>
</tr>
<tr>
<td>GetLastValAndTxtExt</td>
<td>DiagGetLastValAndTxtExt</td>
<td>Get the last (newest) diagnosis message, extended numeric values and text © Chapter 1.6.1.4.2.2.6 “Method Get-xxx-ValAndTxtExt / DiagGet-xxx-ValAndTxtExt: numeric values, extended numeric values and text” on page 3538</td>
</tr>
<tr>
<td>GetPrevValAndTxtExt</td>
<td>DiagGetPrevValAndTxtExt</td>
<td>Get the previous diagnosis message, extended numeric values and text © Chapter 1.6.1.4.2.2.6 “Method Get-xxx-ValAndTxtExt / DiagGet-xxx-ValAndTxtExt: numeric values, extended numeric values and text” on page 3538</td>
</tr>
</tbody>
</table>
**Method Ack / DiagAck: acknowledgement**

This method can be used to acknowledge a diagnosis alarm previously requested by using any Get... / DiagGet... method. Alternatively, you can acknowledge an alarm in Automation Builder.

After acknowledgement, the alarm is deleted from the diagnosis system.

<table>
<thead>
<tr>
<th>Scope</th>
<th>Name for device diagnosis</th>
<th>Name for device diagnosis</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Return</td>
<td>Ack</td>
<td>DiagAck</td>
<td>AC500_DiagTypes.ERROR_ID</td>
<td>Variable containing details of diagnosis alarm to be acknowledged</td>
</tr>
<tr>
<td>Input</td>
<td>Data</td>
<td>Data</td>
<td>AC500_DiagTypes.DIAG_VAL_TYPE</td>
<td></td>
</tr>
</tbody>
</table>

**Example**

System diagnosis: acknowledge first diagnosis message

```vb
VAR
    fbDiag : AC500_Diag.Diag;
    eErrorID_First : AC500_DiagTypes.ERROR_ID; (* return value *)
    sFirstVal : AC500_DiagTypes.DIAG_VAL_TYPE; (* structure of returned values *)
END_VAR

eErrorID_First := fbDiag.Ack(Data := sFirstVal);
```

**Methods Get... / DiagGet...: get and sort diagnosis messages**

All these methods can be used to get the first (oldest), next, last (newest) or previous diagnosis message stored in diagnosis system. The only difference are the details the methods provide. While, e.g., Get-xxx-Val just provides the basic information in numeric format, Get-xxx-ValExt additionally provides this information by the extended diagnosis data of the entry.

The numeric format provided by these methods can be converted into textual format later on if required. The methods Get-xxx-ValAndTxt and Get-xxx-ValAndTxtExt can be used for numeric and textual format in parallel.

All methods may need multiple cycles to process the request. Therefore, they must be called in successive cycles until they return a final result.

1. Call any GetFirst... method until it indicates a final result.
2. If the result is not "NO_ERROR_NO_DATA": Call any GetNext... method as long as its final result is "NO_ERROR".

All diagnosis messages sorted by time, ascending

1. Call any GetLast... method until it indicates a final result.
2. If the result is not "NO_ERROR_NO_DATA": Call any GetPrev... method as long as its final result is "NO_ERROR".

-xxx- = First, Next, Last, Prev. Example: GetFirstVal, DiagGetLastVal.

<table>
<thead>
<tr>
<th>Scope</th>
<th>Name for device diagnosis</th>
<th>Name for device diagnosis</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Return</td>
<td>Get-xxx-Val</td>
<td>DiagGet-xxx-Val</td>
<td>AC500_DiagTypes.ERROR_ID</td>
<td>Variable to write data to</td>
</tr>
<tr>
<td>Inout</td>
<td>Data</td>
<td>Data</td>
<td>AC500_DiagTypes.DIAG_VAL_TYPE</td>
<td></td>
</tr>
</tbody>
</table>

Example

System diagnosis: get values for first diagnosis message

```
VAR
  fbDiag : AC500_Diag.Diag;
  eErrorID_First : AC500_DiagTypes.ERROR_ID;
  sFirstVal : AC500_DiagTypes.DIAG_VAL_TYPE;
  dTimeStamps : DATE_AND_TIME;
  uInt : UNIT;
  eClass : eDiagClass_4_Warning;
  eErrorCode : uInt;
  uIntExtDiag : UNIT;
  hSource : POINTER TO BYTE;
  pConn : POINTER TO IntVar;
END_VAR

  eErrorID_First := fbDiag.GetFirstVal(Data := sFirstVal);
```

Example

Online mode: battery empty or missing

```
PLC_PRG X

PLC_AC500_V3.ApplicationPLC_PRG

Expression | Type | Value | Prepared value | Address | Comment |
-----------|------|-------|----------------|---------|---------|
  fbDiag    | AC500_Diag.Diag  | NO_ERROR | Instance of FB Diag | Return value |
  eErrorID_First | AC500_DiagTypes.ERROR_ID | structure of returned values |
  sFirstVal | AC500_Diag.AC500... | | |
  dTimeStamps | DATE_AND_TIME | | | |
  uInt | UNIT | 356 | | |
  eClass | eDiagClass_4_Warning | | | |
  sDevice | STRING(0) | PLC_AC500_V3 | Name of device |
  eEvent | BOOL | eDiagEvent_occurred | | |
  eHwInterface | TSBWID | eDiagHBwId_CPU | Identifier of hardware interface |
  dwSubSys | DWORD | 365058752 | Any number describing the location within d...
  dwAdditional | DWORD | 0 | Additional number describing detail/location wi...
  uIntError | uINT | 0 | Error code |
  uIntExtDiag | UNIT | 0 | Number of bytes of extended diagnosis data |
  hSource | POINTER TO BYTE | 16#008703D0 | Internal reference needed for test conversion |
  pConn | POINTER TO IntVar | 16#012441D4 | Internal reference needed for test conversion |
```

Method Get-xxx-ValExt / DiagGet-xxx-ValExt: numeric values and extended numeric values

System diagnosis: get numeric values and extended numeric values for first diagnosis message

**Example**

```
VAR
  fbDiag : AC500_Diag.Diag;
  eErrorID_First : AC500_DiagTypes.ERROR_ID;
  sFirstVal : AC500_DiagTypes.DIAG_VAL_TYPE;
  abyExtData : ARRAY[0..1023] OF BYTE;
  wSize : WORD := 1024;
  wLength : WORD;
END_VAR

eErrorID_First := fbDiag.GetFirstValExt(Data := sFirstVal, pExt := ADR(abyExtData), Size := wSize, Length := wLength);
```

**Method Get-xxx-ValAndTxt / DiagGet-xxx-ValAndTxt: numeric values and text**

-xxx- = First, Next, Last, Prev. Example: GetFirstValAndTxt, DiagGetPrevValAndTxt

### Table 1: Function Parameters

<table>
<thead>
<tr>
<th>Scope</th>
<th>Name for device diagnosis</th>
<th>Name for device diagnosis</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Return</td>
<td>Get-xxx-ValExt</td>
<td>DiagGet-xxx-ValExt</td>
<td>AC500_DiagTypes.ERROR_ID</td>
<td></td>
</tr>
<tr>
<td>Inout</td>
<td>Data</td>
<td>Data</td>
<td>AC500_DiagTypes.DIAG_VAL_TYPE</td>
<td>Variable to write data to</td>
</tr>
<tr>
<td>Input</td>
<td>pExt</td>
<td>pExt</td>
<td>POINTER TO BYTE</td>
<td>Address of buffer to copy extended data to</td>
</tr>
<tr>
<td>Input</td>
<td>Size</td>
<td>Size</td>
<td>WORD</td>
<td>Size of buffer to copy extended data to</td>
</tr>
<tr>
<td>Inout</td>
<td>Length</td>
<td>Length</td>
<td>WORD</td>
<td>Size of extended data copied to buffer</td>
</tr>
</tbody>
</table>

### Table 2: Function Parameters

<table>
<thead>
<tr>
<th>Scope</th>
<th>Name for device diagnosis</th>
<th>Name for device diagnosis</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Return</td>
<td>Get-xxx-ValAndTxt</td>
<td>DiagGet-xxx-ValAndTxt</td>
<td>AC500_DiagTypes.ERROR_ID</td>
<td></td>
</tr>
<tr>
<td>Inout</td>
<td>DataVal</td>
<td>DataVal</td>
<td>AC500_DiagTypes.DIAG_VAL_TYPE</td>
<td>Variable to write data to</td>
</tr>
<tr>
<td>Inout</td>
<td>DataTxt</td>
<td>DataTxt</td>
<td>AC500_DiagTypes.DIAG_TXT_TYPE</td>
<td>Variable to write text to</td>
</tr>
</tbody>
</table>
Example System diagnosis: get numeric values and text for first diagnosis message

```plaintext
VAR
    fbDiag: AC500_Dia диаг.; (* instance of FB Diag *)
    eErrorID_First: AC500_Dia DIAG_TYPES.ERROR_ID; (* return value *)
    sFirstVal: AC500_Dia DIAG_VAL_TYPE;
    sFirstTxt: AC500_Dia DIAG_TXT_TYPE; (* structure of returned text *)
END_VAR

eErrorID_First := fbDiag.GetFirstValAndTxt(DataVal := sFirstVal, DataTxt := sFirstTxt);
```

Example Online mode: battery empty or missing

```
METHOD Get-xxx-ValAndTxtExt / DiagGet-xxx-ValAndTxtExt: numeric values, extended numeric values and text

-xxx- = First, Next, Last, Prev. Example: GetLastValAndTxtExt, DiagGetFirstValAndTxtExt

<table>
<thead>
<tr>
<th>Scop e</th>
<th>Name for device diagnosis</th>
<th>Name for device diagnosis</th>
<th>Type</th>
<th>Initial</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Return</td>
<td>Get-xxx-ValAndTxtExt</td>
<td>DiagGet-xxx-ValAndTxtExt</td>
<td>AC500_DiaG_TYPES.ERROR_ID</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Inout</td>
<td>DataVal</td>
<td>DataVal</td>
<td>AC500_DiaG_TYPES.DIAG_VAL_TYPE</td>
<td></td>
<td>Variable to write data to</td>
</tr>
</tbody>
</table>
```

Method Get-xxx-ValAndTxtExt / DiagGet-xxx-ValAndTxtExt: numeric values, extended numeric values and text
<table>
<thead>
<tr>
<th>Scope</th>
<th>Name for device diagnosis</th>
<th>Name for device diagnosis</th>
<th>Type</th>
<th>Initial</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inout</td>
<td>DataTxt</td>
<td>DataTxt</td>
<td>AC500_DiagTypes.DIAG_TXT_TYPE</td>
<td></td>
<td>Variable to write text to</td>
</tr>
<tr>
<td>Input</td>
<td>pExt</td>
<td>pExt</td>
<td>POINTER TO BYTE</td>
<td>0</td>
<td>Address of buffer to copy extended data to</td>
</tr>
<tr>
<td>Input</td>
<td>Size</td>
<td>Size</td>
<td>WORD</td>
<td>0</td>
<td>Size of buffer to copy extended data to</td>
</tr>
<tr>
<td>Inout</td>
<td>Length</td>
<td>Length</td>
<td>WORD</td>
<td>0</td>
<td>Size of extended data copied to buffer</td>
</tr>
</tbody>
</table>

**Example**

System diagnosis: get numeric values, extended numeric values and text of first diagnosis message

```plaintext
VAR
  fbDiag : AC500_Diag.Diag;
  eErrorID_First : AC500_DiagTypes.ERROR_ID;
  sFirstVal : AC500_DiagTypes.DIAG_VAL_TYPE;
  sFirstTxt : AC500_DiagTypes.DIAG_TXT_TYPE;
  abyExtData : ARRAY[0..1023] OF BYTE;
  wSize : WORD := 1024;
  wLength : WORD;
END_VAR

eErrorID_First := fbDiag.GetFirstValExt(DataVal := sFirstVal, DataTxt := sFirstTxt,
  pExt := ADR(abyExtData), Size := wSize, Length := wLength);
```

**Function DiagValToTxt**

Call this function to convert a numeric diagnosis message into a textual one at any time, in case this has not yet been done using a method providing both types when requesting this information.

<table>
<thead>
<tr>
<th>Scope</th>
<th>Name</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Return</td>
<td>DiagValToTxt</td>
<td>AC500_DiagTypes.ERROR_ID</td>
<td></td>
</tr>
<tr>
<td>Inout</td>
<td>DataInVal</td>
<td>AC500_DiagTypes.DIAG_VAL_TYPE</td>
<td>Variable to convert</td>
</tr>
<tr>
<td>Inout</td>
<td>DataOutTxt</td>
<td>AC500_DiagTypes.DIAG_TXT_TYPE</td>
<td>Variable to write text to</td>
</tr>
</tbody>
</table>
**1.6.1.5 Structure of error numbers**

**1.6.1.5.1 Error severity**

<table>
<thead>
<tr>
<th>Error severity</th>
<th>Type</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Fatal errors</td>
<td>Safe operation of the operating system is no longer ensured.</td>
<td>Checksum error in system flash, RAM error</td>
</tr>
<tr>
<td>2</td>
<td>Serious errors</td>
<td>The operating system works correctly, but the error-free execution of the user program is not ensured.</td>
<td>Checksum error in user flash, task cycle times exceeded</td>
</tr>
<tr>
<td>3</td>
<td>Light errors</td>
<td>It depends on the application whether the user program has to be stopped by the operating system or not. The user decides which reaction is to be done.</td>
<td>Flash memory cannot be programmed, I/O module failed</td>
</tr>
<tr>
<td>Error severity</td>
<td>Type</td>
<td>Description</td>
<td>Example</td>
</tr>
<tr>
<td>----------------</td>
<td>------------------</td>
<td>----------------------------------------------------------------------------------------------------------------</td>
<td>---------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>4</td>
<td>Warnings</td>
<td>Errors that occur on peripheral devices or that will have an effect only in the future. The user decides which reactions are to be done.</td>
<td>Short circuit in an I/O module, battery empty/not installed</td>
</tr>
<tr>
<td>11</td>
<td>Parameter error</td>
<td>Error occurred during parameter setting</td>
<td>Different I/O devices in PLC configuration and hardware installation</td>
</tr>
</tbody>
</table>

**Errors with error severity 1 - fatal errors**

Errors with error severity 1 are not entered in the diagnosis system. These errors do not allow normal operation of the PLC. These errors are detected during PLC start-up and stop the PLC immediately.

Examples are RAM errors or checksum errors when starting the firmware.

Such errors are indicated by rapid flashing of the ERR LED.

1.6.1.6 Diagnosis history

As of Automation Builder 2.4.0 / System FW 3.4.0 the diagnosis system has been extended with diagnosis history.

Diagnosis history is the entry of all diagnoses into a file according to their time of occurrence.

The diagnosis history file is in the root directory of the user disk and has the name "DiagHistory.csv". The max. number of entries is 5000. When 5000 entries are reached, the oldest entry is overwritten. The max. size of the extended data is 32 bytes.

An entry consists of following data:

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Comment</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>timestamp</td>
<td>ARRAYDT OF BYTE</td>
<td>RTC time of event in milliseconds consists of diTimestamp in DT format and uiMs milliseconds – see % Chapter 1.6.1.4.1.1 “Structure DIAG_VAL_TYPE” on page 35288STRUCT</td>
<td>1603371910177</td>
</tr>
<tr>
<td>event</td>
<td>BYTE</td>
<td>Event type (1=comes, 2=gone) – see % Chapter 1.6.1.4.1.5 “Enumeration teEvent” on page 3529</td>
<td>1</td>
</tr>
<tr>
<td>class</td>
<td>BYTE</td>
<td>Severity of error event – see % Chapter 1.6.1.4.1.4 “Enumeration teClass” on page 3529</td>
<td>4</td>
</tr>
<tr>
<td>compID</td>
<td>UDINT</td>
<td>Component ID</td>
<td>270540802</td>
</tr>
<tr>
<td>conn</td>
<td>UDINT</td>
<td>Connector</td>
<td>0xb17777ac</td>
</tr>
<tr>
<td>connIdx</td>
<td>UDINT</td>
<td>Connector index</td>
<td>0</td>
</tr>
<tr>
<td>sub</td>
<td>DWORD</td>
<td>SubsystemID: Any number describing detail/location within device, device specific</td>
<td>369098752</td>
</tr>
<tr>
<td>Name</td>
<td>Type</td>
<td>Comment</td>
<td>Example</td>
</tr>
<tr>
<td>-----------------</td>
<td>------------</td>
<td>-------------------------------------------------------------------------</td>
<td>---------</td>
</tr>
<tr>
<td>addl</td>
<td>DWORD</td>
<td>Additional ID: Additional number describing detail/location within device, optional, device specific</td>
<td>0</td>
</tr>
<tr>
<td>error</td>
<td>DWORD</td>
<td>Error code</td>
<td>9</td>
</tr>
<tr>
<td>extended data</td>
<td>ARRAYDT OF BYTE</td>
<td>Extended diagnosis data, max. 32 bytes</td>
<td></td>
</tr>
</tbody>
</table>

As shown in the example data of the diagnosis history file is not easily readable. The entries must be interpreted according to device and/or fieldbus. Therefore the Automation Builder consists a special view for diagnosis history "Chapter 1.5.6.2.3.3 “Diagnosis History” on page 3275.

With the entries CompID, conn and connID, the device generating the event is clearly identified in the device tree.

If the PLC configuration is changed, the values of this entries may be changed also. Therefore, the diagnosis history will be deleted during each download.

1.6.2 Online diagnosis in Automation Builder
1.6.2.1 Short description and overview
To use the diagnosis system in Automation Builder, login to the online mode is required "Chapter 1.6.2.2 “Entering/leaving the online mode” on page 3552. The online diagnosis in Automation Builder consists of a set of partly animated, mostly read only views. They can be invoked by a double-click on a project tree element which shows a circle indicating that this element is able to show diagnosis messages "Chapter 1.6.2.3 “Project tree in online mode” on page 3553.

Available online diagnosis and statistics:
- **Diagnosis messages**
  When the Automation Builder is switched to online mode, incoming diagnosis messages are displayed as plain-text "Chapter 1.6.1.3 “ Diagnosis in Automation Builder” on page 3524.
- **CPU/PLC diagnosis**
- **I/O module diagnosis**
  "Chapter 1.6.2.5 “Live values in views with I/O components” on page 3559.
- **Communication module and fieldbus diagnosis**
  "Chapter 1.6.2.6 “Communication module and fieldbus diagnosis” on page 3560
- **Diagnosis in IEC application**
  "Chapter 1.6.1.4 “Diagnosis in IEC application” on page 3526

For information on the disk status, diagnosis information can be read out with the Function Blocks PmDiskStatus and PmDiskLifetimeUsed "Chapter 1.5.7.4 “Health monitoring” on page 3517. PmDiskStatus PmDiskLifetimeUsed

1.6.2.2 Entering/leaving the online mode
Prerequisite: Set the gateway before entering the online mode "Chapter 1.5.6.2.11 “Gateway configuration” on page 3331. Gateway configuration
Enter the online mode

Right-click the “Application” node and select “Login”.

The Automation Builder project login to online mode updates the latest changes of the project.

![The online mode can be entered or left for each PLC in the project separately.]

Leave the online mode

Right-click the “Application” node and select “Logout”.

When online mode is active, a thread is running on Automation Builder project which sends cyclically a message to the PLC and expects a response. If the PLC does not respond, the online mode is left programmatically.

1.6.2.3 Project tree in online mode

When Automation Builder enters the online mode internally, it shows the state of all configured communication modules.

The connection status can be recognized by a symbol in the device tree:

- Device without diagnosis messages
- Device with diagnosis messages
- Device without diagnosis messages, but with diagnosis messages on at least one device in the branch below
- Device with diagnosis messages and with diagnosis messages on at least one device in the branch below
- Device does not respond to identification message and is not available for online connection
The identification is done in online mode.

- Double-click an element of the device-tree and select “Status” tab. Diagnosis information will be available.
The user will be notified in the device tree with an exclamation mark beside the device having diagnosis messages. The diagnosis messages are provided in the "Diagnosis" tab.

Alarms will be presented with a thunderbolt in the first column of the diagnosis grid.

Acknowledging an alarm

1. Stop diagnosis refreshment by clicking [Stop refresh].
2. Select one or more alarms and click [Acknowledge selected alarms].
Some diagnosis messages contain additional data. Click [View] button to see the additional diagnosis (in hex) for further analysis. If [View] button is not available, no additional data is available for this diagnosis message.

You can copy the additional data to the clipboard with [CTRL] + [C].

When building an IEC application in Automation Builder, diagnosis text lists will be generated and added to the device tree below the diagnosis folder. These text lists contain the device type specific diagnosis texts which are used by the diagnosis functions in the PLC application to show corresponding texts for error numbers.

The text lists will be downloaded automatically to the PLC with the visualization.

If there is a problem with downloading the text lists, make sure that the settings are correct:
1. Right-click on a text list and select “Properties”.
2. Open the “Text List” tab. The check box “Download by visualization” has to be selected.

1.6.2.4 CPU diagnosis views

1.6.2.4.1 Version information

Information on the firmware versions of the processor modules or communication modules, is provided on the “Version information” tab.

Remarks:

- The “Version information” tab displays the version identified on the device and the version provided with Automation Builder.
- The firmware on the devices must match to the Automation Builder version. Upgrade or downgrade to version supplied with Automation Builder is recommended (especially for CPUs) to ensure correct functionality.
- The firmware type can be changed to the type required by the hardware configuration for devices that support changing the firmware type. E.g., the onboard field bus communication modules of PM595 that may be used as PROFINET, Ethernet or EtherCAT communication module.
Firmware version on device matches version supplied with Automation Builder.

Firmware version (or type) on device is different from version supplied with Automation Builder. Upgrade/downgrade to version supplied with Automation Builder is recommended.

Only for communication modules if CPU firmware must be updated first. This happens when CPU firmware has version below 2.5.0.0. Firmware version (or type) on device is different from version supplied with Automation Builder. Upgrade/downgrade to version supplied with Automation Builder is recommended.

Identified device is different from configured device, thus no firmware update is possible. Happens only for Communication Modules.

Firmware of device is not updateable or no newer firmware than the initial version is available.

The [Update Firmware] button to download the new firmware is only enabled if there is updateable firmware.

1.6.2.4.2 PLC shell commands

The PLC shell is used for requesting specific information from the controller. By entering a device-specific command the response is returned in a result window. The PLC shell can be issued without login.

Proceed as follows:

1. Ensure the gateway is configured properly and a connection to the controller can be established.
2. In Automation Builder double-click the PLC node and open the tab “PLC Shell”.
3. Enter “?” in the command line of the tab window. All available PLC commands are listed.

If the gateway is able to establish a connection to the controller, an online connection to the PLC is opened automatically.
1.6.2.4.3 Statistics

**Setting the CPU time**

The PLC always uses “UTC time”.

It can be shown either in UTC or converted to the local time according to the date and time settings of the PC.

1.6.2.5 Live values in views with I/O components

“I/O mapping list” tab: In online mode, all Automation Builder views, which contain I/O component mapping tables, show animated live values which are updated every second.
1.6.2.6 Communication module and fieldbus diagnosis

1.6.2.6.1 Fieldbus commissioning

Common online diagnosis views for all netX-based coupler modules (e.g. CM579-ETHCAT, CM579-PNIO) can be accessed whenever the related PLC is in online mode "Entering/leaving the online mode" on page 3552.

Master/controller modules

Master/controller modules like CM579-ETHCAT or CM579-PNIO, provide the following diagnosis views:

- "Diagnostics main": provides diagnosis messages which are common for all protocols (e.g., protocol state and error)
- "Diagnostics live list": provides a list of connected slaves/devices and their state "PROFINET scan and comparison view" on page 3561
- "Diagnostics eventlog": provides diagnosis messages from the master/controller and its connected slaves/devices
PROFINET scan and comparison view

PNIO_Controller

1. After going online, double-click on “PNIO_Controller (PROFINET-IO-Controller)” in the device tree.
   ➔ The editor “PNIO_Controller” is displayed.

2. Select tab “Diagnostics live list” and click [Scan] to find all hardware devices that exist.
   ➔ The found devices are listed in a table.

3. Click [Compare] to compare the found hardware I/O devices with the current project configuration.

Unknown hardware

If any I/O hardware device is unknown:

● The devices will be marked with a red exclamation mark.
● A message box will be appear for each unknown device.
● Automation Builder generates a message with information about its vendor ID and device ID.
Comparison view

1. To display the comparison view, install the device description for the unknown device.
2. After installing the device description, click [Scan] and click [Compare].

   [Image]

   ⇨ The message box informs you, that the application will go offline to display the comparison view.

3. Click [Yes].
   ⇨ The “Project Comparison - Differences” tab displays the difference between the PROFINET configuration in Automation Builder (left side) and the real hardware configuration (right side).

4. Click [Accept Single] to accept only a part of the differences or [Accept Block] to accept all differences.
   ⇨ After clicking on the Button [Accept Single] or [Accept Block] the found devices will be moved from the right side to the left side.
5. Close tab “Project Comparison - Differences”.
   A message will be displayed to ask if you want to commit the new changes into project.

6. Click [Yes].
   The changes will be saved and the devices will be added to the project.
Slave/device coupler modules

Diagnosis views for slave/device coupler modules like CM589-PNIO:

- "Diagnostics main": provides diagnosis messages which are common for all protocols
- "Diagnostics details": provides protocol specific diagnosis messages

1.6.2.6.2 CI52x Modbus diagnosis

1. Double-click node “CI52x_MODTCP” in the device tree.
2. Select “CI52x Diagnosis” tab.
   - The button [Get Diagnosis] appears in the tab view.
3. Click on the button [Get Diagnosis].
   - One of the following use cases will be displayed:
     - Device not connected
     - No Errors on the device
     - Diagnosis list

Device not connected

If there is no device connected to the project, the following dialog will be displayed:

1. Select tab “Connection Settings” and enter the IP address for the device.
2. Click again button [Get Diagnosis].
No errors on the device  If there are no errors on the device the following dialog will be displayed:

![Error message dialog](image)

Diagnosis list  If the device is not correctly configured the errors will be displayed with “Error Code” and “Code Description”.

![Error code list](image)

### 1.6.3 Diagnosis messages

#### 1.6.3.1 CPU diagnosis

<table>
<thead>
<tr>
<th>Error severity</th>
<th>SubSysteminfo</th>
<th>Byte0</th>
<th>Byte1</th>
<th>Byte2</th>
<th>Byte3</th>
<th>Additional</th>
<th>Error code</th>
<th>Meaning</th>
<th>Remed y</th>
<th>Entry in text list</th>
<th>Diag V3 PLC</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>CPU</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>27</td>
<td>Failed to set parameter &quot;Reboot at powerfail&quot;</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>I/O bus</td>
<td>14</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>Resource failure</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>I/O bus</td>
<td>14</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>Timeout</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>I/O bus</td>
<td>14</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>17</td>
<td>Error setting I/O bus master parameter</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Display</td>
<td>21</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>23</td>
<td>Wrong version of display firmware</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Display</td>
<td>21</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>22</td>
<td>Error at initialization of display</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Battery</td>
<td>22</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>8</td>
<td>Empty or missing</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Error severity</td>
<td>SubSystem info</td>
<td>Additional</td>
<td>Error code</td>
<td>Meaning</td>
<td>Remed y</td>
<td>Entry in text list Diag_ V3_PL C</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>----------------</td>
<td>---------------</td>
<td>------------</td>
<td>------------</td>
<td>---------</td>
<td>--------</td>
<td>---------------------------------</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Byte0</td>
<td>Byte1</td>
<td>Byte2</td>
<td>Byte3</td>
<td>Err_x</td>
<td>Text snippets</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>19</td>
<td>Flash disk</td>
<td>5</td>
<td>Index</td>
<td>0...n</td>
<td>0...n</td>
<td>0</td>
<td>-</td>
<td>0</td>
<td>9</td>
<td>Medium has used 80 % of its spare capacity</td>
</tr>
<tr>
<td>4</td>
<td>19</td>
<td>Flash disk</td>
<td>5</td>
<td>Index</td>
<td>0...n</td>
<td>0...n</td>
<td>0</td>
<td>-</td>
<td>0</td>
<td>10</td>
<td>Medium has almost used its complete spare capacity or is already dead, action required</td>
</tr>
<tr>
<td>11</td>
<td>0</td>
<td>CPU</td>
<td>2</td>
<td>Parameter</td>
<td>0</td>
<td>-</td>
<td>0</td>
<td>-</td>
<td>0</td>
<td>18</td>
<td>At least one parameter not found</td>
</tr>
<tr>
<td>11</td>
<td>0</td>
<td>CPU</td>
<td>2</td>
<td>Parameter</td>
<td>0</td>
<td>-</td>
<td>0</td>
<td>-</td>
<td>0</td>
<td>19</td>
<td>Unable to read at least one parameter value</td>
</tr>
<tr>
<td>11</td>
<td>0</td>
<td>CPU</td>
<td>2</td>
<td>Parameter</td>
<td>0</td>
<td>-</td>
<td>1 or 2</td>
<td>'too big' or 'too small'</td>
<td>0</td>
<td>20</td>
<td>Invalid value of parameter &quot;error LED&quot;</td>
</tr>
<tr>
<td>11</td>
<td>0</td>
<td>CPU</td>
<td>2</td>
<td>Parameter</td>
<td>0</td>
<td>-</td>
<td>1 or 2</td>
<td>'too big' or 'too small'</td>
<td>0</td>
<td>21</td>
<td>Invalid value of parameter &quot;check battery&quot;</td>
</tr>
<tr>
<td>11</td>
<td>0</td>
<td>CPU</td>
<td>2</td>
<td>Parameter</td>
<td>0</td>
<td>-</td>
<td>1 or 2</td>
<td>'too big' or 'too small'</td>
<td>0</td>
<td>26</td>
<td>Invalid value of parameter &quot;Reboot at powerfail&quot;</td>
</tr>
</tbody>
</table>
## 1.6.3.2 I/O bus diagnosis

<table>
<thead>
<tr>
<th>Error severity</th>
<th>SubSysteminfo</th>
<th>Byte0</th>
<th>Byte1</th>
<th>Byte2</th>
<th>Byte3</th>
<th>Additional</th>
<th>Error code</th>
<th>Meaning</th>
<th>Remedy</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Value</td>
<td>Value</td>
<td>Value</td>
<td>Value</td>
<td></td>
<td></td>
<td>Text snippets</td>
<td>Entry in text list Diag_IO_Bus and Diag_S500_IO_Bus</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Meaning</td>
<td>Meaning</td>
<td>Meaning</td>
<td>Meaning</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>Master</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>16129 Critical error, see log</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>Master</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>16130 Fatal error, see log</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>Master</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>16194 Fatal error, not running any more</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>Master</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>16128 Failed Max Wait Run</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>1...20</td>
<td>Module n</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>9480 Module &lt;n&gt;, removed from hot swap terminal unit</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>1...20</td>
<td>Module n</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>9526 Module &lt;n&gt;, module on hot swap terminal unit does not support hot swap functionality</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>1...20</td>
<td>Module n</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>9764 Module &lt;n&gt;, defective hot swap terminal unit</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>1...20</td>
<td>Module n</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>16133 Module &lt;n&gt;, output data size mismatch</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>1...20</td>
<td>Module n</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>16134 Module &lt;n&gt;, input data size mismatch</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>0</td>
<td>Master</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>16145 Error setting I/O bus master parameter</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>0</td>
<td>Master</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>16146 Failed to start the parameterization of modules</td>
<td></td>
</tr>
<tr>
<td>Error severity</td>
<td>SubSysteminfo</td>
<td>Byte0</td>
<td>Byte1</td>
<td>Byte2</td>
<td>Byte3</td>
<td>Additional Error code</td>
<td>Meaning</td>
<td>Remed y</td>
<td>Entry in text list</td>
</tr>
<tr>
<td>----------------</td>
<td>---------------</td>
<td>-------</td>
<td>-------</td>
<td>-------</td>
<td>-------</td>
<td>-----------------------</td>
<td>---------</td>
<td>---------</td>
<td>-------------------</td>
</tr>
<tr>
<td>11</td>
<td>1...20</td>
<td>Module n</td>
<td>0</td>
<td>-</td>
<td>0</td>
<td>-</td>
<td>0</td>
<td>0</td>
<td>16147 Module (&lt;n&gt;), failed setting parameters</td>
</tr>
<tr>
<td>11</td>
<td>1...20</td>
<td>Module n</td>
<td>0</td>
<td>-</td>
<td>0</td>
<td>-</td>
<td>0</td>
<td>0</td>
<td>16149 Module (&lt;n&gt;), no module data</td>
</tr>
<tr>
<td>11</td>
<td>1...20</td>
<td>Module n</td>
<td>0</td>
<td>-</td>
<td>0</td>
<td>-</td>
<td>0</td>
<td>0</td>
<td>16158 Module (&lt;n&gt;), type of present module does not match configuration</td>
</tr>
<tr>
<td>11</td>
<td>0</td>
<td>Master</td>
<td>0</td>
<td>-</td>
<td>0</td>
<td>-</td>
<td>0</td>
<td>0</td>
<td>16159 Configured number of modules differs from found ones</td>
</tr>
<tr>
<td>11</td>
<td>0</td>
<td>Master</td>
<td>0</td>
<td>-</td>
<td>0</td>
<td>-</td>
<td>0</td>
<td>0</td>
<td>16160 At least one module failed during configuration</td>
</tr>
<tr>
<td>11</td>
<td>1...20</td>
<td>Module n</td>
<td>0</td>
<td>-</td>
<td>0</td>
<td>-</td>
<td>0</td>
<td>0</td>
<td>16248 Module (&lt;n&gt;), failed setting expected module</td>
</tr>
<tr>
<td>11</td>
<td>1...20</td>
<td>Module n</td>
<td>0</td>
<td>-</td>
<td>0</td>
<td>-</td>
<td>0</td>
<td>0</td>
<td>16254 Module (&lt;n&gt;), size of parameters expected by module differs from size provided by configuration</td>
</tr>
</tbody>
</table>
### 1.6.3.3 S500 I/O modules diagnosis

<table>
<thead>
<tr>
<th>Error severity</th>
<th>SubSysteminfo</th>
<th>Byte0</th>
<th>Byte1</th>
<th>Byte2</th>
<th>Byte3</th>
<th>Additional</th>
<th>Error code</th>
<th>Meaning</th>
<th>Remedy</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>1...20 Module n</td>
<td>0</td>
<td>-</td>
<td>-</td>
<td>0</td>
<td>0</td>
<td>8482</td>
<td>Timeout, while initializing an I/O module</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>1...20 Module n</td>
<td>0</td>
<td>-</td>
<td>-</td>
<td>0</td>
<td>0</td>
<td>8432</td>
<td>Timeout while initializing an I/O module</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>1...20 Module n</td>
<td>0</td>
<td>-</td>
<td>-</td>
<td>0</td>
<td>0</td>
<td>9249</td>
<td>Timeout while waiting for Reset</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>1...20 Module n</td>
<td>0</td>
<td>-</td>
<td>-</td>
<td>0</td>
<td>0</td>
<td>9258</td>
<td>Module breakdown, communication lost</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>1...20 Module n</td>
<td>1</td>
<td>Channel 0...15</td>
<td>0...15</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>Channel &lt;n&gt;, discrepancy time expired</td>
<td>Check discrepancy time value, channel wiring and sensor.</td>
</tr>
<tr>
<td>3</td>
<td>1...20 Module n</td>
<td>1</td>
<td>Channel 0...15</td>
<td>0...15</td>
<td>0</td>
<td>0</td>
<td>12</td>
<td>Channel &lt;n&gt;, test pulse error</td>
<td>Check wiring and sensor</td>
</tr>
<tr>
<td>3</td>
<td>1...20 Module n</td>
<td>1</td>
<td>Channel 0...15</td>
<td>0...15</td>
<td>0</td>
<td>0</td>
<td>13</td>
<td>Channel &lt;n&gt;, test pulse cross-talk error</td>
<td>Check wiring and sensor. If this error persists, replace I/O module. Contact ABB technical support.</td>
</tr>
<tr>
<td>Error severity</td>
<td>SubSysteminfo</td>
<td>Byte0</td>
<td>Byte1</td>
<td>Byte2</td>
<td>Byte3</td>
<td>Additional</td>
<td>Error code</td>
<td>Meaning</td>
<td>Remedy</td>
</tr>
<tr>
<td>----------------</td>
<td>---------------</td>
<td>-------</td>
<td>-------</td>
<td>-------</td>
<td>-------</td>
<td>------------</td>
<td>------------</td>
<td>---------</td>
<td>--------</td>
</tr>
<tr>
<td></td>
<td>Value</td>
<td>Value</td>
<td>Value</td>
<td>Value</td>
<td>Value</td>
<td>Sub 1_x</td>
<td>Sub 2_x</td>
<td>Value</td>
<td>Mea</td>
</tr>
<tr>
<td></td>
<td>Meaning</td>
<td>Meaning</td>
<td>Meaning</td>
<td>Meaning</td>
<td>Meaning</td>
<td>Text snippets</td>
<td>Entr</td>
<td>Check</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>1...20</td>
<td>Module n</td>
<td>1</td>
<td>Channel</td>
<td>0...15</td>
<td>0</td>
<td>-</td>
<td>0</td>
<td>18</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>1...20</td>
<td>Module n</td>
<td>1</td>
<td>Channel</td>
<td>0...15</td>
<td>0</td>
<td>-</td>
<td>0</td>
<td>25</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>1...20</td>
<td>Module n</td>
<td>1</td>
<td>Channel</td>
<td>0...15</td>
<td>0</td>
<td>-</td>
<td>0</td>
<td>28</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>1...20</td>
<td>Module n</td>
<td>1</td>
<td>Channel</td>
<td>0...3</td>
<td>0</td>
<td>-</td>
<td>0</td>
<td>273</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>1...20</td>
<td>Module n</td>
<td>1</td>
<td>Channel</td>
<td>0...3</td>
<td>0</td>
<td>-</td>
<td>0</td>
<td>311</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>1...20</td>
<td>Module n</td>
<td>1</td>
<td>Channel</td>
<td>0...7</td>
<td>0</td>
<td>-</td>
<td>0</td>
<td>524</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Error severity</td>
<td>SubSystem info</td>
<td>Byte0 Value</td>
<td>Byte1 Value</td>
<td>Byte2 Value</td>
<td>Byte3 Value</td>
<td>Additional Error code</td>
<td>Meaning</td>
<td>Remedy</td>
<td></td>
</tr>
<tr>
<td>----------------</td>
<td>----------------</td>
<td>-------------</td>
<td>-------------</td>
<td>-------------</td>
<td>-------------</td>
<td>----------------------</td>
<td>---------</td>
<td>--------</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Module n</td>
<td>1</td>
<td>Channel 0...7</td>
<td>0...7</td>
<td>0</td>
<td>0</td>
<td>525</td>
<td>Channel &lt;n&gt;, read-back error</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Check I/O module wiring. Restart I/O module, if needed. If this error persists, replace I/O module.</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Module n</td>
<td>1</td>
<td>Channel 0...15</td>
<td>0...15</td>
<td>0</td>
<td>0</td>
<td>530</td>
<td>Channel &lt;n&gt;, cross-talk error</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Check I/O module wiring. Restart I/O module, if needed. If this error persists, replace I/O module.</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Module n</td>
<td>1</td>
<td>Channel 0...15</td>
<td>0...15</td>
<td>0</td>
<td>0</td>
<td>540</td>
<td>Channel &lt;n&gt;, test error</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Module n</td>
<td>1</td>
<td>Channel 0...7</td>
<td>0...7</td>
<td>0</td>
<td>0</td>
<td>555</td>
<td>Channel &lt;n&gt;, internal error</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Module n</td>
<td>1</td>
<td>Channel 0...7</td>
<td>0...7</td>
<td>0</td>
<td>0</td>
<td>1037</td>
<td>Channel &lt;n&gt;, test error</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Module n</td>
<td>0</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>0</td>
<td>8480</td>
<td>Not supported protocol variant</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Module n</td>
<td>0</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>0</td>
<td>8707</td>
<td>PROFIsafe watchdog timed out</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Module n</td>
<td>0</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>0</td>
<td>8708</td>
<td>Overvoltage</td>
<td></td>
</tr>
<tr>
<td>Error severity</td>
<td>SubSysteminfo</td>
<td>Byte0 Value</td>
<td>Byte0 Meaning</td>
<td>Byte1 Value</td>
<td>Byte1 Meaning</td>
<td>Byte2 Value</td>
<td>Byte2 Meaning</td>
<td>Byte3 Value</td>
<td>Byte3 Meaning</td>
</tr>
<tr>
<td>---------------</td>
<td>---------------</td>
<td>-------------</td>
<td>---------------</td>
<td>-------------</td>
<td>---------------</td>
<td>-------------</td>
<td>---------------</td>
<td>-------------</td>
<td>---------------</td>
</tr>
<tr>
<td>3</td>
<td>1...20 Module n</td>
<td>0</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>0</td>
<td>-</td>
<td>0</td>
<td>-</td>
</tr>
<tr>
<td>3</td>
<td>1...20 Module n</td>
<td>0</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>0</td>
<td>-</td>
<td>0</td>
<td>-</td>
</tr>
<tr>
<td>3</td>
<td>1...20 Module n</td>
<td>0</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>0</td>
<td>-</td>
<td>0</td>
<td>-</td>
</tr>
<tr>
<td>3</td>
<td>1...20 Module n</td>
<td>0</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>0</td>
<td>-</td>
<td>0</td>
<td>-</td>
</tr>
<tr>
<td>3</td>
<td>1...20 Module n</td>
<td>0</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>0</td>
<td>-</td>
<td>0</td>
<td>-</td>
</tr>
<tr>
<td>3</td>
<td>1...20 Module n</td>
<td>0</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>0</td>
<td>-</td>
<td>0</td>
<td>-</td>
</tr>
<tr>
<td>3</td>
<td>1...20 Module n</td>
<td>0</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>0</td>
<td>-</td>
<td>0</td>
<td>-</td>
</tr>
<tr>
<td>3</td>
<td>1...20 Module n</td>
<td>0</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>0</td>
<td>-</td>
<td>0</td>
<td>-</td>
</tr>
<tr>
<td>3</td>
<td>1...20 Module n</td>
<td>0</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>0</td>
<td>-</td>
<td>0</td>
<td>-</td>
</tr>
<tr>
<td>3</td>
<td>1...20 Module n</td>
<td>0</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>0</td>
<td>-</td>
<td>0</td>
<td>-</td>
</tr>
<tr>
<td>3</td>
<td>1...20 Module n</td>
<td>0</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>0</td>
<td>-</td>
<td>0</td>
<td>-</td>
</tr>
<tr>
<td>Error severity</td>
<td>SubSysteminfo</td>
<td>Byte0</td>
<td>Byte1</td>
<td>Byte2</td>
<td>Byte3</td>
<td>Additional</td>
<td>Error code</td>
<td>Meaning</td>
<td>Remedy</td>
</tr>
<tr>
<td>----------------</td>
<td>---------------</td>
<td>-------</td>
<td>-------</td>
<td>-------</td>
<td>-------</td>
<td>------------</td>
<td>------------</td>
<td>---------</td>
<td>--------</td>
</tr>
<tr>
<td>3</td>
<td>1...20 Module n</td>
<td>0</td>
<td>-</td>
<td>-</td>
<td>0</td>
<td>0</td>
<td>16137</td>
<td>Overflow diagnosis buffer</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>1...20 Module n</td>
<td>0</td>
<td>-</td>
<td>-</td>
<td>0</td>
<td>0</td>
<td>16138</td>
<td>Process voltage too high</td>
<td>Check process voltage</td>
</tr>
<tr>
<td>3</td>
<td>1...20 Module n</td>
<td>0</td>
<td>-</td>
<td>-</td>
<td>0</td>
<td>0</td>
<td>16139</td>
<td>Process voltage too low</td>
<td>Check process voltage</td>
</tr>
<tr>
<td>3</td>
<td>1...20 Module n</td>
<td>0</td>
<td>-</td>
<td>-</td>
<td>0</td>
<td>0</td>
<td>16146</td>
<td>Plausibility check failed (iParameter)</td>
<td>Check configuration</td>
</tr>
<tr>
<td>3</td>
<td>1...20 Module n</td>
<td>0</td>
<td>-</td>
<td>-</td>
<td>0</td>
<td>0</td>
<td>16147</td>
<td>Checksum error</td>
<td>Non-safety I/O: Replace I/O module. Safety I/O: Check safety configuration and CRCs for I- and F-Parameters.</td>
</tr>
<tr>
<td>3</td>
<td>1...20 Module n</td>
<td>0</td>
<td>-</td>
<td>-</td>
<td>0</td>
<td>0</td>
<td>16148</td>
<td>PROFIsafe communication error</td>
<td>Restart I/O module. If this error persists, contact ABB technical support.</td>
</tr>
<tr>
<td>Error severity</td>
<td>SubSysteminfo</td>
<td>Additional</td>
<td>Error code</td>
<td>Meaning</td>
<td>Remedy</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>----------------</td>
<td>---------------</td>
<td>------------</td>
<td>-----------</td>
<td>---------</td>
<td>--------</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>1...20 Module n</td>
<td>0 - - - 0 - 0</td>
<td>16153</td>
<td>PROFIsafe watchdog timed out</td>
<td>Restart I/O module. If this error persists, increase PROFIsafe watchdog time.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>1...20 Module n</td>
<td>0 - - - 0 - 0</td>
<td>16154</td>
<td>Size of parameters expected by module differs from size provided by configuration</td>
<td>Check master or configuration.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>1...20 Module n</td>
<td>0 - - - 0 - 0</td>
<td>16156</td>
<td>F-Parameter configuration and address switch value do not match.</td>
<td>Check I/O module F-Parameter configuration and module address switch value.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>1...20 Module n</td>
<td>0 - - - 0 - 0</td>
<td>16164</td>
<td>Internal data interchange disturbed</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>1...20 Module n</td>
<td>0 - - - 0 - 0</td>
<td>16168</td>
<td>Different hardware and firmware versions</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>1...20 Module n</td>
<td>0 - - - 0 - 0</td>
<td>16171</td>
<td>Internal error</td>
<td>Replace I/O module</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>1...20 Module n</td>
<td>0 - - - 0 - 0</td>
<td>16174</td>
<td>Sensor voltage too low</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Error severity</td>
<td>SubSysteminfo</td>
<td>Additional</td>
<td>Error code</td>
<td>Meaning</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>---------------</td>
<td>--------------</td>
<td>------------</td>
<td>-----------</td>
<td>---------</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Byte0 Value</td>
<td>Byte1 Value</td>
<td>Byte2 Value</td>
<td>Byte3 Value</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Interpretation</td>
<td>Interpretation</td>
<td>Interpretation</td>
<td>Interpretation</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Byte0 Meaning</td>
<td>Byte1 Meaning</td>
<td>Byte2 Meaning</td>
<td>Byte3 Meaning</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sub 1_x</td>
<td>Sub 2_x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Byte0</td>
<td>Byte1</td>
<td>Byte2</td>
<td>Byte3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Channel</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Channel</td>
<td>0...7</td>
<td>0...7</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Channel</td>
<td>0...7</td>
<td>0...7</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Channel</td>
<td>0...7</td>
<td>0...7</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Channel</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Channel</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Channel</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **Err_x**: Text snippets
- **Remedy**: Entry in text list, Diag_IO_Bus and Diag_S500_IO_Bus_V3_PLC

<table>
<thead>
<tr>
<th>Error severity</th>
<th>SubSysteminfo</th>
<th>Additional</th>
<th>Error code</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Byte0 Value</td>
<td>Byte1 Value</td>
<td>Byte2 Value</td>
<td>Byte3 Value</td>
<td></td>
</tr>
<tr>
<td>Interpretation</td>
<td>Interpretation</td>
<td>Interpretation</td>
<td>Interpretation</td>
<td></td>
</tr>
<tr>
<td>Byte0 Meaning</td>
<td>Byte1 Meaning</td>
<td>Byte2 Meaning</td>
<td>Byte3 Meaning</td>
<td></td>
</tr>
<tr>
<td>Sub 1_x</td>
<td>Sub 2_x</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Byte0</td>
<td>Byte1</td>
<td>Byte2</td>
<td>Byte3</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Channel</td>
<td>0...7</td>
<td>0...7</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>Channel</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>Channel</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>Channel</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>Channel</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

- **Err_x**: Text snippets
- **Remedy**: Entry in text list, Diag_IO_Bus and Diag_S500_IO_Bus_V3_PLC

- **Entry in text list**: Diag_IO_Bus and Diag_S500_IO_Bus_V3_PLC

- **Remedy**: Check channel wiring and sensor power supply.
<table>
<thead>
<tr>
<th>Error severity</th>
<th>SubSysteminfo</th>
<th>Byte0</th>
<th>Byte1</th>
<th>Byte2</th>
<th>Byte3</th>
<th>Additional</th>
<th>Error code</th>
<th>Meaning</th>
<th>Remed y</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Value Meaning</td>
<td>Value Meaning</td>
<td>Value Meaning</td>
<td>Value Meaning</td>
<td></td>
<td></td>
<td>Err_x</td>
<td>Text snippets</td>
<td>Entry in text list</td>
</tr>
<tr>
<td>4</td>
<td>1...20 Module n</td>
<td>1</td>
<td>Channel</td>
<td>0...7</td>
<td>0...7</td>
<td>0</td>
<td>-</td>
<td>0</td>
<td>267</td>
</tr>
<tr>
<td>4</td>
<td>1...20 Module n</td>
<td>1</td>
<td>Channel</td>
<td>0...7</td>
<td>0...7</td>
<td>0</td>
<td>-</td>
<td>0</td>
<td>303</td>
</tr>
<tr>
<td>4</td>
<td>1...20 Module n</td>
<td>1</td>
<td>Channel</td>
<td>0...7</td>
<td>0...7</td>
<td>0</td>
<td>-</td>
<td>0</td>
<td>304</td>
</tr>
<tr>
<td>4</td>
<td>1...20 Module n</td>
<td>1</td>
<td>Channel</td>
<td>0...15</td>
<td>0...15</td>
<td>0</td>
<td>-</td>
<td>0</td>
<td>530</td>
</tr>
<tr>
<td>4</td>
<td>1...20 Module n</td>
<td>1</td>
<td>Channel</td>
<td>0...23</td>
<td>0...23</td>
<td>0</td>
<td>-</td>
<td>0</td>
<td>559</td>
</tr>
<tr>
<td>4</td>
<td>1...20 Module n</td>
<td>1</td>
<td>Channel</td>
<td>0...7</td>
<td>0...7</td>
<td>0</td>
<td>-</td>
<td>0</td>
<td>775</td>
</tr>
<tr>
<td>4</td>
<td>1...20 Module n</td>
<td>1</td>
<td>Channel</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>-</td>
<td>0</td>
<td>796</td>
</tr>
<tr>
<td>4</td>
<td>1...20 Module n</td>
<td>0</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>0</td>
<td>-</td>
<td>0</td>
<td>8482</td>
</tr>
<tr>
<td>4</td>
<td>1...20 Module n</td>
<td>0</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>0</td>
<td>-</td>
<td>0</td>
<td>8483</td>
</tr>
<tr>
<td>Error severity</td>
<td>SubSysteminfo</td>
<td>Byte0 Value</td>
<td>Byte1 Meaning</td>
<td>Byte2 Value</td>
<td>Byte3 Meaning</td>
<td>Additional Err_x</td>
<td>Error code</td>
<td>Meaning</td>
<td>Remedy</td>
</tr>
<tr>
<td>----------------</td>
<td>--------------</td>
<td>-------------</td>
<td>---------------</td>
<td>-------------</td>
<td>---------------</td>
<td>------------------</td>
<td>------------</td>
<td>---------</td>
<td>--------</td>
</tr>
<tr>
<td>4 1...20</td>
<td>Module n</td>
<td>0 - - - 0 - 0</td>
<td>9480</td>
<td>Module removed from hot swap terminal unit</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4 1...20</td>
<td>Module n</td>
<td>0 - - - 0 - 0</td>
<td>9500</td>
<td>Wrong module replugged on hot swap terminal unit</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4 1...20</td>
<td>Module n</td>
<td>0 - - - 0 - 0</td>
<td>9514</td>
<td>Lost communication to module on hot swap</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4 1...20</td>
<td>Module n</td>
<td>0 - - - 0 - 0</td>
<td>9526</td>
<td>Module on hot swap terminal unit doesn't support hot swap functionality</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4 1...20</td>
<td>Module n</td>
<td>0 - - - 0 - 0</td>
<td>9736</td>
<td>Hot swap terminal unit required, but not found</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4 1...20</td>
<td>Module n</td>
<td>0 - - - 0 - 0</td>
<td>9770</td>
<td>Lost communication to hot swap terminal unit</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4 1...20</td>
<td>Module n</td>
<td>0 - - - 0 - 0</td>
<td>16172</td>
<td>Has not passed factory test</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4 1...20</td>
<td>Module n</td>
<td>0 - - - 0 - 0</td>
<td>16173</td>
<td>Process voltage switched off</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11 0</td>
<td>0 - 0 - 0 - 0</td>
<td>16159</td>
<td>Configured number of modules differs from found ones</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11 0</td>
<td>0 - 0 - 0 - 0</td>
<td>16160</td>
<td>At least one module failed during configuration</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### 1.6.3.4 Communication modules diagnosis

#### 1.6.3.4.1 AC500-S: errors from safety CPU and safety I/O modules

<table>
<thead>
<tr>
<th>Table 781: Error messages for safety CPU SM560-S</th>
</tr>
</thead>
<tbody>
<tr>
<td>Severity</td>
</tr>
<tr>
<td>----------</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>Severity</td>
</tr>
<tr>
<td>----------</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>3</td>
</tr>
<tr>
<td>3</td>
</tr>
<tr>
<td>3</td>
</tr>
<tr>
<td>3</td>
</tr>
<tr>
<td>3</td>
</tr>
<tr>
<td>3</td>
</tr>
<tr>
<td>3</td>
</tr>
<tr>
<td>3</td>
</tr>
<tr>
<td>3</td>
</tr>
<tr>
<td>3</td>
</tr>
<tr>
<td>Severity</td>
</tr>
<tr>
<td>----------</td>
</tr>
<tr>
<td>4</td>
</tr>
<tr>
<td>4</td>
</tr>
<tr>
<td>4</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>4</td>
</tr>
<tr>
<td>4</td>
</tr>
<tr>
<td>4</td>
</tr>
<tr>
<td>4</td>
</tr>
<tr>
<td>4</td>
</tr>
<tr>
<td>4</td>
</tr>
<tr>
<td>4</td>
</tr>
<tr>
<td>4</td>
</tr>
<tr>
<td>4</td>
</tr>
<tr>
<td>4</td>
</tr>
<tr>
<td>4</td>
</tr>
<tr>
<td>4</td>
</tr>
<tr>
<td>4</td>
</tr>
<tr>
<td>4</td>
</tr>
<tr>
<td>4</td>
</tr>
</tbody>
</table>

Table 782: Error messages for safety I/O modules (channel or module reintegration is possible)

<table>
<thead>
<tr>
<th>Severity</th>
<th>Error code</th>
<th>Description</th>
<th>Remedy</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>3</td>
<td>Discrepancy time expired</td>
<td>Check discrepancy time value, channel wiring and sensor.</td>
</tr>
<tr>
<td>3</td>
<td>12</td>
<td>Test pulse error</td>
<td>Check wiring and sensor.</td>
</tr>
</tbody>
</table>

PLC Automation with V3 CPUs
Diagnosis and debugging for AC500 V3 products > Diagnosis messages
<table>
<thead>
<tr>
<th>Severity</th>
<th>Error code</th>
<th>Description</th>
<th>Remedy</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>13</td>
<td>Channel test pulse cross-talk error</td>
<td>Check wiring and sensor. If this error persists, replace I/O module. Contact ABB technical support.</td>
</tr>
<tr>
<td>3</td>
<td>25</td>
<td>Channel stuck-at error</td>
<td>Check I/O module wiring. Restart I/O module, if needed. If this error persists, replace I/O module.</td>
</tr>
<tr>
<td>3</td>
<td>28</td>
<td>Channel cross-talk error</td>
<td>Check I/O module wiring. Restart I/O module, if needed. If this error persists, replace I/O module.</td>
</tr>
<tr>
<td>3</td>
<td>260</td>
<td>Measurement overflow at the I/O module</td>
<td>Check channel wiring and sensor power supply.</td>
</tr>
<tr>
<td>3</td>
<td>263</td>
<td>Measurement underflow at the I/O module</td>
<td>Check channel wiring and sensor power supply.</td>
</tr>
<tr>
<td>3</td>
<td>311</td>
<td>Channel value difference too high</td>
<td>Adjust tolerance window for channels. Check channel wiring and sensor configuration.</td>
</tr>
<tr>
<td>3</td>
<td>525</td>
<td>Channel readback error</td>
<td>Check I/O module wiring. Restart I/O module, if needed. If this error persists, replace I/O module.</td>
</tr>
<tr>
<td>3</td>
<td>530</td>
<td>Channel cross-talk error</td>
<td>Check I/O module wiring. Restart I/O module, if needed. If this error persists, replace I/O module.</td>
</tr>
<tr>
<td>3</td>
<td>16138</td>
<td>Process voltage too high</td>
<td>Check process voltage</td>
</tr>
<tr>
<td>3</td>
<td>16139</td>
<td>Process voltage too low</td>
<td>Check process voltage</td>
</tr>
<tr>
<td>3</td>
<td>16148</td>
<td>PROFINet communication error</td>
<td>Restart I/O module. If this error persists, contact ABB technical support.</td>
</tr>
<tr>
<td>3</td>
<td>16153</td>
<td>PROFINet watchdog timed out.</td>
<td>Restart I/O module. If this error persists, increase PROFINet watchdog time.</td>
</tr>
<tr>
<td>3</td>
<td>16171</td>
<td>Internal error in the device</td>
<td>Replace I/O module</td>
</tr>
</tbody>
</table>

Table 783: Error messages for safety I/O modules (channel or module reintegration ist not possible)

<table>
<thead>
<tr>
<th>Severity</th>
<th>Error code</th>
<th>Description</th>
<th>Remedy</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>16146</td>
<td>Plausibility check failed (iParameter)</td>
<td>Check configuration</td>
</tr>
<tr>
<td>3</td>
<td>16147</td>
<td>Checksum error in the I/O module</td>
<td>Check safety configuration and CRCs for I- and F-Parameters.</td>
</tr>
<tr>
<td>3</td>
<td>16154</td>
<td>Parameter value</td>
<td>Check master or configuration</td>
</tr>
<tr>
<td>3</td>
<td>16156</td>
<td>F-Parameter configuration and address switch value do not match.</td>
<td>Check I/O module F-Parameter configuration and module address switch value.</td>
</tr>
</tbody>
</table>
### 16.3.4.2 CM579-PNIO – PROFINET I/O controller diagnosis

Diagnosis data for CM579-PNIO is not displayed in PLC display. In Automation Builder, we recommend to use methods with text output to get diagnosis messages in clear text format. E.g., DiagGetFirstValAndTxt.

Output string:

```
<timestamp>; <error severity>; <device name>; <error location>; error ID <id>: <error text>
```

If you need to access the diagnosis data directly, you have to interpret them manually. Refer to the example to learn how to interpret them correctly "Chapter 16.3.4.2.1 "Manual interpretation of CM579-PNIO diagnosis" on page 3586.

#### For experts: manual interpretation

If you need to access the diagnosis data directly, you have to interpret them manually. Refer to the example to learn how to interpret them correctly "Chapter 16.3.4.2.1 "Manual interpretation of CM579-PNIO diagnosis" on page 3586.

<table>
<thead>
<tr>
<th>Error severity</th>
<th>SubSysteminfo</th>
<th>Additional</th>
<th>Error code</th>
<th>Meaning</th>
<th>Remedy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Word 1 (ADD_SUB_TYPE)</td>
<td>Word 1 (ADD_TYPE)</td>
<td>Word 1 (ADD_TYPE)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sub1_</td>
<td>Sub2_</td>
<td>Add_Word 1_Word 2</td>
<td>Err_x or Err_Word 1_Word 2</td>
<td>Text Entry in text list</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(Word1/2 in hex format)</td>
<td>Diag_PNIO_Controller and Diag_PNIO_Vendor ID_Device ID</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>tbd</td>
<td>tbd</td>
<td>0</td>
<td>1 (general)</td>
<td>Err_Gen_x</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2 (runtime)</td>
<td>Err_Rt_x</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2 (runtime)</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2 (runtime)</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2 (runtime)</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2 (runtime)</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>tbd</td>
<td>tbd</td>
<td>0</td>
<td>3 (configuration)</td>
<td>Err_Cfg_x</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Subslot index (0 – 16#9FFF)</td>
<td>Channel index (0 – 16#7FFF)</td>
<td>1 (USI: 16#8000) channel diagnosis</td>
<td>4 (diagnosis alarm)</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Subslot index (0 – 16#9FFF)</td>
<td>Channel index (16#8000)</td>
<td>1 (USI: 16#8000) channel diagnosis</td>
<td>4 (diagnosis alarm)</td>
<td>0</td>
</tr>
<tr>
<td>Error severity</td>
<td>SubSysteminfo</td>
<td>Additional</td>
<td>Error code</td>
<td>Meaning</td>
<td>Remedy</td>
</tr>
<tr>
<td>---------------</td>
<td>--------------</td>
<td>------------</td>
<td>------------</td>
<td>---------</td>
<td>--------</td>
</tr>
<tr>
<td></td>
<td>Word 2 (bit 16..31)</td>
<td>Word 1 (bit 0..15)</td>
<td>Word 2 (bit 16..31)</td>
<td>Word 1 (bit 0..15)</td>
<td>Word 2 (bit 16...31)</td>
</tr>
<tr>
<td></td>
<td>Sub1_</td>
<td>Sub2_</td>
<td>Add_Word 1_Word 2</td>
<td>Err_x or Err_Word 1_Word 2 (Word1/2 in hex format)</td>
<td>Text</td>
</tr>
<tr>
<td>3</td>
<td>Subslot index (0 – 16#9FFF)</td>
<td>Channel index (0 – 16#7FFF)</td>
<td>2 (USI: 16#8002) extended channel diagnosis</td>
<td>4 (diagnosis alarm)</td>
<td>Extended error Type</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Subslot &lt;subslot idx&gt;, channel &lt;channel idx&gt;, extended channel diagnosis; &lt;error text&gt;</td>
</tr>
<tr>
<td>3</td>
<td>Subslot index (0 – 16#9FFF)</td>
<td>Channel index (16#8000)</td>
<td>2 (USI: 16#8002) extended channel diagnosis</td>
<td>4 (diagnosis alarm)</td>
<td>Extended error Type</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Subslot &lt;subslot idx&gt;, extended channel diagnosis; &lt;error text&gt;</td>
</tr>
<tr>
<td>3</td>
<td>Subslot index (0 – 16#9FFF)</td>
<td>Channel index (0 – 16#7FFF)</td>
<td>3 (USI: 16#8003) qualified channel diagnosis</td>
<td>4 (diagnosis alarm)</td>
<td>Extended error Type</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Subslot &lt;subslot idx&gt;, channel &lt;channel idx&gt;, qualified channel diagnosis; &lt;error text&gt;</td>
</tr>
<tr>
<td>3</td>
<td>Subslot index (0 – 16#9FFF)</td>
<td>Channel index (16#8000)</td>
<td>3 (USI: 16#8003) qualified channel</td>
<td>4 (diagnosis alarm)</td>
<td>Extended error Type</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Subslot &lt;subslot idx&gt;, qualified channel diagnosis; &lt;error text&gt;</td>
</tr>
<tr>
<td>2, 3, 4, 11</td>
<td>Subslot index (0 – 16#9FFF)</td>
<td>Channel index (0 – 16#7FFF)</td>
<td>0</td>
<td>5 (S500 process alarm)</td>
<td>32 bit error code</td>
</tr>
<tr>
<td>2, 3, 4, 11</td>
<td>Subslot index (0 – 16#9FFF)</td>
<td>Channel index (16#8000)</td>
<td>0</td>
<td>5 (S500 process alarm)</td>
<td>32 bit error code</td>
</tr>
<tr>
<td>3</td>
<td>Subslot index (0 – 16#9FFF)</td>
<td>Channel index (0 – 16#7FFF)</td>
<td>Alarm type = (14 .. 30) &amp; (32 ..)</td>
<td>6 (alarm)</td>
<td>Alarm type = (14 .. 30) &amp; (32 ..)</td>
</tr>
<tr>
<td>3</td>
<td>Subslot index (0 – 16#9FFF)</td>
<td>Channel index (16#8000)</td>
<td>Alarm type = (14 .. 30) &amp; (32 ..)</td>
<td>6 (alarm)</td>
<td>Alarm type = (14 .. 30) &amp; (32 ..)</td>
</tr>
<tr>
<td>3</td>
<td>Subslot index (0 – 16#9FFF)</td>
<td>Channel index (0 – 16#7FFF)</td>
<td>Alarm type = 1</td>
<td>6 (alarm)</td>
<td>Alarm type = 1</td>
</tr>
<tr>
<td>3</td>
<td>Subslot index (0 – 16#9FFF)</td>
<td>Channel index (16#8000)</td>
<td>Alarm type = 1</td>
<td>6 (alarm)</td>
<td>Alarm type = 1</td>
</tr>
<tr>
<td>3</td>
<td>Subslot index (0 – 16#9FFF)</td>
<td>Channel index (0 – 16#7FFF)</td>
<td>Alarm type = 2</td>
<td>6 (alarm)</td>
<td>Alarm type = 2</td>
</tr>
<tr>
<td>Error severity</td>
<td>SubSysteminfo</td>
<td>Additional</td>
<td>Error code</td>
<td>Meaning</td>
<td>Remedy</td>
</tr>
<tr>
<td>----------------</td>
<td>---------------</td>
<td>------------</td>
<td>------------</td>
<td>---------</td>
<td>--------</td>
</tr>
<tr>
<td>Sub1_</td>
<td>Sub2_</td>
<td>Add_Word 1_Word 2</td>
<td>Err_x or Err_Word 1_Word 2 (Word1/2 in hex format)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

```
3 Subslot index (0 – 16#9FFF) Channel index (16#8000) Alarm type = 2 6 (alarm) Alarm type = 2 Subslot <subslot idx>, process alarm; <error text>
3 Subslot index (0 – 16#9FFF) Channel index (16#8000) Alarm type = 3 6 (alarm) Alarm type = 3 Subslot <subslot idx>, pull alarm
3 Subslot index (0 – 16#9FFF) Channel index (16#8000) Alarm type = 4 6 (alarm) Alarm type = 4 Subslot <subslot idx>, plug alarm
3 Subslot index (0 – 16#9FFF) Channel index (16#8000) Alarm type = 5 6 (alarm) Alarm type = 5 Subslot <subslot idx>, status alarm
3 Subslot index (0 – 16#9FFF) Channel index (16#8000) Alarm type = 6 6 (alarm) Alarm type = 6 Subslot <subslot idx>, update alarm
3 Subslot index (0 – 16#9FFF) Channel index (16#8000) Alarm type = 7 6 (alarm) Alarm type = 7 Subslot <subslot idx>, redundancy status changed alarm
3 Subslot index (0 – 16#9FFF) Channel index (16#8000) Alarm type = 8 6 (alarm) Alarm type = 8 Subslot <subslot idx>, supervisor controlled alarm
3 Subslot index (0 – 16#9FFF) Channel index (16#8000) Alarm type = 9 6 (alarm) Alarm type = 9 Subslot <subslot idx>, supervisor released alarm
3 Subslot index (0 – 16#9FFF) Channel index (16#8000) Alarm type = 10 6 (alarm) Alarm type = 10 Subslot <subslot idx>, wrong submodule plugged alarm
3 Subslot index (0 – 16#9FFF) Channel index (16#8000) Alarm type = 11 6 (alarm) Alarm type = 11 Subslot <subslot idx>, wrong submodule returned alarm
3 Subslot index (0 – 16#9FFF) Channel index (0 – 16#7FFF) Alarm type = 12 6 (alarm) Alarm type = 12 Subslot <subslot idx>, channel <channel idx>, diagnosis disappeared alarm
3 Subslot index (0 – 16#9FFF) Channel index (16#8000) Alarm type = 12 6 (alarm) Alarm type = 12 Subslot <subslot idx>, diagnosis disappeared alarm
```
<table>
<thead>
<tr>
<th>Error severity</th>
<th>SubSystem info</th>
<th>Additional data</th>
<th>Error code</th>
<th>Meaning</th>
<th>Remedy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sub1_</td>
<td>Sub2_</td>
<td>Add_Word 1_Add_Word 2</td>
<td>Err_x or</td>
<td>Text</td>
<td>Entry in text list</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Err_Word 1_Word 2</td>
<td></td>
<td>Diag_PNIO_Controller and Diag_PNIO_Vendor ID_Device ID</td>
</tr>
<tr>
<td>3</td>
<td>Subslot index (0 – 16#9FFF)</td>
<td>Channel index (16#8000)</td>
<td>Alarm type = 13</td>
<td>Alarm type = 13</td>
<td>Subslot &lt;subslot idx&gt;, port data changed alarm</td>
</tr>
<tr>
<td>3</td>
<td>Subslot index (0 – 16#9FFF)</td>
<td>Channel index (16#8000)</td>
<td>Alarm type = 31</td>
<td>Alarm type = 31</td>
<td>Used module pulled alarm</td>
</tr>
</tbody>
</table>
Manual interpretation of CM579-PNIO diagnosis

For better understanding, we show the manual interpretation of CM579-PNIO diagnosis with an example.

System: AC500 CM579-PNIO + CI501-PNIO + optional S500 I/O inserted as PROFINET standard device

Error: discrepancy time expired (class 3, error ID 3) at channel 4 of first attached S500 I/O device on CI501-PNIO

For comparison: If a method with text output is used, e.g. DiagGetFirstValAndTxt the text output will be the string: Timestamp; E3; device name; subslot 1, channel 4, extended channel diagnosis; error ID 3: discrepancy time expired (class 3, error ID 3)

In Automation Builder the following error entry data is displayed:

Analyze the data in the following order: Element “dwAdditional” for the type of diagnosis, element “Error Code”, element “SubSysteminfo”.

Type of diagnosis

   Convert the given value from decimal to hexadecimal format.
   “dwAdditional” = 131076 = 16#20004

2. Interpretation:
   Word 2 = 2
   Word 1 = 4

3. Generate error text Add_Word 1_Word 2 = Add_4_2

4. Look up which error type it is.
   Add_4_2 = extended channel diagnosis

Data analysis

1. Analyze element “Error Code”:
   “dwErrorCode” = 196867 = 16#30103 →
   Word 2 (extended error type) = 16#0003
   Word 1 (error type) = 16#0103
   ➔ Error text: Err_Word 1 (hex)_Word 2 (hex)
   ➔ Err_0103_0003 →
   Error ID 3 – discrepancy time expired (class 3, error ID 3)

2. Analyze “SubSysteminfo”
   “dwSubSysteminfo” = 65540 = 16#10004 →
   Word 2 (subslot index) = 1
   Word 1 (channel index) = 4
### 1.7 Engineering interfaces and tools

#### 1.7.1 Export and import interfaces

**1.7.1.1 Exporting and importing ECAD data (PBF)**

Automation Builder provides an ECAD interface for exchanging the PLC configuration data with EPLAN Electric P8 and Zuken E3. This feature removes double data entry between electrical engineering in the ECAD tool and the control logic programming in Automation Builder by synchronizing the PLC hardware including topology and I/O signals between these tools.

Automation Builder - ECAD interface supports various flexible workflows:

- Enables PLC hardware planning and configuration in the ECAD tool and allows importing the exported data from the ECAD tool through the PBF file (process integration bus interchange format) into the Automation Builder project with diff and merge functionality, providing full control on selective import/merge.
- Enables PLC hardware configuration in Automation Builder and allows exporting the configuration to the ECAD tool through a PBF file.
- Supports bi-directional roundtrip engineering with loss less data exchange between Automation Builder and the ECAD tool.

Automation Builder uses the rack information to identify the relations between:

- PLC and devices plugged to IO bus or extension bus.
- Fieldbus slave and attached IO devices.

It is recommended to assign the PLC, IO devices, communication modules and fieldbus slaves properly to the rack in the ECAD project. If the rack information is missing, devices will be imported to the device pool and must be arranged manually in the Automation Builder project or mapped to already existing devices.
1.7.1.1 Requirements on EPLAN electric P8

- EPLAN Electric P8 with PLC and Bus Extension. It is recommended to use version 2.3 or later.
- Use of appropriate part data and macros for ABB devices. This can be achieved by getting the part data and macros from the EPLAN data portal.

1.7.1.2 Importing PLC data from the ECAD tool

You can create a new Automation Builder PLC project from the existing PLC hardware configuration in your ECAD tool, by importing the exported PBF file to Automation Builder.

1. From the main menu, select "Project ➔ Import ➔ ECAD (PBF)".
2. From the file system, select the PBF file.
   Automation Builder starts importing the devices and its associated signals from the PBF file. After a successful import, the result is displayed in the Project Compare – Differences view. You can now decide and selectively merge the differences. See 'Compare' Command in Chapter 1.3.1.25.2.4.18 "Command 'Compare'" on page 905
3. Select the DevicePool node and click “Accept Block” to accept the complete PLC structure in the ECAD tool.
4. Select the PLC node and click “Accept Block” to accept all child device nodes.
   The DevicePool node holds all devices coming from the ECAD tool without any hierarchy information. The missing hierarchy information can be defined after closing the editor.
5. Close the Project Compare – Differences view to accept the changes.

The I/O signals assigned to I/O devices in the PBF file are imported and allocated to IO devices. I/O signals can be viewed in I/O mapping editor of the I/O devices.
1.7.1.1.3 Importing third party devices

Prerequisite: To import third party devices from ECAD to Automation Builder, install third party fieldbus devices (for example, GSD, GSDML and EDS files) using “Tools ➔ Device repository” in Automation Builder.

1. From the main menu, select “Project ➔ Import ➔ ECAD (PBF)”.
2. From the file system, select the ECAD pbf file which consists of third party devices.

.existsSync(). When the device identifier of the third party device installed in Automation Builder does not match with the device identifier of the device imported from ECAD, an error window is shown with the devices which are failed to import with error ID 14.

To import third party devices, it is required to assign ECAD identifier (PLC type designation/order number) in Automation Builder in “Tools ➔ Device ECAD data”. Click the link in the Import window to see the error messages in a text file.

3. Click “Continue” in the Import window to import valid devices to the project that are imported successfully or click “Cancel” to cancel the import process.
4. In Automation Builder, click “Tools ➔ Device ECAD data”.
5. In the Device ECAD data editor, add the ECAD identifier for the devices shown in the import errors window with error ID 14, to enable these devices for export and import.

SuccessListener(). Also, add the ECAD identifiers for all devices which need to support export/import in ECAD.

6. Reimport the pbf file to import the third party devices.

1.7.1.1.4 Exporting PLC data to ECAD tool

1. Open the existing PLC project.
2. In the device tree, right-click “PLC ➔ Export ➔ ECAD (PBF)”.
3. Select the desired location in the file system to save the PBF file.

The ECAD user can import the exported PBF file from Automation Builder and can use the imported PLC data for electrical engineering purpose. If the user modifies imported PLC data in the ECAD project, the data can be imported back to the Automation Builder project which supports the round trip engineering efficiently with loss less synchronization of the data.
1.7.1.1.5 Exporting third party devices

1. Right-click on a PLC device, click “Export” and select “ECAD (PBF)”.
2. Save the file to the desired location in the file system.
   
   If the third party devices does not contain assigned ECAD identifiers, a message is displayed showing which devices cannot be exported.
   
   To add ECAD identifiers to the devices, see Importing third party devices Chapter 1.7.1.1.3 “Importing third party devices” on page 3589.
   
   After adding ECAD identifiers to the third party devices, execute “Export” to export the devices including third party devices.

1.7.1.1.6 Importing ECAD PLC data to existing AB project

Automation Builder ECAD interface supports concurrent engineering by importing the ECAD data to the existing Automation Builder PLC project.

1. From the main menu, select “Project ➔ Import ➔ ECAD (PBF)”.
2. Select the PBF file which has been created during the export from the ECAD tool.
3. Select the PLC from the list and click “OK”.
   
   A dialog window is displayed if the Automation Builder project provides PLCs of the identical type as defined in the PBF file.
   
   By selecting “None” in the dialog window a new PLC is defined in the ECAD tool.
4. In the Project Compare – Differences view, click [ ] to merge device signals.
   
   The differences between the current PLC hardware configuration in Automation Builder and the ECAD PLC data are displayed.
5. Select the differences as desired and click “Accept Single” to accept the selected difference block.
6. Close the Project Compare – Differences view to accept the changes.

1.7.1.1.7 Arrange or map devices imported to the device pool

Devices that are imported to the device pool because of missing hierarchy information (mainly rack information) must be arranged manually in the Automation Builder project or mapped to already existing devices.

Arrange devices imported to the device pool

Arrange the unassigned devices in the DevicePool to the PLC hardware structure by drag-and-drop.

Map devices imported to the device pool

If the devices are already added to the Automation Builder project prior to the import, you have to map the instances of the same type manually (one instance in the Automation Builder project tree and one instance in the DevicePool).

After mapping the devices, you can selectively merge the device parameter or signal in the difference view.

To map pool devices, proceed as follows:
1. In the device tree, select the Device Pool node, click “Project” and select “Map pool devices”.

2. Map the device pool instances of identical types in the project from the drop-down list and click “OK”.

   Pool devices which are mapped are removed from the device pool and mapped to the corresponding Automation Builder device. Differences between the signals of the mapped I/O devices are displayed. e.g. AI523_1 device:

1.7.1.8 Limitations

The following limitations are considered when working with the Automation Builder ECAD interface:

- The scope of a PBF file is limited to one single PLC including all connected devices.
- There is no representation of XC variants of devices in Automation Builder. Therefore, always use the standard variant for export. This might lead to part data mismatch when importing into the ECAD tool.
- In reimport or round trip import cases, if any changes are made in ECAD by adding a new communication module with connecting to one of the PLC slot or replacing existing communication module, then those device changes to the communication modules are not displayed as connected to PLC slots during the import in Automation Builder diff and import, instead those CM modules are added under the device pool. After merging and importing is completed, to work with device pool devices ⇒ Chapter 1.7.1.1.7 “Arrange or map devices imported to the device pool” on page 3590.
- IO mapping data cannot be imported for IO devices plugged to an EtherCAT slave when they are imported individually to the device pool because of missing hierarchy information. After arranging the devices properly in the device tree, the import can be done again to import also the IO mapping data.

1.7.1.2 Exporting and importing I/O mapping (CSV)

The I/O module mappings of an Automation Builder project can be exported to CSV for bulk editing in MS Excel or other documentation purposes. I/O mappings can be exported at single I/O module level or at PLC level.
Further, the I/O module mappings can be imported with the option of displaying differences and merging each single changed or import all signals at once by overwriting existing I/O module signals.

1.7.1.2.1 Exporting IO mapping data to CSV

To export I/O mappings to a CSV signal list, proceed as follows:

1. In the device tree, right-click “PLC ➔ Export ➔ IO mapping (CSV)”.
2. Save the **IO mappings CSV** to the desired location in the file system.

   If the CSV signal list has been exported successfully, a success message is displayed. The status of the export is shown in the dialog.

3. In the export dialog, click the link to open the exported **IO mapping CSV** file in MS Excel.

   ![Excel Template]

   *The template can only be opened if MS Excel is installed and configured to open .csv files.*

4. In the IO mapping (CSV) file, change **Variable** and **Description** fields to edit I/O mappings.

   ![Excel File]

   *Do not modify other field’s data in IO mapping (CSV) file.*
1.7.1.2 Importing I/O mapping data from CSV

To import an edited I/O mapping (CSV) file, proceed as follows:

1. From the main menu, select “Project ➔ Import ➔ I/O mapping (CSV) ➔ Open”.
2. A CSV signal list import dialog is displayed.
   - With “YES”, all I/O mappings will be imported without difference view. With “NO”, the difference view is displayed with the I/O mapping differences.
3. In the Project Compare – Differences view, click [ ] to merge I/O mappings.
4. Select the signal row for which the difference is to be accepted. Select the Variable field and click “Accept Single” to merge the I/O mappings.
5. Close the Project compare – Differences view to accept the changes and merge the I/O mappings with the Automation Builder project.

1.7.1.3 Exporting and importing device list (CSV)

The Automation Builder project devices can be exported to CSV for bulk device renaming or adding device tag labels to devices in MS Excel or other documentation purposes. A devices export is only possible at PLC level.

Automation Builder provides importing devices in bulk based on device type, instance and hierarchy information provided in the CSV file.

1.7.1.3.1 Exporting device list to CSV

To export a CSV device list, proceed as follows:

1. In the device tree, right-click “PLC ➔ Export ➔ Device list (CSV)”.
2. Select the desired location in the file system to save the Device list (CSV).
   - If the CSV device list is exported successfully, a success message is displayed.
3. In the Export dialog, click the link to open the exported CSV device list. The exported CSV device list consists of all devices connected to the PLC that is exported. Each row represents a device with its device type and hierarchy information.

1.7.1.3.2 Creating CSV device list

To create the devices in CSV, use the device list template provided in Automation Builder.

- In the main menu, click “Tools ➔ Create CSV Device list”.
- The device list template is opened in the MS Excel.

The template can only be opened if MS Excel is installed and configured to open .csv files.

In this file, add each device in a separate row with device information like Device Type (Order Num or Device Type Name) and instance details (name, tag) and hierarchy information (parent Device name, parent Device Tag, position). The mandatory information required to import CSV is only Device Type. All other fields are optional. After editing the device list CSV file, save it in the file system and close.
1.7.1.3.3 Importing a device list from CSV

To import devices from CSV in bulk, proceed as follows:

1. From the main menu, click “Project ➔ Import ➔ Device list (CSV)”.
2. Select the device list CSV file from the file system and click “Open” in the Import dialog. All devices that are defined in the CSV are imported. The Project Compare – Differences view displays the current project and the project that has been updated by the import file.
3. Select the desired devices and click “Accept Block” to accept all the devices and its child device nodes or “Accept Single” to accept only a single device.
4. After closing the Project Compare – Differences view, the devices are imported to the Automation Builder project.

⇒ The devices (except PLC) are placed under the device pool if the valid device hierarchy information is not provided in the CSV device list file. By drag-and-drop devices can be assigned to the desired PLC hardware structure ⇒ Chapter 1.7.1.1.7 “Arrange or map devices imported to the device pool” on page 3590.

If a device tag is provided for a device in CSV, it appears next to each device node in the device tree.

1.7.1.3.4 Renaming devices

To rename the devices, proceed as follows:

1. In the device tree, right-click “PLC ➔ Export ➔ Device list (CSV)”.
2. Select the desired location from the file system to save the CSV device list.
3. Rename the device names in the column **Device Name**:

4. Click "Project ➔ Import ➔ Device list (CSV)".

5. Select the updated CSV file from the file system.

Open the **Project Compare – Differences** view. If only the device names have been changed in the CSV file, the difference view does not show the changes.

6. Close the **Project Compare – Differences** view. The Renamed Devices dialog is displayed with the current name and the new name provided in the CSV file.

7. In the Rename Devices window, select the desired devices and click “OK”. The device names are updated in the Automation Builder project.
1.7.2 CODESYS Security Agent

1.7.2.1 Integration in CODESYS Development System

The CODESYS Security Agent is an add-on product for CODESYS. It simplifies the configuration and control of security-relevant aspects in the development system.

At this time, you can configure and create certificates of the controlling device with the CODESYS Security Agent. After installation, a new "Device" tab is added to the “Security-Screen” view in the development system. You can then configure encrypted communication with the controller, as well as encrypt the boot application, download, and online change.

A description of the standard security functions in CODESYS Development System are located in the standard CODESYS online help under the keywords "security", "encryption", and "security screen".

See also

● Chapter 1.7.2.2 “Encrypted communication with devices via controller certificates” on page 3597
● Chapter 1.7.2.3 “Encrypting boot applications with certificates” on page 3598

1.7.2.2 Encrypted communication with devices via controller certificates

Requirement: A digital signature for certificate exchange is configured. Refer to the standard CODESYS online help “Encryption and Signing with Certificates”.

We assume that the controller still has not certificate that is intended for encrypted communication. With the following steps, you generate this kind of certificate and encrypt communication:

1. Configure the active path to the controlling device.
2. Open the “Security-Screen” view by double-clicking the symbol in the status bar or by clicking “View ➔ Security-Screen”.
3. Click the button to update the list of available devices and their certificate store.
4. Select the corresponding device entry on the left side.
   ➤ On the right side, there is still no license entry for the “Encrypted Communication” use case.
5. Select “Encrypted Communication” on the right side and click the button to generate a new certificate on the device.
   Change the default key length to 4096. Otherwise an error occurs that is only visible in the log of the PLC.
   ➤ The certificate is generated and displayed in the table with its properties. The symbol before “Encrypted Communication” appears now as follows: 🛡.
6. In this step, you activate encrypted communication with the controller:

Open the “Security-Screen” view from CODESYS (“Users” tab). Activate the option “Force encrypted communication” (“Security level”).

➤ As of now, communication with all controllers is possible only as long as the certificate is valid on the controller and you have a key for it.

The connecting line between the development system, the gateway, and the controller is displayed in yellow in the “Communication” tab of the device editor of the controller.

As an alternative to the option “Force encrypted communication” that was just described and applies to all controllers, you can also encrypt communication with a specific controller only. To do this, open the “Communication” tab in the device editor of the controller. In the drop-down list “Device”, click “Encrypted communication”. 
7. Then log in again to the controller.
   A dialog opens, prompting that the certificate of the controller is not signed by a trusted source. In addition, the dialog displays information about the certificate and prompts whether to install it as a trustworthy certificate in the local store in the "Controller certificates" folder.

8. Confirm the dialog.
   The certificate is installed in the local store and you log in to the controller.
   In the future, the communication with the controller is encrypted automatically with this control certificate.

See also

- Chapter 1.7.2.4.1 “View ‘Security-Screen’ - ‘Devices’” on page 3599
- Chapter 1.7.2.1 “Integration in CODESYS Development System” on page 3597

1.7.2.3 Encrypting boot applications with certificates

Aim: You want to encrypt the boot application with a certificate from the controller in order to make sure that it cannot be exchanged. To do this, a corresponding certificate must be created on the controller and installed in the Windows Certificate Store of your computer.

Requirement: The active path to the controller is configured. A digital signature for certificate exchange is configured. Refer to the standard CODESYS online help “Encryption and Signing with Certificates”.

1. Open the “Security-Screen” view by double-clicking the symbol in the status bar or by clicking “View → Security-Screen”.
2. Click the button “Refresh the list of available devices and their certificate store”.
3. Select the device entry on the left side.
4. Select “Encrypted Application” on the right side and click the button “Generate a new certificate on the device”.
   The certificate is created and listed in the table with the symbol.
5. Change the default key length to 4096. Otherwise an error occurs that is only visible in the log of the PLC.
   Double-click the certificate entry.
   The default Windows “Certificate” dialog opens.
6. Click the “Install certificate” button in the “General” tab.
   The “Certificate import assistant” opens.
7. In the “Certificate memory” dialog, select the option “Save all certificates in the following memory location”, and then select the folder “Controller Certificates” as the “Certificate memory”.
   The controller certificate is imported to the directory “Controller certificates” and is now available for encryption of download, online change, and boot application.
8. Follow the steps below if you want the boot application of your project, as well as downloads and online changes, to always be encrypted.
9. Open the “Users” tab in the “Security-Screen”. Activate the option “Force encryption of downloads, online changes, and boot applications” in the “Security-Level” area.
10. Open the “Project” tab and double-click the application entry in the area “Encryption of boot application, download, and online change”.
    The “Properties” dialog of the application opens.
11. Select the “Encryption” tab and select “Encryption with certificates” as the “Encryption technology”. Then click [ ]. Note: If the option “Force encryption of downloads, online changes, and boot applications” is activated in the “Security screen”, then “Encryption with certificates” is already preselected.

12. In the “Certificate selection” dialog, select the corresponding certificate from the “Controller certificates” folder and click [ ].

13. Click “OK” to confirm the dialog.

→ The certificate is displayed in the properties dialog.

14. Confirm the “Properties” dialog of the application.

→ The certificate is displayed in the “Security screen” view (“Project ” tab, “Encryption of boot application, download, and online change”): The boot application, download, and online change are encrypted.

15. Now transferring the boot application, download, and online change are possible, as long as the certificate configured for it and the signature are valid.

See also
- ☞ Chapter 1.7.2.1 “Integration in CODESYS Development System” on page 3597

### 1.7.2.4 Reference User Interface

#### 1.7.2.4.1 View 'Security-Screen' - 'Devices'

**Symbol:** 🚀

**Function:** This tab allows for the configuration and the transfer of controller certificates for encrypted communication with the controller.

**Call:** Menu bar: “View”.

**Requirement:** The CODESYS Security Agent add-on product is installed.

The “Devices” tab shows all PLC devices configured in the project and their certificate store. If the communication path to the controller is configured, then you see the certificates stored in the memory. Here you can create and configure new certificates on the controller.

<table>
<thead>
<tr>
<th>Left side: “Information”</th>
<th>Devices and certificate store</th>
</tr>
</thead>
<tbody>
<tr>
<td>Shows the individual devices 🎨 as expandable nodes, each with the controller-specific 🛡️ certificate store below it.</td>
<td>For example, there are the following categories for the CODESYS Control Win V3:</td>
</tr>
<tr>
<td>For example, there are the following categories for the CODESYS Control Win V3:</td>
<td>• Individual certificates: Certificates with the associated, private key to which you have access.</td>
</tr>
<tr>
<td>• Trusted certificates: Certificates that have been created by a trusted certificate source.</td>
<td>• Untrustworthy certificates: Certificates that you have defined specifically as not trusted.</td>
</tr>
</tbody>
</table>
| • Certificates in quarantine: Certificates that do not fulfill the criteria of the categories above. | Toolbar (left side): 🔄: Refresh the display

Download: Transfer the selected certificate to the PLC
If the active path to the controller is set and a device node is selected, then every use case for controller certificates are displayed on the right side.

- **“OPC UA Server”:** Encrypted communication over an OPC UA server
- **“Encrypted Communication”:** Encrypted communication between the development system and the controller
- **“Encrypted Application”:** Encryption of the boot application
- **“Web Server”:** Encrypted communication with the web server

As long as a certificate is not available for one of these use cases, it is displayed with the icon & and as “(not available)”.

When a certificate store is selected on the left side, all certificates in it are displayed on the right side with the following information:

- **“Information”:** Use case (Currently the affected controller component is displayed: for example, “CmpSecureChannel”)
- **“Created for”:** Name of the computer for which the certificate was created (for example, “MyLocalPC”)
- **“Created by”:** Name of the computer on which the certificate was created (for example, “MyLocalPC”)
- **“Valid as of”:** Date (for example, “20-07-2017 15:09:29”)
- **“Valid until”:** Date (for example, “20-07-2022 00:00:00”)
- **“Thumbprint”:** Checksum from specific properties of the certificate for purposes of identification (for example, “279e1a46b86bd636c8e6f19fd51c222469ec49a8”) Double-clicking a certificate entry opens the default Windows “Certificate” dialog. In this way, you can import a controller certificate in the Windows Certificate Store to the directory “Controller certificates” so that it is available for encryption of download, online change, and boot application.

If multiple certificates are available for one use case, then the system follows the steps below to determine the certificate that is used:

- Certificate that was created directly by the user (currently not supported)
- Filtering of existing certificates by:
  - 1. Subject (user of the certificate)
  - 2. Key usage
  - 3. Extended key usage
  - 4. Valid time stamp
- Dividing of detected, valid certificates as "signed" and "self-signed"
- Filtering of signed certificates, and the self-signed certificates by the following criteria:
  - 1. Longest validity period
  - 2. Strongest key

**Drag&Drop:** Moving of the certificate to another certificate store of the same device

Double-clicking a certificate entry opens the default Windows dialog for displaying all certificate information.

**Toolbar (right side)**

- **Creation of a new certificate for a specific use case**
  The dialog “Certificate settings” opens for configuring the “Validity period” of the certificate and the “Key length” for the private key. “OK” saves the specified values in the CODESYS options. The values are set again at the next operation.

As long as the certificate is being created, ”(computing)” is shown after the use case. You cannot cancel the creation operation, but you can close and continue working with the “Security-Screen”.

- **Upload and save the selected certificate to the local file system.**
- **Delete the selected certificate.**
1.7.3 CODESYS Static Analysis

CODESYS Static Analysis already helps to write more readable code when programming in CODESYS and to detect contradictory or unsupported settings. In particular, potential sources of error can be identified, such as test code or pointers that have not been checked for 0 before dereferencing. With specific checks, you can make sure that the code is portable. Example: The analysis should report the use of language resources for object orientation because the code is to run on platforms that do not support object orientation.

The analysis checks the source code of the CODESYS project and reports any deviations from certain coding rules, naming conventions, or permitted keywords and identifiers. CODESYS Static Analysis is based on the rule set defined in the PLCopen Coding Guidelines and extends it with additional test options.

You can display the detected deviations as errors or warnings in the message view before the project is downloaded to the target system.

You activate static analysis either explicitly by clicking “Build ➤ Run static analysis”, or you let it execute automatically at each code generation. You activate the automatic execution in the “Static Analysis” dialog of the project settings. In this dialog, you also configure what is to be checked in detail. You can use pragma statements to exclude individual parts of the code from the check.

To evaluate the code quality, you can also display selected metrics that CODESYS Static Analysis detects in your code in a separate view. An example of this is the McCabe metric, which measures the cyclomatic complexity and indicates the number of execution paths that can be processed during code execution.

NOTICE!
The analysis is performed only for the code of the applications in the current project. Libraries are not taken into consideration.

The CODESYS development system contains a light version of static analysis that is extended by CODESYS Static Analysis.

See also
- § Chapter 1.7.3.2.2.1 “Dialog ‘Static Analysis’ - ‘Settings’” on page 3606
- § Chapter 1.7.3.2.1 “Menu Commands” on page 3605
- § Chapter 1.7.3.3.1 “Pragmas and Attributes” on page 3611

1.7.3.1 Configuring and Running Static Analysis

Using a basic sample project below, you will find the most important steps and options for configuring and running a static analysis.

Requirements: CODESYS Static Analysis is installed.

Sample project

If you want to reproduce the example project, create a standard project and insert the POU below the application in the device tree. Then configure the communication settings for the connection to your local CODESYS Control Win V3.

FUNCTION_BLOCK fb1
VAR_INPUT
    iVar_fb1in1 : INT;
    iVar_fb1in2 : INT;
    rVar_fb1in3 : REAL;
END_VAR
VAR_OUTPUT
    iVar_fb1out:INT;
Checking for compliance to rules

1. Click “Project ➔ Project Settings” and select the “Static Analysis” category. Click the “Rules” tab.
   ⇧ A list is displayed containing all possible rule checks. They are organized in a tree structure by topical category. The rule number is added in parentheses (for example, “Unused variables (33)” in category “Unused objects”).

2. Click the check box of the first line a few times (“Rules” node).
   ⇧ Clicking toggles the activation status. The check boxes in the entire tree have a red or orange check mark, or no check mark at all.

3. In this way, activate all entries with a red check mark. This means that CODESYS Static Analysis should report any detected rule violations as errors.

4. Click “Build ➔ Static analysis ➔ Run static analysis”.
   ⇧ Errors are reported in the message view. The message texts are tagged with a § and begin with the error number "SA<rule number>".

5. Double-click the first message SA0033: Unused variables 'iVar_fb2out'.
   ⇧ The focus moves to the declaration part of function block fb2 and the relevant variable is selected. The variable is declared, but not used. This is checked in Rule 33 (“Unused variables”). In the code, the relevant locations are underlined with a wavy line.

6. Now test how to perform static analysis automatically. Click “Project ➔ Project Settings”, “Static Analysis” category. On the “Settings” tab, select the “Perform static analysis automatically” option. Click “OK” to close the dialog.

7. Click “Online ➔ Login”.
   ⇧ A dialog prompt indicates that compile errors exist. The errors reported by the code analysis are displayed again in the message view.
8. Now deactivate all rules for static analysis in the project settings. In the "Unused objects" category, explicitly activate Rule SA0035 ("Unused input variables (35)") with an orange-colored check mark to report a warning. See the tooltip for the rule text: "This rule corresponds to the following PLCopen rules: CP24"). Click "OK" to close the project settings.

9. Click "Build ➔ Generate code".
   The analysis is performed automatically. Two errors are reported in the message view:
   $ SA0035: Unused input variable 'iVar_fb1in2' and $ SA0035: Unused input variable 'iVar_fb1in3'.

10. Double-click the message and comment or remove the declaration. Perform the code analysis again.
   No error messages are displayed.

See also
- § Chapter 1.7.3.2.2.1 “Dialog 'Static Analysis' - 'Settings'” on page 3606
- § Chapter 1.7.3.2.2.2 “Dialog 'Static Analysis' - 'Rules'” on page 3607

Checking for compliance to defined naming conventions

1. Click “Project ➔ Project Settings” and select the “Static Analysis” category. Click the “Naming Conventions” tab.
   You see a table in a tree structure that is divided into expandable categories of variables and program blocks.

2. Expand the “Prefixes for variables” - “Prefixes for types” category: In the “Prefix” column, specify I for “INT (14)”.
   Expand the “Prefixes for POUss” - “Prefixes for POU type” category: In the “Prefix” column, specify the prog for “PROGRAM (122)” and fb for “FUNCTIONBLOCK (103)”. 

3. Click the “Naming conventions (2)” tab. Select the “First character after prefix should be an upper case letter” option. Clear all other options.

4. Click “Build ➔ Static analysis ➔ Run static analysis”.
   Error messages:
   - NC0102: Invalid name 'PLC_PRG': Expect prefix 'prog' because PLC_PRG does not have the required prefix
   - First character after prefix should be uppercase: 'ivar_fb1in2' because iVar_fb1in2 : INT; in fb1
   - NC0014: Invalid variable name P_fSampleProperty: Expect prefix 'i' because this integer variable does not have the required prefix

See also
- § Chapter 1.7.3.2.2.3 “Dialog 'Static Analysis' - 'Naming Conventions'” on page 3608
- § Chapter 1.7.3.2.2.6 “Dialog 'Static Analysis' - 'Naming conventions (2)'” on page 3610
Checking for forbidden symbols

1. Click “Project ➔ Project Settings” and select the “Static Analysis” category. Click the “Forbidden symbols” tab.
   - A line editor allows for specifying character strings that should not to be used in the code.
2. As an example, double-click the blank line and type in the invalid character string PRO directly. Double-click the next blank line and click to open the input assistance. From “Standard Types”, select “REAL”. Click “OK” to close the dialog.
3. Click “Build ➔ Static analysis ➔ Run static analysis”.
   - The error message Forbidden symbol 'REAL' and Forbidden symbol 'PRO' are displayed in the message view. Double-click the message text to jump to the relevant line of code.

See also
- Chapter 1.7.3.2.2.5 “Dialog ‘Static Analysis’ - ‘Forbidden symbols’” on page 3609

Displaying metrics

CODESYS Static Analysis performs selected tests on the code, and you can display the results in a view.

1. Click “Project ➔ Project Settings” and select the “Static Analysis” category. Click the “Metrics” tab.
   - The metrics that CODESYS Static Analysis applies to the code are listed in a table.
2. For this example, activate the “Number of inputs variables” metric and specify the permitted value range: lower limit 1 and upper limit 2. Activate some more metrics, for example “Code size” and “Number of calls”.
3. Click “Build ➔ Static analysis ➔ View standard metrics”.
   - The view includes a table with a line for each “Program unit” of the sample program. For each activated metric, there is a column showing the measured values. Values that are outside of the value range defined in the settings are highlighted in red. In the case of this specific example, this is at least the “PLC_PRG/Inputs” field because the number of input variables in this POU is greater than the defined upper limit of 2.

See also
- Chapter 1.7.3.2.2.4 “Dialog ‘Static Analysis’ - ‘Metrics’” on page 3609
- Chapter 1.7.3.2.1 “Menu Commands” on page 3605

1.7.3.2 Reference, User Interface

1.7.3.2.1 Menu Commands......................................................................................... 3605
1.7.3.2.2 Dialogs........................................................................................................ 3606
1.7.3.2.1 Menu Commands

1.7.3.2.1.1 Command 'Run Static Analysis'.................................................... 3605
1.7.3.2.1.2 Command 'View Standard-Metrics'.............................................. 3605

Command 'Run Static Analysis'

Symbol: §

Function: The command starts the static analysis for the active application and displays the results in the message view.

Call: Menu bar: “Build ➔ Static Analysis”

During the code analysis, CODESYS generates code just like the “Build ➔ Generate Code” command. The results of the analysis are displayed as errors § and warnings § in the message view (“Build” category). The numbers refer to the corresponding rules as they are defined in the project settings. The syntax for the displayed messages is “SA<rule number:<rule text>”.

See also
● Chapter 1.7.3.2.2 “Rules” on page 3616

Command 'View Standard-Metrics'

Symbol: 

Function: The command starts the static analysis for the active application and displays the metrics for all POUs in a table.

Call: Menu bar: “Build ➔ View Standard Metrics”

The metrics (code numbers) to be displayed are activated in the project settings. You can access the configuration by clicking “Configure” in the context menu of the displayed table. If a value is outside of the configured upper and lower limits, then the field in the table is highlighted in red.

See also
● Chapter 1.7.3.2.2.4 “Dialog 'Static Analysis’ - 'Metrics’” on page 3609

Commands in the context menu

The following commands are provided in the context menu of the table:

● “Calculate”: The values are refreshed.
● “Copy Table”: The table is copied to the clipboard. The separators are tabs.
● “Print Table”: The default dialog for setting up a print job opens.
● “Export Table”: The table is exported as a CSV file. The separators are semicolons.
● “Kiviat Diagram”: Requirement: At least three metrics have defined upper and lower limits. A radar chart is created for the selected POU. This visualizes the quality of POU code with respect to a given standard.

Each metric is depicted as an axis with its origin at the center (value 0) which radiates outward into three concentric ring zones. The inner ring zone represents the value range below the lower limit defined for the metric. The outer ring represents the value range above the upper limit. The axes of the metrics are distributed uniformly around the circle.

The current values of the individual metrics on the axes are connected by a line. In the ideal case, the complete line is located in the middle zone.

● “Configure”: The table for selecting the desired metrics opens. This corresponds to the table in the project settings.
● “Open POU”: The POU opens in the editor.
The name of the metric is displayed at the end of the respective axis and the name of the POU is displayed in the upper right corner of the diagram.

See also
- Chapter 1.7.3.2.2.4 “Dialog 'Static Analysis' - 'Metrics'” on page 3609

### 1.7.3.2 Dialogs

| 1.7.3.2.2.1 | Dialog 'Static Analysis' - 'Settings' | 3606 |
| 1.7.3.2.2.2 | Dialog 'Static Analysis' - 'Rules' | 3607 |
| 1.7.3.2.2.3 | Dialog 'Static Analysis' - 'Naming Conventions' | 3608 |
| 1.7.3.2.2.4 | Dialog 'Static Analysis' - 'Metrics' | 3609 |
| 1.7.3.2.2.5 | Dialog 'Static Analysis' - 'Forbidden symbols' | 3609 |
| 1.7.3.2.2.6 | Dialog 'Static Analysis' - 'Naming conventions (2)' | 3610 |

The dialogs for the configuration of static code analysis are located in "Project -> Project Settings", in the "Static Analysis" category. The requirement is that a CODESYS project has to be open.

### Dialog 'Static Analysis' - 'Settings'

**Function:** In the dialog, you select automatic static analysis, and save or load the project settings for static analysis as a CSA file.

**Call:** Menu bar: “Project ➔ Project Settings”, “Static Analysis” category, “Settings” tab

**Requirement:**
- The CODESYS Static Analysis package is installed.
- A project is open.
Table 784

<table>
<thead>
<tr>
<th>“Perform static analysis automatically”</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>☑: CODESYS performs the code check automatically at each code generation (for example, when the “Build ➔ Generate code” command is executed or before a download.</td>
<td>☐: The code check is not performed automatically, but it can be performed explicitly by means of the “Build ➔ Static analysis ➔ Run static analysis” command.</td>
<td></td>
</tr>
</tbody>
</table>

“Load”

Opens the “Load Static Analysis Configuration” dialog for selecting the project settings for the static analysis as a CSA file in the file system. When you click the “Open” button, the selected CSA file is loaded.

“Save”

Open the “Save Static Analysis Configuration” dialog for saving all project settings in the “Static Analysis” category as a CSA file in the file system.

See also

- ☇ Chapter 1.7.3.1 “Configuring and Running Static Analysis” on page 3601

Dialog ‘Static Analysis’ - ‘Rules’

**Function:** In the dialog, you select the rules that are checked during the static analysis of the source code of a project.

**Call:** Menu bar: “Project ➔ Project Settings”, “Static Analysis” category, “Rules” tab

**Requirement:** A project is open.

This tab shows a tree structure of all rules that can be checked during static analysis. By default, every rule is activated, with the exception of SA0016, SA0024, SA0073, SA0101, SA0105, SA0106, SA0133, SA0134, SA0150, SA0162, and all strict IEC rules.

Each rule has a unique number. When the rule is checked and a violation is detected, the rule number and an error description are shown in the message view in the “Build” category in the following format: SA<rule number>, where SA stands for ”Static Analysis” (example: "SA003" for rule 3).

- The list of available rules can be extended by specific plug-ins.

Table 785

<table>
<thead>
<tr>
<th>Some rules that are activated in the dialog can be deactivated temporarily in the application by applying a pragma.</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>When you click the check box, the setting toggles between ☐, ☑, and ☑</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

When you activate or deactivate a parent node, all child rules are also activated or deactivated, respectively.

- The rule is not checked.

- If the result of the check is positive, then an error ( § ) for the static analysis is displayed in the message view.

- If the result of the check is positive, then a warning ( § ) for the static analysis is displayed in the message view.

See also

- ☇ Chapter 1.7.3.2.2 “Rules” on page 3616
- ☇ Chapter 1.7.3.3.1 “Pragmas and Attributes” on page 3611
- ☇ “Checking for compliance to rules” on page 3602
Dialog 'Static Analysis' - 'Naming Conventions'

**Function:** In the dialog, you define the prefixes for the data types and scopes of variables, as well as prefixes for POUs and user-defined data types (DUTs). Static analysis checks compliance with the naming conventions. When a convention is not observed, the static analysis reports an error message in the “Messages” view.

**Call:** Menu bar: “Project ➔ Project Settings”, “Static Analysis” category, “Naming Conventions” tab

**Requirement:**
- The CODESYS Static Analysis package is installed.
- A project is open.

The error messages are displayed in the following format: § NC<prefix-convention number><message text>. NC stands for "Naming Convention".

For example, the error message "§NC0102: Invalid name..." means a violation of naming convention 102 for POUs of type PROGRAM.

| **Names** | Nodes and elements for which a prefix can be defined.  
The number in parentheses after each element (for example, “PROGRAM (102)”) is the prefix convention number that is reported in the case of noncompliance with a naming convention. |
| **Prefix** | Input field of the prefix  
Multiple prefixes can be specified by means of comma separation.  
Examples of prefixes: prog, func, fu  
Regular expressions (RegEx) are also possible for prefixes. To do this, an @ must be prepended.  
Example: @x[a-zA-Z]: The name must begin with x and may contain one character from the scope a-zA-Z. |
| **Prefixes for variables** | Organizational node for all data types and scopes of variables for which a prefix can be defined |
| **Prefixes for POUs** | Organizational node for all POU types and method scopes for which a prefix can be defined |
| **Prefixes for DUTs** | Organizational node for all DUT data types for which a prefix can be defined |
For variables of type “Alias” and POUs of type “Property”, \{\text{datatype}\} can be defined as a prefix. Then the code analysis reports errors for all variables that do not have the prefix \{\text{datatype}\} for the data type of the alias or property. In the definition, \{\text{datatype}\} can also be combined with other prefixes.

See also:
● “Checking for compliance to defined naming conventions” on page 3603
● Chapter 1.7.3.2.2.6 “Dialog ‘Static Analysis’ - ‘Naming conventions (2)’” on page 3610
● Chapter 1.7.3.3.1 “Pragmas and Attributes” on page 3611

Dialog ‘Static Analysis’ - ‘Metrics’

Function: In the dialog, you select the metrics to be displayed for each POU in the “Standard metrics” view by means of the “Build ➔ Static Analysis ➔ View standard-metrics” command.

Call: Menu bar: “Project ➔ Project Settings”, “Static Analysis” category, “Metrics” tab

Requirement:
● The CODESYS Static Analysis package is installed.
● A project is open.

The “Code size”, “Variable size”, “Stack size”, and “Calls” metrics are reported only for POUs from libraries which are integrated in the project.

Violations of the upper and lower limits of the activated metrics can be reported as build errors by means of static analysis rule SA0150.

<table>
<thead>
<tr>
<th>“Metrics”</th>
<th>All selectable metrics are displayed in the column.</th>
</tr>
</thead>
</table>
| “Active”  | ☑: The metric is displayed for each POU in the “Standard metrics” view following the “Build ➔ Static Analysis ➔ View standard metrics” command.  
☐: The metric is not displayed in the “Standard metrics” view following the “Build ➔ Static Analysis ➔ View standard metrics” command. |
| “Lower limit” | Lower value from which the “Metric” is displayed |
| “Upper Limit” | Upper value to which the “Metric” is displayed |

See also:
● “Displaying metrics” on page 3604
● Chapter 1.7.3.3.2.52 “SA0150: Violations of lower or upper limits or the metrics” on page 3682

Dialog ‘Static Analysis’ - ‘Forbidden symbols’

Function: In the dialog, you define the keywords and symbols that must not be used in the project code.

Call: Menu bar: “Project ➔ Project Settings”, “Static Analysis” category, “Forbidden symbols” tab
Requirement:
- The CODESYS Static Analysis package is installed.
- A project is open.

<table>
<thead>
<tr>
<th>Input line</th>
<th>Double-clicking the line opens the line editor for specifying a keyword or symbol.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>: The Input Assistant opens for selecting the keyword or symbol.</td>
</tr>
</tbody>
</table>

See also
- “Checking for forbidden symbols” on page 3604

Dialog 'Static Analysis' - 'Naming conventions (2)'

Function: In the dialog, you define the check options for using prefixes. The dialog is an extension of the “Naming Conventions” tab.

Call: Menu bar: “Project ➔ Project Settings”, “Static Analysis” category, “Naming conventions (2)” tab

Requirement:
- The CODESYS Static Analysis package is installed.
- A project is open.

<table>
<thead>
<tr>
<th>“First character after prefix should be an upper case letter”</th>
<th>Static analysis reports an error for a variable when the first character of the variable name after the defined prefix is not an uppercase letter.</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Recursive prefixes for combinable data types”</td>
<td>Variables of combined data types (for example, POINTER OF POINTER TO INT or REFERENCE TO ARRAY) necessarily have to be combined prefixes that correspond to the naming conventions defined on the “Naming Conventions” tab.</td>
</tr>
<tr>
<td></td>
<td>Example: On the “Naming Conventions” tab, the prefix p was defined for the data type POINTER and the prefix i was defined for the data type INT. Static analysis reports errors for all variables of type POINTER OF POINTER TO INT which do not have the prefix ppi.</td>
</tr>
<tr>
<td>“Combine scope prefix with data type prefix”</td>
<td>As its namespace, a variable must have the prefix defined on the “Naming Conventions” tab followed by the defined prefix for its data type.</td>
</tr>
<tr>
<td></td>
<td>Example 1: On the “Naming Conventions” tab, the following prefixes were defined: g for “VAR_GLOBAL” and r for data type “REAL”. The code analysis reports errors for global REAL variables that do not have the prefix gr_.</td>
</tr>
<tr>
<td></td>
<td>Example 2: In the application, there exists an alias s_StringAlias and a variable sMyAlias.</td>
</tr>
<tr>
<td></td>
<td>TYPE s_StringAlias : STRING;</td>
</tr>
<tr>
<td></td>
<td>END_TYPE</td>
</tr>
<tr>
<td></td>
<td>VAR</td>
</tr>
<tr>
<td></td>
<td>sMyAlias : s_StringAlias;</td>
</tr>
<tr>
<td></td>
<td>END_VAR</td>
</tr>
<tr>
<td></td>
<td>On the “Naming Conventions” tab, the prefix al_{datatype} is defined for the “Alias” type. The prefix s is defined for the “STRING” type.</td>
</tr>
<tr>
<td></td>
<td>The code analysis reports the error NNC0033: Invalid variable name 'sMyAlias'. Expect prefix 'al_s'.</td>
</tr>
</tbody>
</table>

See also
- Chapter 1.7.3.2.2.3 “Dialog 'Static Analysis' - 'Naming Conventions’” on page 3608
- “Checking for compliance to defined naming conventions” on page 3603
1.7.3.3 Reference, Programming

1.7.3.3.1 Pragmas and Attributes

CODESYS Static Analysis provides pragmas for activating or deactivating individual rules or naming conventions for static code analysis.

Requirement: The rules or conventions are activated or defined in the project settings.

Rules that are deactivated in the project settings cannot be activated by means of pragmas or attributes.

Rule SA0004 cannot be deactivated by means of a pragma.

See also

- Chapter 1.7.3 “CODESYS Static Analysis” on page 3601

Pragma ‘analysis’

This pragma is used to deactivate the code rules for individual code lines of a POU. You deactivate code rules by specifying the rule numbers with a prepended minus sign (“-”). A prepended plus sign (“+”) activates the rule. You can specify any number of rules in the pragma.

Insert location: Deactivation: With \{analysis - ...\} before the first code line where the code analysis is deactivated. Activation: With \{analysis + ...\} after the last line of the deactivation.

Syntax:

Deactivation of rules:

\{analysis -<rule number> ( , -<additional rule number> )* \}  
* : optional none, one or more additional rule numbers

Activation of rules:

\{analysis +<rule number> ( , +<additional rule number> )* \}  
* : none, one or more additional rule numbers

Example

Rule 24 is deactivated for two lines and then reactivated. As a result, rule 24 is not checked in these lines so that nTest:=DINT#99 is allowed for example.

\{analysis -24\}
nTest := 99;
iVar := INT#2;
{analysis +24}

Deactivating multiple rules:

\{analysis -10, -24, -18\}

See also

- Chapter 1.7.3.1 “Configuring and Running Static Analysis” on page 3601
Attribute 'analysis'

The pragma deactivates specific rules for an entire programming object. You deactivate the code rules by specifying the rule numbers with a prepended minus sign ("-".) You can specify any number of rules in the pragma.

**Insert location:** In the declaration part of a POU, in the first line.

**Syntax:**

```
{attribute 'analysis' := '-<rule number> ( , -<additional rule number> )* '}
```

* : none, one or more additional rule numbers

**Example**

Rules 33 and 31 are deactivated for the entire structure:

```plaintext
{attribute 'analysis' := '-33, -31'}
```

```
TYPE My_Structure :
STRUCT
  iLocal : INT;
  uiLocal : UINT;
  udiLocal : UDINT;
END_STRUCT
END_TYPE
```

Rule 100 is deactivated for the array:

```plaintext
{attribute 'analysis' := '-100'}
```

```
PROGRAM PLC_PGR
VAR
  aBigData: ARRAY[1..10000] OF DWORD;
  aBigDATA_2: ARRAY[1..10000] OF DWORD;
END_VAR
```

See also
- Chapter 1.7.3.1 "Configuring and Running Static Analysis" on page 3601

Attribute 'naming'

The pragma marks the code lines that are excluded from the analysis of naming convention. An off is assigned to the pragma attribute before the first code line where the code analysis is deactivated. An on is assigned after the last line. When an omit is assigned, only the next code line is ignored.

**Insert location:** Deactivation: In the declaration part of POUs and DUTs, above the affected lines. Activation: Below the affected lines.

**Syntax:**

```
{attribute 'naming' := '<switch state>'}
```

*switch state*: on | off | omit

- on : naming is switched on
- off : naming is switched off
- omit : only next code line is switched off
Defined naming conventions: 1) INT variable names must be prepended with "int" as the identifier prefix, for example "intVar1". 2) Program names must begin with "prog".

For the code presented below, the static analysis issues messages only for the following variables: cccVar, aVariable, and bVariable.

```plaintext
VAR
{attribute 'naming' := 'off'}
iVarA : INT;
iVarB : INT;
{attribute 'naming' := 'on'}
iVarC : INT;
END_VAR

VAR
...
{attribute 'naming' := 'omit'}
iVarC : INT;
...
END_VAR

{attribute 'naming' := 'omit'}
PROGRAM PLC_PRG
VAR
...
END_VAR

{attribute 'naming' := 'omit'}
PROGRAM DoSomethingA
VAR
{attribute 'naming' := 'on'}
iVarA : INT;
iVarB : INT;
...
VAR_END
```

See also
- [Chapter 1.7.3.1 “Configuring and Running Static Analysis” on page 3601](#)

**Attribute 'nameprefix'**

The attribute defines a prefix for variables of a structured data type. The prefix must be prepended to the identifier of variables that are declared by this type.

**Insert location:** In the line before the declaration of a structured data type

**Syntax:**

```plaintext
{attribute 'nameprefix' := '<prefix>'}
```
Example

In the following example, Static Analysis issues a message for pB because the variable name does not begin with "point".

```plaintext
{attribute 'nameprefix' := 'point'}
TYPE DATAPoint :
  STRUCT
    iX: INT;
    iY: INT;
  END_STRUCT
END_TYPE

PROGRAM PLC_PRG
VAR
  pointA : DATAPoint;
  pB : DATAPoint;
END_VAR
pointA.iX := 1;
pointA.iY := 10;
pB.iX := 2;
pB.iY := 20;

Error message after static analysis: "Invalid variable name 'pB'. Expect prefix 'point'"
```

See also

- Chapter 1.7.3.1 "Configuring and Running Static Analysis" on page 3601

Attribute 'analysis:report-multiple-instance-calls'

The pragma marks a function block for checking for rule 105: Only function blocks with this pragma are checked whether the function block instances are called more than one time. If rule 105 is deactivated in the project settings, then the pragma does not have any effect.

Insert location: First line in the declaration part of a function block.

Syntax:

```plaintext
{attribute 'analysis:report-multiple-instance-calls'}
```
FUNCTION_BLOCK FB_DoA
VAR_INPUT
END_VAR
VAR_OUTPUT
END_VAR
VAR
  iA : INT;
END_VAR
  iA := iA + 1;
FUNCTION_BLOCK FB_DoB
VAR_INPUT
END_VAR
VAR_OUTPUT
END_VAR
VAR
  iB : INT;
END_VAR
  iB := iB + 1;
PROGRAM PLC_PRG
VAR
  fbA : FB_DoA;
  fbB : FB_DoB;
ND_VAR
  fbA();
  fbB();  // SA0105
  fbA();
  fbB();  // SA0105
--> SA0105: Instance 'fbB' called more than once

See also
- % Chapter 1.7.3.2.46 “SA0105: Multiple instance calls” on page 3668
- % Chapter 1.7.3.1 “Configuring and Running Static Analysis” on page 3601
1.7.3.3.2 Rules

1.7.3.3.2.1 SA0001: Unreachable code................................................................. 3617
1.7.3.3.2.2 SA0002: Empty objects........................................................................ 3617
1.7.3.3.2.3 SA0003: Empty statements................................................................. 3618
1.7.3.3.2.4 SA0004: Multiple write access on output......................................... 3618
1.7.3.3.2.5 SA0006: Write access from several tasks......................................... 3619
1.7.3.3.2.6 SA0007: Address operator on constants........................................... 3620
1.7.3.3.2.7 SA0008: Check subrange types......................................................... 3620
1.7.3.3.2.8 SA0009: Unused return values............................................................ 3621
1.7.3.3.2.9 SA0010: Arrays with only one component.......................................... 3622
1.7.3.3.2.10 SA0011: Useless declarations........................................................... 3622
1.7.3.3.2.11 SA0012: Variable which could be declared as constants.................. 3623
1.7.3.3.2.12 SA0013: Declarations with the same variable name.......................... 3623
1.7.3.3.2.13 SA0014: Assignment of instances.................................................... 3624
1.7.3.3.2.14 SA0015: Access to global data via FB_Init....................................... 3625
1.7.3.3.2.15 SA0016: Gaps in structures............................................................... 3625
1.7.3.3.2.16 SA0017: Non-regular assignments..................................................... 3626
1.7.3.3.2.17 SA0018: Unusual bit access.............................................................. 3626
1.7.3.3.2.18 SA0020: Possibly assignment of truncated value to REAL variable..... 3627
1.7.3.3.2.19 SA0021: Transporting the address of a temporary variable.................. 3628
1.7.3.3.2.20 SA0022: (Possibly) unassigned return value....................................... 3628
1.7.3.3.2.21 SA0023: Too big return values........................................................... 3629
1.7.3.3.2.22 SA0024: Untyped literals / constants............................................... 3629
1.7.3.3.2.23 SA0025: Unqualified enumeration constants...................................... 3630
1.7.3.3.2.24 SA0026: Possible truncated strings.................................................. 3630
1.7.3.3.2.25 SA0027: Multiple uses of identifiers............................................... 3631
1.7.3.3.2.26 SA0028: Overlapping memory areas.................................................. 3631
1.7.3.3.2.27 SA0029: Notation in code different to declaration............................. 3632
1.7.3.3.2.28 Unused Objects................................................................................... 3632
1.7.3.3.2.29 SA0034: Enumerations with incorrect assignment............................ 3635
1.7.3.3.2.30 SA0037: Write access to input variable............................................. 3636
1.7.3.3.2.31 SA0038: Read access to output variable............................................. 3636
1.7.3.3.2.32 SA0040: Possible division by zero..................................................... 3637
1.7.3.3.2.33 SA0041: Detect possible loop invariant code.................................... 3638
1.7.3.3.2.34 SA0042: Usage of different access paths.......................................... 3639
1.7.3.3.2.35 SA0043: Use of a global variable in only one POU............................. 3639
1.7.3.3.2.36 SA0044: Declarations with reference to interface............................... 3640
1.7.3.3.2.37 Conversions....................................................................................... 3641
1.7.3.3.2.38 Use of Direct Addresses..................................................................... 3646
1.7.3.3.2.39 Rules for Operators............................................................................. 3648
1.7.3.3.2.40 Rules for Statements.......................................................................... 3658
1.7.3.3.2.41 SA0095: Assignments in conditions..................................................... 3664
1.7.3.3.2.42 SA0100: Variables greater than <n> bytes.......................................... 3665
1.7.3.3.2.43 SA0101: Names with invalid length.................................................. 3666
1.7.3.3.2.44 SA0102: Access to program/fb variables from the outside.................. 3666
1.7.3.3.2.45 SA0103: Concurrent access on not atomic data.................................. 3667
1.7.3.3.2.46 SA0105: Multiple instance calls........................................................ 3668
1.7.3.3.2.47 SA0106: Virtual method calls in FB_INIT......................................... 3669
1.7.3.3.2.48 SA0107: Missing formal parameters................................................. 3670
1.7.3.3.2.49 Checking Strict IEC Rules................................................................. 3671
1.7.3.2.50 SA0140: Statements commented out......................................... 3679
1.7.3.2.51 Possible Use of Uninitialized Variables..................................... 3679
1.7.3.2.52 SA0150: Violations of lower or upper limits or the metrics............ 3682
1.7.3.2.53 SA0160: Recursive calls.......................................................... 3682
1.7.3.2.54 SA0161: Unpacked structure in packed structure...................... 3682
1.7.3.2.55 SA0162: Missing comments..................................................... 3684
1.7.3.2.56 SA0163: Nested comments..................................................... 3685
1.7.3.2.57 SA0164: Multiline comments................................................. 3685
1.7.3.2.58 SA0165: Tasks calling other POUs than programs..................... 3686
1.7.3.2.59 SA0166: Max. number of input/output/in-out variables............. 3687
1.7.3.2.60 SA0167: Temporary function block instances........................... 3687

SA0001: Unreachable code

Detects lines of code that are not executed, for example due to a RETURN or CONTINUE statement

Justification: Unreachable code should always be avoided. The test often indicates that test code still exists which should be removed.

Importance: High

PLCopen rule: CP2

Example

PROGRAM PLC_PRG
VAR
  xReturn_Before_End: BOOL;
  xContinue_In_Loop_FUN: BOOL;
  iCounter: INT;
END_VAR

xContinue_In_Loop_FUN := FALSE;
FOR iCounter := INT#0 TO INT#5 BY INT#1 DO
  CONTINUE;
  xContinue_In_Loop_FUN := FALSE;
END_FOR

--> SA0001: Unreachable code detected in 'PLC_PRG'

See also

● ⇐ Chapter 1.7.3.1 “Configuring and Running Static Analysis” on page 3601

SA0002: Empty objects

Detects POUs, GVLs, data type declarations, or interfaces that do not contain any code

Justification: Empty objects should be avoided. They are often a sign that an object has not been implemented completely. Exception: In some cases, no code is specified in the body of a function block when it should be used by interfaces only. In other cases, a method is created only because it is required by an interface without a sensible implementation being possible for the method. No matter the case, this kind of situation should be commented.

Importance: Medium

See also

● ⇐ Chapter 1.7.3.1 “Configuring and Running Static Analysis” on page 3601
SA0003: Empty statements

Detects lines of code that have a semicolon (;) but not a statement

Justification: An empty statement can be a sign for missing code. Exceptions: There are good reasons for using empty statements. For example, in a CASE statement it could make sense to explicitly program out all cases, even those where there is nothing to do. This would simply be to document that the case was not forgotten.

Importance: Low

Examples

```plaintext
CASE value OF
1: DoSomething();
2:; // nothing to do
3: DoSomethingElse();
END_CASE
```

See also

- § Chapter 1.7.3.1 “Configuring and Running Static Analysis” on page 3601

SA0004: Multiple write access on output

Detects outputs that are written to more than one location

Justification: The maintainability is degraded when an output is written in different locations in the code. Then it is uncertain which write access is the one that actually has an effect in the process. Good practice is to calculate the output variables in auxiliary variables and assign the calculated value at one location at the end of the cycle.

Importance: High

PLCopen rule: CP12

An error is not issued when an output variable (VAR_IN_OUT) is written in different branches of IF and CASE statements.

A pragma cannot deactivate this rule.
Example

VAR_GLOBAL
  g_xVar AT %QX0.0 : BOOL ;
  g_iTest AT %QW0 : INT ;
END_VAR

PROGRAM PLC_PRG
  IF g_iCondition < INT#0 THEN
    g_xVar := TRUE;
    g_iTest := INT#12;
  END_IF
  CASE g_iCondition OF
    INT#1:
      g_xVar := FALSE;
    INT#2:
      g_iTest := INT#11;
    ELSE
      g_xVar := TRUE;
      g_iTest := INT#9;
  END_CASE
  --> SA0004: Multiple write access on output '%QX0.0'
  --> SA0004: Multiple write access on output '%QW0'

See also

● § Chapter 1.7.3.1 “Configuring and Running Static Analysis” on page 3601

SA0006: Write access from several tasks

Detects variables that are written by more than one task

Justification: A variable that is written in multiple tasks may change its value unexpectedly. This can lead to confusing situations. String variables (and on some 32-bit systems also 64-bit integer variables) can even reach an inconsistent state if the variable is written to two tasks simultaneously.

Exception: In specific cases, it may be necessary for several tasks to write a variable. For example, use semaphores to make sure that access does not lead to an inconsistent state.

Importance: High

PLCopen rule: CP10

Example

VAR_GLOBAL
  g_iTemp1: INT;
END_VAR

PROGRAM PLC_PRG    // Controlled by MainTask
  g_iTemp1 := g_iTemp1 + INT#2;
END_PROGRAM

PROGRAM PLC_PRG_1  // Controlled by SubTask
  g_iTemp1 := g_iTemp1 - INT#3;
END_PROGRAM

  --> SA0006: Concurrent write access to 'g_iTemp1' in Tasks MainTask, SubTask

See also

● § Chapter 1.7.3.1 “Configuring and Running Static Analysis” on page 3601
SA0007: Address operator on constants

Detects lines of code where the operator ADR is applied for a constant

Justification: Using a pointer to a constant variables overrides the CONSTANT property of the variable. The variable can be changed by means of the pointer without any notification from the compiler.

Exception: In rare cases, it might be useful to pass a pointer to a constant to a function. However, you have to make sure that this function does not change the transferred value. Whenever possible, use VAR_IN_OUT CONSTANT.

Importance: High

When the “replace constants” option is selected in the “Compiler options” of the project settings, the address operator is not permitted for scalar constants (integer, BOOL, REAL) and a compile error is issued. (Constant strings, structures, and arrays always have an address.)

Example

PROGRAM PLC_PRG
VAR CONSTANT
  c_iValue : INT := INT#15;
END_VAR
VAR
  poiValue : POINTER TO INT;
END_VAR
poiValue := ADR(c_iValue); // SA0007
--> SA0007: Address to constant variable 'c_iValue'

See also
● Chapter 1.7.3.1 “Configuring and Running Static Analysis” on page 3601

SA0008: Check subrange types

Detects out-of-range violations of subrange types. Assigned literals are already checked by the compiler. When constants are assigned, then the values must be within the defined range. When variables are assigned, then the data types must be identical.

Justification: If subrange types are used, then make sure that this subrange is not exited. The compiler checks for these kinds of subrange violations only for assignments of constants.

Importance: Low

The check is not performed for CFC objects because the code structure does not allow for it.
Example

VAR_GLOBAL
   iVarGlob:INT;
END_VAR

PROGRAM PLC_PRG
VAR
   iSubr1: INT (INT#1..INT#10);
   iSubr2: INT (INT#1..INT#1000);
   iCount: INT;
   by_SubType : BYTE (BYTE#0..BYTE#11);
   iVar : INT (-4095..4095);
END_VAR

iSubr1 := nCount;              // SA0008
iSubr1 := subr2;               // SA0008
iSubr1 := gvl.iVarGlob;        // SA0008
//byBYTE_SubType := BYTE#123;  //already detected by compiler,
error "Cannot convert type..."

--> SA0008: Subrange variable 'iSubr1' maybe out of allowed range

See also

● 溘 Chapter 1.7.3.1 “Configuring and Running Static Analysis” on page 3601

SA0009: Unused return values

Detects function, method and property calls in which the return value is not used

Justification: When a function or method returns a return value, it should also be evaluated. The return value often indicates whether or not the function was executed successfully. If not, then you will not be able to identify later whether the return value was forgotten or if it is actually not needed.

Exception: If a return value is irrelevant to the call, then you can document this and omit the assignment. Error returns should never be ignored.

Importance: Medium

PLCopen rule: CP7 / CP17

Example

FUNCTION Return_BOOL : BOOL
VAR_INPUT
END_VAR
VAR
   xTest : BOOL;
END_VAR
xTest := FALSE;
Return_BOOL := xTest;

PROGRAM PLC_PRG
Return_BOOL ();  // SA0009

--> SA0009: Ignoring return value of 'Return_Bool'

See also

● 溘 Chapter 1.7.3.1 “Configuring and Running Static Analysis” on page 3601
SA0010: Arrays with only one component

Detects arrays with only one element

Justification: An array with one element can be replaced by a base-type variable. Access to this variable is considerably faster than access by index to the variable.

Exception: The length of an array is often determined by a constant and is a parameter for a program. Then the program can work with arrays of different lengths and does not have to be changed if the length is only 1. This kind of situation should be documented accordingly.

Importance: Low

Example

```pascal
PROGRAM PLC_PRG
VAR
  aoiEmpty : ARRAY [22..22] OF INT;
  aorEmpty : ARRAY [1..1] OF REAL;
END_VAR

aoiEmpty;
aorEmpty;

--> SA0010: Vacuous array element in variable 'aoiEmpty'
--> SA0010: Vacuous array element in variable 'aorEmpty'
```

See also

- Chapter 1.7.3.1 “Configuring and Running Static Analysis” on page 3601

SA0011: Useless declarations

Detects structures or enumerations with only one component

Justification: This kind of declaration can be confusing for the reader. A structure with only one element can be replaced by an alias type. An enumeration with only one element can be replaced by a constant.

PLCopen rule: CP22 / CP24

Importance: Low

Example

```pascal
TYPE SingleStruct : STRUCT
  iPart : INT;
END_STRUCT
END_TYPE

TYPE myUnion :
  UNION
    lrValue : LREAL;
END_UNION
END_TYPE

TYPE SingleEnum :
  (
    OnlyOne := 1
  );
END_TYPE

--> Useless declaration 'SingleStruct'
--> Useless declaration 'myUnion'
--> Useless declaration 'SingleEnum'
```
See also

- § Chapter 1.7.3.1 “Configuring and Running Static Analysis” on page 3601

SA0012: Variable which could be declared as constants

Detects variables that are not accessed with write permission and therefore could be declared as constants

Justification: If a variable is written only at the declaration point and is otherwise used only for reading, then the static analysis assumes that the variable should also not be changed. Firstly, a declaration as a constant results in checking that the variable is not changed when the program is changed. Secondly, the declaration as a constant may result in faster code.

NOTICE!

If multiple applications exist in one project, then only the objects below the currently active application are affected. If there is only one application, then the objects in the common POU pool are also affected.

Importance: Low

Example

PROGRAM PLC_PRG
VAR
  iVar : INT := INT#17;
  iTest : INT;
END_VAR
iTest := iTest + iVar;   // SA0012: iVar could be declared as constant

--> SA0012: Variable 'iVar' could be declared as constant

See also

- § Chapter 1.7.3.1 “Configuring and Running Static Analysis” on page 3601

SA0013: Declarations with the same variable name

Detects variables with names that are already used by other variables (for example, global and local variables with the same name). Also detects variables with names of functions, actions, methods, or properties which are used in the same access scope. Variables are also detected that are declared in a GVL in the “Devices” view or in the POU pool. For this, however, the GVL of the “POUs” view have to be used in the application program.

Justification: The same names can be confusing when reading the code, and they can cause errors if the wrong object is accessed unintentionally. We recommend that you use naming conventions to avoid these situations.

PLCopen rule: N5 / N9

Importance: Medium
Example

```plaintext
VAR_GLOBAL
   xVar1 : BOOL;
   iVar3 : INT;
END_VAR

PROGRAM PLC_PRG
VAR
   xVar1 : BOOL;  // SA0013
   iVar3 : INT;   // SA0013
END_VAR

xVar1 := NOT GVL.xVar1;
iVar3 := iVar3 + INT#2;
iVar3 := GVL.iVar3;

--> SA0013: Declaration of 'iVar1' hides symbol 'GVL.iVar1'
--> SA0013: Declaration of 'xVar3' hides symbol 'GVL.xVar3'
```

Example

The function block **POU** has the action **ACT** and the method **METH**.

```plaintext
FUNCTION_BLOCK POU
VAR
   ACT : UINT;  // SA0013
   METH : BYTE; // SA0013
END_VAR

--> SA0013: Declaration of 'ACT' hides symbol 'POT.ACT'
--> SA0013: Declaration of 'METH' hides symbol 'POT.METH'
```

See also

- % Chapter 1.7.3.1 “Configuring and Running Static Analysis” on page 3601

**SA0014: Assignment of instances**

Detects assignments to function block instances. In the case of instances with pointer or reference variables, these assignments are potentially risky.

**Justification:** This is a performance warning. When an instance is assigned to another instance, all elements and subelements are copied from the one instance to the other instance. Pointers to data are also copied, but not their referenced data, so that the target instance and the source instance contain the same data after the assignment. Depending on the size of the instances, this kind of assignment could last a long time. For example, if an instance should be passed to a function for processing, then it is much more efficient to pass a pointer to the instance. If you want to selectively copy values from one instance to another, then a copy method is useful: `inst_First.Copy_From(inst_Second)`.

**Importance:** Medium
PROGRAM PLC_PRG
VAR
    inst_First : My_FB;
    inst_Second : My_FB;
END_VAR
inst_First();
inst_Second := inst_First;  // SA0014

--> SA0014: Assignment of instances

See also
- ☑ Chapter 1.7.3.1 “Configuring and Running Static Analysis” on page 3601

SA0015: Access to global data via FB_Init

Detects the access of a function block to global variables by means of the method FB_Init.
The value of this variable depends on the order of initializations.

Justification: Depending on the declaration location of the POU instance, an uninitialized variable could be accessed if the rule is violated.

Importance: High

Example

VAR_GLOBAL
    g_xTest1 : BOOL;
    g_iTest3 : INT;
END_VAR

METHOD PUBLIC fb_init : BOOL
VAR_INPUT
    bInitRetains : BOOL; // If TRUE, the retain variables are initialized (warm start / cold start)
    bInCopyCode : BOOL; // If TRUE, the instance afterwards gets moved into the copy code (online change)
END_VAR

g_xTest1 := NOT g_xTest1;       // SA0015
g_iTest3 := g_iTest3 + INT#1;   // SA0015

--> SA0015: FB_Init method of function block 'POU' accesses global data

See also
- ☑ Chapter 1.7.3.1 “Configuring and Running Static Analysis” on page 3601

SA0016: Gaps in structures

Detects gaps in structures or function blocks that are caused by the alignment requirements of the currently set target system. If possible, you should remove the gaps by resorting the structure elements or filling them with a dummy element. If this is not possible, then you can deactivate the rule for the affected structures by means of the analysis pragma.

Justification: Due to different alignment requirements on different platforms, there may be a different layout in the memory for these kinds of structures. Then the code can perform differently, depending on the platform.

Importance: Low
Example

PROGRAM PLC_PRG
VAR
  myStruct : Unpadded_Structure;
END_VAR
myStruct.iTest := 0;

TYPE Unpadded_Structure :
  STRUCT
    xTest : BOOL;
    iTest : INT; // SA0016
    byTest : BYTE;
    wTest : WORD;
  END_STRUCT
END_TYPE

--> SA0016: Structure 'Unpadded_Structure' must be padded (pack-mode=8)

See also
  ● § Chapter 1.7.3.1 “Configuring and Running Static Analysis” on page 3601

SA0017: Non-regular assignments

Detects assignments to pointers that are neither addresses (ADR operator, pointer variables) nor constants 0

Justification: If a pointer contains a value that is not a valid address, then an access violation exception occurs when dereferencing the pointer.

Importance: High

Example

PROGRAM PLC_PRG
VAR
  pInt : POINTER TO INT;
  dwAddress : DWORD;
END_VAR

dwAddress := dwAddress + DWORD#1;
pInt := dwAddress; // SA0017

--> SA0017: Non-regular assignment

See also
  ● § Chapter 1.7.3.1 “Configuring and Running Static Analysis” on page 3601

SA0018: Unusual bit access

Detects bit access to signed variables. However, the IEC 61131-3 standard permits only bit access and bit shift operations on bit fields. See also the strict rules SA0147 and SA0148.

Justification: Signed data types should not be used as bit fields and the other way around. The IEC 61131-3 standard does not provide for this kind of access, and therefore you should comply with this rule when you write portable code.

Importance: Medium
Exception for flag enumerations: When an enumeration is declared as a flag by means of the {attribute 'flags'} pragma attribute, the SA0018 error is not issued for bit access with the OR, AND or NOT operators.

Example

PROGRAM PLC_PRG
VAR
    iTemp1 : INT;
    diTemp3 : DINT;
    uliTemp4 : UUINT;
    siTemp5 : SINT;
    usiTemp6 : USINT;
    byTemp2 : BYTE;
END_VAR
iTemp1.3 := TRUE;     // SA0018
diTemp3.4 := TRUE;    // SA0018
uliTemp4.18 := FALSE; // no error because this is an unsigned data type
siTemp5.2 := FALSE;   // SA0018
usiTemp6.3 := TRUE;   // no error because this is an unsigned data type
byTemp2.5 := FALSE;   // no error because the byte is a bitfield
--> SA0018: Unusual bit access

See also
- % Chapter 1.7.3.1 “Configuring and Running Static Analysis” on page 3601
- % Chapter 1.7.3.2.49.12 “SA0147: Unusual shift operation - strict” on page 3676
- % Chapter 1.7.3.2.49.13 “SA0148: Unusual bit access - strict” on page 3677

SA0020: Possibly assignment of truncated value to REAL variable

Detects operations on integer variables for which a truncated value could be assigned to a REAL data type variable

Justification: Static analysis issues an error when the result of an integer calculation is assigned to a REAL or LREAL variable. The programmer should be alerted to a possible incorrect interpretation of this kind of assignment: lrealvar := dintvar1 * dintvar2. Because the value range of LREAL is greater than that of DINT, one could assume that the result of the calculation could always be represented in LREAL. But that is not the case. The processor calculates the result of the multiplication as an integer and then casts the result to LREAL. An overflow in the integer calculation would be lost. To work around the problem, the calculation has to be done as a REAL operation: lreal_var := TO_LREAL(dintvar1) * TO_LREAL(dintvar2).

Importance: High

Example

PROGRAM PLC_PRG
VAR
    rx : LREAL;
    di : DINT;
END_VAR
rx := di * di                              // SA0020
rx := TO_LREAL(di) * TO_LREAL(di) // No message
--> SA0020: Possibly assignment of truncated value to REAL variable
See also

● § Chapter 1.7.3.1 “Configuring and Running Static Analysis” on page 3601

SA0021: Transporting the address of a temporary variable

Detects address assignments of temporary variables (on the stack) to non-temporary variables

Justification: Local variables of a function or method are created on the stack and they exist only while the function or method is being processed. If a pointer points to this kind of variable after processing the method or function, then you can use this pointer to access undefined memory, or to access an incorrect variable in another function. This situation should be avoided at all costs.

Importance: High

Example

FUNCTION TempVarInFUNC : DWORD
VAR
  uiTemp : UINT;
END_VAR
TempVarInFUNC := ADR(uiTemp);  // SA0021

PROGRAM PLC_PRG
VAR
  dwTest : DWORD;
END_VAR
dwTest := TempVarInFUNC();

--> SA0021: Transporting address of temporary variable to outer scope symbol

See also

● § Chapter 1.7.3.1 “Configuring and Running Static Analysis” on page 3601

SA0022: (Possibly) unassigned return value

Detects all functions and methods that include an execution thread without an assignment to the return value

Justification: An unassigned return value in a function or method is an indication of missing code. Even if the return value always has a default value, it is always useful to assign it again explicitly to avoid confusion.

Importance: Medium

Example

FUNCTION FUN : DINT
VAR_INPUT
  bTest : BOOL;
END_VAR

IF bTest THEN
  RETURN;
ENDIF
FUN := 99;

--> SA0022: (Possibly) unassigned return value
SA0023: Too big return values

Detects structured return values that fill more than 4 bytes of memory and therefore are too large. Moreover, return values of type `STRING` are identified (regardless of the size of the used memory).

Justification: This is a performance warning. If large values are returned as the result of a function, method, or property, then the processor copies them multiple times when executing the code. This can lead to runtime problems and should be avoided whenever possible. Performance can be improved by passing a structured value as `VAR_IN_OUT` to a function or method and filling it in the function or method.

Importance: Medium

Example

```plaintext
TYPE LargeStructure :
  STRUCT
    a : LINT;
    b : BOOL;
  END_STRUCT
END_TYPE

TYPE SmallStructure :
  STRUCT
    a : INT;
    b : BYTE;
  END_STRUCT
END_TYPE

FUNCTION Large_Return_Value_FUNC : LargeStructure   // SA0023
FUNCTION Small_Return_Value_FUNC : SmallStructure   // OK

--> SA0023: Structured return value
```

SA0024: Untyped literals / constants

Detects untyped literals and constants.

Justification: CODESYS assigns types for literals depending on their use. In some cases, this can cause unexpected problems, which should be resolved better with a typed literal. For example: `dw := ROL(DWORD#1, i)`

Importance: Low
Example

PROGRAM PLC_PRG
VAR
  iTemp1 : INT = 10; // SA0024
diTemp2 : DINT;
liTemp3 : LINT;
rTemp4 : REAL;
lrTemp5 : LREAL;
END_VAR
iTemp1 := iTemp1 + INT#34;
diTemp2 := diTemp2 + 23; // SA0024
liTemp3 := liTemp3 + 124; // SA0024
rTemp4 := rTemp4 + 1.1; // SA0024
lrTemp5 := lrTemp5 + 3.4; // SA0024
---> SA0024: Untyped literal found

See also

- Chapter 1.7.3.1 “Configuring and Running Static Analysis” on page 3601

SA0025: Unqualified enumeration constants

Detects enumeration constants for which a qualified name does not prepend the enumeration
Justification: Qualified access makes the code more readable and easier to maintain. Without
forcing qualified variable names, an additional enumeration could be inserted when the program
is extended. This enumeration contains a constant with the same name as an existing enumera-
tion (see the example below: "red"). This would result in ambiguous access to this piece of
code. We recommend to always use only enumerations with the {attribute 'qualified-only'}.
Importance: Medium

Example

TYPE COLOR
  (red, green, blue);
END_TYPE

PROGRAM PLC_PRG
enumVar : COLOR;
enumVar := COLOR.red; // SA0025
enumVar := red; // SA0025
---> SA0025: Enumeration constant 'red' not qualified

See also

- Chapter 1.7.3.1 “Configuring and Running Static Analysis” on page 3601

SA0026: Possible truncated strings

Detects string assignments and string initializations that do not use sufficient string length
Justification: When strings of different lengths are assigned, a string could be truncated. This
can have unexpected results.
Importance: Medium
Example

PROGRAM PLC_PRG
VAR
    strVar1 : STRING[10];
    strVar2 : STRING[6];
END_VAR

strVar2 := strVar1; // SA0026

--> SA0026: Truncation of string 'abcdefghi'
--> SA0026: Possible truncation of string 'strVar1'

See also

●  § Chapter 1.7.3.1 “Configuring and Running Static Analysis” on page 3601

SA0027: Multiple uses of identifiers

Detects multiple uses of a name/identifier for a variable or an object (POU) within the scope of a project

Justification: Same names can be confusing when reading the code. They can cause errors if the wrong object is accessed accidentally. Define and follow naming conventions to avoid any situation like this.

The following cases are detected:

● The name of an enumeration is identical to the name of another enumeration in the application or in an integrated library.
● The name of a variable is identical to the name of another object in the application or in an integrated library.
● The name of a variable is identical to the name of an enumeration constant in an enumeration in the application or in an integrated library.
● The name of an object is identical to the name of another object in the application or in an integrated library.

Importance: Medium

Example

The Standard library is integrated in the project and provides the TON function.

PROGRAM PLC_PRG
VAR
    ton : INT;
END_VAR

--> Variable name 'ton' in 'PLC_PRG' already used for an object in library 'standard,...'

See also

●  § Chapter 1.7.3.1 “Configuring and Running Static Analysis” on page 3601

SA0028: Overlapping memory areas

Detects the lines of code where two or more variables reserve the same memory

Justification: When two variables reserve the same memory, the code may behave with unexpected results. This situation should be avoided at all costs. If you cannot avoid using a value in different interpretations (for example, one time as DINT and another time as REAL), then you should define a UNION. You can also use a pointer to access a value with a different type without the value being converted.
Importance: High

**Example**

```
PROGRAM PLC_PRG
VAR
 iVar AT %QB21: INT;
 dwVar2 AT %QD5: DWORD;
END_VAR

--> The following variables access the same memory:
--> SA0028: iVar AT %QB21
--> SA0028: dwVar2 AT %QD5
```

See also
- § Chapter 1.7.3.1 “Configuring and Running Static Analysis” on page 3601

**SA0029: Notation in code different to declaration**

Detects the code locations where the notation of an identifier is different from the notation in its declaration

Justification: The IEC 61131-3 standard defines identifiers as not case-sensitive. This means that a variable declared as "varx" can also be used as "VaRx" in the code. However, this is confusing and misleading and should be avoided.

Importance: Medium

**Example**

```
A POU PLC_PRG and a POU fnc (function) exist in the device tree.

PROGRAM PLC_PRG
VAR
 iVar: INT;
 _123test_var_: INT;
END_VAR

ivar := iVar + 1;                   // SA0029
_123TEST_var_ := _123test_var_;        // SA0029
Fnc();                              // SA0029

--> SA0029: Notation in code (ivar) must equal declaration (iVar)
--> SA0029: Notation in code (_123TEST_var_) must equal declaration (_123test_var_)
--> SA0029: Notation in code (Fnc) must equal declaration (fnc)
```

See also
- § Chapter 1.7.3.1 “Configuring and Running Static Analysis” on page 3601

**Unused Objects**

- 1.7.3.2.28.1 SA0031: Unused signatures.......................................................... 3633
- 1.7.3.2.28.2 SA0032: Unused enumeration constants................................. 3633
- 1.7.3.2.28.3 SA0033: Unused variables......................................................... 3633
- 1.7.3.2.28.4 SA0035: Unused input variables............................................... 3634
- 1.7.3.2.28.5 SA0036: Unused output variables............................................. 3634
SA0031: Unused signatures
Detects programs, function blocks, functions, data types, interfaces, methods, properties, and actions that are not called within the compiled program code

Justification: Unused objects unnecessarily increase the size of the project and can be confusing when reading the code.

Importance: Low

PLCopen rule: CP2

If multiple applications exist in a project, then only the objects below the currently active applications are affected. If there is only one application, then the objects in the POU pool are also affected.

See also
● § Chapter 1.7.3.1 “Configuring and Running Static Analysis” on page 3601

SA0032: Unused enumeration constants
Detects enumeration constants that are not used in the compiled program code

Justification: Unused enumeration constants unnecessarily increase the size of the enumeration definition and can be confusing when reading the program.

PLCopen rule: CP24

Importance: Low

If multiple applications exist in a project, then only the objects below the currently active applications are affected. If there is only one application, then the objects in the common POU pool are also affected.

Example

TYPE My_Enum :
  (one := 1, two := 2);
END_TYPE

--> SA0032: Unused enumeration constant 'one'
--> SA0032: Unused enumeration constant 'two'

See also
● § Chapter 1.7.3.1 “Configuring and Running Static Analysis” on page 3601

SA0033: Unused variables
Detects variables that are declared but not used within the compiled program code

Justification: Unused variables make a program less readable and maintainable. Unused variables unnecessarily fill memory and unnecessarily waste runtime during initialization.

Importance: Medium

PLCopen rule: CP22 / CP24
For GVL variables: If multiple applications exist in a project, then only the objects below the currently active applications are affected. If there is only one application, then the objects in the common POU pool are also affected.

**Example**

```plaintext
PROGRAM PLC_PRG
VAR
    iCounter1 : INT;
    iCounter2 : INT; // SA0035
END_VAR

iCounter1 := 100;
--> SA0035: Unused Variable 'iCounter2'
```

See also

- ⇢ *Chapter 1.7.3.1 “Configuring and Running Static Analysis” on page 3601*

**SA0035: Unused input variables**

Detects input variables that are not used by any function block instance

Justification: Unused variables make a program less readable and maintainable. Unused variables unnecessarily fill memory and unnecessarily waste runtime during initialization.

Importance: Medium

PLCopen rule: CP24

**Example**

```plaintext
FUNCTION_BLOCK AFB
VAR_INPUT
    iIn1: INT;
    iIn2: INT;
END_VAR

VAR_OUTPUT
    iOut1: INT;
END_VAR

PROGRAM PLC_PRG
VAR
    Fb1: AFB;
END_VAR

Fb1(iIn1 := 99)
--> SA0035: Unused input 'iIn2'
```

See also

- ⇢ *Chapter 1.7.3.1 “Configuring and Running Static Analysis” on page 3601*

**SA0036: Unused output variables**

Detects output variables that are not used by any function block instance

Justification: Unused variables make a program less readable and maintainable. Unused variables unnecessarily fill memory and unnecessarily waste runtime during initialization.
Example

```plaintext
FUNCTION_BLOCK AFB
VAR_INPUT
  iIn1: INT;
  iIn2: INT;
END_VAR
VAR_OUTPUT
  iOut1: INT;
END_VAR
PROGRAM PLC_PRG
VAR
  Fb1: AFB;
END_VAR
Fb1(iIn1 := 99)
--> SA0036: Unused output 'iOut1'
```

See also
- § Chapter 1.7.3.1 “Configuring and Running Static Analysis” on page 3601

**SA0034: Enumerations with incorrect assignment**

Detects values that are assigned to an enumeration variable. Only defined enumeration constants of an enumeration variable are permitted to be assigned.

Justification: A variable of the enumeration type should have only the intended values, otherwise the code that uses this variable may not work correctly. We recommend to always use enumerations with the `{attribute 'strict'}`. Then the compiler already checks the correct use of the enumeration components.

Importance: High

Example

```plaintext
TYPE COLOR :
  (  
    Red := 0,
    Green,
    Yellow
  );
END_TYPE

PROGRAM PLC_PRG
VAR
  eColor1: COLOR;
END_VAR

eColor1 := COLOR.Red;
eColor1 := 1;     // SA0034
--> SA0034: Use enumeration value instead of 'INT#1'
```

See also
- § Chapter 1.7.3.1 “Configuring and Running Static Analysis” on page 3601
SA0037: Write access to input variable

Detects input variables (VAR_INPUT) that are accessed with write permission within the POU

Justification: According to the IEC 61131-3 standard, an input variable must not be changed within a POU. This kind of access is also a cause for errors and makes the code poorly maintainable. This is an indication that a variable is used as both an input variable and an auxiliary variable. This kind of dual use should be avoided.

Importance: Medium

Example

```plaintext
VAR_GLOBAL
  g_xGlob AT %QX0.0 : BOOL;
END_VAR

PROGRAM PLC_PRG
VAR_INPUT
  xVarIn1:BOOL;
  xVarIn2:BOOL;
END_VAR
VAR
  iCondition : INT;
END_VAR

iCondition := iCondition + INT#1;
CASE iCondition OF
  INT#1:
    g_xGlob := xVarIn1;
  INT#2:
    g_xGlob := xVarIn2;
ELSE
  g_xGlob := FALSE;
  xVarIn1 := FALSE;  // SA0037
END_CASE

--> SA0037: Write access to input variable 'xVarIn1'
```

See also

- Chapter 1.7.3.1 “Configuring and Running Static Analysis” on page 3601

SA0038: Read access to output variable

Detects output variables (VAR_OUTPUT) that are accessed with read permission within the POU

Justification: According to the IEC 61131-3 standard, it is prohibited to read an output within a POU. This is an indication that the output is not only used as an output but also as a temporary variable for intermediate results. This kind of dual use should be avoided.

Importance: Low
Example

```plaintext
VAR_GLOBAL
    g_xGlob AT %QX0.0 : BOOL ;
    g_iGlob AT %QW1 : INT ;
END_VAR

PROGRAM PLC_PRG
VAR_OUTPUT
    xVarOut1:BOOL;
    xVarOut2:INT;
    xVarOut3:INT;
END_VAR
VAR
    iCondition : INT;
END_VAR

iCondition := iCondition + INT#1;
CASE iCondition OF
    INT#1:
        xVarOut1 := g_xGlob;
        xVarOut2 := g_iGlob;
    INT#2:
        xVarOut3 := xVarOut2; // SA0038
    ELSE
        xVarOut1 := FALSE;
        g_xGlob := xVarOut1; // SA0038
        xVarOut2 := INT#0;
        xVarOut3 := INT#-1;
END_CASE

--> SA0038: Read access to output variable 'xVarOUT2'
--> SA0038: Read access to output variable 'xVarOUT1'
```

See also

- ¶ Chapter 1.7.3.1 “Configuring and Running Static Analysis” on page 3601

SA0040: Possible division by zero

Detecets code locations where there is possible division by zero

Justification: Division by zero should never occur, and a variable denominator should always be checked for 0 first.

Importance: High
**Example**

```plaintext
VAR_GLOBAL
    g_iVar AT %QW1 : INT;
END_VAR

PROGRAM PLC_PRG
VAR
    iCounter : INT;
    iSumme : INT;
    iMid : INT;
    iVal1 : INT := INT#2;
    iVal2 : INT;
    iVal3 : INT := INT#3;
    iVal4 : INT := INT#4;
    iVal5 : INT;
END_VAR

IF iVal2 <> 0 THEN
    iVal1 := iVal1 / iVal2;  // no error
END_IF;

iMid := iSumme / iCounter;  // SA0040
iCounter := iCounter + INT#1;
iSumme := g_iVar + iSumme;
IF iMid < INT#100 THEN
    iVal1 := iVal1 / iVal2;  // SA0040
END_IF

--> SA0040: Possible division by zero
--> SA0040: Possible division by zero
```

See also

- ☀ Chapter 1.7.3.1 “Configuring and Running Static Analysis” on page 3601

**SA0041: Detect possible loop invariant code**

Detects assignments in loops that calculate the same value for each loop cycle. These lines of
code could possibly be inserted outside of the loop.

Justification: This is a performance warning. Code that is executed in a loop, but does the same
thing in each loop cycle, can be executed outside of the loop.

Importance: Medium

**Example**

```plaintext
PROGRAM PLC_PRG
VAR
    iCounter, iVar1, iVar2: INT;
END_VAR

FOR iCounter := 0 TO 10 DO
    iVar1 := 100;        // SA0041
    iVar2 := iVar2 + iVar1;
END_FOR

--> SAN0041: Possible loop invariant code 'iVar1 := 100'
```

See also

- ☀ Chapter 1.7.3.1 “Configuring and Running Static Analysis” on page 3601
SA0042: Usage of different access paths

Detects the usage of different access paths for the same variable

Justification: Different access to the same element decreases the readability and maintainability of a program. We recommend the consistent usage of `{attribute 'qualified-only'}` for libraries, global variable lists, and enumerations. This forces a fully qualified access.

Importance: Low

Example

```plaintext
VAR_GLOBAL
    iTemp:INT;
    instPOU:POU;
END_VAR

FUNCTION_BLOCK POU
VAR
    a:INT;
END_VAR
    a := INT#1;

PROGRAM SA0042
VAR
    ptiTemp:POINTER TO INT;
    sTemp:STRING;
END_VAR
    ptiTemp := ADR(iTemp);
    ptiTemp^:= INT#1;
    iTemp:= INT#2;                       // SA0042 - direct access
    on variable
    GVL.iTemp := INT#2;                  // SA0042 - access on
    variable via GVL
    sTemp := CONCAT( 'ab', 'cd');        // SA0042 - direct access on
    function
    sTemp := Standard.CONCAT( 'ab', 'cd'); // SA0042 - access on
    function via Standard
    instPOU();                           // SA0042 - direct access
    on POU instance
    GVL.instPOU();                       // SA0042 - access via GVL

--> SA0042: Different access paths for 'CONCAT'
--> SA0042: Different access paths for 'Standard.CONCAT'
--> SA0042: Different access paths for 'instPOU'
--> SA0042: Different access paths for 'GVL.instPOU'
--> SA0042: Different access paths for 'iTemp'
--> SA0042: Different access paths for 'GVL.iTemp'
```

See also

- Chapter 1.7.3.1 “Configuring and Running Static Analysis” on page 3601

SA0043: Use of a global variable in only one POU

Detects the use of a global variable in only a single POU

Justification: A global variable that is used in only one location should also only be declared at this location.

Importance: Medium
PLCopen rule: CP26

**Example**

```plaintext
VAR_GLOBAL
  g_xVar AT %QX0.0 : BOOL;
  g_iTest AT %QW1 : INT;
  g_wTest AT %QW2 : WORD;
END_VAR

PROGRAM prog1
VAR
  iCondition : INT;
  bTemp :BOOL;
END_VAR

  IF iCondition < INT#0 THEN
    bTemp := g_xVar; // SA0043 - g_xVar only read in this POU
  ELSEIF iCondition = INT#0 THEN
    bTemp := g_xVar; // SA0043 - g_xVar only read in this POU
  ELSE
    bTemp := g_xVar; // SA0043 - g_xVar only read in this POU
    g_wTest := WORD#4; // g_wTest used also in prog2 -> OK
  END_IF

PROGRAM prog2
VAR
  iCondition : INT;
END_VAR

  CASE iCondition OF
    INT#1:
      g_iTest := WORD_TO_INT(g_wTest); // SA0043 - g_iTest only written in this POU
    INT#2:
      g_iTest := INT#2;                 // SA0043 - g_iTest only written in this POU
    ELSE
      g_iTest := INT#3;                 // SA0043 - g_iTest only written in this POU
  END_CASE

--> SA0043: Global variable 'g_xVar' only used in 'prog1'
--> SA0043: Global variable 'g_iTest' only used in 'prog2'
```

See also

- Chapter 1.7.3.1 “Configuring and Running Static Analysis” on page 3601

**SA0044: Declarations with reference to interface**

Detects declarations with `REFERENCE TO` interfaces and declarations of `VAR_IN_OUT` variables with interfaces (implicitly implemented by means of `REFERENCE TO`)

Justification: An interface type is always implicitly a reference to an instance of a function block that implements this interface. A reference to an interface is therefore a reference to a reference and can result in unwanted behavior.

Importance: High
**Example**  

ITF is an interface that is defined in the project.

```plaintext
PROGRAM PLC_PRG
VAR
    inst:POU;
    itf_inst1 : ITF;
    itf_ref : REFERENCE TO ITF; // SA0044
END_VAR FUNCTION_BLOCK POU
VAR_INPUT
    inst_itf2 : ITF;
END_VAR
VAR_OUTPUT
    inst_itf3 : ITF;
END_VAR
VAR_IN_OUT
    inst_itf4 : ITF;           // SA0044
END_VAR

--> SA0044: Reference to interface 'itf_ref'
--> SA0044: Reference to interface 'itf4_ref'
```

See also

- § Chapter 1.7.3.1 “Configuring and Running Static Analysis” on page 3601

**Conversions**

1.7.3.2.37.1 SA0019: Implicit pointer conversions........................................ 3641
1.7.3.2.37.2 SA0130: Implicit expanding conversions.................................... 3642
1.7.3.2.37.3 SA0131: Implicit narrowing conversions.................................. 3643
1.7.3.2.37.4 SA0132: Implicit signed/unsigned conversions........................... 3644
1.7.3.2.37.5 SA0133: Explicit narrowing conversions.................................. 3644
1.7.3.2.37.6 SA0134: Explicit signed/unsigned conversions........................... 3645

**SA0019: Implicit pointer conversions**

Detects implicitly generated pointer conversions

Justification: In CODESYS, pointers are not strictly typed and they can be assigned to each other in any way. This is often used and therefore not reported by the compiler. However, it can also accidentally cause unexpected access. If you assign a POINTER TO BYTE to a POINTER TO DWORD, then you can unintentionally overwrite memory using the latter pointer. Therefore, always check this rule and block the message for cases in which you intentionally want to access a value with a different type.

Implicit data type conversions are reported with a different message.

Importance: High

PLCopen rule: CP25

Exception: BOOL <-> BIT
Example

PROGRAM PLC_PRG
VAR
  pINT : POINTER TO INT;
  byteVar : BYTE;
END_VAR

pINT := ADR(byteVar);

--> SA0019: Implicit conversion from pointer to 'POINTER TO BYTE'
to pointer to 'POINTER TO INT'

See also

● Chapter 1.7.3.1 “Configuring and Running Static Analysis” on page 3601

SA0130: Implicit expanding conversions

Detects implicit conversions from smaller data types to larger data types

Justification: The compiler permits any assignments of different types when the value range of the source type is completely contained within the value range of the target type. However, the compiler will build a conversion into the code as late as possible. For an assignment of type lint := dint * dint, the compiler performs the implicit conversion only after multiplication: lint := TO_LINT(dint * dint). An overflow is therefore truncated. To prevent this, you can already convert the elements: lint := TO_LINT(dint) * TO_LINT(dint). Therefore, it may be useful to report locations where the compiler implements implicit conversions in order to check whether these are exactly what is intended. Furthermore, explicit conversions can be used to improve portability to other systems when those systems have more restrictive type checks.

Importance: Low
Example

PROGRAM PLC_PRG
VAR
  byTemp : BYTE;
  usiTemp : USINT;
    uiTemp : UINT;
    iTemp : INT;
    udiTemp : UDINT;
  diTemp : DINT;
  uliTemp : ULINT;
    liTemp : LINT;
    lwTemp : LWORD;
    lrTemp : LREAL;
END_VAR

liTemp := iTemp;       // SA0130
uliTemp := usiTemp;    // SA0130
lwTemp := udiTemp;     // SA0130
lrTemp := byTemp;      // SA0130
diTemp := uiTemp;      // SA0130

byTemp.5 := FALSE;     // OK (BIT_BOOL conversion)

--> SA0130: Implicit widening conversion from type 'INT' to type 'LINT'
--> SA0130: Implicit widening conversion from type 'USINT' to type 'ULINT'
--> SA0130: Implicit widening conversion from type 'UDINT' to type 'LWORD'
--> SA0130: Implicit widening conversion from type 'BYTE' to type 'LREAL'
--> SA0130: Implicit widening conversion from type 'UINT' to type 'DINT'

See also

● "Chapter 1.7.3.1 “Configuring and Running Static Analysis” on page 3601"

SA0131: Implicit narrowing conversions

Detects implicit conversions from larger data types to smaller data types

Justification: This message is obsolete now because it is already reported as a warning by the compiler.

Importance: Low

Example

PROGRAM PLC_PRG
VAR
  rTemp : REAL;
  lrTemp : LREAL;
END_VAR
rTemp := lrTemp;    // SA0131

--> SA0131: Implicit narrowing conversion from type 'LREAL' to type 'REAL'

See also

● "Chapter 1.7.3.1 “Configuring and Running Static Analysis” on page 3601"
SA0132: Implicit signed/unsigned conversions

Detects implicit conversions from signed data types to unsigned data types or the other way around.

This message is obsolete now because it is already reported as a warning by the compiler.

Importance: Low

Example

```plaintext
PROGRAM PLC_PRG
VAR
  byTest : BYTE;
  udiTest: UDINT;
  ulktest: ULIINT;
  wTest : WORD;
  lwTest : LWORD;
  siTest : SINT;
  iTest : INT;
  diTest : DINT;
  liTest : LINT;
END_VAR
liTest := ulktest;    // SA0132
udiTest:= diTest;     // SA0132
siTest := byTest;     // SA0132
wTest := iTest;       // SA0132
lwTest := siTest;     // SA0132
---> SA0132: Implicit signed/unsigned conversion from type 'ULINT' to type 'LINT'
---> SA0132: Implicit signed/unsigned conversion from type 'DINT' to type 'UDINT'
---> SA0132: Implicit signed/unsigned conversion from type 'BYTE' to type 'SINT'
---> SA0132: Implicit signed/unsigned conversion from type 'INT' to type 'WORD'
---> SA0132: Implicit signed/unsigned conversion from type 'SINT' to type 'LWORD'
```

See also

- § Chapter 1.7.3.1 “Configuring and Running Static Analysis” on page 3601

SA0133: Explicit narrowing conversions

Detects explicit conversions from a larger data type to a smaller data type

Justification: A large number of type conversions may indicate that you have chosen the wrong data types for variables. For this reason, there are programming guidelines that require an explicit justification for data type conversions.

Importance: Low
Example

PROGRAM SA0133
VAR
    siVar:SINT;
    diVar:DINT;
    liVar:LINT;
    byVar:BYTE;
    uiVar:UINT;
    dwVar:DWORD;
    lwVar:LWORD;
    rVar:REAL;
    lrVar:LREAL;
END_VAR
siVar := LINT_TO_SINT(liVar);     // SA0133
byVar := DINT_TO_BYTE(diVar);     // SA0133
siVar := DWORD_TO_SINT(dwVar);    // SA0133
uiVar := LREAL_TO_UINT(lrVar);    // SA0133
rVar := LWORD_TO_REAL(lwVar);     // SA0133

--> SA0133: Explicit narrowing conversion from type 'LINT' to type 'SINT'
--> SA0133: Explicit narrowing conversion from type 'DINT' to type 'BYTE'
--> SA0133: Explicit narrowing conversion from type 'DWORD' to type 'SINT'
--> SA0133: Explicit narrowing conversion from type 'LREAL' to type 'UINT'
--> SA0133: Explicit narrowing conversion from type 'LWORD' to type 'REAL'

See also

- § Chapter 1.7.3.1 “Configuring and Running Static Analysis” on page 3601

SA0134: Explicit signed/unsigned conversions

Detects explicit conversions from signed data types to unsigned data types and the other way around.

Justification: Excessive use of type conversions may indicate that you have chosen the wrong data types for variables. For this reason, there are programming guidelines that require an explicit justification for data type conversions.

Importance: Low
Example

```plaintext
PROGRAM PLC_PRG
VAR
  byVar : BYTE;
  udiVar : UDINT;
  uliVar : ULLINT;
  lwVar : LWORD;
  wVar : WORD;
  siVar : SINT;
  iVar : INT;
  diVar : DINT;
  liVar : LINT;
END_VAR
liVar := ULLINT_TO_LINT(uliVar);
udiVar := DINT_TO_UDINT(diVar);
siVar := BYTE_TO_SINT(byVar);
wVar := INT_TO_WORD(iVar);
lwVar := SINT_TO_LWORD(siVar);

--> SA0134: Explicit signed/unsigned conversion from type 'ULLINT'
to type 'LINT'
--> SA0134: Explicit signed/unsigned conversion from type 'DINT' to
type 'UDINT'
--> SA0134: Explicit signed/unsigned conversion from type 'BYTE' to
type 'SINT'
--> SA0134: Explicit signed/unsigned conversion from type 'INT' to
type 'WORD'
--> SA0134: Explicit signed/unsigned conversion from type 'SINT' to
type 'LWORD'
```

See also

- Chapter 1.7.3.1 “Configuring and Running Static Analysis” on page 3601

Use of Direct Addresses

1.7.3.2.38.1 SA0005: Invalid addresses and data types............................. 3646
1.7.3.2.38.2 SA0047: Accesses to direct address....................................... 3647
1.7.3.2.38.3 SA0048: AT-declarations on direct addresses......................... 3647

SA0005: Invalid addresses and data types

Detects invalid addresses and data type specifications. Valid size prefixes in addresses: X for
BOOL B for 1-byte data types, W for 2-byte data types, and D for 4-byte data types.

Justification: Variables located on direct addresses should preferably be associated with an
address that corresponds to their data type width. It can be confusing for the reader of the code,
for example, if a DWORD is assigned to a BYTE address.

Importance: Low
Example

PROGRAM Check_Address_Type_PRG
VAR
  iVar AT %QB0 : INT ;   // OK e. g.: %QW0
  xTest AT %QW1 : BOOL ; // OK e. g.: %QX1.0
END_VAR

iVar := iVar + INT#1;
xTest := NOT xTest;

--> SA0005: Invalid address for data type 'iVar'

See also

● % Chapter 1.7.3.1 “Configuring and Running Static Analysis” on page 3601

SA0047: Accesses to direct address

Detects direct address access in the implementation code

Justification: Symbolic programming is always preferable. A variable has a name that can also have a meaning. An address cannot indicate what it is used for.

Importance: High
PLCopen rule: N1 / CP1

Example

PROGRAM PLC_PRG
VAR
  xVar : BOOL;
  byVar : BYTE;
END_VAR

xVar := %IX0.0;
%QX0.0 := xVar;
%MX0.1 := xVar;
%MB1 := byVar;

--> Access to direct address '%IX0.0'
--> Access to direct address '%QX0.0'
--> Access to direct address '%MX0.1'
--> Access to direct address '%MB1'

See also

● % Chapter 1.7.3.1 “Configuring and Running Static Analysis” on page 3601

SA0048: AT-declarations on direct addresses

Detects AT declarations on direct addresses

Justification: The use of direct addresses in the code is problematic because the address then appears in multiple locations: first in the controller configuration where the assignment of a physical object to an address is defined, and second in the program where variables are assigned to these addresses. If the addresses are relocated because the configuration is changed, then you have to reassign variables to addresses at a completely different location in the program. This is a cause of error and results in poorer readability and maintainability of the code. Therefore, it is best to perform all assignments in the I/O mapping of the device editor.

Importance: High
PLCopen rule: N1 / CP1

Note: We recommend that you use direct addresses ONLY in the “I/O Mapping” tab of the device editor.

Example

```plaintext
PROGRAM PLC_PRG
VAR
  xVar1 AT %IX0.0 : BOOL;
  byVar1 AT %IB1 : BYTE;
  xVar2 AT %QX0.0 : BOOL;
END_VAR

--> SA0048: Declaration uses direct address '%IX0.0'
--> SA0048: Declaration uses direct address '%IB1'
--> SA0048: Declaration uses direct address '%QX0.0'
```

See also

- § Chapter 1.7.3.1 “Configuring and Running Static Analysis” on page 3601

Rules for Operators

1.7.3.2.39.1  SA0051: Comparison operations on BOOL variables.......... 3648
1.7.3.2.39.2  SA0052: Unusual shift operation.................................. 3649
1.7.3.2.39.3  SA0053: Too big bitwise shift.................................. 3649
1.7.3.2.39.4  SA0054: Comparisons of REAL/LREAL for equality / inequality................................................................. 3650
1.7.3.2.39.5  SA0055: Unnecessary comparisons of unsigned operands.... 3651
1.7.3.2.39.6  SA0056: Constant out of valid range.......................... 3651
1.7.3.2.39.7  SA0057: Possible loss of decimal places.................... 3652
1.7.3.2.39.8  SA0058: Operations on enumeration variables............... 3652
1.7.3.2.39.9  SA0059: Comparison operations always returning TRUE or FALSE................................................................. 3654
1.7.3.2.39.10 SA0060: Zero used as invalid operand....................... 3654
1.7.3.2.39.11 SA0061: Unusual operation on pointer....................... 3654
1.7.3.2.39.12 SA0062: Uses of TRUE or FALSE in expressions............ 3655
1.7.3.2.39.13 SA0063: Possibly not 16-bit-compatible operations........ 3655
1.7.3.2.39.14 SA0064: Addition of pointer.................................. 3656
1.7.3.2.39.15 SA0065: Incorrect pointer addition to base size.......... 3656
1.7.3.2.39.16 SA0066: Uses of temporary results.......................... 3657

**SA0051: Comparison operations on BOOL variables**

Detects comparison operations on variables of type **BOOL**

Justification: CODESYS permits these kinds of comparison, but they are very unusual and can be confusing. The IEC 61131-3 standard does not provide for these comparisons. By avoiding them, you increase the portability of the code to other development systems.

Importance: Medium
Example

PROGRAM PLC_PRG
VAR
    xBool1, xBool2 : BOOL;
    xResult : BOOL;
END_VAR

xResult := xBool1 > xBool2;  // SA0051
xBool1 := NOT xBool1;           // OK!
xBool2 := xBool2 XOR xBool1;    // OK!

--> SA0051: Comparison operations on BOOL variables

See also

● % Chapter 1.7.3.1 “Configuring and Running Static Analysis” on page 3601

SA0052: Unusual shift operation

Detects shift operations (bit shift) on signed variables. In the case of shift operations on bit field data types (Byte, DWORD, LWORD, WORD), an error is not reported.

Justification: CODESYS permits shift operations on signed data types. However, these operations are unusual and can be confusing. The IEC 61131-3 standard does not provide for these kinds of operations. Therefore, they should be avoided in order to increase the portability of the code to other development systems.

Importance: Medium

Example

PROGRAM PLC_PRG
VAR
    iTemp : INT;
    dwTemp1 : DWORD;
    byTemp2 : BYTE;
    diTemp3 : DINT;
    siTemp4 : SINT;
    liTemp5 : LINT;
END_VAR

//the following lines each will cause an  SA0052:
iTemp := SHL(iTemp, BYTE#2);
diTemp3 := SHR(diTemp3, BYTE#4);
siTemp4 := ROL(siTemp4, BYTE#2);
liTemp5 := ROR(liTemp5, BYTE#2);

//no error SA0052 because DWORD and BYTE are bit field data types:
dwTemp1 := SHL(dwTemp1, BYTE#3);
byTemp2 := SHR(byTemp2, BYTE#1);

---> SA0052: Unusual shift operation

See also

● % Chapter 1.7.3.1 “Configuring and Running Static Analysis” on page 3601

SA0053: Too big bitwise shift

Detects whether or not the data type width of the operand has been exceeded in the case of a bitwise shift (bit shift) of operands
Justification: If a shift operation exceeds the data type width, then a constant 0 is generated. If a rotation shift exceeds the data type width, then it is difficult to read. Therefore, the rotation value should be shortened.

Importance: High

Example

```plaintext
PROGRAM PLC_PRG
VAR
  byTemp1 : BYTE;
  wTemp2 : WORD;
  dwTemp3 : DWORD;
  lwTemp4 : LWORD;
END_VAR
byTemp1 := SHR(byTemp1, BYTE#25);
wTemp2 := SHL(wTemp2, BYTE#45);
dwTemp3 := ROR(dwTemp3, BYTE#78);
lwTemp4 := ROL(lwTemp4, BYTE#111);
--> SA0053: Too big bitwise shift
```

See also

- Chapter 1.7.3.1 “Configuring and Running Static Analysis” on page 3601

SA0054: Comparisons of REAL/LREAL for equality / inequality

Detects whether or not the comparison operators = (equality) and <> (inequality) compare the operands of type REAL or LREAL

Justification: REAL/LREAL values are implemented as floating-point numbers according to the IEEE 754 standard. This standard implies that specific, apparently simple decimal numbers cannot be represented with precision. As a result, there may be different representations as LREAL for the same decimal number.

Consider the following lines of code:

```plaintext
lr11 := 1.1;
lr33 := 3.3;
lrVar1 := lr11 + lr11;
lrVar2 := lr33 - lr11;
botest := lrVar1 = lrVar2;
```

In this case, `botest` returns FALSE, even if the variables `lrVar1` and `lrVar2` both return the monitoring value of 2.2. This is not an error of the compiler, but a property of the floating point units of all conventional processors. You can avoid this by specifying a minimum value by which the values may differ: `botest := ABS(lrVar1 - lrVar2) < 0.1;`

Exception: A comparison with 0.0 is not reported by this analysis. For the 0, there is an exact representation in the IEEE 754 standard, and therefore the comparison functions normally as expected. Therefore, for better performance, it makes sense to permit a direct comparison here.

Importance: High

PLCopen rule: CP54
Example

PROGRAM PLC_PRG
VAR
    rTest1 : REAL;
    rTest2 : REAL;
    lrTest3 : LREAL;
    lrTest4 : LREAL;
    xResult : BOOL;
END_VAR

//the following lines each will cause an SA0054:
    xResult := rTest1 = rTest1;
    xResult := rTest1 = rTest2;
    xResult := rTest1 <> rTest2;
    xResult := lrTest3 = lrTest3;
    xResult := lrTest3 = lrTest4;
    xResult := lrTest3 <> lrTest4;
//the following lines each will not cause an SA0054:
    xResult := rTest1 > rTest2;
    xResult := lrTest3 < lrTest4;

--> SA0054: Comparisons of REAL/LREAL for equality / inequality

See also
● Chapter 1.7.3.1 “Configuring and Running Static Analysis” on page 3601

SA0055: Unnecessary comparisons of unsigned operands

Detects unnecessary comparisons with unsigned operands. An unsigned data type is never less than zero. This can be used as a sign check.

Justification: A comparison detected with this check yields a constant result and is an indication of an error in the code.

Importance: High

Example

PROGRAM PLC_PRG
VAR
    byTest: BYTE;
END_VAR

WHILE byTest >= 0 DO
    byTest := byTest - 1;
END_WHILE;

--> SA0055: Unnecessary comparisons of unsigned operands

See also
● Chapter 1.7.3.1 “Configuring and Running Static Analysis” on page 3601

SA0056: Constant out of valid range

Detects literals (constants) outside of the valid range of the operator

Justification: The message is issued for an operation that returns the constant TRUE or FALSE. This is an indication of a programming error.

Importance: High
Example

PROGRAM PLC_PRG
VAR
  byTestVar: BYTE;
END_VAR

WHILE byTestVar >= 0 DO
  byTestVar := byTestVar + 1;
END_WHILE

--> SA0056: Constant out of valid range

See also

- Chapter 1.7.3.1 “Configuring and Running Static Analysis” on page 3601

SA0057: Possible loss of decimal places

Detects statements with possible loss of decimal places

Justification: A piece of code of the following type (diTemp2 := 1 rTemp1 := 
TO_REAL(diTemp2 / DINT#2)) can cause a misinterpretation. The author or reader of this line of code can assume that the division would be performed as a REAL operation, and in this case the result would be REAL#0.5. However, this is not true. It is an integer operation. The result is cast to REAL and rTemp1 gets the value REAL#0. To avoid this, use a cast to make sure that the operation is performed as a REAL operation: rTemp1 := TO_REAL(diTemp2) / REAL#2.

Importance: Medium

Example

PROGRAM PLC_PRG
VAR
  rTemp1 : REAL;
  diTemp2 : DINT;
  liTemp3 : LINT;
END_VAR

diTemp2 := diTemp2 + DINT#11;
rTemp1 := DINT_TO_REAL(diTemp2 / DINT#3); // SA0057
rTemp1 := DINT_TO_REAL(diTemp2) / REAL#3.0;
liTemp3 := liTemp3 + LINT#13;
rTemp1 := LINT_TO_REAL(liTemp3 / LINT#7); // SA0057
rTemp1 := LINT_TO_REAL(liTemp3) / REAL#7.0;

--> SA0057: Possible loss of decimal places

See also

- Chapter 1.7.3.1 “Configuring and Running Static Analysis” on page 3601

SA0058: Operations on enumeration variables

Detects operations on variables of the enumeration data type Assignments are permitted.

Justification: Enumerations should not be used as ordinary integer values. You can also define an alias data type or use a subrange type.

Importance: Medium
Exception: If an enumeration is tagged with the pragma \{attribute 'strict'\}, then the compiler already reports this kind of operation.

If an enumeration is declared as a flag by the pragma \{attribute 'flags'\}, then an error is not issued for \texttt{AND}, \texttt{OR}, \texttt{NOT}, or \texttt{XOR} operations.

**Example**

```plaintext
TYPE My_Enum :
 (   red := 1, blue := 2, green := 3, black := 4
 );
END_TYPE

PROGRAM PLC_PRG
VAR
   iTemp1 : INT;
   abc : My_Enum;
END_VAR

iTemp1 := iTemp1 + INT#1;
abc := My_Enum.red; // OK
iTemp1 := My_Enum.black / My_Enum.blue; // SA0058
iTemp1 := My_Enum.green / My_Enum.red; // SA0058

--> SA0058: Operations on enumeration variables
```

**Example with a pragma \{attribute 'flags'\}**

```plaintext
{attribute 'flags'} // declaring the enumeration as a "flag"
TYPE Flags :
 (   Unknown := 16#00000001,
      Stopped := 16#00000002,
      Running := 16#00000004
 ) DWORD;
END_TYPE

PROGRAM PLC_PRG
VAR
   iTemp1 : INT;
   abc : Flags;
   batate : BYTE;
   dwFlags : DWORD;
   dwState : DWORD;
END_VAR

// OK for the following
IF (dwFlags AND Flags.Unknown) <> DWORD#0 THEN
   dwState := dwState AND Flags.Unknown;
ELSIF (dwFlags OR Flags.Stopped) <> DWORD#0 THEN
   dwState := dwState OR Flags.Running;
END_IF
```

See also

- ≫ Chapter 1.7.3.1 “Configuring and Running Static Analysis” on page 3601
SA0059: Comparison operations always returning TRUE or FALSE

Detects comparisons with literals that always have the result TRUE or FALSE, and can already be processed during the compile.

Justification: An operation that consistently yields TRUE or FALSE is an indication of a programming error.

Importance: High

Example

PROGRAM PLC_PRG
VAR
  byTemp1 : BYTE;
END_VAR

WHILE byTemp1 <= 255 DO
  byTemp1 := byTemp1 + 1;
END_WHILE;

--> SA0059: Relational operator '<=' always evaluates 'TRUE'

See also

● Chapter 1.7.3.1 “Configuring and Running Static Analysis” on page 3601

SA0060: Zero used as invalid operand

Detects operations where an operand with the value "0" causes an invalid or a nonsense operation.

Justification: This kind of expression could be an indication of a programming error. In any case, it unnecessarily wastes runtime.

Importance: Medium

Example

PROGRAM PLC_PRG
VAR
  byTemp1 : BYTE;
  wTemp2 : WORD;
  dwTemp3 : DWORD;
END_VAR

byTemp1 := byTemp1 + 0;
wTemp2 := wTemp2 - WORD#0;
dwTemp3 := dwTemp3 * DWORD#0;

--> SA0060: Zero used as invalid operand

See also

● Chapter 1.7.3.1 “Configuring and Running Static Analysis” on page 3601

SA0061: Unusual operation on pointer

Detects operations one variables of type POINTER TO which are not = (equality), <> (inequality), + (addition), or ADR.
In CODESYS, pointer arithmetic is generally permitted and can also be used appropriately. Therefore, the addition of a pointer with an integer value is considered a common operation on pointers. This makes it possible to use a pointer to process an array of variable length. All other (unusual) operations with pointers are reported with SA0061.

Importance: High
PLCopen rule: E2 / E3

**Example**

```plaintext
PROGRAM PLC_PRG
VAR
  piTemp : POINTER TO INT;
  iTemp : INT;
END_VAR

iTemp := iTemp + INT#1;
piTemp := ADR(iTemp);
piTemp := piTemp * DWORD#5; // SA0061
piTemp := piTemp / DWORD#2; // SA0061
piTemp := piTemp MOD DWORD#3; // SA0061
piTemp := piTemp + DWORD#1;
piTemp := piTemp - DWORD#1; // SA0061

--> SA0061: Unusual operation on pointer
```

See also

- Chapter 1.7.3.1 “Configuring and Running Static Analysis” on page 3601

**SA0062: Uses of TRUE or FALSE in expressions**

Detects the use of the literals TRUE or FALSE in expressions

Justification: This kind of expression is obviously unnecessary and may indicate an error. In any case, the expression unnecessarily affects the runtime.

Importance: Medium

**Example**

```plaintext
PROGRAM PLC_PRG
VAR
  xTemp1, xTemp2 : BOOL;
END_VAR
xTemp1 := xTemp1 AND NOT TRUE;
xTemp2 := xTemp1 OR TRUE;
xTemp2 := xTemp1 OR NOT FALSE;
xTemp2 := xTemp1 AND FALSE;

--> Uses of TRUE or FALSE in expressions
```

See also

- Chapter 1.7.3.1 “Configuring and Running Static Analysis” on page 3601

**SA0063: Possibly not 16-bit-compatible operations**

Detects 16-bit operations with temporary results. Background: On 16-bit systems, 32-bit temporary results can be truncated. Example: (int+10) can exceed 16 bits.

Justification: In the very rare case that you have to write code which should run on a 16-bit processor as well as on a 32-bit processor, this message should help to prevent any problems.
Importance: Low

Example

PROGRAM PLC_PRG
VAR
    iVar : INT;
END_VAR
iVar := (iVar + 10) / 2;

--> SA0063: Compatibility for 16 Bit - Possible truncated intermediate result

See also
- Chapter 1.7.3.1 “Configuring and Running Static Analysis” on page 3601

SA0064: Addition of pointer

Detects the addition of pointers

Justification: In CODESYS, pointer arithmetic is generally permitted and can also be used appropriately. However, it is also a source of errors. Therefore, programming rules exist that generally prohibit pointer arithmetic. This test can check such a requirement.

Importance: Medium

Example

PROGRAM PLC_PRG
VAR
    iTest:INT;
    ariTest:ARRAY[0..10] OF INT;
    {attribute 'analysis':='-111'}
    piTest:POINTER TO INT;
    i:INT;
END_VAR

piTest := ADR(ariTest[0]);            // OK
piTest^:= 0;
piTest := ADR(ariTest) + SIZEOF(INT); // SA0064
piTest^:= 1;
piTest := ADR(ariTest) + 6;           // SA0064
piTest^:= 3;
piTest := ADR(ariTest[10]);
FOR i:=0 TO 10 DO
    piTest^ := i;
    piTest := piTest + 2;               // SA0064
END_FOR

--> SA0064: Addition of pointer

See also
- Chapter 1.7.3.1 “Configuring and Running Static Analysis” on page 3601

SA0065: Incorrect pointer addition to base size

Detects pointer additions for which the value to be added does not match the base size of the pointer. Only literals of the base size can be added. Also multiplication products of the base size cannot be added.
Justification: In CODESYS (in contrast to C and C++), when adding a pointer with an integer value, only this integer value is added as the number of bytes, and not the integer value multiplied by the base size. Example in ST:

```plaintext
pINT := ADR(array_of_int[0])
pINT := pINT + 2 ;  // In CODESYS, pINT then points to array_of_int[1]
```

This code would function differently in C:

```plaintext
short* pShort
pShort = &(array_of_short[0])
pShort = pShort + 2;  // In C, pShort then points to array_of_short[2]
```

Therefore, in CODESYS, you should always add a multiple of the base size of the pointer to a pointer. Otherwise, the pointer may point to non-aligned memory which (depending on the processor) can lead to an alignment exception when accessing it.

Importance: High

### Example

```plaintext
VAR
  pudiTest:POINTER TO UDINT;
  udiTest:UDINT;
  prTest:POINTER TO REAL;
  rTest:REAL;
END_VAR

pudiTest := ADR(udiTest) + 4;                  // OK
pudiTest := ADR(udiTest) + ( 2 + 2 );          // OK
pudiTest := ADR(udiTest) + SIZEOF(UDINT);      // OK
pudiTest := ADR(udiTest) + 3;                   // SA0065
pudiTest := ADR(udiTest) + 2*SIZEOF(UDINT);    // SA0065
pudiTest := ADR(udiTest) + ( 3 + 2 );          // SA0065
prTest := ADR(rTest);
prTest := prTest + 4;                          // OK
prTest := prTest + ( 2 + 2 );                  // OK
prTest := prTest + SIZEOF(REAL);               // OK
prTest := prTest + 1;                          // SA0065
prTest := prTest + 2;                          // SA0065
prTest := prTest + 3;                          // SA0065
prTest := prTest + ( SIZEOF(REAL) - 1 );       // SA0065
prTest := prTest + ( 1 + 4 );                  // SA0065
```

```plaintext
 --> SA0065: Incorrect pointer addition to base size
```

See also

- “Chapter 1.7.3.1 “Configuring and Running Static Analysis” on page 3601

### SA0066: Uses of temporary results

Detects the use of temporary results in statements with a data type that is less than the registry size. The implicit cast in this case may lead to unwanted results.

Justification: For performance reasons, CODESYS performs operations on the register width of the processor. Intermediate results are not truncated. This can lead to misinterpretations as in the following case:

```plaintext
usintTest := 0; xError := usintTest - 1 <> 255;
```

In CODESYS, `xError` is TRUE in this case because the operation `usintTest - 1` is typically executed as a 32-bit operation and the result is not cast to the byte size. Then the value `16#ffffffff` (not equal to 255) is located in the registry. To avoid this, you have to cast the intermediate result explicitly:

```plaintext
xError := TO_USINT(usintTest - 1) <> 255;
```
NOTICE!

If this message is activated, then many less problematic locations in the code will be reported. Although a problem can only occur when the operation produces an overflow or underflow in the data type, the static analysis cannot differentiate between the individual locations.

If you include an explicit typecast in all reported locations, then the code will be much slower and less readable.

Importance: Low

Example

PROGRAM PLC_PRG
VAR
  byTest:BYTE;
  liTest:LINT;
  xError:BOOL;
END_VAR

//type size smaller than register size;
byTest := 0;
IF (byTest - 1) <> 255 THEN //use of temporary result + implicit casting -> SA0066
  xError := TRUE;
ELSE
  xError := FALSE;
END_IF

//type size equal to or bigger than register size;
liTest := 0;
IF (liTest - 1) <> -1 THEN // use of temporary result and no implicit casting -> OK
  xError := TRUE;
ELSE
  xError := FALSE;
END_IF

--> SA0066: Use of temporary result: (byTest - USINT #1)

See also
-
  Chapter 1.7.3.1 “Configuring and Running Static Analysis” on page 3601

Rules for Statements

1.7.3.2.40.1  SA0072: Invalid uses of counter variable........................................ 3659
1.7.3.2.40.2  SA0073: Uses of inadequate counter variable.............................. 3659
1.7.3.2.40.3  SA0080: Loop index variable for array index exeeds array range.............................. 3659
1.7.3.2.40.4  SA0081: Upper border is not a constant........................................ 3660
1.7.3.2.40.5  SA0075: Missing ELSE................................................ 3661
1.7.3.2.40.6  SA0076: Missing enumeration constant................................... 3662
1.7.3.2.40.7  SA0077: Type mismatches with CASE expression....................... 3663
1.7.3.2.40.8  SA0078: Missing CASE branches......................................... 3663
1.7.3.2.40.9  SA0090: Return statement before end of function.................... 3664
SA0072: Invalid uses of counter variable

Detects the use of a counter variable in a **FOR** loop

Justification: Manipulation of the counter variable in a **FOR** loop can easily result in an infinite loop. To prevent the execution of the loop for specific values of the counter variable, use **CONTINUE** or simply an **IF**.

Importance: High

PLCopen rule: L12

**Example**

```plaintext
PROGRAM PLC_PRG
VAR_TEMP
  iIndex : INT;
END_VAR
FOR iIndex := INT#0 TO INT#20 BY INT#1 DO
  iIndex := iIndex - INT#1;
END_FOR
--> SA0072: Invalid use of counter variable 'iIndex'
```

See also
- Chapter 1.7.3.1 “Configuring and Running Static Analysis” on page 3601

SA0073: Uses of inadequate counter variable

Detects the use of non-temporary variables in **FOR** loops.

Justification: This is a performance warning. A counter variable is always initialized each time a POU is called. You can create this variable as a temporary variable (**VAR_TEMP**). Access to it may be faster and the variable does not take up any permanent memory.

Importance: Medium

PLCopen rule: CP21 / L13

**Example**

```plaintext
PROGRAM PLC_PRG
VAR
  nIndex : INT;
  iVar : INT;
END_VAR
FOR nIndex := INT#0 TO INT#20 BY INT#1 DO
  iVar := iVar + nIndex;
END_FOR
--> SA0073: Inadequate counter variable
```

See also
- Chapter 1.7.3.1 “Configuring and Running Static Analysis” on page 3601

SA0080: Loop index variable for array index exceeds array range

Detects the **FOR** statements where the index variable is used to access an array index and exceeds the range of the array index.
Justification: Arrays are typically processed in **FOR** loops. The start and end value of the counter variable should typically match (or at least not exceed) the upper and lower bounds of the array. A typical cause of error is detected here when array bounds are changed and constants are not carefully used; or when a different value is used accidentally in the **FOR** loop than in the array declaration.

Importance: High

**Example**

```plaintext
PROGRAM PLC_PRG
VAR
  iIndex1, iIndex2, iIndex3 : INT;
  arWord : ARRAY[1..100] OF WORD;
  arararINT : ARRAY[1..9, 1..9, 1..9] OF INT;
  arUSINT : ARRAY[0..99] OF USINT;
END_VAR

//1 violation of the rule (lower range is exceeded): SA0080
FOR iIndex1 := INT#0 TO INT#100 DO
  arWord[iIndex1] := INT_TO_WORD(iIndex1);
END_FOR

//6 violations (lower and upper range is exceeded for each array dimension): 3SA0080
FOR iIndex2 := INT#0 TO INT#10 DO
  arararINT[iIndex2, iIndex2, iIndex2] := iIndex2;
END_FOR

//1 violation (upper range is exceeded by the end result of the index), previous expressions on index are not evaluated -> OK
FOR iIndex3 := INT#0 TO INT#50 DO
  arUSINT[iIndex3 * INT#2] := INT_TO_USINT(iIndex3);
END_FOR

--> SA0080: Loop index range of 'Index1' exceeds array range
--> SA0080: Loop index range of 'Index2' exceeds array range
```

See also

- Chapter 1.7.3.1 “Configuring and Running Static Analysis” on page 3601

**SA0081: Upper border is not a constant**

Detects the **FOR** statements where the upper bound is not defined with a constant value

Justification: If the upper bound of a loop is a variable value, then it is no longer possible to see how often a loop is executed. This can result in serious problems at runtime. The worst case is an infinite loop.

Importance: High
Example

PROGRAM PLC_PRG
VAR
  i:INT;
  iBorder1: INT := 10;
  iBorder2: INT := 10;
  iCounter: INT;
END_VAR
VAR CONSTANT
  ciBorder:INT := 10;
END_VAR
FOR i:=0 TO 10 DO    //OK
  iCounter := i;
END_FOR
FOR i:=0 TO ciBorder DO // OK
  iCounter := i;
END_FOR
FOR i:=0 TO iBorder1 DO    // SA0081
  iCounter := i;
END_FOR
FOR i:=0 TO iBorder2 DO    // SA0081
  iCounter := i;
  IF iCounter = 10 THEN
    iBorder2 := 50;
  END_IF
END_FOR
--> SA0081: Upper border of a for loop must be a constant value

See also

- Chapter 1.7.3.1 “Configuring and Running Static Analysis” on page 3601

SA0075: Missing ELSE

Detects CASE statements without an ELSE branch

Justification: Defensive programming requires the inclusion of an ELSE branch in every CASE statement. If there is nothing to do in the ELSE branch, then include a comment to indicate this. It is then clear to the reader of the code that the case was not simply forgotten.

Importance: Low

PLCopen rule: L17
Example

PROGRAM PLC_PRG
VAR
  iVar : INT;
  xTemp : BOOL;
END_VAR

iVar := iVar + INT#1;
CASE iVar OF
  INT#1:
  xTemp := FALSE;
  INT#2:
  xTemp := TRUE;
END_CASE

--> SA0075: Missing ELSE in CASE statement

See also

● § Chapter 1.7.3.1 “Configuring and Running Static Analysis” on page 3601

SA0076: Missing enumeration constant

Detects whether or not an enumeration variable is used as a condition and not all enumeration values are treated as CASE branches.

Justification: Defensive programming requires the processing of all possible values of an enumeration. If an action is not required for a particular enumeration value, then you should add a comment to indicate this explicitly. It is then clear to the reader of the code that the value was not simply forgotten.

Importance: Low

Example

TYPE My_Enum :
  (red := 1, blue := 2, green := 3, black := 4);
END_TYPE

PROGRAM PLC_PRG
VAR
  iVar : My_Enum;
  xTemp : BOOL;
END_VAR

iVar := My_Enum.black;
CASE iVar OF
  My_Enum.red:
  xTemp := FALSE;
  My_Enum.blue, My_Enum.green:
  xTemp := TRUE;
  ELSE
  xTemp := NOT xTemp;
END_CASE

--> SA0076: Missing enumeration constant 'black' in CASE statement

See also

● § Chapter 1.7.3.1 “Configuring and Running Static Analysis” on page 3601
SA0077: Type mismatches with CASE expression

Detects code locations where the data type of a condition does not match that of the CASE branch.

Justification: If the data types between the CASE variable and the CASE itself do not match, then this could indicate an error.

Importance: Low

Example

```
TYPE My_Enum :
   (eins := 1, zwei := 2, drei := 3, vier := 4);
END_TYPE

PROGRAM PLC_PRG
VAR
diVar : DINT;
xTemp : BOOL;
END_VAR
diVar := diVar + DINT#1;
CASE diVar OF
   DINT#1:
      xTemp := FALSE;
      My_Enum.zwei, DINT#3: //SA0077
      xTemp := TRUE;
      ELSE
      xTemp := NOT xTemp;
   END_CASE
```

See also

- *Chapter 1.7.3.1 “Configuring and Running Static Analysis” on page 3601*

SA0078: Missing CASE branches

Detects CASE statements without CASE branches and only one ELSE statement.

Justification: A CASE statement without cases wastes execution time and it is difficult to read.

Importance: Medium

Example

```
PROGRAM PLC_PRG
VAR
   iVar : INT;
xTemp : BOOL;
END_VAR
iVar := iVar + INT#1;
//in the following the case descriptions are missing:
CASE iVar OF
   ELSE
      xTemp := NOT xTemp;
END_CASE
```

--> SA0077: Type mismatches with CASE expression

--> SA0078: CASE-Missing CASE branches
SA0090: Return statement before end of function

Detects whether or not the `RETURN` statement is not the last statement in a function, method, property, or program.

Justification: A `RETURN` in the code results in worse maintainability, testability, and readability of the code. A `RETURN` in the code is easily overlooked. Before each `RETURN`, it is often forgotten to insert code that should always be executed when exiting a function.

Importance: Medium

PLCopen rule: CP14

Example

```plaintext
FUNCTION FUN : DINT
VAR_INPUT
  bTest : BOOL;
END_VAR

IF bTest THEN
  RETURN;
END_IF
FUN := 99;

--> SA0090: Return statement before end of function
```

See also

● § Chapter 1.7.3.1 “Configuring and Running Static Analysis” on page 3601

SA0095: Assignments in conditions

Detects assignments in conditions of `IF`, `CASE`, or `REPEAT` constructs

Justification: An assignment (:=) and a comparison (=) can easily be mistaken. As a result, an assignment in a condition can easily be unintentional, and it is therefore reported. This can also confuse the reader of the code.

Importance: High
Example

```plaintext
PROGRAM PLC_PRG
VAR
  iCond1:INT := INT#1;
iCond2:INT := INT#2;
xCond:BOOL := FALSE;
iVar : INT;
END_VAR

IF INT_TO_BOOL(iCond1 := iCond2) THEN                      // SA0095
  iCond1 := INT#1;
iCond2 := INT#2;
ELSIF (iCond1 := 11) = 11 THEN                             // SA0095
  iCond1 := INT#1;
iCond2 := INT#2;
END_IF

IF xCond := TRUE THEN                                      // SA0095
  xCond := FALSE;
END_IF

IF (xCond := FALSE) OR (iCond1 := iCond2) = 12 THEN        // SA0095
  xCond := FALSE;
iCond1 := INT#1;
iCond2 := INT#2;
END_IF

IF (iVar := iVar + 1) = 120 THEN                           // SA0095 (can be valid, but is not reparable very well
  iVar := 0;
END_IF

WHILE (xCond = TRUE) OR (iCond1 := iCond2) = 12 DO         // SA0095
  xCond := FALSE;
END_WHILE

// Error: assignment in repeat loop
REPEAT
  xCond := FALSE;
UNTIL
  (xCond = TRUE) OR (iCond1 := iCond2) = 12              // SA0095
END_REPEAT

--> SA0095: Assignment in condition: '...'
```

See also

- § Chapter 1.7.3.1 “Configuring and Running Static Analysis” on page 3601

SA0100: Variables greater than <n> bytes...

Detects variables that use more than n bytes, where n is defined by the current configuration.
Default value: 1024 bytes. The value can be changed by double-clicking the line.

Justification: Some programming guidelines specify a maximum size for a single variable. This can be checked with this.

Importance: Low
**Example**

```plaintext
PROGRAM PLC_PRG
VAR
   aobyTest : ARRAY [0..1024] OF BYTE;
END_VAR

aobyTest[INT#0] := aobyTest[INT#0] + BYTE#1;
--> SA0100: Variable 'aobyTest' greater 1024 bytes
```

See also

- "Chapter 1.7.3.1 “Configuring and Running Static Analysis” on page 3601"

**SA0101: Names with invalid length**

Detects names with invalid lengths. In the “Project Settings”, double-click the rule entry to open a dialog where you can define the length of the name and define any exception.

Justification: Some programming guidelines specify a minimum length for variable names. This analysis can be used to check compliance.

Importance: Low

PLCopen rule: N6

```plaintext
Example

PROGRAM PLC1               // SA0101
VAR
   iVar1: INT;            // SA0101
END_VAR

--> SA0101: Incorrect length of name 'PLC1'
```

See also

- "Chapter 1.7.3.1 “Configuring and Running Static Analysis” on page 3601"

**SA0102: Access to program/fb variables from the outside**

Detects external access to local variables of programs or function blocks

Justification: CODESYS permits external read access to local variables of programs or function blocks. This contradicts the principle of data encapsulation (hiding data) and does not comply with the IEC 61131-3 standard.

Importance: Medium
Example

PROGRAM PLC_PRG
VAR
  iCounter : INT;
  afb_Instance : AFB;
  bfb_Instance : BFB;
END_VAR
iCounter := A_PRG.iLocal;                // SA0102
iCounter := bfb_Instance.iLocal;         // SA0102
A_PRG();

FUNCTION_BLOCK AFB
VAR_INPUT
END_VAR
VAR_OUTPUT
END_VAR
VAR
  iLocal: INT;
END_VAR
METHOD METH : INT
VAR_INPUT
END_VAR
iLocal := iLocal + 1;

FUNCTION_BLOCK BFB EXTENDS AFB
VAR_INPUT
END_VAR
VAR_OUTPUT
END_VAR
VAR
END_VAR
METHOD METH : INT
VAR_INPUT
END_VAR
iLocal := iLocal + 1;

PROGRAM A_PRG
VAR
  iLocal: INT;
END_VAR
iLocal := iLocal + 1;

--> SA0102: Access to program/fb variable 'iLocal' from the outside

See also
● “Chapter 1.7.3.1 “Configuring and Running Static Analysis” on page 3601

SA0103: Concurrent access on not atomic data

Detects whether or not non-atomic variables (for example, with data type STRING, WSTRING, ARRAY, STRUCT, FB instances, 64-bit data types) are used in more than one task

Justification: When there is no synchronization during access, inconsistent values can be read when reading in one task and writing in another task at the same time.

Importance: Medium
For some data types, especially 64-bit integers, it depends on the platform whether or not access is atomic. Static analysis reports a problem only when the controller does not support atomic access to 64-bit integer data types.

This rule does not apply in the following cases:

- If the target system has a floating point unit (FPU), then access of multiple tasks to LREAL variables is not detected.
- If the target system is a 64-bit processor or the corresponding target setting is set for the target device, then the rule does not apply to 64-bit data types.

**Example**

The project contains both programs, PRG1 and PRG2: The program PRG1 is called by the task MainTask_1. The program PRG2 is called by the task MainTask_2.

```
GVL
VAR_GLOBAL
   lrTest : LREAL;   // Since the target system has an FPU, SA0103 does apply.
   lint1 : LINT;
   sTest : STRING;   // SA0103
   wsTest : WSTRING; // SA0103
END_VAR

PROGRAM PRG1
GVL.lrTest := 5.0;
GVL.sTest := 'welt';
GVL.wsTest := "welt";
GVL.lint1 := 99;

PROGRAM PRG2
GVL.lrTest := 5.0;
GVL.sTest := 'hallo';
GVL.wsTest := "hallo";
GVL.lint1 := 88;

--> SA0103: Concurrent access on not atomic data 'sTest'
--> SA0103: Concurrent access on not atomic data 'wsTest'
```

See also

- § Chapter 1.7.3.1 "Configuring and Running Static Analysis" on page 3601

**SA0105: Multiple instance calls**

Detects the instances of function blocks that are called multiple times. To do this, the function blocks have to be marked with the pragma `attribute 'analysis:report-multiple-instance-calls'`.

Justification: Some function blocks are designed in such a way that they can be called only one time in the cycle. This test checks whether or not a call is made in multiple locations.

Importance: Low

PLCopen rule: CP16 / CP20
Example

```plaintext
// {attribute 'analysis:report-multiple-instance-calls'} Deactivated
FUNCTION_BLOCK FB_DoA
  VAR_INPUT
  END_VAR
  VAR_OUTPUT
  END_VAR
  VAR
    iA : INT;
  END_VAR
  iA := iA + 1;

{attribute 'analysis:report-multiple-instance-calls'}
FUNCTION_BLOCK FB_DoB
  VAR_INPUT
  END_VAR
  VAR_OUTPUT
  END_VAR
  VAR
    iB : INT;
  END_VAR
  iB := iB + 1;

PROGRAM PLC_PRG
  VAR
    fbA : FB_DoA;
    fbB : FB_DoB;
    ND_VAR
    fbA();
    fbB();    // SA0105
    fbA();
    fbB();    // SA0105
    --> SA0105: Instance 'fbB' called more than once
```

See also

- Chapter 1.7.3.3.1.5 “Attribute 'analysis:report-multiple-instance-calls’” on page 3614

SA0106: Virtual method calls in FB_INIT

Detects method calls in the FB_Init method of a base function block, which are overwritten by a function block derived from a base function block.

Justification: In these cases, it could be that the variables in the overwritten methods are not initialized in the base FB.

Importance: High
The function block FB_A includes the methods `FB_Init` and `Meth_MyInit`. FB_Init calls Meth_MyInit for initialization. The function block FB_B is derived from FB_A. PLC_PRG calls FB_B's Meth_MyInit and therefore uses its mbMyDintB variable before it has been initialized. FB_B's Meth_MyInit overwrites FB_A's Meth_MyInit.

```plaintext
FUNCTION_BLOCK FB_A
  VAR
    mbMyDintA : DINT;
  END_VAR
FUNCTION_BLOCK FB_B EXTENDS FB_A
  VAR
    mbMyDintB : DINT;
  END_VAR
  METHOD FB_Init : BOOL
    VAR_INPUT
    bInitRetains:BOOL;
    bInCopyCode:BOOL;
    END_VAR
    mbMyDintA := 123;
    diDummy := Meth_MyInit();
  METHOD Meth_MyInit : DINT
    VAR_INPUT
    END_VAR
    mbMyDintB := 123; // access to member of FB_B
  PROGRAM PLC_PRG
    VAR
      g_BInst : FB_B;
      xVar : BOOL;
    END_VAR
    xVar := g_BInst.fb_init(TRUE, TRUE);
    // this instruction causes the following order of initializations:
    // FB_A.fb_init
    // FB_B.Meth_MyInit
    // FB_B.fb_init
    // FB_B.Meth_MyInit
    --> SA0106: Virtual method call 'Meth_MyInit' in FB_INIT
```

See also

- "Chapter 1.7.3.3.1.5 “Attribute 'analysis:report-multiple-instance-calls'” on page 3614"

### SA0107: Missing formal parameters

Detects whether or not formal parameters are missing.

Justification: Code becomes more readable when formal parameters are specified in the call.

Importance: Low
FUNCTION FUNA : BOOL
VAR_INPUT
    bDo: BOOL;
    bInit: BOOL;
    bManual : BOOL;
END_VAR
VAR
    iInit: INT;
    iLocal: INT;
    iManual: INT;
END_VAR
IF bInit = TRUE THEN
    iInit := iInit + 1;
END_IF
IF bDo = TRUE THEN
    iLocal := iLocal + 1;
END_IF
IF bManual = TRUE THEN
    iManual:= iManual + 1;
END_IF
FUNA := TRUE;
PROGRAM PLC_PRG
VAR
END_VAR
FUNA(bInit := TRUE, bDo := TRUE, bManual := FALSE); // OK
FUNA(TRUE, TRUE, bManual:= FALSE);                      // SA0107
--> SA0107: Missing formal parameter for input 'TRUE'

See also
● & Chapter 1.7.3.3.1.5 “Attribute 'analysis:report-multiple-instance-calls” on page 3614

Checking Strict IEC Rules

1.7.3.3.2.49.1 SA0111: Pointer variables................................................................. 3672
1.7.3.3.2.49.2 SA0112: Reference variables................................................................. 3672
1.7.3.3.2.49.3 SA0113: Variables with data type WSTRING........................................ 3672
1.7.3.3.2.49.4 SA0114: Variables with data type LTIME............................................. 3673
1.7.3.3.2.49.5 SA0115: Variables with data type UNION............................................. 3673
1.7.3.3.2.49.6 SA0117: Variables with data type BIT.................................................. 3673
1.7.3.3.2.49.7 SA0119:Object-oriented features..................................................... 3674
1.7.3.3.2.49.8 SA0120: Program calls................................................................. 3674
1.7.3.3.2.49.9 SA0121: Missing VAR_EXTERNAL declarations..................................... 3675
1.7.3.3.2.49.10 SA0122: Array index defined as expression................................. 3676
1.7.3.3.2.49.11 SA0123: Usages of INI, ADR or BITADR........................................... 3676
1.7.3.3.2.49.12 SA0147: Unusual shift operation - strict........................................ 3676
1.7.3.3.2.49.13 SA0148: Unusual bit access - strict............................................... 3677
1.7.3.3.2.49.14 SA0118: Initialisations not using constants..................................... 3678
1.7.3.3.2.49.15 SA0124: Pointer dereferences in declarations.............................. 3678
1.7.3.3.2.49.16 SA0125: References in initializations............................................ 3678
SA0111: Pointer variables

Detects variables of type `POINTER TO`

Justification: The IEC 61131-3 standard does not permit pointers.

Importance: Low

Example

```plaintext
VAR
    piTemp : POINTER TO INT;
    pbyTemp : POINTER TO BYTE;
END_VAR

--> SA0111: Data type POINTER not allowed
```

See also

- Chapter 1.7.3.3.1.5 “Attribute 'analysis:report-multiple-instance-calls’” on page 3614

SA0112: Reference variables

Detects variables of type `REFERENCE TO`

Justification: The IEC 61131-3 standard does not permit references.

Importance: Low

Example

```plaintext
VAR
    ref_int : REFERENCE TO INT;
    ref_dw : REFERENCE TO DWORD;
END_VAR

--> Data type REFERENCE not allowed
```

See also

- Chapter 1.7.3.3.1.5 “Attribute 'analysis:report-multiple-instance-calls’” on page 3614

SA0113: Variables with data type `WSTRING`

Detects variables of type `WSTRING`

Justification: Not all systems support `WSTRING`. The code is more easily portable without `WSTRING`.

Example

```plaintext
VAR
    wstrTemp : WSTRING;
END_VAR

--> SA0113: Data type WSTRING not allowed
```

See also

- Chapter 1.7.3.3.1.5 “Attribute 'analysis:report-multiple-instance-calls’” on page 3614
SA0114: Variables with data type LTIME

Detects variables of type LTIME.

Justification: Not all systems support LTIME. The code is more easily portable without LTIME.

Importance: Low

Example

```
VAR
  ltVar : LTIME; // SA0114
END_VAR
```

--> SA0114: Data type LTIME not allowed

See also

- Chapter 1.7.3.3.1.5 “Attribute ‘analysis:report-multiple-instance-calls’” on page 3614

SA0115: Variables with data type UNION

Detects declarations of a UNION data type and variable declarations of the UNION type.

Justification: The IEC 61131-3 standard does not include unions. The code is more easily portable without unions.

Importance: Low

Example

```
TYPE u1: UNION
  lrTemp : LREAL;
  liTemp : LINT;
END_UNION
END_TYPE

PROGRAM PLC_PRG
VAR
  uVar: u1;
END_VAR
```

--> SA0115: Unions not allowed

See also

- Chapter 1.7.3.3.1.5 “Attribute ‘analysis:report-multiple-instance-calls’” on page 3614

SA0117: Variables with data type BIT

Detects variable declarations of data type BIT (possible within structure definitions).

Justification: The IEC 61131-3 standard does not include the data type BIT. The code is more easily portable without BIT.

Importance: Low
Example

```plaintext
TYPE Struct1 :
  STRUCT
    bitVar : BIT;
    iVar : INT;
    bVar : BOOL;
  END_STRUCT
END_TYPE

--> SA0117: Variables with data type BIT
```

See also

- Chapter 1.7.3.3.1.5 “Attribute 'analysis:report-multiple-instance-calls’” on page 3614

SA0119: Object-oriented features

Detects the use of object-oriented features, such as function block declarations with `EXTENDS` and `IMPLEMENTS`, or property and interface declarations. This rule is useful when you write code that is intended to be ported to other IEC 61131-3-compliant systems.

Justification: Not all systems support object-oriented programming. The code is more easily portable without object-orientation.

Importance: Low

Example

```plaintext
//Function block extended by another and implementing an interface:
FUNCTION_BLOCK POU EXTENDS CTD IMPLEMENTS ITF        //SA0119
...

// Declaration parts of property methods assigned to a function block:
POU.Prop.Get        //SA0119
POU.Prop.Set        //SA0119

--> SA0119: Object-oriented features not allowed
```

See also

- Chapter 1.7.3.3.1.5 “Attribute 'analysis:report-multiple-instance-calls’” on page 3614

SA0120: Program calls

Detects program calls

Justification: According to the IEC 61131-3 standard, programs can be called in the task configuration only. The code is more easily portable when you do not call programs from other locations.

Importance: Low
Example

PROGRAM prog_control
VAR
END_VAR

PROGRAM PLC_PRG
VAR
END_VAR

prog_control();
--> SA0120: Program call to 'prg_control' not allowed

See also

- Chapter 1.7.3.3.1.5 “Attribute 'analysis:report-multiple-instance-calls'” on page 3614

SA0121: Missing VAR_EXTERNAL declarations

Detects the use of a global variable in function blocks without them being declared there as “VAR_EXTERNAL”

Justification: According to the IEC 61131-3 standard, access to global variables is permitted only by an explicit import by means of a VAR_EXTERNAL declaration.

Importance: Low
PLCopen rule: CP18

Example

VAR_GLOBAL
  iGlob1:INT;
END_VAR

PROGRAM PLC_PRG
VAR
ivar:INT;
END_VAR
ivar:=iGlob1;       // SA0121

--> SA0121: EXTERNAL declaration required for variable 'iGlob1'

Example: Avoid error

VAR_GLOBAL
  iGlob1:INT;
END_VAR

PROGRAM PLC_PRG
VAR
ivar:INT;
END_VAR
VAR_EXTERNAL
  iGlob1:INT;
END_VAR
ivar:=iGlob1;       // OK
SA0122: Array index defined as expression

Detects the use of expressions in the declaration of array indexes

Justification: Not all systems permit expressions as array limits.

Importance: Low

Example

```plaintext
PROGRAM PLC_PRG
VAR CONSTANT
c_iValue : INT := INT#15;
END_VAR
VAR
arr: ARRAY[0..c_iValue + 1] OF INT;
END_VAR

--> SA0122: Only constants allowed for array definition 'arr'
```

SA0123: Usages of INI, ADR or BITADR

Detects the use of the CODESYS-specific operators INI, ADR, and BITADR

Justification: CODESYS-specific operators prevent the portability of code.

Importance: Low

Example

```plaintext
PROGRAM PLC_PRG
VAR
  uiTemp: UINT;
  TempVarInFUNC: DWORD;
END_VAR

TempVarInFUNC := ADR(uiTemp);        //SA0123

--> SA0123: Operator 'ADR' not allowed
```

SA0147: Unusual shift operation - strict

Detects bit shift operations that are not made to bit field data types (BYTE, WORD, DWORD, LWORD)

Justification: The IEC 61131-3 standard permits bit access only to bit field data types. However, the CODESYS compiler also permits bit shift operations with unsigned data types.

Importance: Low
See also the strict rule SA0018.

**Example**

```plaintext
PROGRAM PLC_PRG
VAR
  in_byte : BYTE := 16#45;   // 2#01000101
  in_word : WORD := 16#0045; // 2#0000000001000101
  in_uint : UINT;
  in_dint : DINT;
  erg_byte : BYTE;
  erg_word : WORD;
  erg_uint : UINT;
  erg_dint : DINT;
  n: BYTE := 2;
END_VAR

  erg_byte := SHL(in_byte,n);  // no error because BYTE is a bit field
  erg_word := SHL(in_word,n);  // no error because WORD is a bit field
  erg_uint := SHL(in_uint,n);  // SA0147
  erg_dint := SHL(in_dint,n);  // SA0147

  --> SA0147: Unusual shift operation - strict
```

See also

- % Chapter 1.7.3.3.1.5 “Attribute ‘analysis:report-multiple-instance-calls’” on page 3614
- % Chapter 1.7.3.3.2.17 “SA0018: Unusual bit access” on page 3626

**SA0148: Unusual bit access - strict**

Detects bit access that is not made to bit field data types (BYTE, WORD, DWORD, and LWORD). The IEC 61131-3 standard permits only bit access to bit field data types. However, the CODESYS compiler also permits bit access to unsigned data types.

**Example**

```plaintext
PROGRAM PLC_PRG
VAR
  iTemp1 : INT;
  diTemp3 : DINT;
  uliTemp4 : ULINT;
  siTemp5 : SINT;
  usiTemp6 : USINT;
  byTemp2 : BYTE;
END_VAR

  iTemp1.3 := TRUE;     // SA0148
  diTemp3.4 := TRUE;    // SA0148
  uliTemp4.18 := FALSE; // SA0148
  siTemp5.2 := FALSE;   // SA0148
  usiTemp6.3 := TRUE;   // SA0148
  byTemp2.5 := FALSE;   // no error because BYTE is a bit field

  --> SA0148: Unusual bit access - strict
```
SA0118: Initialisations not using constants

Detects initializations that do not assign constants

Justification: Initializations should be constant if possible and should not refer to other variables. In particular, you should avoid function calls during initialization because this can allow access to uninitialized data.

Importance: Medium

Example

```plaintext
PROGRAM PLC_PRG
VAR
 dwTemp : DWORD := 22;
dwTest : DWORD := dwTemp;       // SA0118
dwVar : DWORD := TempVarInFUNC(); // SA0118
END_VAR

--> SA0118: Initialisations not using constants
```

See also

● “Chapter 1.7.3.3.1.5 “Attribute ‘analysis:report-multiple-instance-calls’” on page 3614

SA0124: Pointer dereferences in declarations

Detects pointer dereferences that are used for initialization in the declaration part

Justification: Pointers and references should not be used for initializations because this can lead to access violations if the pointer has not been initialized.

Importance: Medium

Example

```plaintext
FUNCTION_BLOCK FB_Test
VAR_INPUT
 refStruct: REFERENCES TO ST_Test;
END_VAR
VAR
 xPointer : BOOL := refStruct.a; // SA0124
 iCount : INT;
END_VAR

--> SA0124: Dereference access in initialisation
```

See also

● “Chapter 1.7.3.3.1.5 “Attribute ‘analysis:report-multiple-instance-calls’” on page 3614

SA0125: References in initializations

Detects reference variables that are used for initialization in the declaration part

Justification: Pointers and references should not be used for initializations because this can lead to access violations if the pointer has not been initialized.

Importance: Medium
Example

PROGRAM PLC_PRG
VAR
  xRef: REFERENCE TO INT;
  iCount: INT := xRef;
END_VAR

--> SA0125: Reference used in initializations

See also

● § Chapter 1.7.3.3.1.5 “Attribute 'analysis:report-multiple-instance-calls’” on page 3614

SA0140: Statements commented out

Detects commented-out statements

Justification: Code is often commented out for debugging purposes. When this kind of comment is released, it is not always clear at a later time whether the code should be deleted, or whether it has been commented out for debugging purposes and unintentionally not uncommented.

Importance: High

PLCopen rule: C4

Example

PROGRAM PLC_PRG
VAR
  iValue1: INT;
  iValue2: INT;
END_VAR

iValue1 := 100;
iValue2 := 200;
// iValue2 := 300;

--> SA0140: Statement commented out:: iValue2 := 300

See also

● § Chapter 1.7.3.3.1.5 “Attribute 'analysis:report-multiple-instance-calls’” on page 3614

Possible Use of Uninitialized Variables

1.7.3.3.2.51.1 SA0039: Possible null-pointer deferences.............................. 3679
1.7.3.3.2.51.2 SA0046: Possible use of not initialised interface..................... 3680
1.7.3.3.2.51.3 SA0145: Possible use of not initialised reference................... 3681

SA0039: Possible null-pointer deferences

Detects code locations where a null pointer is possibly dereferenced

Justification: A pointer should be checked before each dereferencing to make sure it is not equal to zero. Otherwise an access violation may occur at runtime.

Importance: High
Example

PROGRAM PLC_PRG
VAR
  ptiVar1:POINTER TO INT;
  ptiVar2:POINTER TO INT;
  ptiVar3:POINTER TO INT;
  iVar:INT;
  iCount :INT;
  iCondition: INT;
END_VAR

iCount := iCount + INT#1;
ptiVar1 := ADR(iVar);
ptiVar1^ := iCondition;  // OK - valid reference
ptiVar2^ := iCondition;  // SA0039 - null pointer dereferenciation
iVar := ptiVar3^;        // SA0039 - null pointer dereferenciation

--> SA0039: Possible null pointer dereference 'ptiVar2^'
--> SA0039: Possible null pointer dereference 'ptiVar3^'

See also

- Chapter 1.7.3.3.1.5 "Attribute 'analysis:report-multiple-instance-calls" on page 3614

SA0046: Possible use of not initialised interface

Detected the use of interfaces that were not initialized before being used

Justification: An interface reference should be checked for <> 0 before it is used. Otherwise an access violation may occur during access.

Importance: High
Example

// declaration of INTERFACE ITF and assigned METH2:
METHOD METH2 : BOOL
VAR_INPUT
  iInput2:INT;
END_VAR

// declaration of INTERFACE Master_ITF1 and assigned METH:
METHOD METH : BOOL
VAR_INPUT
  iInput:INT;
END_VAR

PROGRAM PLC_PRG
VAR
  instPOU:POU;
  instITF:ITF;
  instMasterITF1:Master_ITF1;
  instMasterITF2:Master_ITF2;
  iDummy:INT;
  xDummy:BOOL;
  instNoInitITF:ITF;
  instNoInitITF2:ITF;
  instNoInitMasterITF1:Master_ITF1;
  instNoInitMasterITF2:Master_ITF2;
END_VAR

instITF := instPOU;
xDummy := instITF.METH(iInput := iDummy);             // OK
instMasterITF1 := instPOU;
xDummy := instMasterITF1.METH(iInput := iDummy);        // OK

xDummy := instNoInitITF.METH(iInput := INT#1);        // SA0046
xDummy := instNoInitITF2.METH(iInput2 := INT#2);       // SA0046
xDummy := instNoInitMasterITF1.METH(iInput := INT#3);  // SA0046
iDummy := instNoInitMasterITF2.Prop;                  // SA0046

IF instNoInitITF <> 0 THEN
  instNoInitITF.Prop;          // OK, weil das Interface nicht 0
  sein kann
END_IF

--> SA0046: Possible use of not initialised interface
  'instNoInitITF'

See also

- Chapter 1.7.3.3.1.5 “Attribute 'analysis:report-multiple-instance-calls'” on page 3614

SA0145: Possible use of not initialised reference

Detects any reference variables that may not be initialized before use and are not checked by
the operator __ISVALIDREF. This rule is applied in the implementation part of POUs. Rule
SA0124 applies to the declaration.
Justification: A reference should be checked for its validity before access because an access violation may occur during access.

Importance: High

Example

PROGRAM PLC_PRG
VAR_INPUT
  ref_iTest : REFERENCE TO INT;
END_VAR

ref_iTest := 99;         // SA0145
IF __ISVALIDREF(ref_iTest) THEN
  ref_iTest := 88;
END_IF

--> SA0145: Possible use of not initialised reference 'ref_iTest'

See also

- § Chapter 1.7.3.3.1.5 “Attribute ‘analysis:report-multiple-instance-calls’” on page 3614
- § Chapter 1.7.3.3.2.49.15 “SA0124: Pointer dereferences in declarations” on page 3678

SA0150: Violations of lower or upper limits or the metrics

Detects the POUs that violate the activated metrics at the lower or upper limits

Justification: Code that complies with certain metrics is easier to read, easier to maintain, and easier to test.

Importance: High

PLCopen rule: CP9

Example

Initial situation: The “Number of calls” metric is selected in “Project Settings ➔ Static Analysis ➔ Metrics”. Lower limit: 0; upper limit: 3. Prog_1 is called five times.

When running the static analysis, the “SA0150: Metric violation for Prog_1. Report for metric calls (5) > 2” error is issued in the message view, in the “Build” category.

See also

- § Chapter 1.7.3.3.1.5 “Attribute ‘analysis:report-multiple-instance-calls’” on page 3614

SA0160: Recursive calls

Detects recursive calls in actions, methods, and properties of function blocks. Also detects possible recursions from virtual function calls and interface calls.

Justification: Recursions lead to non-deterministic behavior and are therefore a source of errors.

Importance: Medium

PLCopen rule: CP13
Example

The following method Call is assigned to the function block FB_Test:

```plaintext
FUNCTION_BLOCK FB_Test
VAR
  bParameter: BOOL;
END_VAR

METHOD Call : BOOL
VAR_INPUT
END_VAR
Call := THIS^.Call();  //SA0160
```

The program PLC_PRG calls FB_Test:

```plaintext
PROGRAM PLC_PRG
VAR
  fbTest : FB_Test;
  bValue : BOOL;
END_VAR
bValue := fbTest.bParameter;
fbTest.Call();
--> SA0160: Recursive call detected: 'PLC_PRG -> FB_Test.Call -> FB_Test.Call
```

See also

- Chapter 1.7.3.3.1.5 “Attribute ’analysis:report-multiple-instance-calls’” on page 3614

SA0161: Unpacked structure in packed structure

Detects unpacked structures that are used in packed structures

Justification: The compiler typically sets an unpacked structure to an address that allows aligned access to all elements within the structure. If you create this structure in a packed structure, then aligned access is no longer possible. Furthermore, access to an element in the unpacked structure can lead to a misalignment exception.

Importance: High
The structure `structSingleDataRecord` is packed, but it contains the unpacked structures `struct4Byte` and `struct9Byte`.

```plaintext
{attribute 'pack_mode' := '1'}
TYPE structSingleDataRecord :
  STRUCT
    str9ByteData: struct9Byte;    (* 9 BYTE *)
    str4ByteData: struct4Byte;    (* 4 BYTE *)
    udi1: UDINT;
    udi2: UDINT;
    udi3: UDINT;
    usi4: USINT;
  END_STRUCT
END_TYPE (* 9 BYTE *)
TYPE struct9Byte :
  STRUCT
    usiRotorSlots: USINT;        (* 1 BYTE *)
    uiMaxCurrent: UINT;            (* 2 BYTE *)
    usiVelocity: USINT;            (* 1 BYTE *)
    uiAcceleration: UINT;        (* 2 BYTE *)
    uiDeceleration: UINT;        (* 2 BYTE *)
    usiDirectionChange: USINT;    (* 1 BYTE *)
  END_STRUCT
END_TYPE TYPE struct4Byte :
  STRUCT
    //udiDummy : UDINT;
    rRealDummy : REAL;
  END_STRUCT
END_TYPE
```

--> SA0161: Declaration of an unpacked struct 'struct9ByteData'
inside a packed struct 'structSingleDataRecord'

--> SA0161: Declaration of an unpacked struct 'struct4ByteData'
inside a packed struct 'structSingleDataRecord'

See also

- ☞ Chapter 1.7.3.3.1.5 “Attribute 'analysis:report-multiple-instance-calls'” on page 3614

**SA0162: Missing comments**

Detects uncommented locations in the program

Justification: Complete commenting is required by many programming guidelines, and it increases the readability and maintainability of the code.

Importance: Low

PLCopen rule: C2

Comments are required in the following cases:

- Declaration of variables (Comments are located either above the declaration or to the right of the declaration.)
- Declaration of programs, function blocks, or methods (Comments are located above the declaration in the first line.)
Example

```plaintext
PROGRAM PLC_PRG
VAR
   iVar: INTEGER;
END_VAR

---> SA0162: Missing comment for 'PLC_PRG'
---> SA0162: Missing comment for 'iVar'
```

See also

- Chapter 1.7.3.3.1.5 “Attribute 'analysis:report-multiple-instance-calls” on page 3614

**SA0163: Nested comments**

Detects nested comments

Justification: Nested comments should be avoided because they are difficult to read.

Importance: Low

PLCopen rule: C3

Example

```plaintext
{attribute 'do-analysis'}
(* That is
(* nested comment 1 *)
*)
PROGRAM PLC_PRG
VAR
(* That is
// nested comment 2
comment *)
   iVar: INTEGER;
   iVar2: INTEGER;

(* That is
(* nested comment 3 *) *)
   iVar3: POINTER TO DWORD;
   hugo: INTEGER;
END_VAR

(* That is
// nested comment 4
comment *)

iVar := iVar + 1;

(* That is
(* nested comment 5 *)
*)

(* Not that one *)

---> SA0163: Nested comment 'nested comment 1'
---> SA0163: Nested comment 'nested comment 2'
---> SA0163: Nested comment 'nested comment 3'
---> SA0163: Nested comment 'nested comment 4'
---> SA0163: Nested comment 'nested comment 5'
```
SA0164: Multiline comments

Detects multiline comments that are coded as (* comment *). Only single-line comments that are coded as // comment are permitted.

Justification: Some programming guidelines prohibit multiline comments in code because the beginning and end of a comment could get lost and the closing comment bracket could be deleted by accident.

Importance: Low

PLCopen rule: C5

Example

```plaintext
{attribute 'do-analysis'}
(*
  This is a multi-line comment     // SA0164
*)
PROGRAM PLC_PRG
VAR
  // This is a single line comment
  a: DINT;
END_VAR
(* This is not a single line comment *)     // SA0164
a := a + 1;
```

SA0165: Tasks calling other POUs than programs

Detects tasks that call function blocks or functions instead of a program

Justification: This rule is part of the PLCopen Coding Guidelines. Therefore, compliance is also checked in CODESYS. We do not see any problems with data consistency in CODESYS if tasks would call POUs other than programs. However, problems can occur if the code is to be ported to other platforms.

Importance: Low

PLCopen rule: CP16

Tasks are inserted below the task configuration. The POUs to be called are configured in the tasks. The POUs must be the “Program” type. The “Function block” and “Function” types are not permitted.
Example

SA0166: Max. number of input/output/in-out variables...

Detects whether or not a defined number of input variables (VAR_INPUT), output variables (VAR_OUTPUT) or VAR_IN_OUT variables is exceeded in a POU. In the “Project Settings”, double-click the rule entry to open a dialog where you define the maximum number.

Justification: This is about checking individual programming guidelines. Many programming guidelines provide for a maximum number of POU parameters. Too many parameters make the code unreadable and the POUs difficult to test.

Importance: Medium

PLCopen rule: CP23

Example

FUNCTION_BLOCK FB1
    VAR_INPUT
        xIn    : BOOL;
    END_VAR
    VAR_IN_OUT
        xInOut1 : BOOL;
        xInOut2 : BOOL;
    END_VAR
    --> SA0166: Too many VAR_IN_OUT variables in POU 'FB1'

See also

● § Chapter 1.7.3.1.5 “Attribute 'analysis:report-multiple-instance-calls’’ on page 3614

SA0167: Temporary function block instances

Detects function block instances that are declared as temporary variables. This affects instances that are declared in a method or function or as VAR_TEMP, and therefore are reinitialized in each processing cycle or for each POU call.

Justification: Function blocks have a state that is usually maintained over multiple PLC cycles. An instance on the stack exists only for the duration of the function call. Therefore, it rarely makes sense to create an instance as a temporary variable. Secondly, function block instances are often large and need a lot of space on the stack (which is usually restricted to controllers). Thirdly, the initialization and often also the scheduling of a function block can take a long time.

See also

● § Chapter 1.7.3.1.5 “Attribute 'analysis:report-multiple-instance-calls’’ on page 3614
Importance: Medium

Examples

PROGRAM PLC_PRG
VAR
END_VAR
VAR_TEMP
  _yafb: AFB;
END_VAR

FUNCTION Fun : INT
VAR_INPUT
END_VAR
VAR
  funafb: AFB;
END_VAR

METHOD METH : INT
VAR_INPUT
END_VAR
VAR
  methafb: AFB; // SA0167
END_VAR

--> SA0167: Temporary function block instance: 'methafb'

See also
● % Chapter 1.7.3.3.1.5 “Attribute 'analysis:report-multiple-instance-calls'” on page 3614

1.7.4 CODESYS Profiler
1.7.4.1 Overview

CODESYS Profiler is a program analysis tool which is integrated in CODESYS for detecting inefficient programming in your application source code.

The term profiling means taking measurements for the runtime behavior of POUs of the application. This is possible by means of the instrumentation of the code, or on multicore systems by means of sampling.

The "Sampling" method of measurement is not supported yet in V1.3.0.0.

In addition to profiling, CODESYS Profiler also supports a measurement of code coverage, which detects the number of processed statements per POU. This method also works by instrumenting the code.

The measurements for profiling and code coverage have to be activated explicitly.

You can use the measurement data to help plan a course of action for optimizing the source code on both a structural and algorithmic level.

Overview of the integration of CODESYS Profiler in CODESYS:

Insert a “Profiler” object below the application. In the editor of this object, configure the desired measurement method and view the measurement results in online mode. Moreover, in the "Online ➔ Profiler" menu, the "Profiler Watch List" is provided for configuring and displaying the POUs to be measured.
1.7.4.2 Executing the Profiling Measurements

NOTICE!
The implicitly added IEC code for a Profiler measurement method results in a change which slows down the evaluated program.

When a profiling or code coverage measurement is active, an online is generally not possible. Moreover, some online features are not available, such as flow control and breakpoints.

Enabling or disabling a Profiler measurement method, or changing any settings in the Profiler editor each requires a download. Then an online change is no longer possible.

NOTICE!
POUs from uncompiled libraries are also displayed on the “Online” tab of the Profiler editor and in the “Profiler Watch List”.

CODESYS Profiler provides different profiling methods for measuring the processing times of programs for various kinds of problems.

- **Instrumentation**
  Profiler measures the times by using additional code which is generated into the program. The recording and processing of the measured values takes place on the controller and therefore can also be performed without connecting the development system to the controller (in contrast to the other measuring methods).

- **Sampling (for multicore systems only)**
  This method is not supported yet in V1.3.0.0. Profiler pauses a task at random times and evaluates the call stack of the task. The Profiler sampling measures the frequency of the POU execution, not durations of time. The recorded call stacks are transferred in cycles to the development system for processing. Therefore, this method only works when the development system is in online mode.

- **Profiler Watch List**
  Profiler measures the execution time of individual POUs by means of execution points which are set in the POUs while the program is running.

See the help pages for each method to find the right one for your application.

See also

- Chapter 1.7.4.2.1 “Profiling by Code Instrumentation” on page 3690
- Chapter 1.7.4.2.3 “Profiling by Sampling” on page 3693
- Chapter 1.7.4.2.2 “Using the Profiler Watch List” on page 3692
1.7.4.2.1 Profiling by Code Instrumentation

NOTICE!
The implicitly added IEC code for a Profiler measurement method, as well as the breakpoints for POUIs in the watch list, result in a change and slows down the evaluated program.

When a profiling or code coverage measurement is active, an online is generally not possible. Moreover, some online features are not available, such as flow control and breakpoints. There are no restrictions for the watch list in this respect.

Enabling or disabling a Profiler measurement method, or changing any settings in the Profiler editor each requires a download. Then an online change is no longer possible. There are no restrictions for the watch list in this respect.

The recording and processing of the measured values takes place on the controller and therefore can also be performed without connecting the development system to the controller (in contrast to the other measuring methods).

With this profiling method, CODESYS Profiler inserts additional code into the entry and exit of each function in the program. The runtimes of a entire task cycle are measured and stored.

The method is useful for detecting individual outliers in the task runtimes. To do this, select the “Record max cycle” recording mode.

When the entire project is instrumented, this can have a considerable impact on task runtimes and memory consumption. Therefore, select only a part of the application POUIs for instrumentation whenever possible. One recommended approach is to first select only those POUIs that are called directly by the task. Determine the hot spots of the runtime for these POUIs and then based on these select the called POUIs for measurement.

NOTICE!
- POUIs which are not selected for measurement are not shown on the measurement list, the call tree, or the backtrace tabs. Therefore, it is possible that a call tree “POU1 -> POU3” is displayed, although there is no direct call of POU3 by POU1. This is because there are POU calls which are not selected for measurement between POU1 and POU3.
- The "own time" output in the measurement for a POU can also contain times of called POUIs which are not selected for measurement.

Requirement:
- A CODESYS project application with multiple POUIs is open in offline mode.
- A task is defined to control POU calls.
- The application is the active application and can be compiled without errors.
- A connection to the standard controller is configured in the communication settings, and the controller is running.

1. To activate and deactivate profiling programmatically, you define a Boolean variable in the application. (This is optional for the example described here. Activation can also be performed by a menu command.)
2. Click “Add Object” to add a “Profiler” object below the application in the device tree.
3. Double-click the object.
   - The object editor opens. The 🔎 “Settings” tab is in focus.
4. Set the “Method” to “Instrumentation”. Select the “Task” with the POU calls that you want to perform time measurements on.

5. Under “Recording”, select the task for controlling the POUs to be instrumented.
   For “Condition”, click and select the Boolean variable from your application that you want to use to switch the value recording on and off. Note: Using this kind of variable is optional. If you leave the field blank, then every cycle is recorded.

6. Under “Instrumentation parameters”, set the “Recording mode” to “Record next cycle”. Leave the required “Buffer size” (the maximum number of POU calls that are recorded per task cycle) as the default “10000”.
   Under “Snapshot appearance”, select the “Time format” for displaying the recording.

7. In “POU selection”, select the POUs that should be instrumented. Note that you could also select “Instrument whole application”, which would include all POUs of the project as well as the POUs of the integrated libraries.

8. Click “Online ➔ Login” to download the application to the controller.
   ➔ CODESYS displays in the status bar.

9. Click “Debug ➔ Start” to start the application.
   ➔ The project runs and you see the current variables values in the usual monitoring view.

10. If you have configured a Boolean variable as a condition to start profiling, then now set this variable to TRUE.

   Background information about recording: Because you selected the “Record next cycle” option in this example, only the first cycle is recorded when the application is started and the profiling. No additional cycle is recorded. An additional measurement is performed only when a recording is uploaded from the controller. This uploading is done either by opening the “Online” tab for the first time or by clicking the “Refresh this snapshot” button. For information about the alternative “Record max cycle” option, see the "Tab 'Settings’” chapter in the reference part of the help.

   Now look at the measurement results. Open the editor of the Profiler object and its “Online” tab.
   ➔ The “Online” tab includes four subtabs which can be selected from the menu on the left side. In addition, you see an area with buttons and settings for the profiling that is currently running. A description of the tabs is located in the “Tab 'Online’” chapter in the reference part of the help.

Detailed information about a POU call in the “Properties” dialog

1. On the “Online” tab, click the “Call Tree” subtab.
   ➔ You see the hierarchy of all calls that have originated from the task specified in the “Settings” dialog.

2. Select one of the calls and click “Properties” in the context menu.
   ➔ The “Properties” dialog for the called POU opens with detailed information, such as “Own time” and “Standard deviation”.

See also
● Chapter 1.7.4.5.1.3 “Tab 'Online’” on page 3697
● Chapter 1.7.4.5.1.2 “Tab 'Settings’” on page 3696
● Chapter 1.7.4.5.2.2 “Command 'Properties’” on page 3702
### 1.7.4.2.2 Using the Profiler Watch List

**NOTICE!**

The implicitly added IEC code for a Profiler measurement method, as well as the breakpoints for POUs in the watch list, result in a change and slows down the evaluated program.

When a profiling or code coverage measurement is active, an online is generally not possible. Moreover, some online features are not available, such as flow control and breakpoints. There are no restrictions for the watch list in this respect.

Enabling or disabling a Profiler measurement method, or changing any settings in the Profiler editor each requires a download. Then an online change is no longer possible. There are no restrictions for the watch list in this respect.

In CODESYS, the Profiler Watch List is provided with CODESYS Profiler. In this view, you can combine a selection of POUs or POU instances whose runtimes and calls that you want Profiler to measure. The configuration is possible in both online mode and offline mode.

When you select a POU for the Profiler Watch List, a watch point is set implicitly at the first and last break point position of the POU. The time difference between the two watch points on the controller is measured. In the case of a function block, the measurement is also performed and displayed for all POU instances.

**NOTICE!**

The Profiler Watch List can have a very different impact on the task runtime, depending on how often the implicitly set execution points are started.

When a function block instance is entered in the watch list, the execution points for all instances of the corresponding function block are reached. However, time is measured only for the specified instance.

The Profiler Watch List is suitable for measuring the runtime of individual POUs in runtime mode. The major advantage of this method is that it does not require any special code and it is always available.

**Requirement:** A project exists with an application which calls different program and function blocks as well as function block instances.

- In this example, you configure the list in offline mode.

1. Open the project and click “Online ➔ Profiler ➔ Profiler Watch List”.
   - The empty “Profiler Watch List” opens.

2. Drag one of the application POUs from the device tree to the watch list.
   - The POU is inserted into the first line of the list. The “POU” column contains the POU name, and the “Application” column contains the name of the respective application.

3. Add all additional required POUs, either by dragging them to the list or by double-clicking a field in the “POU” column to open the Input Assistant.

Now view the list in online mode:

1. Download the project to the controller and start it. Open the “Profiler Watch List”.
   - You see measurement results about the call durations and number of calls for each POU. See the help page for the “Profiler Watch List”.

---

**Configuring watch lists**

- Requirement: A project exists with an application which calls different program and function blocks as well as function block instances.

- In this example, you configure the list in offline mode.

1. Open the project and click “Online ➔ Profiler ➔ Profiler Watch List”.
   - The empty “Profiler Watch List” opens.

2. Drag one of the application POUs from the device tree to the watch list.
   - The POU is inserted into the first line of the list. The “POU” column contains the POU name, and the “Application” column contains the name of the respective application.

3. Add all additional required POUs, either by dragging them to the list or by double-clicking a field in the “POU” column to open the Input Assistant.

Now view the list in online mode:

1. Download the project to the controller and start it. Open the “Profiler Watch List”.
   - You see measurement results about the call durations and number of calls for each POU. See the help page for the “Profiler Watch List”.

---

3692 3ADR010583, 1, en_US 2020/12/10
2. Click “Display Mode” in the context menu of the watch list and change to another data type or unit.
   ⇒ All current results in the list are displayed continuously in the selected format, as is the case for monitoring.

3. Click “Open POU” in the context menu of the watch list.
   ⇒ The POU currently in focus in the list is opened in the editor. You see the solid green circle of the watch point at the beginning and end of the implementation code that is implicitly set to enable the Profiler measurement.

4. Drag another POU to the watch list.
   ⇒ The POU is included immediately in the measurement.

See also
● % Chapter 1.7.4.5.2.9 “Command 'Profiler' - 'Profiler Watch List’” on page 3705
● % Chapter 1.7.4.1 “Overview” on page 3688

1.7.4.2.3 Profiling by Sampling

This method is not supported yet in V1.3.0.0.

1.7.4.3 Measuring Code Coverage

NOTICE!

The implicitly added IEC code for a Profiler measurement method results in a change which slows down the evaluated program.

Enabling or disabling a Profiler measurement method, or changing any settings in the Profiler editor each requires a download. Then an online change is no longer possible.

In addition to profiling, CODESYS Profiler also supports the measurement of code coverage. This determines which of the statements in a POU are executed when processing. The part of the total number of executed instructions is termed as "coverage". Coverage is expressed as a percentage.

In the “Code Coverage” view, select the POUs in offline mode that should be used to perform the measurement. In online mode, you will continuously see the current values of the code coverage.

Requirements
● A CODESYS project application with multiple POUs is open in offline mode.
● A task is defined to control POU calls.
● The application is the active application and can be compiled without errors.
● A connection to the standard controller is configured in the communication settings, and the controller is running. The project is in offline mode.

Settings in offline mode
1. In the device tree, double-click the “Profiler” object to open the editor. Set the method to “Code Coverage”.
   ⇒ The POUs of the application are displayed in the editor in a tree structure under "POU selection".
2. Select the POUs that should be measured by code coverage.

Viewing measurement results in online mode

1. In the device tree, double-click the “Profiler” object to open the editor.

   ⊳ The view opens with the measurement results for each of the instrumented POUs, including the total number of statements and the number of statements not executed. The code coverage is displayed as a percentage with a gray bar in the “Coverage” column.

   You can sort the table according to this column either ascending or descending by clicking a column heading.

2. Note the menu bar above the results table.

   ● “Open POU” for opening the POU selected in the results table in its own editor
   ● “Reset” for restarting the measurement
   ● “Save Snapshot” for creating a snapshot

   ⊳ If the open POU is programmed in ST, then the processed code is displayed in green and the unprocessed code is displayed in red.

See also

● ☞ Chapter 1.7.4.1 “Overview” on page 3688
● ☞ Chapter 1.7.4.5.1.3 “Tab ‘Online’” on page 3697
● ☞ Chapter 1.7.4.5.1.4 “Tab ‘Snapshots’” on page 3701
● ☞ Chapter 1.7.4.5.2.1 “Command ‘Open POU’” on page 3702

1.7.4.4 Creating Snapshots

In online mode, a snapshot of all current measurement results can be created on the tab during a Profiler measurement. All created snapshots are managed on the “Snapshots” tab of the Profiler editor.

1. For example: Configure a profiling measurement for your application using the instrumentation method.

2. Start the application to the controller, and start the measurement.

   ⊳ The measurement results are displayed in the Profiler editor on the various tabs.

3. Click “Save snapshot”. For the instrumentation and sampling methods, the command button for this is located on the left side of the “Online” tab. For the code coverage method, the command button is located on the toolbar of the tab.

   ⊳ The “Save Snapshot” dialog opens.

4. Type in the name for the snapshot and include an optional description. Click “OK” to confirm.

5. Open the “Snapshots” tab. Note that this tab is available only after the first snapshot has been created.

   ⊳ The recently created snapshot is listed with the “Name” which you assigned to it and the “Timestamp”. It is listed below the node of the name of the applied Profiler method (“Sampling” in the current example).

6. Click the line of the snapshot.

   ⊳ The snapshot opens on the right side of the tab. In this view, the same functionalities (for example, open POU, copy table, and export) are available as on the “Online” tab.

7. On the left side of the tab, right-click the created snapshot.

   ⊳ Here you see the commands for deleting and editing the snapshot.
8. On the “Profiler” - “Snapshots” tab, double-click the line of the snapshot.

   ➔ The snapshot opens in a new tab named <name, time stamp> on the same level as the Profiler editor. The usual functionalities are available here as well. However, if a POU has been modified after the snapshot, then you cannot open it in the red/green view.

See also

- % Chapter 1.7.4.5.1.4 “Tab ‘Snapshots’” on page 3701
- % Chapter 1.7.4.5.1.3 “Tab ‘Online’” on page 3697
- % Chapter 1.7.4.5.2.4 “Command ‘Edit Snapshot’” on page 3703
- % Chapter 1.7.4.5.2.3 “Command ‘Delete Snapshot’” on page 3703

1.7.4 Reference, User Interface

1.7.4.5 Objects

1.7.4.5.1 Objects

1.7.4.5.1.1 Object ‘Profiler’…………………………………………………………………………… 3695
1.7.4.5.1.2 Tab ‘Settings’……………………………………………………………………………… 3696
1.7.4.5.1.3 Tab ‘Online’………………………………………………………………………………… 3697
1.7.4.5.1.4 Tab ‘Snapshots’…………………………………………………………………………… 3701

Object ‘Profiler’

Symbol: $\text{设计器}$

You can insert the “Profiler” object below an application in the device tree. Double-click the object to open the editor for configuring the settings and viewing the measurement results in online mode. Created snapshots are also managed on an editor tab.

“Method”: Setting of the method of measurement

The "Sampling" measurement method of is not supported yet in V1.3.0.0.

- No profiling
- Instrumentation
- Code coverage

The “Settings” tab has different contents depending on the selected method. If measurements have already been saved, then the “Snapshots” tab is also displayed.

In online mode, the “Online” tab is available for showing the measurement results.

See also

- % Chapter 1.7.4.5.1.2 “Tab ‘Settings’” on page 3696
- % Chapter 1.7.4.5.1.3 “Tab ‘Online’” on page 3697
- % Chapter 1.7.4.5.1.4 “Tab ‘Snapshots’” on page 3701
Tab 'Settings'

The “Settings” tab in the editor of the Profiler object is used to configure the profiling in online mode. It contains the setting options that correspond to the set method.

Table 786: Recording

<table>
<thead>
<tr>
<th>“Task”</th>
<th>In online mode, CODESYS Profiler records the runtime behavior for the POUs that are called by this task.</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Condition”</td>
<td>(Boolean expression) The value TRUE is a requirement for recording. For example, you can use the Input Assistant to set a Boolean variable for this. If the field is blank, then every cycle is recorded.</td>
</tr>
<tr>
<td>“Instrumentation parameters”</td>
<td>Available only when the profiling method is set to “Instrumentation”</td>
</tr>
<tr>
<td>● “Instrument whole application”: All POUs of the application are instrumented, including the POUs from the referenced, compiled libraries. In this case, the “POU selection” is not available in the lower part of the tab.</td>
<td></td>
</tr>
<tr>
<td>● “Recording mode”</td>
<td>– “Record next cycle”: When the application starts, the first PLC cycle is recorded. No additional cycle is recorded. An additional measurement is performed only after a recording is uploaded from the controller either by opening the “Online” tab for the first time, or by clicking “Refresh this snapshot”.</td>
</tr>
<tr>
<td></td>
<td>– “Record max cycle”: Every cycle is recorded. The result of a recording is saved when the cycle has lasted longer than the last saved recording. When a recording is uploaded from the controller, the longest recording to date is deleted and the recording starts all over again.</td>
</tr>
<tr>
<td>● “Buffer size”: Maximum number of individual recordable run times per cycle. These are the runtimes of all POU calls for the task specified above. For a correct configuration of the buffer size, also note the selected “Recording mode”.</td>
<td></td>
</tr>
<tr>
<td>“Sampling parameters”</td>
<td>Available only when the profiling method is set to “Sampling”</td>
</tr>
<tr>
<td>● “Profiler task group”: The available task groups in their multicore system can be selected.</td>
<td></td>
</tr>
<tr>
<td>● “Sampling interval”: Time period when a random measurement should be taken</td>
<td></td>
</tr>
<tr>
<td>● “Maximum depth of call stack”: Selection between 20 and 100</td>
<td></td>
</tr>
</tbody>
</table>
Table 787: Snapshot appearance

| "Time format"     | 
|-------------------|----------------------------------|
| "tick"            | Time unit defined by the processor |
| "ms"              | Milliseconds                      |
| "µs"              | Microseconds                      |
| "ms/µs"           | Milliseconds or microseconds (depending on size) |

Note: Implicit code calculates the approximate assignment of the time formats "ms" and "µs" to system ticks. On the "Online" tab, on the "Overview" subtab, you can see the calculated value displayed in the "Time" entry. Example: 1 ms = 3398810 Ticks.

Determining critical limits

- "Critical limit total time"
- "Critical limit end of call chain"
- "Critical limit high own time"

Critical upper limits for the times measured in the current cycle. If the time of a call in the cycle exceeds the percentage defined here for the total cycle time, then this is indicated in the Profiler results with a symbol at the beginning of the respective lines.

- Total time: Time spent by the POU call, including all calls from this POU
- End of the call chain: Last call in the call tree. In this case: total time = "Own time"
- Own time: Time spent by the POU call, excluding the time spent on all POU calls made by this POU itself

Table 788: POU selection

<table>
<thead>
<tr>
<th>In tree structure:</th>
<th>Available only when the &quot;Instrumentation&quot; or &quot;Code coverage&quot; profiling method is set and the &quot;Instrument whole application&quot; option is not selected</th>
</tr>
</thead>
<tbody>
<tr>
<td>POUs of the application</td>
<td>Selection of the desired POUs</td>
</tr>
<tr>
<td>POUs from the POU pool</td>
<td>POUs from integrated libraries can only be instrumented all together with all POUs of the application by selecting the &quot;Instrument whole application&quot; option.</td>
</tr>
</tbody>
</table>

See also

- § Chapter 1.7.4.5.1.1 "Object 'Profiler'" on page 3695

Tab 'Online'

This tab is displayed in the editor of the Profiler object in online mode only. It displays the measurement results for the selected method. The display is not refreshed automatically. Instead, it shows a "snapshot" on request.

Profiling by instrumentation or sampling

When the profiling method is set to "Sampling" or "Instrumentation", the "Online" tab includes additional subtabs on the left side. In addition, there is an area with buttons for creating snapshots and controlling the current measurement process. The context menu also provides helpful commands.

Table 789: Tab 'Online' - 'Overview'

<table>
<thead>
<tr>
<th>General information about the recording</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Header</strong></td>
</tr>
<tr>
<td><strong>Device</strong></td>
</tr>
<tr>
<td><strong>Application</strong></td>
</tr>
<tr>
<td>-----------------</td>
</tr>
<tr>
<td><strong>Compiler version</strong></td>
</tr>
<tr>
<td><strong>Number of instrumented POU s</strong></td>
</tr>
<tr>
<td><strong>Task</strong></td>
</tr>
<tr>
<td><strong>Cycle</strong></td>
</tr>
<tr>
<td><strong>Duration</strong></td>
</tr>
<tr>
<td><strong>Time</strong></td>
</tr>
<tr>
<td><strong>Start time of the measured cycle</strong></td>
</tr>
<tr>
<td><strong>Current time on device</strong></td>
</tr>
<tr>
<td><strong>Recording mode</strong></td>
</tr>
<tr>
<td><strong>Buffer size</strong></td>
</tr>
<tr>
<td><strong>Recording condition</strong></td>
</tr>
</tbody>
</table>

**Table 790: Tab 'Online' - 'Call Tree'**

This hierarchical tree structure shows all calls that have been made to the selected task during the measurement period. The top node is the task, and all call chains are displayed below it.

Calls in libraries are not displayed.

Each node corresponds to a specific POU call and provides the following information about the current and previous measured times and number of calls.

- Percentage of time that has elapsed in this call to the time that has elapsed in all calls (example: 34.05%)
- Name and type of POU (example: PLC_PRG (PRG))
- Total time elapsed in the call (example: 85.531 µs)
- Number of calls of this POU in this cycle in this call tree (example: 22 Calls)
- Average, minimum, and maximum execution time of this call (example: Avg: 0.254 µs Min: 0.194 µs Max: 0.3µs)

Example:
Table 791: Tab 'Online' - 'List'

All instrumented POUs are listed. For each POU, you see the number of “Samples” and the “Time (ms/µs)” of all recorded calls.

The “Functions called by <POU name>” list in the lower part of the view always shows the POU calls of the POU currently selected in the list above. Double-clicking a line in the lower list selects the corresponding entry in the upper list.

<table>
<thead>
<tr>
<th>“POU Name”</th>
<th>Name and type of POU (example: PLC_PRG (PRG))</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Time (ms/µs)”</td>
<td>Total time elapsed in the call (example: 85.53 µs)</td>
</tr>
<tr>
<td>“Average time”</td>
<td>Average, minimum, and maximum execution time of this call in &quot;ms&quot; or &quot;µs&quot; (example: Avg: 0.254 µs, Min: 0.194 µs Max: 0.3µs)</td>
</tr>
<tr>
<td>“Min. time (ms/µs)”</td>
<td></td>
</tr>
<tr>
<td>“Max. Time”</td>
<td></td>
</tr>
<tr>
<td>“Own time (ms/µs)”</td>
<td>Time that the POU call has spent, excluding the time spent by all POU calls from this POU</td>
</tr>
<tr>
<td>“Own Time (%)”</td>
<td>Percentage of the own time to the total time</td>
</tr>
<tr>
<td>“Calls”</td>
<td>Number of calls of this POU in this cycle in this call tree (example: 22 Calls)</td>
</tr>
<tr>
<td>“Standard deviation”</td>
<td>Standard deviation of the average execution time (example: 0.128 µs)</td>
</tr>
</tbody>
</table>

Example:
Table 792: Tab 'Online' - 'Back Traces'

This is a reverse view of the call tree. This means that you can trace all calls from a POU call to the beginning of the call chain.

- Percentage of own time to total time (example: 19.35%)
- Name and type of POU (example: MID (FUN))
- Own time, or portion of the own time of the call to the own time of the node (example: 9.466μs of 9.752μs)
- Number of calls in this call tree in this cycle, and the percentage of calls to the total number of calls (example: 4 Calls of 5)

Example:

![Call Tree Example]

Table 793: Actions for the current measurement

<table>
<thead>
<tr>
<th>Action</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Refresh this snapshot&quot;</td>
<td>The current measurements are done and displayed. The display is not refreshed automatically.</td>
</tr>
<tr>
<td>&quot;Save snapshot&quot;</td>
<td>The “Save Snapshot” dialog opens for typing in a name and description for the snapshot. After you click “OK”, the current measurement results are saved and can be called again in the “Snapshots” tab.</td>
</tr>
<tr>
<td>For the “Sampling” method only (not supported at this time): “Reset”</td>
<td>The current measurement is reset and can be restarted.</td>
</tr>
<tr>
<td>For the “Sampling” method only (not supported at this time): “Stop”, “Pause”</td>
<td>Button for starting, pausing, and stopping, the measurement process</td>
</tr>
</tbody>
</table>
| For the “Sampling” method only (not supported at this time): “Sampling interval” | Time period between measurements  
A random value is generated in the range between 0 and the specified time span. After this time, the task to be measured is paused and recorded. The remaining time until the specified time span elapses until the next random value is generated. This means that it is measured within the time span, but not before the full time span has been reached.  
At a sampling interval of 1 ms, 100 measurements should be performed in 100 ms. |
Measurement of code coverage

When code coverage is used, the “Online” tab shows which of the statements are executed in the selected POUs and which are not. In contrast to the profiling methods, explicit refreshing of the measured values is not necessary here. However, the measurement can be repeated.

Table 794: Table with measurement results

| “Name” | The names of the POUs selected for the measurement are shown in the tree structure. The parent objects act as nodes (for example, the name of the application to which they belong). |
| “Number of statements” | Total number of statements contained in the POU |
| “Statements not executed” | Number of statements contained in the POU but not executed |
| “Coverage (%)*” | Percentage of statements in the POU that are executed (example: 75 for 75%) |

Table 795: Buttons above the measurement value table

| “Open POU” | The POU selected in the view is opened in its editor. Statements that have been processed are displayed in green. Statements that have not been processed are displayed in red. The POU editor also opens when you double-click a row in the table. |
| “Reset” | The measurement results are reset to 0 and the measurement is repeated in the next cycle. |
| “Save Snapshot” | The “Save Snapshot” dialog opens for typing in a name and description for the snapshot. After you click “OK”, the current measurement results are saved and can be called again in the “Snapshots” tab. |

Context menu

The following commands are available in the context menu depending on the selected location in the various displays of the measurement results:

- “Open POU”
- “Export”
- “Copy Table”
- “Properties” (in the call tree only)

See also

- Chapter 1.7.4.5.1.1 “Object ‘Profiler’” on page 3695
- Chapter 1.7.4.5.1.2 “Tab ‘Settings’” on page 3696
- Chapter 1.7.4.2.1 “Profiling by Code Instrumentation” on page 3690
- Chapter 1.7.4.3 “Measuring Code Coverage” on page 3693
- Chapter 1.7.4.5.2.1 “Command ‘Open POU’” on page 3702
- Chapter 1.7.4.5.2.7 “Command ‘Export’” on page 3704
- Chapter 1.7.4.5.2.8 “Command ‘Copy Table’” on page 3704
- Chapter 1.7.4.5.1.4 “Tab ‘Snapshots’” on page 3701
- Chapter 1.7.4.5.2.3 “Command ‘Delete Snapshot’” on page 3703
- Chapter 1.7.4.5.2.4 “Command ‘Edit Snapshot’” on page 3703

Tab ‘Snapshots’

This tab is displayed only after a measurement has already been recorded as a snapshot. The “Save snapshot” button on the “Online” tab of the Profiler editor is used to record snapshots.

The recorded snapshots are displayed in a tree structure. The nodes correspond to the profiling method in which they were created.
The recorded measurement results are displayed on the right for the time recording selected in the tree. Double-click the time recording in the tree to display it in a separate window. You can work with the same functionalities and commands as in online mode of the measurement. However, if a POU has been modified after the snapshot, then you cannot open it in the red/green view. If a POU has been deleted, then it cannot be opened ever again.

Commands for deleting and editing (name, description) are available in the context menu of a selected time recording.

See also

- Chapter 1.7.4.5.1.1 “Object ‘Profiler’” on page 3695
- Chapter 1.7.4.5.2.3 “Command ‘Delete Snapshot’” on page 3703
- Chapter 1.7.4.5.2.4 “Command ‘Edit Snapshot’” on page 3703
- Chapter 1.7.4.5.1.3 “Tab ‘Online’” on page 3697
- Chapter 1.7.4.4 “Creating Snapshots” on page 3694

## 1.7.4.5.2 Menu Commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Symbol</th>
<th>Function</th>
<th>Call</th>
</tr>
</thead>
<tbody>
<tr>
<td>'Open POU'</td>
<td>![Symbol]</td>
<td>The command opens the selected POU in its editor.</td>
<td>On the “Online” tab, the context menu of a selected line on the “Call Tree” subtab, on the “List” subtab, or on the “Back Traces” subtab</td>
</tr>
<tr>
<td>'Properties'</td>
<td>![Symbol]</td>
<td>The “Properties” dialog is opened for the selected POU.</td>
<td>On the “Online” tab, the context menu of a selected line on the “Call Tree” subtab</td>
</tr>
</tbody>
</table>

**Dialog ‘Properties for ‘<POU>’**

The dialog shows recording results for the call of the POU that is currently selected in the call tree.

- **‘This call’**: Call of the POU by the POU or task located above in the tree
- **‘All calls’**: All calls of the POU by the task
<table>
<thead>
<tr>
<th><strong>POU Type</strong></th>
<th>Type of POU (examples: function, program)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Code Size</strong></td>
<td>Code size of selected POU</td>
</tr>
<tr>
<td><strong>Location</strong></td>
<td>Name of the project that contains the active application</td>
</tr>
<tr>
<td><strong>Task</strong></td>
<td>Name of the responsible task</td>
</tr>
<tr>
<td><strong>Number of calls</strong></td>
<td>Number of calls of the POU in the current cycle</td>
</tr>
<tr>
<td><strong>Total time</strong></td>
<td>Total time for the POU call, including the time for all calls that the POU and its called POUs have performed</td>
</tr>
<tr>
<td><strong>Own time</strong></td>
<td>Total time for the POU call, excluding the time for all calls that the POU and its called POUs have performed</td>
</tr>
<tr>
<td><strong>Average time</strong></td>
<td>Average total time measured in the current cycle for the call of the selected POU</td>
</tr>
<tr>
<td><strong>Min. time</strong></td>
<td>Minimum total time measured in the current cycle for the call of the selected POU</td>
</tr>
<tr>
<td><strong>Max. Time</strong></td>
<td>Maximum total time measured in the current cycle for the call of the selected POU</td>
</tr>
<tr>
<td><strong>Standard deviation</strong></td>
<td>Standard deviation of the call time in the current cycle from the average time</td>
</tr>
</tbody>
</table>

See also

- Chapter 1.7.4.5.1.3 “Tab ‘Online’” on page 3697

**Command 'Delete Snapshot’**

Symbol: ✗

**Function:** The selected snapshot is deleted.

**Call:** Context menu of a selected snapshot entry on the “Snapshots” tab

The deleted snapshot cannot be restored.

See also

- Chapter 1.7.4.5.1.4 “Tab ‘Snapshots’” on page 3701

**Command 'Edit Snapshot’**

Symbol: 📝

**Function:** The name and description of the selected snapshot can be modified.

**Call:** Context menu of a selected snapshot entry on the “Snapshots” tab

The command opens the “Edit Snapshot” dialog for changing the name and description. The dialog corresponds to the “Save Snapshot” dialog.

See also

- Chapter 1.7.4.5.1.4 “Tab ‘Snapshots’” on page 3701
- Chapter 1.7.4.5.1.3 “Tab ‘Online’” on page 3697

**Command 'Print Table’**

Symbol: 📈

**Function:** The command opens the default dialog for configuring a print job. You can print the table with the current measurement results of the code coverage.

**Call:** Context menu of a code coverage measurement on the “Online” tab
See also
● § Chapter 1.7.4.5.1.3 “Tab ‘Online’” on page 3697

Command 'Times'

**Function:** The command opens the dialog "Times for <POU>". The measured time is displayed for each POU call.

**Call:** On the “Online” tab, the context menu of a selected line of a POU which is called multiple times on the “Call Tree” subtab

Dialog 'Times for <POU>

The dialog shows the measured duration for each individual POU call at the selected position in the call tree.

Additional information includes the following:

- **“POU Type”:** For example, Function
- **“Code Size”:** POU size (in bytes), for example, 360 Bytes
- **“Location”:** Location where the POU is defined (application, library, etc.), for example, standard, 3.5.5.0 (system)

See also
● § Chapter 1.7.4.5.1.3 “Tab ‘Online’” on page 3697

Command 'Export'

**Function:** The command opens the standard dialog for saving a file to the local file system. You can save the current list of Profiler results as a CSV file.

**Call:** On the “Online” tab, the context menu on the “List” subtab

See also
● § Chapter 1.7.4.5.1.3 “Tab ‘Online’” on page 3697

Command 'Copy Table'

**Symbol:** ⌘

**Function:** The command saves the current list of profiler results to the clipboard.

**Call:** On the “Online” tab, the context menu on the “List” subtab

See also
● § Chapter 1.7.4.5.1.3 “Tab ‘Online’” on page 3697
NOTICE!
The implicitly added IEC code for a Profiler measurement method, as well as
the breakpoints for POU's in the watch list, result in a change and slows down
the evaluated program.

When a profiling or code coverage measurement is active, an online is gener-
ally not possible. Moreover, some online features are not available, such as flow
control and breakpoints. There are no restrictions for the watch list in this
respect.

Enabling or disabling a Profiler measurement method, or changing any settings
in the Profiler editor each requires a download. Then an online change is no
longer possible. There are no restrictions for the watch list in this respect.

For each POU and each POU instance of an application in this watch list, CODESYS Profiler
detects at runtime the frequency and duration that it is called.

You can fill the list with the required POUs and instances in offline mode or online mode:
- By dragging a POU from the device tree or editor to the list
- By specify an entry manually (supported by the Input Assistant)
- By clicking “Add to Profiler Watch List” in the context menu of a POU in the “Devices” or
“POUs” view

<table>
<thead>
<tr>
<th>“Call”</th>
<th>Name of the POU whose calls are analyzed (example: PLC_PRG). For input options, see above.</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Instance”</td>
<td>Name POU instance</td>
</tr>
<tr>
<td>“Application”</td>
<td>Path of the application for which the analysis should be performed (example: Device_XY.Application)</td>
</tr>
<tr>
<td>“Count”</td>
<td>Number of calls since the measurement was started (example: “24”)</td>
</tr>
<tr>
<td>“Duration &lt;unit&gt;”</td>
<td>Total duration of all calls in the current cycle (example: 51 µs)</td>
</tr>
<tr>
<td>“Max. Duration &lt;unit&gt;”</td>
<td>Maximum duration of calls in a cycle since beginning the measurement (example: 764 µs)</td>
</tr>
<tr>
<td>“Min. Duration &lt;unit&gt;”</td>
<td>Minimum duration of calls in a cycle since beginning the measurement (example: 200 µs)</td>
</tr>
<tr>
<td>“Sum Duration &lt;unit&gt;”</td>
<td>Total duration of all calls since beginning the measurement (example: 204876 µs)</td>
</tr>
<tr>
<td>“Avg. Duration &lt;unit&gt;”</td>
<td>Average duration of calls in a cycle since beginning the measurement (example: 48 µs)</td>
</tr>
</tbody>
</table>

Context menu in online mode or offline mode when the focus is in a table cell:
- “Reset” Resets the measured times of the selected lines to zero
- “Delete” Deletes the selected lines
“Open POU” | Opens the POU of the selected lines in the editor
---|---
Submenu “Display Mode” | For setting the time data type or the unit directly for displaying the duration:
- “LTIME” (example: LTIME#43ms13µs44ns)
- “TIME” (example: T#201ms)
- “ms” (example: 201)
- “µs” (example: 1347)
- “ns” (example: 597811)

See also
- Chapter 1.7.4.2.2 “Using the Profiler Watch List” on page 3692
- Chapter 1.7.4.5.2.10 “Command ‘Add to Profiler watch list’” on page 3706

Command ‘Add to Profiler watch list’
Symbol: 
**Function:** This command inserts the selected POU to the Profiler watch list.
**Call:** Context menu of the “Devices” or “POUs” view
**Requirement:** A POU is selected in the “Devices” or “POUs” view.
See also
- Chapter 1.7.4.2.2 “Using the Profiler Watch List” on page 3692

1.7.5 Drive composer pro integration

Drive Composer Pro is a start-up and maintenance tool for ABB’s common architecture drives. The tool is used to view and set drive parameters, and to monitor and tune process performance.

Drive Composer Pro provides:
- Setting parameters,
- taking local control of the drive from the PC,
- event logger handling
- control diagrams,
- fast monitoring,
- working with multiple drives on the PC tool network,
- macro script editing for parameters and much more.

1. Add “Drive Composer Pro” object into the tree via add object dialog.
2. Open the “Drive Composer Pro” with double-click on the object.

In the following section important functions are described.
1. Import of FSO backup files (*.dcsafety) and Drive Parameters backup files (*.dcparamsbak) into Automation Builder project via the Drive Composer Pro object in the device tree.

2. View of integrated FSO backup files and Drive Parameters backup files in Automation Builder project - refer to image below.

Drive Composer Pro can't be launched directly with integrated “FSO backup files” but they have to be loaded manually via context menu on the drive in Drive Composer Pro → “Safety Settings”.
1. Select the FSO and Drive Parameters backup files.
2. Export the selected file by clicking [Export].
   ➞ Select the desired storage path.
Remove of backup files

1. Select the FSO and Drive Parameter backup files from Automation Builder project.
2. Remove the selected files by clicking [Remove].
1. Open the “Crane_follower12.dcparsmbak” with double-click.

2. The “Drive Composer Pro” starts automatically.

Standard Drive Parameter backup files (*.dcparsmbak) are automatically displayed under “File Drives”.

3. Saved changes in the standard drive parameter backup file are automatically updated in the Automation Builder project.

### 1.7.6 CODESYS SVN

SVN integration in CODESYS

CODESYS SVN allows for the development of CODESYS projects under version control by Apache™ Subversion®. CODESYS SVN provides an SVN client integrated in CODESYS. The objects of your project are versioned in a central SVN repository.

As a rule, the SVN repository should be created in a server configuration and located on a server. For testing purposes, you can create a local SVN repository where you can access per file://.

Professional Version Control requires a valid license and can be installed using the Automation Builder Installer or the Automation Builder Installation Manager.
1.7.6.1 Getting Started

The following steps are required in order to develop your CODESYS project with CODESYS SVN with version control by Apache™ Subversion®:

1. Install the CODESYS SVN package in CODESYS.
2. Install an SVN server.
3. Create an SVN repository.
4. Open your CODESYS project in CODESYS.
5. Import the CODESYS project into the SVN project archive.
   ➔ The CODESYS project is saved in the SVN repository.
6. To edit and further develop the project with SVN version control, the project is edited in CODESYS and then committed to the SVN repository.

A detailed description of these individual steps is located in the following sections.
See also
● § Chapter 1.7.6.3 “Using an SVN Repository” on page 3711
● § Chapter 1.7.6.4 “Using Working Copies” on page 3713

1.7.6.2 Version control

What is version control? Apache™ Subversion® (SVN) is a tool for version and revision management of current and previous versions of files, such as source code, websites, and documentation. Apache™ Subversion® is a registered trademark of the Apache Software Foundation.

Revision management (also known as version control, version management, and source code management) is the management of changes to documents, programs, and other information that is stored as computer files. Version control is employed frequently in software development when a team of employees works on the same files.

Tasks
● Co-writing of changes in revisions: At any time, you can show who made which changes at which time.
● Restoring of old revisions of individual files: At any time, you can reverse accidental changes to files.
● Archiving of special revisions of a project: At any time, you can revert to older versions.
● Coordination of common access of developers to data
● Development of a project simultaneously in multiple branches

Script Engine SVN Add-on API
CODESYS SVN provides a scripting-interface for SVN.
See also
● Script Engine SVN AddOn API Reference

1.7.6.3 Using an SVN Repository

An SVN repository usually saves information as a file system tree, a hierarchy of files, and directories. Any number of clients connects to the SVN repository and reads or writes changes to the files in revisions.
Creating an SVN repository

**NOTICE!**
Consult with your IT specialists for more information, for example how to create an SVN repository. For production purposes, we recommend a strictly dedicated administrative SVN server.

We recommend that you create the suggested default directory structure in the SVN repository.

See also


---

Creating an SVN repository for testing purposes

**NOTICE!**
Use the file:// access method for testing purposes only.

For testing purposes, you can create a local SVN repository without installing your own server. The SVN repository is accessed via file:// and provides the same functionality as a server.

Requirement: The SVN client TortoiseSVN 1.9 is installed on the development system.

1. Create a new, empty folder on your local file system. The test repository will be created there.
   - **Example:** D:\SVN repository
2. Click **“TortoiseSVN ➔ Create repository here”**.
   - The dialog “Create repository” opens.
3. Click “Create directory tree”.
   - The SVN repository is created.

See also
- Documentation TortoiseSVN
  [Documentation TortoiseSVN](#)

---

Creating a test repository with TortoiseSVN

Accessing the SVN repository

<table>
<thead>
<tr>
<th><strong>Table 797: SVN repository URLs</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>file:///</td>
</tr>
<tr>
<td>http://</td>
</tr>
<tr>
<td>https://</td>
</tr>
<tr>
<td>svn://</td>
</tr>
<tr>
<td>svn+ssh://</td>
</tr>
</tbody>
</table>
Import the project into the SVN repository.

1. Open the CODESYS project that you want to save in the SVN repository.
   
   ⇒ Example: A.project is open.

2. Click “Project ➔ SVN ➔ Import project to SVN”.
   
   ⇒ The “Browse SVN repository” dialog opens.

3. Select the directory file:///D:/SVN repository/trunk in the directory tree.

4. Select the command.
   
   ⇒ The “Create remote directory” dialog opens.

5. Specify the URL for the new directory.
   
   Note: Because the new directory should contain the CODESYS project, specify the project name with extension here.
   
   ⇒ file:///D:/SVN20repository/trunk/A.project

6. Click “OK” to close the dialog.

7. Select the new project and click “OK” to exit the “Browse SVN repository” dialog.
   
   ⇒ The “Import Project to SVN” dialog opens. The directory file:///D:/SVN repository/trunk/A.project is specified in “URL of SVN repository”.

See also

● § Chapter 1.7.6.5.1 “Overlay Icons” on page 3714

1.7.6.4 Using Working Copies

Checking out a project

You can copy CODESYS projects to your development system that are saved in the SVN repository.

1. Open CODESYS.

2. Click “Project ➔ SVN ➔ Checkout”.
   
   ⇒ The “Checkout” dialog opens.

3. Specify the URL of the SVN repository and select a project in the SVN repository tree.
   
   If a CODESYS project has the extension .project or _project, then it is recognized automatically as a project at checkout. If it has the extension .library or _library, then it is recognized as a library project.

4. In “Checkout to”, specify the name and location of the working copy on your development system.

5. Click “OK” to close the dialog.
   
   ⇒ The project opens in CODESYS. In the object tree of the project, the SVN link is shown with overlaid icons. Now the project is saved as a working copy on your development system.

See also

● § Chapter 1.7.6.5.1 “Overlay Icons” on page 3714
Update the working copy before you start editing, especially if the project is revised by a team. This is how you avoid conflicts.

1. Open the working copy.
2. Click “Project ➔ SVN ➔ Update project” (symbol: 🔄).  
   ⇒ You working copy is current.
3. Revise your project.
4. Click “Project ➔ SVN ➔ Edit SVN working copy”.  
   ⇒ The dialog opens. There you can browse your changes.
5. Close the dialog.
6. If necessary, you can click “SVN ➔ Revert” in the context menu.  
   ⇒ The file is reverted back to the base revision and your changes are discarded.
7. If necessary, you can click “Compare” in the context menu of an edited object.  
   ⇒ The compare dialog opens. You can resolve any conflicts here.
8. Close the compare dialog.
9. Click “Project ➔ SVN ➔ Commit project” (symbol: ✨).  
   ⇒ The “Commit” dialog opens.
    ⇒ Your changes are saved in the SVN repository as a revision with a revision number.

See also
- Chapter 1.7.6.5.2.1 “Command ‘SVN Repository Browser’” on page 3716

For projects in version CODESYS SVN V4.1.0.0 and later, the working directory (working copy) has a new format.

If you open a project that was created with V4.0.4.0 or earlier, then the project is updated automatically to the new format when it is opened.

If you open a project that was created with V4.0.4.0 or earlier and the project is based on an older SVN version of 1.7.x or earlier, then you are prompted whether or not CODESYS should update the format. If you decline the update, then the SVN link of the project is deactivated. You can still load and edit the project.

The update does not have an effect on saving to the SVN server. You can also checkout projects with earlier versions of the client. The new format affects only the local working directory.

See also

1.7.6.5 Reference, User Interface
1.7.6.5.1 Overlay Icons

Every object in CODESYS has a status value in the SVN repository. This status value is displayed in the object tree (in the “POUs”, “Devices”, or “Modules” views) for each object by overlay icons.
Table 798: Overlay icons

<table>
<thead>
<tr>
<th>Icon</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>+</td>
<td>Object is planned to be added to the SVN repository.</td>
</tr>
<tr>
<td>▲</td>
<td>Object conflicted</td>
</tr>
<tr>
<td>×</td>
<td>Object deleted</td>
</tr>
<tr>
<td>■</td>
<td>Object modified</td>
</tr>
<tr>
<td>□</td>
<td>Object with modification in the metadata</td>
</tr>
<tr>
<td>☐</td>
<td>Object with modifications in the memory format</td>
</tr>
<tr>
<td>⊿</td>
<td>Object normal</td>
</tr>
<tr>
<td>⚠</td>
<td>Object write-protected (read-only)</td>
</tr>
<tr>
<td>⚭</td>
<td>Object locked</td>
</tr>
<tr>
<td>✗</td>
<td>Object with deleted subobjects</td>
</tr>
<tr>
<td>🐸</td>
<td>Object ignored on commit</td>
</tr>
<tr>
<td>👁</td>
<td>External object</td>
</tr>
<tr>
<td>☐</td>
<td>Ignored object</td>
</tr>
<tr>
<td>🌯</td>
<td>Unversioned object</td>
</tr>
<tr>
<td>🗿</td>
<td>Object with modified subobjects</td>
</tr>
<tr>
<td>☢</td>
<td>The object is not saved in the SVN repository. It will be created again when loaded from SVN.</td>
</tr>
<tr>
<td>🌋</td>
<td><code>SVN_VERSION_INFO</code> temporarily unavailable, for example as with interface libraries</td>
</tr>
<tr>
<td>🚂</td>
<td>The status of the object is not updated.</td>
</tr>
<tr>
<td>⚡</td>
<td>The object was modified on the server (Update available).</td>
</tr>
<tr>
<td>🚨</td>
<td>The object was locked on the server by another user (or in another working directory).</td>
</tr>
<tr>
<td>⚠️</td>
<td>Tree conflict by changes to the structure of the project</td>
</tr>
</tbody>
</table>
1.7.6.5.2 Commands

1.7.6.5.2.1 Command 'SVN Repository Browser'........................................... 3716
1.7.6.5.2.2 Command 'Edit SVN working copy'........................................... 3717
1.7.6.5.2.3 Command 'Import project to SVN'........................................... 3720
1.7.6.5.2.4 Command 'Checkout'.............................................................. 3721
1.7.6.5.2.5 Command 'Commit', Command 'Commit project'............................ 3722
1.7.6.5.2.6 Command 'Compare'............................................................... 3725
1.7.6.5.2.7 Command 'Compare with HEAD revision'.................................... 3725
1.7.6.5.2.8 Command 'Compare with revision'............................................ 3725
1.7.6.5.2.9 Command 'Compare to remote project...'.................................... 3726
1.7.6.5.2.10 Command 'Include externals to project', Command 'Include externals'................................................................. 3727
1.7.6.5.2.11 Command 'Ignore on commit'.................................................. 3729
1.7.6.5.2.12 Command 'SVN Info'............................................................... 3729
1.7.6.5.2.13 Command 'Show properties'................................................... 3730
1.7.6.5.2.14 Command 'Get lock'............................................................... 3730
1.7.6.5.2.15 Command 'Steal locks'............................................................ 3731
1.7.6.5.2.16 Command 'Release lock'......................................................... 3731
1.7.6.5.2.17 Command 'Release locks recursively'...................................... 3731
1.7.6.5.2.18 Command 'Show log', Command 'Show project log'........................ 3731
1.7.6.5.2.19 Command 'Revert', Command 'Revert project'............................ 3733
1.7.6.5.2.20 Command 'Revert to revision', Command 'Revert project to revision'................................................................. 3734
1.7.6.5.2.21 Command 'Update', Command 'Update project'............................. 3734
1.7.6.5.2.22 Command 'Update to revision'.................................................. 3735
1.7.6.5.2.23 Command 'Disconnect project from SVN'.................................... 3736
1.7.6.5.2.24 Command 'Switch'................................................................. 3736
1.7.6.5.2.25 Command 'Un-Ignore on commit'............................................. 3737
1.7.6.5.2.26 Command 'SVN Cleanup'......................................................... 3737
1.7.6.5.2.27 Command 'Clear authentication data'........................................ 3738
1.7.6.5.2.28 Command 'Merge changes'....................................................... 3738
1.7.6.5.2.29 Command 'Connect to existing project'..................................... 3739
1.7.6.5.2.30 Command 'Resolve conflict'.................................................... 3740
1.7.6.5.2.31 Command 'Work in offline mode'.............................................. 3740
1.7.6.5.2.32 Command 'Copy (Branch/Tag)'............................................... 3740
1.7.6.5.2.33 Command 'Pending changes'.................................................... 3741

Command 'SVN Repository Browser'

Symbol: 

Function: This command opens the SVN repository browser. The contents of an SVN repository is shown in a tree structure here. You can search through the repository in the browser.

Call: Menu bar: “Project ➔ SVN”.

Depending on the selected object, the following commands are available in the context menu:

- “Show log”
- “Checkout”
- “Create folder”
- “Copy to”
- “Rename”
- “Delete”

Double-clicking the object with the right mouse button opens the log dialog.
**Dialog 'SVN Repository Browser'**

<table>
<thead>
<tr>
<th>“URL”</th>
<th>URL in SVN repository</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Example: <a href="https://svnserver/repository/trunk/ControlABC.project">https://svnserver/repository/trunk/ControlABC.project</a></td>
</tr>
<tr>
<td></td>
<td>Tip: As soon as a valid SVN repository is specified, you can browse and select a specific project by means of the adjacent button.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><img src="image" alt="button" /></th>
<th>Opens the dialog “Select revision”. The button is labeled with the currently selected revision:</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="icon" /></td>
<td>“HEAD”: Top revision (latest). Preset</td>
</tr>
<tr>
<td><img src="image" alt="icon" /></td>
<td>“3”: Revision number of the selected revision</td>
</tr>
<tr>
<td><img src="image" alt="icon" /></td>
<td>“23.12.2016 11:59:59 (UTC)”: Change date of the selected revision (UTC)</td>
</tr>
<tr>
<td><img src="image" alt="icon" /></td>
<td>Note: The dialog provides the same options as the “Revision” group.</td>
</tr>
</tbody>
</table>

| ![icon](image) | Updates the browser view by rescanning the SVN repository. |
| ![icon](image) | Navigates the URL address up by one folder. |

**Left area**

Directory tree in the SVN repository. Project nodes are shown in bold.

![Diagram](image)

Note: In this view, you can directly edit the project name and the name of the superordinate folder.

**Right area**

List of objects of the selected directory

**“Close”**

Closes the dialog

See also

- ![icon](image) Chapter 1.7.6.5.3.3 “Dialog 'Select revision’” on page 3745

**Command 'Edit SVN working copy'**

**Symbol:** ![icon](image)

**Function:** This command opens the dialog “Edit SVN working copy” and displays the working copy in a browser from the SVN view.

**Call:** Menu bar: “Project ➔ SVN”. 
The functionality of the browser allows for:

- Access to and actions on objects that are not displayed in the “Devices” view.
- Actions on objects that can lead to exceptions in the “Devices” view.
- Editing of global objects that are modified, in conflict, or blocked.

**Dialog ‘Edit SVN working copy’**

<table>
<thead>
<tr>
<th>Table 799: “Edit SVN working copy: &lt;project name&gt; - &lt;project URL&gt;”</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>“Path in SVN repository”</strong></td>
</tr>
<tr>
<td><strong>“Name of object”</strong></td>
</tr>
<tr>
<td>Example: Application</td>
</tr>
<tr>
<td><strong>“Node type”</strong></td>
</tr>
<tr>
<td><strong>“Text status”</strong></td>
</tr>
<tr>
<td>“modified”</td>
</tr>
<tr>
<td>“added”</td>
</tr>
<tr>
<td>“deleted”</td>
</tr>
<tr>
<td>“non-versioned”</td>
</tr>
<tr>
<td>“Conflicted”</td>
</tr>
<tr>
<td><strong>“Property status”</strong></td>
</tr>
<tr>
<td>“modified”</td>
</tr>
<tr>
<td>“added”</td>
</tr>
<tr>
<td>“deleted”</td>
</tr>
<tr>
<td>“Conflicted”</td>
</tr>
<tr>
<td>“normal”</td>
</tr>
<tr>
<td><strong>“Revision”</strong></td>
</tr>
<tr>
<td><strong>“Conflict information”</strong></td>
</tr>
<tr>
<td><strong>“Lock”</strong></td>
</tr>
<tr>
<td>Example: b.mayer</td>
</tr>
<tr>
<td><strong>“Lock comment”</strong></td>
</tr>
<tr>
<td><strong>“URL”</strong></td>
</tr>
</tbody>
</table>

**Table 800: Menu commands**

<table>
<thead>
<tr>
<th>Menu commands</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Select ➔ All</td>
<td>Selects all files.</td>
</tr>
<tr>
<td>Select ➔ None</td>
<td>Deselects all files.</td>
</tr>
<tr>
<td>Select ➔ Modified</td>
<td>Selects the modified files.</td>
</tr>
<tr>
<td>Select ➔ Conflicted</td>
<td>Selects the conflicted files.</td>
</tr>
<tr>
<td>Select ➔ Locked</td>
<td>Selects the locked files.</td>
</tr>
<tr>
<td>Update ➔ Project</td>
<td>Updates the working copy. Changes made by others are added from the SVN repository to your working copy.</td>
</tr>
<tr>
<td>Command</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>“Update  ➔ Selected nodes”</strong></td>
<td>Updates only the selected files.</td>
</tr>
<tr>
<td><strong>“Update  ➔ Selected nodes and children”</strong></td>
<td>Updates the selected files and subordinate files.</td>
</tr>
<tr>
<td><strong>“Reset”</strong></td>
<td>Discards your changes to the working copy. Then the object corresponds to the revision in the repository.</td>
</tr>
<tr>
<td><strong>“Delete  ➔ Selected nodes”</strong></td>
<td>Deletes the selected objects from the working copy.</td>
</tr>
<tr>
<td><strong>“Commit”</strong></td>
<td>Commits your changes to the SVN repository. Any locked objects will be unlocked.</td>
</tr>
<tr>
<td><strong>“Commit  ➔ Project”</strong></td>
<td>Commits all files in the project.</td>
</tr>
<tr>
<td><strong>“Commit  ➔ Selected nodes”</strong></td>
<td>Commits only the selected files.</td>
</tr>
<tr>
<td><strong>“Commit  ➔ Selected nodes and children”</strong></td>
<td>Commits the selected files and subordinate files.</td>
</tr>
<tr>
<td><strong>“Locks  ➔ Revalidate all”</strong></td>
<td>Checks the validity of locks in the working copy. Any invalid locks will be unlocked.</td>
</tr>
<tr>
<td><strong>“Locks  ➔ Release locks”</strong></td>
<td>Releases the lock.</td>
</tr>
<tr>
<td><strong>“Locks  ➔ Acquire locks”</strong></td>
<td>Locks the object from editing by others.</td>
</tr>
<tr>
<td><strong>“Locks  ➔ Steal locks”</strong></td>
<td>Locks the file for you and removes the lock of another user. Tip: Avoid stealing a lock because the changes made by another user can be lost.</td>
</tr>
<tr>
<td><strong>“Conflicts  ➔ Mark as resolved”</strong></td>
<td>Indicates a displayed conflict in the SVN repository as marked and resolved. Note: Select the command if you edited and resolved the displayed conflict. Then you can commit changes again.</td>
</tr>
<tr>
<td><strong>“Conflicts  ➔ Resolve using theirs”</strong></td>
<td>Resolves the conflict: In the SVN repository, the changes are accepted that were committed by other users. Your changes are discarded.</td>
</tr>
<tr>
<td><strong>“Conflicts  ➔ Resolve using mine”</strong></td>
<td>Resolves the conflict: In the SVN repository, the changes to your working copy are accepted and the changes by other users are discarded.</td>
</tr>
<tr>
<td><strong>“Show log”</strong></td>
<td>Opens the dialog “Log - Application”. The history of the selected node is shown here. The previous revisions are displayed with the respective actions.</td>
</tr>
<tr>
<td><strong>“Change location”</strong></td>
<td>Changes the storage location of the selected object within the working copy. Example: You can resolve a tree conflict by saving the local object to another location. Then update the parent object to apply it to the locked children.</td>
</tr>
<tr>
<td><strong>“Update”</strong></td>
<td>Updates the browser view by rescanning the working copy.</td>
</tr>
<tr>
<td><strong>“Cleanup”</strong></td>
<td>Executes an SVN cleanup operation on the working copy.</td>
</tr>
</tbody>
</table>
See also

- Chapter 1.7.6.5.2.21 “Command 'Update', Command 'Update project'” on page 3734
- Chapter 1.7.6.5.2.19 “Command 'Revert', Command 'Revert project'” on page 3733
- Chapter 1.7.6.5.2.5 “Command 'Commit', Command 'Commit project'” on page 3722
- Chapter 1.7.6.5.2.30 “Command 'Resolve conflict', Command 'Resolve project conflict'” on page 3740
- Chapter 1.7.6.5.2.14 “Command 'Get lock'” on page 3730
- Chapter 1.7.6.5.2.16 “Command 'Release lock'” on page 3731
- Chapter 1.7.6.5.2.15 “Command 'Steal locks'” on page 3731
- Chapter 1.7.6.5.2.24 “Command 'Switch'” on page 3736
- Chapter 1.7.6.5.2.18 “Command 'Show log', Command 'Show project log'” on page 3731
- Chapter 1.7.6.5.2.26 “Command 'SVN Cleanup'” on page 3737

**Command 'Import project to SVN'**

**Symbol:** 🔄

**Function:** This command opens the dialog “Import project to SVN” in order to import a CODESYS project to the SVN repository.

**Call:** Menu bar: “Project ➔ SVN”.

**Requirement**

- You have access to an SVN repository and you know its URL.
- You have read access to the entire project.

See also

- User and access management in Protect and save project

**Dialog 'Import Project to SVN'**

<table>
<thead>
<tr>
<th>“URL of SVN repository”</th>
<th>URL of the SVN repository with the new project folder where the files are imported. Example: <a href="https://svnserver/repository/trunk/ControlABC.project">https://svnserver/repository/trunk/ControlABC.project</a></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Tip: When importing libraries, specify the extension .library or _library. For projects, specify the extension .project or _project. Then the project type is recognized automatically at checkout and the options are set accordingly in the “Checkout” dialog.</td>
</tr>
<tr>
<td>“Import message”</td>
<td>Text for use as log message Example: Control project for customer A</td>
</tr>
<tr>
<td>“Recent Messages”</td>
<td>Opens the “Recent Messages” dialog. There you can reuse the last log messages.</td>
</tr>
<tr>
<td>“Generate SVN_VERSION_INFO”</td>
<td>If the object SVN_VERSION_INFO is not created automatically during the import operation. Therefore, the project does not obtain any global constants or variables for the project metadata.</td>
</tr>
<tr>
<td>“OK”</td>
<td>Creates the current project in the SVN repository and imports the project objects. The local project in CODESYS Development System is linked to the SVN repository. Overlay icons show this in the object trees.</td>
</tr>
</tbody>
</table>
Command 'Checkout'

Symbol: 

**Function:** This command opens the “Checkout” dialog. Here you can checkout a project stored in the SVN repository as a working copy.

**Call:** Menu bar: “Project ➔ SVN”.

**Dialog 'Checkout'**

<table>
<thead>
<tr>
<th>Table 801: “URL of SVN repository”</th>
</tr>
</thead>
<tbody>
<tr>
<td>URL of the project in the SVN repository</td>
</tr>
<tr>
<td>Example: <a href="https://svnserver/repository/trunk/ControlABC.project">https://svnserver/repository/trunk/ControlABC.project</a></td>
</tr>
<tr>
<td>Tip: As soon as a valid SVN repository is specified, you can click the adjacent button or use the options to browse in “Revision” and select a specific project.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Table 802: “Checkout to”</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Name” Name of the working copy</td>
</tr>
<tr>
<td>Example: ControlABC.project</td>
</tr>
<tr>
<td>“Location” Storage location of the working copy</td>
</tr>
<tr>
<td>Example: /D:/svn/repository/trunk/ControlABC.project</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Table 803: “Checkout as”</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Project” The project is saved as a CODESYS project &quot;&lt;project name&gt;.project&quot;.</td>
</tr>
<tr>
<td>“Library” The project is saved as a CODESYS library file &quot;&lt;project name&gt;.library&quot;.</td>
</tr>
<tr>
<td>“Auto-detect” CODESYS attempts to recognize the project type by means of the extension. The current implementation checks whether the URL of the project ends with &quot;_library&quot; or &quot;.library&quot;. In this case, the project is recognized as a library or a project.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Table 804: “Checkout options”</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Omit externals”: [✓] Externals (external objects) are not copied to the working directory.</td>
</tr>
</tbody>
</table>
Table 805: “Revision”

For a description, refer to the section “Dialog ‘Select revision’”.

Note: The group provides the same options as the “Revision” dialog.

| “OK”          | Checks out the project from the SVN repository, saves it locally to the specified location, and opens it in CODESYS as the primary project. |

If files were encrypted when imported to the SVN repository, or if they have been committed, then note the following:

When committing to the SVN repository, the information about an encrypted project file is included. However, the type of encryption is not included (password, Wibu security key, X509 certificate). Therefore, it may be necessary to encrypt the working copy again in the project settings. In this case, a dialog opens when exiting the command to notify you of this. Then you are able to switch directly to the project settings.

See also
- Chapter 1.7.6.5.3.3 “Dialog ‘Select revision’” on page 3745
- Chapter 1.7.6.5.2.1 “Command ‘SVN Repository Browser’” on page 3716
- “Version control with Subversion”, Section “Revision identifier”

Command ‘Commit’, Command ‘Commit project’

Symbol: ✅

Function: This command commits changes that were made in CODESYS to the SVN repository. The “Commit” dialog opens for this purpose.

Call:
- Context menu: “SVN” to commit this object exactly.
- Menu bar: “Project ➔ SVN ➔ Commit project” to commit all changes in the project at the same time.

Requirement: At least one object was edited. The edited object is overlaid in the object tree with the icon.

When you execute the command, the block on the objects to be committed is lifted automatically.

Dialog ‘Commit’

Table 806: “Commit to: <URL project/object>”

<table>
<thead>
<tr>
<th>URL in SVN repository</th>
<th>Example: file://D:/SVN repository/trunk/ControlABC.project</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Log message”</td>
<td>Comment the change in a log message.</td>
</tr>
<tr>
<td></td>
<td>Example: Bug fix error 123</td>
</tr>
<tr>
<td>“Recent Messages”</td>
<td>Opens the dialog “Recent Messages” to display the last log messages. You can click a log message to accept it.</td>
</tr>
</tbody>
</table>
### Table 807: “Changes made (double-click on object for compare, right-click on object for more operations)”

| **Object** | List of objects that were changed and can therefore be committed. The SVN URLs mirror the hierarchy of the object in the SVN repository. The objects are highlighted in color according to the object status:  
  - Blue: Modified  
  - Green: Added  
  - Dark red: Deleted  
  - Red: Conflicted  
  - Black: Non-versioned (not in SVN repository)  
  - Gray: Excluded from commit  
  - Note: These objects are displayed when the option “Show non-versioned objects” is activated.  
  - Note: This is the case when the option “Ignore during commit” is activated.  
  - Double-click an object in order to open the compare dialog. The revision of the working copy is compared with the base revision. The compare dialog also opens when you click “Compare” in the context menu.  
  - Right-click an object in order to open the context menu.  
  - Note: This opens a list of objects when “Commit project” was clicked. For the “Commit” command applied to a specific object, only this object is listed (if modified) and its modified children.  
  - “Object”: The object is activated for the commit.  
  - Example: ![Device|Plc Logic|Application|PLC_PRG](image)  |
| **Text status** | Object status in CODESYS  
  - “Modified”  
  - “Added”  
  - “Deleted”  
  - “Non-versioned”  
  - “Conflicted”  |
| **Property status** | Status of the metadata of the object  
  - “Modified”  
  - “Added”  
  - “Deleted”  
  - “Conflicted”  
  - “Normal”  |
| **Lock** | If the object has a lock, then it is shown here the user who applied the lock.  
  - Example: b.mayer  |
| **Description** | Display of the log message  |
| **Select/deselect all** | ☑: All objects in the list are activated.  |

- “Keep locks”: ☑: Your locked object keeps remains in lock status after the commit.
- “Keep change lists”: ☑: The change list also remains after the commit.
- ☓: The change list is not deleted after the commit.
- “Update after commit (recommended)”: ☑: The object/project is updated after the commit. Select this check box to ensure that the project is up-to-date and to prevent conflicts resulting from mixed revisions of working copies.

See also: SVN help
| Button “Update project” | Updates the project.  
| Tip: Prevent conflicts by committing a previously updated project/object. |
| **“OK”** | Checks the working copy first. Starts the commit of changes when the working copy is current.  
| Keyboard shortcut [Ctrl]+[Enter] | Opens a dialog when the working copy is outdated. You can then select the following:  
| ● “Abort the commit, I want to investigate the issue.”  
| ● “Yes, I want to update this project now.”  
| ● “Continue with the commit, I know what I do.”  
| Note: The history of the commit is displayed in the “Messages” view.  
| The messages are highlighted in color:  
| ● Blue: Commit a change  
| ● Green: Add an object  
| ● Dark red: Delete/replace an object  
| ● Black: Other messages (summary) |

### Handling external objects

*If the external object is in the same SVN repository, then changes in this external object are listed in the commit dialog and committed together with the internal project. If an external object is in another SVN repository, then you are notified about changes in the external project and you have to commit these separately.*

*An external object has the “externals” property.*

See also

- § Chapter 1.7.6.5.2.6 “Command 'Compare’” on page 3725
- SVN help: [http://svnbook.red-bean.com/en/1.7/svn.basic.in-action.html#svn.basic.in-action.mixedrevs](http://svnbook.red-bean.com/en/1.7/svn.basic.in-action.html#svn.basic.in-action.mixedrevs)

### Context menu

*(right-click on object)*

| “Compare” | Opens the compare dialog to compare the working copy with the top-level revision. |
| “Compare with HEAD version” | Opens the compare dialog to compare the working copy with the HEAD revision. |
| “Compare with revision” | The list entries are highlighted in color according to the object status:  
| ● Blue: Modified  
| ● Green: Added  
| ● Dark red: Deleted  
| ● Red: Conflicted  
| ● Black: Non-versioned (not in SVN repository)  
| Note: These objects are displayed when the option “Show non-versioned objects” is activated.  
| ● Gray: Excluded from commit  
| Note: This is the case when the option “Ignore during commit” is activated for the object. |
| “Revert” | Discards your changes to the working copy. Then the object corresponds to the revision in the SVN repository. |
| “Show log” | Shows the version history of the selected object. |
“Properties” | Opens the “SVN Properties” dialog. The properties are displayed there and you can edit them.
---|---
Move to change list | Note: This command has not been implemented yet.

Command 'Compare'

Symbol: 

**Function:** This command opens a tab that shows the result of the comparison of your working copy and the BASE revision. The base revision is the top-level revision in the SVN repository.

**Call:**
- Menu bar: “Project ➔ SVN”.
- Context menu

**Requirement:** The object is versioned, it was modified locally, and it does not contain any conflicts.

Multiple tabs can be open at the same time with the comparison of different objects.

**Comparison by object type**

The comparison dialog makes use of the functionality of the CODESYS command “Project ➔ Compare”. In this way, objects are compared according to their object type.

See also
- Project compare with the 'Compare' command

Command 'Compare with HEAD revision'

Symbol: 

**Function:** This command opens a tab that shows the result of the comparison of your working copy and the HEAD revision. The HEAD revision is the top-level revision in the branch. You can revert specific changes that were committed to the HEAD revision.

**Call:** Context menu: “SVN”

**Requirement:** The object is versioned and not conflicted.

Multiple tabs can be open at the same time with the comparison of different objects.

**Comparison by object type**

The comparison dialog makes use of the functionality of the CODESYS command “Project ➔ Compare”. In this way, objects are compared according to their object type.

See also
- Project compare with the 'Compare' command

Command 'Compare with revision'

Symbol: 

---
**Function:** This command opens the “Project log” dialog or “Log - <object>” where the version history is displayed from the project or an object of the CODESYS project. Here you can select a revision. A tab opens and shows the result of the comparison of your working copy and the revision.

**Call:** Context menu: “SVN”

**Requirement:** The object is versioned and not conflicted.

Multiple tabs can be open at the same time with the comparison of different objects.

<table>
<thead>
<tr>
<th>Comparison by object type</th>
</tr>
</thead>
<tbody>
<tr>
<td>The comparison dialog makes use of the functionality of the CODESYS command “Project → Compare”. In this way, objects are compared according to their object type.</td>
</tr>
</tbody>
</table>

See also
- “Tab ‘Project log’, Dialog ‘Log - <object>’” on page 3732
- Project compare with the ‘Compare’ command

**Command 'Compare to remote project...’**

**Symbol:** 🌐

**Function:** This command opens the dialog “Select Remote Project for Comparison”.

**Call:** Menu bar: “Project → SVN”.

See also
- Project compare with the ‘Compare’ command

**Dialog 'Select Remote Project for Comparison’**

**Table 808: “URL of SVN repository”**

<table>
<thead>
<tr>
<th>URL of the project in the SVN repository that is compared.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example: file:///D:/SVN repository/trunk/ControlDEF.project</td>
</tr>
</tbody>
</table>

As soon as a valid SVN repository is specified, you can click the adjacent button or use the options to browse in “Revision” and select a project.

<table>
<thead>
<tr>
<th>15</th>
</tr>
</thead>
</table>
| The label on the button corresponds to the selected revision:
  - “HEAD”: Top revision (latest).
  - “15”: Revision number of the selected revision
  - “23.12.2016 11:59:59 (UTC)”: Change date of the selected revision (UTC)

After clicking the button, the dialog “Select revision” opens.

Note: The dialog provides the same options as the “Revision” group.

<p>| |</p>
<table>
<thead>
<tr>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Opens the dialog “Browse SVN repository” to search the SVN repository.</td>
</tr>
</tbody>
</table>

**Table 809: “Checkout options”**

| “Omit externals”: | ☑: External objects are not compared. |
Table 810: "Revision"

<table>
<thead>
<tr>
<th>Options for selecting a specific revision</th>
</tr>
</thead>
<tbody>
<tr>
<td>Note: the current valid selection is also displayed next to the SVN repository URL.</td>
</tr>
<tr>
<td>&quot;HEAD&quot;</td>
</tr>
<tr>
<td>● The HEAD revision is selected. This is the latest revision (top revision) within a branch.</td>
</tr>
<tr>
<td>&quot;Revision&quot;</td>
</tr>
<tr>
<td>● A specific revision is selected by the revision number. Example: 3</td>
</tr>
<tr>
<td>&quot;Date&quot;</td>
</tr>
<tr>
<td>● The specific revision is selected by the modification date. Example: 12/23/2016 11:59:59</td>
</tr>
<tr>
<td>&quot;Use UTC Time&quot;</td>
</tr>
<tr>
<td>✔ Modification date in universal time.</td>
</tr>
</tbody>
</table>

Table 811: "compare options"

<table>
<thead>
<tr>
<th>&quot;Ignore Whitespace&quot;</th>
</tr>
</thead>
<tbody>
<tr>
<td>✔ No comparison of whitespace characters. Semantically relevant whitespaces, such as in strings, are compared anyway.</td>
</tr>
<tr>
<td>&quot;Ignore Comments&quot;</td>
</tr>
<tr>
<td>✔ No comparison of comments.</td>
</tr>
<tr>
<td>&quot;Ignore Properties&quot;</td>
</tr>
<tr>
<td>✔ No comparison of properties. Folders, the property &quot;Exclude from build&quot;, and POU images are not compared.</td>
</tr>
<tr>
<td>See: Dialog 'Properties'</td>
</tr>
</tbody>
</table>

See also
- Chapter 1.7.6.5.2.1 “Command ‘SVN Repository Browser’” on page 3716
- Dialog 'Properties'

Command 'Include externals to project', Command 'Include externals'

Symbol: ➔

Function: These commands open the dialog “Include externals”.

Call:
- Menu bar: “Project ➔ SVN”.
- Context menu: “SVN”

Requirement: An object is selected in the object tree. The external objects are linked below that. If you have selected nothing or the project root directory, then the command “Include externals to project” is available. If you have selected an object, then the command “Include externals” is available.

The same external objects cannot be linked multiple times at different locations in the same project. This leads to problems in CODESYS because of conflicts with the internal identification of the object.

Dialog 'Include externals'
Table 812: “URL of SVN repository”

| **URL of the external object that is linked. The object to be linked is versioned and can have sub-objects.** |
| External objects are located at another location in the SVN repository than the project. It can even be in another SVN repository. |
| **Example:** file:///D:/SVN repo A/trunk/DSTest.project/GlobalTextList |
| **Note:** The objects that should be linked below the selected object must have a matching object type. For example, only a task can be linked below the “Task configuration” object. |

| **Opens the dialog “Select revision”** |
| Here you can select a revision. |
| The button is labeled with the currently selected revision: |
| ● **“HEAD”**: Top revision (latest). Preset |
| ● **“15”**: Revision number of the selected revision |
| ● **“23.12.2016 11:59:59 (UTC)”**: Change date of the selected revision (UTC) |
| **Note:** The dialog provides the same options as the “Revision” group. |

| **Opens the “SVN repository browser” dialog** |
| Here you can browse the SVN repository. |

Table 813: “Revision”

| Options for selecting a revision |
| Note: the current valid selection is also displayed in the buttons next to the SVN repository URL. |
| **“HEAD”** | ● Latest revision (top revision) selected in a branch. |
| **“Revision”** | ● A specific revision by the revision number. |
| Example: 3 |
| **“Date”** | ● A specific revision by the modification date. |
| Example: 12/23/2016 11:59:59 |
| **“Use UTC Time”**: | [✓]: Modification date in universal time. |

| **“OK”** |
| Adds the external object and its sub-objects with the property `svn:externals` to your project (below the selected object). The working copy is updated and the external object is overlaid with the symbol. |
| Example: [Source](external device Source) |
| **Note:** If the linking fails (for example when adding a device below a task configuration), then the complete operation fails and reverts back. |
| **Note:** Renaming or moving individual external objects is permitted only within an external tree, whereby it is not permitted to move the top object. |
| To move a complete tree, you have to remove it and link it to another location. |
“... You should seriously consider using explicit revision numbers in all of your externals definitions. Doing so means that you get to decide when to pull down a different snapshot of external information, and exactly which snapshot to pull. Besides avoiding the surprise of getting changes to third-party repositories that you might not have any control over, using explicit revision numbers also means that as you backdate your working copy to a previous revision, your externals definitions will also revert to the way they looked in that previous revision, which in turn means that the external working copies will be updated to match the way they looked back when your repository was at that previous revision. For software projects, this could be the difference between a successful and a failed build of an older snapshot of your complex codebase. ...”

This is a quote from: http://svnbook.red-bean.com/nightly/en/svn.advanced.externals.html.

Command 'Ignore on commit'

**Function:** This command identifies an object and adds it to the "ignore-on-commit" list. Then it is deactivated in the commit dialog by default.

**Call:** Menu bar: “SVN”

**Requirement:** At least one object is available that is not in the change list ignore-on-commit.

Objects of the "ignore-on-commit" list are overlaid with the symbol in the object tree. By default, they are not selected in the commit dialog, unless a dependency of a selected object requires it. These objects can always be selected manually in the dialog.

See also

- Chapter 1.7.6.5.2.25 “Command 'Un-Ignore on commit'” on page 3737

Command 'SVN Info'

**Function:** This command provides information about the selected object in the SVN repository. The "SVN Information" dialog opens for this purpose.

**Call:** Context menu: “SVN”

**Requirement:** A versioned object (with SVN link) is selected in the object tree.

Dialog 'SVN Information'

**Example**

Name: Device_4\Plc Logic\Application\PLC_PRG
URL: file:///D:/SVN repository/trunk/ControlABC.project/Device/Plc Logic/Application/PLC_PRG/svnobj
Repository Root: file:///D:/SVN repository/
Repository UUID: 185325d7-73eb-e54b-ab50-206aa23c8b42
Revision: 29
Node Kind: File
Schedule: Normal
Last Changed Author: a.mayer
Last Changed Rev: 8
Last Changed Date: 17.01.2017 12:33:51
Text Last Updated: 17.01.2017 12:33:51
Checksum: d5fb4d91ebea06f26bcb15942724d57932b6a3
Command 'Show properties'

Symbol: 

Function: This command opens the "SVN Properties" dialog. Here you can edit the properties of the versioned object.

Call: Context menu: “SVN”

Requirement: A versioned, unlocked object is selected.

Dialog 'SVN Properties'

Table 814: “properties for: <object name>”

<table>
<thead>
<tr>
<th>“Name”</th>
<th>Name of the property</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Example: myprop:customer-number</td>
</tr>
<tr>
<td></td>
<td>Note: SVN has some reserved properties. Example: svn:mime-type</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>“Value”</th>
<th>Example: 1234</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Double-click in the field to edit the value.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>“Add”</th>
<th>Opens a dialog to define another property with its value.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>“Remove”</th>
<th>Deletes the selected property.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>“Show binary properties”</th>
<th>☑: The binary properties are also displayed.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>“Reset”</th>
<th>Resets the changes displayed in green.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>“OK”</th>
<th>Accepts the changes.</th>
</tr>
</thead>
</table>

See also
- http://svnbook.red-bean.com

Command 'Get lock'

Symbol: 

Function: This command locks the object explicitly for you. The “Lock Message” dialog opens for this purpose.

Call: Context menu: “SVN”

Requirement: The versioned object is not locked (not overlaid with the symbol).

Dialog 'Lock Message'

<table>
<thead>
<tr>
<th>“Enter the reason why you lock the object:”</th>
<th>Lock message</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Example: Locked for processing task 123</td>
</tr>
</tbody>
</table>

Button “Recent Message” Shows message in the dialog that have already been used. There you select one in order to use the lock message.

<table>
<thead>
<tr>
<th>“Recursive”</th>
<th>☑: The object is locked with all subordinate child objects.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>“OK”</th>
<th>Locks the object</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>When the lock is successful, the object (in the object tree) is overlaid with the symbol.</td>
</tr>
</tbody>
</table>
Command 'Steal locks'

Symbol: ⚒

**Function:** This command steals the lock of the object. The “Lock Message” dialog opens for this purpose.

**Call:** Context menu: “SVN”

**Requirement:** The versioned object is locked by someone else (overlaid with the ⚒ symbol).

**Dialog 'Lock Message'**

<table>
<thead>
<tr>
<th>“Enter the reason why you lock the object:”</th>
<th>Lock message</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Example:</strong> a.mayer had to steal the lock because the changes need to be implemented so urgently.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>“Recent Message”</th>
<th>Shows message in the dialog that have already been used. There you select one in order to use the lock message.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>“Recursive”</th>
<th>☑: The lock is stolen by the object and all subordinate child objects.</th>
</tr>
</thead>
</table>

| “OK” | Steals the lock. When the stolen lock is successful, the object (in the object tree) is overlaid with the ⚒ symbol. |

Command 'Release lock'

Symbol: ⚒

**Function:** This command releases the lock of an object.

**Call:** “Context menu ➔ SVN”

**Requirement:** The object is locked.

Command 'Release locks recursively'

Symbol: ⚒

**Function:** This command releases the lock of an object explicitly with all of its subordinate objects.

**Call:** “Context menu ➔ SVN”

**Requirement:** The object is locked.

Command 'Show log', Command 'Show project log'

Symbol: ⚒

**Function:** These commands open the tab “Project log” or “Log - <object>”. The version history of the project or an object of the CODESYS project is displayed in the tab.

**Call:**
- Menu bar: “Project ➔ SVN”.
- “Context menu ➔ SVN”

If you select nothing or the base node in the object tree, then the history of the entire project is displayed (“Show project log”). If you select one or more objects, then the history of these elements is displayed (“Show log”).

Multiple tabs can be open at the same time with the version history of different objects.
Tab 'Project log', Dialog 'Log - <object>'

Upper area
- "Revision": Revision number
- "Author"
- "Date"
- "Message": Message entered at commit

List of all revisions of the project or the selected objects in the information. The first 100 revisions are displayed by default. The "Next 100" and "All" buttons are provided for displaying more or all revisions.

Several commands are available in the context menu of each revision. These context menu commands are described below.

Middle area
Display of the "Message" of the revision that is selected in the upper area.

Lower area
- "Action"
- "Path": Object path in SVN
- "Copy from path"
- "Copy from revision"

List of actions that were performed on the objects of the project in the selected revision:

"Hide unrelated changed paths" ☑: All changes of this revision are hidden that do not have any relevance to the object.

"Stop on copy/rename" ☑: If the object was copied from another location in the SVN repository, then no more log messages are retrieved. This is especially beneficial when branches or tags are monitored and only changes within the branch are relevant.

"Filter/Range" Opens the "Filter" dialog

"All" All revisions are listed.

"Next 100" The next 100 revisions are listed.

Table 815: Dialog "Filter"

- "Revision range" The displayed revisions can be filtered by "Head", "Revision", or "Date". ☑: The option fields for "Start revision" and "End revision" are editable. "Use UTC time": Date display in universal time.
  For more detailed information, refer to the description "Dialog 'Select revision'".

- "Message contains" Display of revision logs that contain a special text in the "Message"

- "Author contains" Display of revision logs of the specified author

- "Path contains" Display of revision logs of the specified path

Table 816: Context menu commands of the revisions

- "Compare with base working copy" Compares the selected revision of the object with the base working copy (without local changes).

- "Com with working copy" Compares the selected revision of the object with the working copy.

- "Compare with HEAD revision" Compares the selected revision of the object with the HEAD revision.

- "Compare with previous revision" Compares the selected revision of the object with the previous revision.

- "Update item to revision" Updates the object to the selected revision.
  Note: Changes of the project by this command cannot be committed.
  For VSS users: This is comparable to loading an older version without checkout. To revert a previous commit, the command "Revert to this revision" has to be used.
Revert to this revision

Reverts the object to the selected revision. This command does not have an effect on the SVN repository as long as the changes are not committed. Internally, SVN reverts the merges for all changes that were made after the selected revision in order to revert the changes of the preceding commits.

Edit author

Opens a dialog for changing the author of the revision.

Edit log message

Opens a dialog for changing the log message of the revision.

Revision properties

Opens the dialog “Revision properties” where the properties are displayed. In the dialog, you can activate the “Add” and “Remove” properties and the option “Show binary properties”.

Create branch/tag from this revision

Creates a branch or tag from the selected revision.

Browse SVN repository

Opens the “SVN repository browser” dialog.

Copy to clipboard

Copies log details of the selected revision to the clipboard. This is the revision number, author, date of revision, log message, and the list of changes objects for each revision.

See also

- % Chapter 1.7.6.5.2.6 “Command 'Compare'” on page 3725
- % Chapter 1.7.6.5.2.7 “Command 'Compare with HEAD revision'” on page 3725
- % Chapter 1.7.6.5.2.8 “Command 'Compare with revision'” on page 3725
- % Chapter 1.7.6.5.2.1 “Command 'SVN Repository Browser'” on page 3716
- % Chapter 1.7.6.5.3.3 “Dialog 'Select revision'” on page 3745

Command 'Revert', Command 'Revert project'

Symbol: 

Function: This command opens the “Revert” dialog. In the dialog, select the objects whose local changes should be reverted, and those that are reverted to the state of the base revision of the working copy.

Call:

- Menu bar: “Project ➔ SVN”.
- “Context menu ➔ SVN”

If you select nothing or the main node in the device tree, then all modified objects are listed in this dialog (“Revert project”). If you selected one or more objects, then only the changes to this object are listed and recursively their sub-objects (“Revert”).

Dialog ‘Revert’

- Group externals
  - The external definitions are grouped by their external storage locations.
- Keep locks
  - The lock is retained for all files that are modified by the revert command.
- Select/deselect all

When external objects are deleted, CODESYS SVN cannot restore this data in SVN offline mode. The user is prompted how to proceed:

- Switch back to SVN online mode and call the external objects.
- Connect now to the SVN server one time in order to complete the current operation, but afterwards switch back to SVN offline mode.
- Skip the retrieval of the external objects. They can be fetched later by updating the project.
See also
- Chapter 1.7.6.5.2.20 “Command ‘Revert to revision’, Command ‘Revert project to revision’” on page 3734

Command ‘Revert to revision’, Command ‘Revert project to revision’

Symbol:  ⚫

Function: This command opens the “Select revision” dialog. In this dialog, you select the revision to which the project or the selected objects revert.

Call:
- “Project ➔ SVN”
- “Context menu ➔ SVN”

If nothing or the base node is marked in the object tree, then the entire project is reverted to a specific revision (“Revert project to revision”). If one or more objects are selected, then these objects and their sub-objects are reverted (“Revert to revision”).

Dialog 'Select revision'

For a description of the dialog, refer to the section "Select revision".

See also
- Chapter 1.7.6.5.3.3 “Dialog ‘Select revision”” on page 3745
- Chapter 1.7.6.5.2.19 ‘Command ‘Revert’, Command ‘Revert project’” on page 3733
- Chapter 1.7.6.5.2.18 ‘Command ‘Show log’, Command ‘Show project log’” on page 3731

Command 'Update', Command 'Update project'

Symbol:  ⚫

Function: This command commits changes in the SVN repository to the project. The update is performed with the HEAD revision.

Call:
- Menu bar: “Project ➔ SVN”.
- “Context menu ➔ SVN”

If nothing or the main node is selected, then the entire project is updated (“Update project”). If one or more objects are selected, then these objects and their sub-objects are updated (“Update SVN”).
The following cases are possible:

- Projects are added to the project that are present in the SVN repository, but not in the project. In this case, the message "Added <object>" is issued to the message view.
- Objects that no longer exist in the SVN repository, but are present in the project locally (and not marked as "added"), are treated according to the Subversion standard procedure: If local changes are present, then the object remains in the project as unversioned. If there are no local changes, then the object is also deleted locally because the user can retrieve the object from an older version at any time. In this case, "Deleted object" is issued to the message view.
- Versioned objects that exist in both the SVN repository and the project are updated if they are different. Three cases to observe:
  - No local changes have been made since the last update: In this case, the local object is overwritten by the contents from the SVN repository. The message "Object updated" is issued to the message view.
  - Local changes have been made since the last update and the corresponding object type can be merged. When versions have been merged successfully, the message "Objects merged" is issued to the message view. If the command is not executed successfully, then the object is marked as "Conflicted object" in the object tree and the message "Conflicted object" is issued.
  - Local changes have been made since the last update and the corresponding object type cannot be merged. In this case, the object is marked as "Conflicted object" in the object tree and the message "Conflicted object" is issued.

If only some of the objects are updated, it may be that objects with the same name already exist. For example, this situation can come from moving objects to a folder.

For this conflict, you can react in the following ways:

- Do nothing and leave the conflict-causing objects as they are.
- Update (and remove) the conflicting objects in order to correct the conflict.
- Update the entire project in order to remove all conflicting objects and correct the conflict.

See also

- Chapter 1.7.6.5.2.22 “Command 'Update to revision'” on page 3735

Command 'Update to revision'

Symbol: 

SFunction: This command opens the “Update” dialog. In the dialog, the revision is defined for updating the project.

Call:

- "Project ➔ SVN"
- “Context menu ➔ SVN"

If you select nothing or the base node in the object tree, then the entire project is updated to a revision (“Update project to revision”). If you select one or more objects, then these objects are updated and their sub-objects are updated recursively (“Update to revision”). As an option, you can define that the sub-objects are not updated.

The behavior of the updating process (for example merging of conflicts) is similar to the “Update project” and “Update” commands.

Dialog 'Update'

| “HEAD” | : This command behaves the same as the “Update” and “Update project” commands. |
| “Revision” | : The revision to which was last updated is selected by the revision number. |
| | : Opens the dialog “Log” for selecting the revision. |
“Date”  • The revision to which was last updated is selected by the modification date.
  “Use UTC time”:  • The date is displayed in universal time.

“Recursive”  • Default setting. The selected part is updated recursively. This means that all elements below the selected object are also updated.

“Omit external objects”  • External objects are not updated.

See also
- † Chapter 1.7.6.5.2.21 ‘Command 'Update', Command 'Update project' ” on page 3734
- † Chapter 1.7.6.5.3.3 “Dialog 'Select revision’” on page 3745

Command 'Disconnect project from SVN'

Symbol:  

Function: This command deletes all connections of the current project to SVN by converting the project into a non-versioned project.

Call: Menu bar: “Project ➔ SVN”.

Because this operation cannot be reversed, the operation must be confirmed before the command is executed.

Use the command "Connect to existing project" to connect to the SVN repository again at a later time.

See also
- † Chapter 1.7.6.5.2.29 “Command ‘Connect to existing project’” on page 3739

Command 'Switch'

Symbol:  

Function: This command opens the “SVN switch” dialog. In this dialog, you specify a URL in the SVN repository to which the current working copy of the project is updated. The command switches a project from a branch or tag to another.

Call: Menu bar: “Project ➔ SVN”.

Requirement: The project is versioned.

Dialog 'SVN switch'

<table>
<thead>
<tr>
<th>“From”</th>
<th>Current SVN URL of the project</th>
</tr>
</thead>
<tbody>
<tr>
<td>“To”</td>
<td>Input field for the target URL in SVN</td>
</tr>
</tbody>
</table>

- ”HEAD”: The “Select revision” dialog opens.
-  The “SVN Repository Browser” dialog opens. There you select the target URL in the SVN repository.

See also
- † Chapter 1.7.6.5.2.1 “Command ‘SVN Repository Browser’” on page 3716
Command 'Un-Ignore on commit'

**Function:** This command removes an unversioned object from the ignore list so that the object is checked by default on commit.

**Call:** Context menu: “SVN”

**Requirement:** The command “Ignore on commit” was executed for the object. The object is marked with the © symbol.

See also

- © Chapter 1.7.6.5.2.11 “Command 'Ignore on commit'” on page 3729

Command 'SVN Cleanup'

**Function:** This command opens the “SVN Cleanup” dialog. In the dialog, you define actions that are performed when cleaning up the SVN working copy.

**Call:** Menu bar: “Project ➔ SVN”.

Dialog 'SVN Cleanup'

### Table 817

<table>
<thead>
<tr>
<th>“Internal SVN working copy”</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>“Update time stamps (speeds up SVN status display)”</strong></td>
</tr>
<tr>
<td><strong>“Vacuum cached pristine copies (may reduce the size of your project file)”</strong></td>
</tr>
<tr>
<td><strong>“Clear work queue and force unlock of SVN internal data structures (emergency only)!”</strong></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>“Project contents”</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>“Revert all local changes (use with care!)”</strong></td>
</tr>
<tr>
<td><strong>“Release all locks”</strong></td>
</tr>
<tr>
<td><strong>“Revalidate all locks against the repository (they could have been stolen)”</strong></td>
</tr>
<tr>
<td><strong>“Status caches”</strong></td>
</tr>
</tbody>
</table>
Command 'Clear authentication data'

**Function**: This command opens the “CODESYS” dialog. In this dialog, define the caches that will be deleted.

**Call**: Menu bar: “Project ➔ SVN”.

Dialog 'CODESYS'

The authentication memory contains the authentication data of all SVN repositories for which the user has selected for saving the authorization data. This memory is deleted completely by this command.

<table>
<thead>
<tr>
<th>“Clear the shared on-disk cache.”</th>
<th>The data saved on the computer is deleted.</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Clear the RAM cache of this instance.”</td>
<td>The data saved in the RAM is deleted.</td>
</tr>
</tbody>
</table>

The authentication data saved on the computer is stored in %APPDATA%\Subversion\auth. This memory path is also used for most other Subversion client applications (for example, TortoiseSVN and AnkhSVN). Therefore, deleting the authentication data affects these applications as well.

Command 'Merge changes'

**Symbol**: ⌘

**Function**: This command opens the “Merge” dialog. In this dialog, you determine the revisions with the changes to be merged with the working copy of the project.

**Call**: Menu bar: “Project ➔ SVN”.

**Requirement**: The project is linked to SVN.

Dialog 'Merge’“”

| “Kind of merge” | ● “Sync/Reintegrate/Symmetric merge”: Synchronizes all missing changes from trunk (or a different branch) into this branch.  
● “Cherry pick”: Integrates specifically selected revisions from one branch to another branch. This is necessary, for example, if any error trapping has to be ported back to an older version. |
|-----------------|--------------------------------------------------------------------------------------------------|
| “Merge source” | SVN URL of the SVN repository  
● Input field  
● “HEAD”: HEAD revision  
● [ ] Dialog “SVN Repository Browser” opens for selecting the SVN repository. |
| “Define start and end revision” | Select this option to merge a cohesive range of revisions with the working copy. |
| “Start revision” | Defines the range of revisions that are merged with the working copy:  
● “HEAD”: HEAD revision  
● “Revision”: Start and end revision of the range  
● “Date”: Date of the start and end revisions |

3ADR010583, 1, en_US 2020/12/10
"Define revision range" | Select this option to merge individual revisions with the working copy. You can also highlight the individual revisions in the "Log" dialog.
Note: When defining ranges, CODESYS SVN behaves like other graphical clients, such as Tortoise SVN), and not like the command-line client. Example: For a range of 4–7, revisions 4, 5, 6, and 7 are merged.
See also: Merging a Range of Revisions

"Dry run (simulation)" | ☑: This command is executed without changing the working copy. Files that are changed during an actual merge are displayed, as well as ranges where conflicts occur.

"Record only" | ☑: The revision is marked as "merged" without actually performing the merge.

"Ignore ancestry " | ☑: SVN uses path-based differences only, not history-based differences.

See also
- % Chapter 1.7.6.5.3.3 “Dialog ‘Select revision’” on page 3745
- % Chapter 1.7.6.5.2.1 “Command ‘SVN Repository Browser’” on page 3716

Command ‘Connect to existing project’

Symbol: ➔

Function: This command opens the “Connect to SVN repository” dialog. In the dialog, you define the URL and the revision of the SVN repository with which the unversioned project is connected.

Call: Menu bar: “Project ➔ SVN”.

Requirement: The project is disconnected from SVN.

NOTICE!
Only users who have read access to the entire project (see the CODESYS user and access management) can import the project into the SVN repository or can link to an existing database project.

NOTICE!
This command functions reliably only when the project has already been imported into SVN and then disconnected with the command “Disconnect project from SVN”.

Dialog ‘Connect to SVN repository’

"URL of existing project" | URL of the SVN repository
  "HEAD": Selection of the revision in the “Select revision” dialog
  ☑: Selection of the SVN repository in the “SVN Repository Browser”

"Checkout options" | “Omit externals”: External objects are not checked out.

"Revision" | ● “HEAD”: HEAD revision
    ● “Revision”: Number of the revision
    ● “Date”: Date of the revision
    “Use UTC time”: ☑: Date display in universal time.
Command 'Resolve conflict'

Symbol:  

**Function:** This command opens the "<object>" dialog. In the dialog, the conflicts are displayed and functions for resolving conflicts are prepared in order to merge changes.

**Call:** Context menu of the object.

**Requirement:** The object has a conflict that has occurred by updating the object with local changes.

**Dialog '<object>**'

<table>
<thead>
<tr>
<th>Action</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Compare&quot;</td>
<td>The local objects are displayed on the left side, and the version from the SVN repository is displayed on the right side.</td>
</tr>
<tr>
<td>&quot;Use mine&quot;</td>
<td>A local change is used.</td>
</tr>
<tr>
<td>&quot;Use yours&quot;</td>
<td>A change of the version from the SVN repository is changed.</td>
</tr>
<tr>
<td>&quot;Apply&quot;</td>
<td>All changes are accepted that you made in this dialog. The status of the object is changed.</td>
</tr>
<tr>
<td>&quot;Cancel&quot;</td>
<td>Cancels all changes that you made in this dialog. But the object keeps the conflicted status.</td>
</tr>
</tbody>
</table>

Command 'Work in offline mode'

**Function:** This command switches to SVN offline mode. In SVN offline mode, the implicit locking and all commands that access the SVN repository are not possible.

**Call:**
- Menu bar: "Project ➔ SVN".
- Context menu: "SVN"

**Requirement:** The project is linked to SVN.

When switching back to SVN online mode, all present locks on the working copy are checked against the server. If this locking is invalid, then it is released.

**Uses case**

The user on a machine wants to make changes to the project without disconnecting the connection. At the moment, there is no connection to the server. Despite this, when automatic locking is activated, work is possible because the SVN offline mode deactivates the automatic lock temporarily.

Command 'Copy (Branch/Tag)'

Symbol:  

**Function:** This command opens the "SVN Copy Branch/Tag" dialog. There you can "Branch" or "Tag" a revision of your project. A specific revision of your project is saved there at this position. A branch is normally used in order to save changes isolated in one version. A tag is used for marking a specific state, for example a shipping version. Internally, it is copied not in the actual sense, but more refers to the revision.

**Call:** Menu bar: "Project ➔ SVN".
**Requirement:** The project is versioned.

**Dialog 'SVN Copy (Branch/Tag)'**

Table 818: “SVN repository”

<table>
<thead>
<tr>
<th>“From”</th>
<th>SVN path of the current project</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example:</td>
<td><a href="https://svnserver/repository/trunk/ControlABC.project">https://svnserver/repository/trunk/ControlABC.project</a></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>“To”</th>
<th>Target path in the SVN repository for the copy operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example of tag:</td>
<td><a href="https://svnserver/repository/tags/V4.4.4.4/ControlABC.project">https://svnserver/repository/tags/V4.4.4.4/ControlABC.project</a></td>
</tr>
</tbody>
</table>

: Dialog “SVN Repository Browser” opens for selecting the target path.

Table 819: “Log message”

<table>
<thead>
<tr>
<th>Input field</th>
<th>Comment the change in a log message.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example:</td>
<td>Tag for version 4.4.4.4 created.</td>
</tr>
</tbody>
</table>

| “Recent Messages”                 | Opens the dialog “Recent Messages” to display the last log messages. You can click a log message to accept it. |

Table 820: “Create copy from”

<table>
<thead>
<tr>
<th>“Working copy (including local changes)”</th>
<th>The new branch/tag refers to the working copy including all local changes. The local changes are committed to the SVN repository for this purpose.</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Base revision of working copy (&lt;revision number&gt;)”</td>
<td>The new branch/tag refers to the base revision of your working copy whose revision number is displayed in the parentheses. If the working copy already contains local changes, then these are not committed to the SVN repository.</td>
</tr>
<tr>
<td>“HEAD revision of the repository”</td>
<td>The new branch/tag refers to the HEAD revision of your project.</td>
</tr>
<tr>
<td>“Specific revision in SVN repository”</td>
<td>The new branch/tag refers to a revision that is displayed on the adjacent button. Click the button to change the revision. The “dialog opens.”.</td>
</tr>
</tbody>
</table>

| “Switch to new location ” | [ ] After the dialog is confirmed, the working copy switches to the new branch/tag. |

| “OK”                         | The target path is created (as a new tag ../repository/tags/V4.4.4.4 or as a new branch ../repository/branches/new_feature). Then the revision specified in “Create copy from” is copied there. |

See also

- [Chapter 1.7.6.5.2.1 “Command 'SVN Repository Browser'” on page 3716](#)

**Command 'Pending changes’**

Symbol: ☑
**Function:** This command opens the “Pending changes” view. All objects are listed there that have changed from the base revision.

**Call:** “View ➔ Pending changes”

**View ‘Pending changes’**

The modified objects are shown in the lower half of the view. You can use the commands “Commit”, “Revert”, and “Update” on these objects. When you do this, you can select individual objects or all objects. Commands for the comparison and display of the version history are available in the context menu of a selected object.

Double-clicking the object opens the project comparison.

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Select”</td>
<td>Selection or clearing of all objects</td>
</tr>
<tr>
<td>“Commit”</td>
<td>Commits local changes to the SVN repository.</td>
</tr>
<tr>
<td>“Revert”</td>
<td>Resets the local changes to the state of the base revision of the working copy.</td>
</tr>
<tr>
<td>“Update”</td>
<td>This command commits changes in the SVN repository to the project. The update is performed with the HEAD revision.</td>
</tr>
<tr>
<td>“Keep locks”</td>
<td>Lock is not released automatically after commit.</td>
</tr>
<tr>
<td>“Recent Messages”</td>
<td>Shows the last used log messages. You can click a log message to accept it.</td>
</tr>
<tr>
<td>“Messages”</td>
<td>Comment the change in a log message. Example: Bug fix error 123</td>
</tr>
</tbody>
</table>

### 1.7.6.5.3 Dialogs

1.7.6.5.3.1 Dialog 'SVN Settings'........................................................................................................ 3742
1.7.6.5.3.2 Dialog 'Project Settings' - 'SVN Settings'........................................................................ 3744
1.7.6.5.3.3 Dialog 'Select revision'..................................................................................................... 3745
1.7.6.5.3.4 Dialog 'Subversion Authentication'....................................................................................... 3745
1.7.6.5.3.5 Dialog 'Automatic locking failed'......................................................................................... 3748

**Dialog 'SVN Settings'**

**Tab ’General’**

Symbol:  

**Function:** This tab includes the basic settings for CODESYS SVN.

**Call:** Menu bar: “Tools ➔ Options”.

---

**Double-clicking the object opens the project comparison.**

---
Table 821: “Automatic locking and merging”

<table>
<thead>
<tr>
<th>“Merge”</th>
<th>Behavior for the commands “Update”, “Merge”, or “Switch”, when both sides (working copy and SVN repository) have changed from the base version.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>● “Mark all colliding changes as conflicts”: The objects are not merged automatically. All changes with a conflict are marked as &quot;With conflict&quot;, even if some of them can be merged automatically.</td>
</tr>
<tr>
<td></td>
<td>● “Merge mergeable changes, mark the others as conflicts”: Changes that can be merged are merged automatically. All others are marked as &quot;With conflict&quot;.</td>
</tr>
<tr>
<td></td>
<td>● “Merge mergeable changes, ask the user for the others”: Changes that can be merged are merged automatically. The user is prompted for all others.</td>
</tr>
<tr>
<td></td>
<td>● “Always ask the user, even for mergeable changes”: For all changed objects, the user is prompted, even if some of them can be merged automatically.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>“Locks”</th>
<th>Behavior such as CODESYS SVN objects when they are changed locally.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>● “Always try to lock before modification”: All objects are locked before they are changed, even if they can be merged.</td>
</tr>
<tr>
<td></td>
<td>● “Only lock the objects which don’t support merging”: Only those objects are locked that cannot be merged automatically.</td>
</tr>
<tr>
<td></td>
<td>● “Never acquire a lock automatically”: No objects are locked, not even if they can be merged automatically.</td>
</tr>
</tbody>
</table>

| “Marker” | ● “Use conflict markers when merging objects”: If objects with conflicts exist that cannot be merged, then these conflicts are marked in the source code with conflict markers. In addition, the object itself is marked as being merged successfully (no conflict). |
|          | ● “Leave non-mergeable objects as conflicted”: No conflict marker is set. Objects that cannot be merged remain in the status "With conflict". |

| “Prompt the user when automatic locking fails.” | ☑: If it is not possible, to lock the object, then the dialog “Automatic locking failed” opens (see dialog description). |

**Example of conflict markers**

```plaintext
Foo();

>>>>>>>>>>>>>>>>>
I := I + 1;

=============
I := I + 2;

<<<<<<<<<<<
```

Table 822: “Server check”

<table>
<thead>
<tr>
<th>“Check server for updates and locks”</th>
<th>☑: CODESYS SVN checks in the specified time interval that objects have been updated on the server. In addition, it checks whether objects are locked or locks have been stolen.</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Check interval (minutes)”</td>
<td>Example: 10</td>
</tr>
</tbody>
</table>

Table 823: “Ignore for comparison”

<table>
<thead>
<tr>
<th>Ignore whitespace</th>
<th>☑: Whitespace differences between the current project and the reference project are ignored.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ignore comments</td>
<td>☑: Comments in the programming code are excluded from the comparison.</td>
</tr>
<tr>
<td>Ignore Properties</td>
<td>☑: Object properties are excluded from the comparison.</td>
</tr>
</tbody>
</table>
Some of the SVN options can be overwritten by the project-specific settings.

Project-specific settings are defined in the menu “Project ➔ Project settings”, category “SVN Settings”.

See also

● Chapter 1.7.6.5.3.5 “Dialog 'Automatic locking failed’” on page 3748

Tab 'SSH'

Symbol: 🌐

**Function:** This tab contains the settings for the SSH protocol.

**Call:** Menu bar: “Tools ➔ Options”.

Table 824: “SSH client implementation”

<table>
<thead>
<tr>
<th>Setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;libssh2 (recommended)&quot;</td>
<td>CODESYS SVN uses Libssh2 for establishing a connection via SSH protocol. This is the recommended setting.</td>
</tr>
<tr>
<td>&quot;SharpPlink (backwards compatibility)&quot;</td>
<td>CODESYS SVN uses plink.exe for establishing a connection with SSH servers. This option is required only for communication with outdated servers that support the deprecated SSH-1 protocol.</td>
</tr>
</tbody>
</table>

The SSH configuration can be overwritten by means of the environment variable SVN_SSH or server-specific by means of the SVN configuration file.

See also

● Tunneling via SSH

Dialog 'Project Settings' - 'SVN Settings'

Symbol: 🌐

**Function:** The behavior of the integrated SVN version control system is configured in this dialog.

**Call:** Menu bar: “Project ➔ Project Settings” (“SVN Settings”).

**Requirement:** A project is open.

Table 825: “Automatic locking and merging”

<table>
<thead>
<tr>
<th>Setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Merge&quot;</td>
<td>Behavior for the commands “Update”, “Merge”, or “Switch”, when both sides (working copy and SVN repository) have changed from the base version.</td>
</tr>
<tr>
<td>&quot;Locks&quot;</td>
<td>Behavior such as CODESYS SVN objects when they are changed locally.</td>
</tr>
<tr>
<td>&quot;Marker&quot;</td>
<td>Behavior for conflicts</td>
</tr>
</tbody>
</table>
Table 826: “Settings SVN version info”

| “Create SVN_VERSION_INFO constants for IEC access” | ☑: The object SVN_VERSION_INFO is created and includes global constants or variables for the project metadata. |
|                                                  | ☐: The object SVN_VERSION_INFO is not available. |

When you activate the option, the object is created automatically. When you deactivate the option, the object is removed from the project automatically.

See also
- ☛ Chapter 1.7.6.5.3.1 “Dialog 'SVN Settings'” on page 3742

**Dialog 'Select revision'**

**Function:** This dialog shows the currently selected revision. You can edit the selection there.

<table>
<thead>
<tr>
<th>“Revision”</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>“HEAD”</td>
<td>☑: The latest revision (top revision) within a branch is displayed.</td>
</tr>
<tr>
<td>“Revision”</td>
<td>☑: A specific revision is displayed by the revision number.</td>
</tr>
</tbody>
</table>

Example: 3

Tip: Click to show the revisions. Then the “Log” dialog opens to display the revisions and the associated actions. The revision that you select there is applied.

<table>
<thead>
<tr>
<th>“Date”</th>
<th>☑: A specific revision is checked out by the modification date. This is the highest revision at the given time (the last revision before that time).</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Example: 12/23/2016 11:59:59</td>
</tr>
</tbody>
</table>

Tip: See section "Revision identifiers" in "Version control with Subversion"

<table>
<thead>
<tr>
<th>“Use UTC Time”</th>
<th>☑: Modification date in universal time is used.</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Reset recursively”</td>
<td>☑: All objects below the selected object are also reset.</td>
</tr>
</tbody>
</table>

The action fails if
- Objects have been moved in or out of the hierarchy below
- Objects outside of the hierarchy would be changed by implicit dependencies

See also
- ☛ Chapter 1.7.6.5.2.18 “Command ‘Show log’, Command ‘Show project log’” on page 3731
- "Version control with Subversion", Section "Revision identifier"

**Dialog 'Subversion Authentication'**

The dialogs are used for authenticating the server/client connection. A server or client authentication is performed depending on the initial situation and protocol.
Overview of possible protocols and dialogs

- **svn://**: The SVN protocol; either unencrypted or SSL/TLS encrypted
  - Can prompt for user name and password (even for an unencrypted connection)
  - Can prompt for a server certificate from the dialog for authentication in order to confirm the server if a certificate is unknown, defective, or invalid (for TLS/SSL encryption)
  - As an alternative or in addition to the user name and password prompt, the client can also be authenticated with client certificates (for TLS/SSL encryption). The dialogs for authentication open with the client certificate.

- **http://**: SVN via http, unencrypted
  - Can prompt for user name and password

- **https://**: SVN via http, SSL/TLS encrypted.
  - Can prompt for user name and password
  - Can prompt for a server certificate from the dialog for authentication in order to confirm the server if a certificate is unknown, defective, or invalid.
  - As an alternative or in addition to the user name and password, the client can also be authenticated with client certificates. The dialogs for authentication open with the client certificate.

- **svn+ssh://**: The SVN protocol, encrypted through an SSH tunnel. SSH (Secure Shell) is the usual networking tool in Linux/Unix for accessing other computers.
  - Can prompt for user name and password
  - Prompts for server certificate in the dialog for authentication if the server is still unknown in order to be sure that it is the correct server.

### Initial situation
CODESYS (as a client) receives an unknown or defective server certificate.

This dialog shows information about the certificate. There you can confirm the identity of the server.

<table>
<thead>
<tr>
<th>&quot;Authentication area&quot;</th>
<th>Connection that is secured</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Example: <a href="https://svn">https://svn</a> repository:443</td>
</tr>
</tbody>
</table>

#### Table 827: “Certificate information” (for SSL/TLS connections)

<table>
<thead>
<tr>
<th>&quot;Host name&quot;</th>
<th>Example: svn repository</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Thumbprint&quot;</td>
<td></td>
</tr>
<tr>
<td>&quot;Valid from&quot;</td>
<td></td>
</tr>
<tr>
<td>&quot;Valid to&quot;</td>
<td></td>
</tr>
<tr>
<td>&quot;Issuer&quot;</td>
<td>Example: 3S-Smart Software Solutions GmbH</td>
</tr>
<tr>
<td>&quot;Certificate&quot;</td>
<td></td>
</tr>
</tbody>
</table>

#### Table 828: “SSH server key information” (for SSH connections)

| "Key type" |
| "Key size (bits)" |
| "Key thumbprint" |
### "Save information to RAM"
- The certificate is saved to the working memory. Then the client recognizes in the current CODESYS session for future connections.
- If you restart CODESYS, then you have to accept the certificate again.

### "Save to disk"
- The certificate is saved on the computer and it is available for future connections.
- If you restart CODESYS, then the saved certificate is used.

### "OK"
- Authenticates and established the connection.

---

The certificate memory is secured cryptographically and distributed with other SVN clients.

See also
- **Version Control with Subversion**

---

**Dialog for authentication with a client certificate**

Initial situation: The SVN server requires a client certificate for authentication.

In this dialog, you select the client certificate in order to confirm the identity.

<table>
<thead>
<tr>
<th>&quot;Authentication area&quot;</th>
<th>Connection that is secured</th>
<th>Example: <a href="https://svn">https://svn</a> repository:443</th>
</tr>
</thead>
</table>

**Table 829: "The SSL server requires a client certificate file."**

<table>
<thead>
<tr>
<th>&quot;File&quot;</th>
<th>Client certificate file</th>
</tr>
</thead>
</table>

---

**Dialog for authentication with a pass phrase**

Initial situation: The SVN server is configured so that it demands a client certificate for authentication. The applied certificate is protected by a pass phrase.

<table>
<thead>
<tr>
<th>&quot;Authentication area&quot;</th>
<th>Connection that is secured</th>
<th>Example: <a href="https://svn">https://svn</a> repository:443</th>
</tr>
</thead>
</table>

---
**Table 830: “A pass phrase is needed to unlock the certificate.”**

<table>
<thead>
<tr>
<th>“Pass phrase”</th>
<th>Example: ***</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Save information to RAM”</td>
<td>✓: The pass phrase is saved to the working memory. Then the client recognizes in the current CODESYS session for future connections. If you restart CODESYS, then you have to accept the certificate again.</td>
</tr>
<tr>
<td>“Save to disk”</td>
<td>✓: The pass phrase is saved on the computer and it is available for future connections. If you restart CODESYS, then the saved certificate is used.</td>
</tr>
<tr>
<td>“OK”</td>
<td>Authenticates with client certificates by means of a pass phrase and establishes the connection.</td>
</tr>
</tbody>
</table>

**Dialog for authentication with a user name and password**

Initial situation: The SVN server is configured so that it demands a user name and password for authentication.

<table>
<thead>
<tr>
<th>“Authentication area”</th>
<th>Connection that is secured</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example: <a href="https://svn">https://svn</a> repository:443</td>
<td></td>
</tr>
<tr>
<td>“User name”</td>
<td>Example: a.mayr</td>
</tr>
<tr>
<td>“Password”</td>
<td>Example: ***</td>
</tr>
<tr>
<td>“Save information to RAM”</td>
<td>✓: Saved to the working memory. Then the client recognizes in the current CODESYS session for future connections. If you restart CODESYS, then you have to accept the certificate again.</td>
</tr>
<tr>
<td>“Save to disk”</td>
<td>✓: Saved on the computer and it is available for future connections. If you restart CODESYS, then the saved certificate is used.</td>
</tr>
<tr>
<td>“OK”</td>
<td>Establishes the connection and authenticates it.</td>
</tr>
</tbody>
</table>

**Dialog 'Automatic locking failed'**
The dialog shows a list of all objects for which an automatic locking was not possible. In the options you define how CODESYS SVN will resolve the conflict.
Table 831: "Automatic Locking and Merging"

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Try to steal the lock for the affected objects”</td>
<td>These options are displayed if another user has locked the object.</td>
</tr>
<tr>
<td>“Activate the &quot;Offline Mode&quot; to temporarily suppress locking”</td>
<td>These options are displayed if another user has locked the object.</td>
</tr>
<tr>
<td>“Update the affected objects to the newest revision”</td>
<td>These options are displayed if there exists a more current version of the object on the server.</td>
</tr>
<tr>
<td>“Update the whole project to the newest revision”</td>
<td>These options are displayed if there exists a more current version of the object on the server.</td>
</tr>
<tr>
<td>“Activate the &quot;Offline Mode&quot; to temporarily suppress locking”</td>
<td>These options are displayed if no connection can be established to the server.</td>
</tr>
<tr>
<td>“SVN Project Settings”</td>
<td>Opens the SVN project settings dialog (menu “Project ➔ Project Settings”). There you can change the settings for the automatic locking.</td>
</tr>
<tr>
<td>“SVN Settings”</td>
<td>Opens the general SVN project settings dialog (menu “Tools ➔ Options”).</td>
</tr>
</tbody>
</table>

See also

- Chapter 1.7.6.5.3.1 “Dialog 'SVN Settings’” on page 3742
- Chapter 1.7.6.5.3.2 “Dialog 'Project Settings' - 'SVN Settings’” on page 3744

1.7.6.4 Objects

1.7.6.4.1 Object ‘SVN_VERSION_INFO’ ................................................................. 3749

Object ‘SVN_VERSION_INFO’

Symbol: 

The object contains the SVN metadata of the project as global constants or variables in a variable list. It is located in the “POUs” view. You can specifically retrieve the data of the global constants or variables by the application. By calling specific data, you can also reduce the memory usage on the controller.

The SVN metadata is provided for this purpose, subdivided over multiple global variable lists (GVLs):

- “SVN_VERSION_INFO”
- “SVN_Info_Summary”
- “SVN_Info_SummaryW”
- “SVN_Info_URI”
- “SVN_Info_Revisions”
- “SVN_Info_Flags”
- “SVN_info_LastChange”

The SVN_VERSION_INFO object is created automatically when a project is imported to a SVN repository. To do so the option “Create SVN_VERSION_INFO” in the dialog “Import project to SVN” must be activated.

Furthermore you can create the object or remove it from the project with the option “Generate SVN_VERSION_INFO constants for IEC Access” (Dialog “Project ➔ Project Settings”, category “SVN Settings”).
Table 832: Global Constants

<table>
<thead>
<tr>
<th>Name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MINREVISION</td>
<td>LINT</td>
<td>Lowest revision number of the working copy</td>
</tr>
<tr>
<td>MAXREVISION</td>
<td>LINT</td>
<td>Highest revision number of the working copy</td>
</tr>
<tr>
<td>PARTIAL</td>
<td>BOOL</td>
<td>TRUE: The working copy is incomplete. Example: Cancellation during the last update due to a network error or a checkout.</td>
</tr>
<tr>
<td>MODIFIED</td>
<td>BOOL</td>
<td>TRUE: Local changes were made.</td>
</tr>
<tr>
<td>SWITCHED</td>
<td>BOOL</td>
<td>TRUE: Parts of the project were branched (with the “Switch” command).</td>
</tr>
<tr>
<td>VERSION</td>
<td>STRING</td>
<td>Version identification, similar to Apache™ Subversion® (subversion.exe)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Example: 12:34M, means MINREVISION = 12, MAXREVISION = 34, MODIFIED = TRUE</td>
</tr>
<tr>
<td></td>
<td></td>
<td>For more information, refer to the documentation for Apache™ Subversion®.</td>
</tr>
<tr>
<td>CLEAN</td>
<td>BOOL</td>
<td>TRUE: The version is clean. This is the case when MINREVISION is equal to MAXREVISION, the working copy is complete, and non-versioned, and is was not switched.</td>
</tr>
<tr>
<td>URL</td>
<td>WSTRING</td>
<td>SVN-URL of the project</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Example: <a href="https://svnserver/repository/trunk/ControlABC.project">https://svnserver/repository/trunk/ControlABC.project</a></td>
</tr>
</tbody>
</table>

If a controller does not support the data type WSTRING, then a compiler error is issued when accessing the object SVN_VERSION_INFO.

See also
- § Chapter 1.7.6.5.3.2 “Dialog ‘Project Settings’ - ‘SVN Settings’” on page 3744
- § Chapter 1.7.6.5.2.3 “Command ‘Import project to SVN’” on page 3720

1.7.7 Subversion

1.7.7.1 Project Version Control with Subversion

Introduction
Automation Builder projects can be stored in Subversion (SVN) repositories by using the Project Version Control. The Project Version Control can be used to track changes on a project and to have access to historic versions of the whole project or objects in the project. It is possible to hold different versions of a project in branches and to compare these versions. The Project Version Control enables multiple engineers to work collaboratively on the same project.

Basic knowledge
Make yourself familiar with the concepts of SVN.
This manual about Project Version Control is additionally to the following information and describes mainly the specific behavior of Subversion in Automation Builder.
● Homepage of Subversion: http://subversion.apache.org/
● Online user manual for Subversion: http://svnbook.red-bean.com/
● Documentation on SVN integration in Automation Builder: Refer to subfolder CODESYS SVN.

1.7.7.1 Preconditions

**Automation Builder**

- In Automation Builder, the Project Version Control must be installed.
- A valid license for the Project Version Control must be activated.
- All collaborating users working on the same project need:
  - Automation Builder installed in the same version with the same features.
  - License for same edition.
  - Same set of optional third party device descriptions.
  - Same set of optional customer specific packages.

**SVN server**

The Project Version Control can be used in combination with an SVN server in version 1.6 or newer, the repository format should be 1.5, 1.6 or 1.7. Newer repository formats are not yet supported.

The usage of local repositories in the local file system or even on a network share is strongly discouraged.

1.7.7.2 Working with Project Version Control

- All objects in the device tree or POU tree are represented by an object in the SVN repository, there might be hidden objects that are not visible in the tree but that exist in SVN.
- The smallest unit in the SVN repository is one object including all its data like name, parameters, device identification.
- Objects are identified in the SVN repository by their name. Renaming one object in Automation Builder means to delete it from the SVN repository and add a new one to the SVN repository. Renaming an object causes a break in the history of that object.
- By default objects are locked before they are changed to prevent other users from changing the object. The locking strategy can be changed in the user options.
- Objects can be compared to other versions of the same object, many differences/changes between the current object in the Automation Builder project and the compared object can be merged into the object in the Automation Builder project. Merging changes could be used to resolve conflicts in case concurrent changed can not be avoided.

- To ensure consistency it is required and also enforced that some changes can be committed or reverted only together.
  - All changes to device objects in the hardware tree that are sub-nodes to the same top level device. Note: Objects that are not devices are excluded, e.g. the application node.
  - All changes below the AC500 PLC application node.
- Most SVN operations can not be performed while other external applications like CODESYS or Panel Builder work on files that are embedded in Automation Builder project.
- Some operations like changing the target or updating the project to the latest device (description) versions do a recursive lock of the whole AC500 PLC. If the lock can’t be obtained the operation is aborted.
- Some objects contain internal data that has no meaning to the end user but is also important. Changes on such data are not shown in the compare dialog or are summarized by a placeholder like "There are hidden changes".
- Including externals is not supported.
1.7.7.1.3 Recommendations on Working with Project Version Control

Be collaborative
- Multiple users that work collaboratively on the same project should agree on their responsibility for certain parts of the project where they do changes to avoid conflicts and tree conflicts.
- Agree on locking strategy used by all users working on the same project.
- Distribute the work between multiple users meaningfully.
  - It is suggested to setup the hardware structure at first before other users checkout a project to work on it and limit structure changes in the hardware tree to the minimum.
  - Before adding objects, especially top level objects, users should agree that only one user adds objects at top level or below the same parent, or agree on unique names for the objects to add. The default naming scheme for new objects bears the risk of name conflicts. These conflicts could be resolved only by reverting the changes of the user who later tried to commit the changes.

Be careful
- The SVN integration (and also project compare) gives lot of power to the user, users should be sure to do only things they fully understand. Especially by merging changes incomplete it is possible to create inconsistent data.
- Adding devices, removing devices or even changing parameters can have side effects to other devices, do not change objects/parameters to their original state by merging that were not done explicitly.
- Commit changes frequently to SVN.
  - To release locks that you don’t longer need.
  - To reduce the risk of conflict with co-workers.
  - To keep the sets of changes to commit small.
- Do frequent updates when collaborating in a team.
  - To be up-to date.
  - To keep the sets of changes to get from SVN small.
  - To reduce the risk of losing work results in case of conflicts.
- To avoid conflicts, it is suggested to stay with the default setting to automatically lock objects before doing changes. Consider explicit recursive locks of sub-trees where you plan bigger changes.
- Prefer a clean checkout over using the switch command to change between different branches.
- Do not use the switch command to change between unrelated projects, this could corrupt the Automation Builder project (local copy, not in SVN) easily.
- Commit local changes to the SVN repository before creating a branch.

Be effective
- Give objects good/correct names after adding them and use renaming of objects already committed to SVN sparely to maintain a continuous history in the SVN repository.
- The goal to revert only single changes of all changes done that must be committed/reverted together, could be achieved by using project compare or the object compare dialog.
- If changes can’t be committed to the SVN repository because of locks hold by other users, it is possible to create a branch, use the switch command to change to this branch and commit the changes there. The branch and base line could be merged together later.

1.7.7.1.4 Known Issues and Troubleshooting

Not all changes are shown for all objects, but hidden changes are also important. The device pool may be changed as side effect of several operations, including opening the project.

When a project was corrupted (by performing an update that tried to add an AC500 Communication Module) it is possible to save this project and merge changes to a project that has been cleanly checked out by project compare.
1.7.7.2 SVN Support Examples

1.7.7.2.1 Importing Automation Builder Project to SVN Repository

1. In the Automation Builder main menu, go to “Project ➔ SVN ➔ Import Project to Subversion”.
2. Enter user credentials and click “OK”.
3. Select SVN server repository to import Automation Builder project and click “OK”.

⇒ The Automation Builder project is imported into the selected repository and connected automatically to the repository. The imported project nodes are identified with green indicators.

1.7.7.2.2 Logging in User2

1. In the Automation Builder main menu, go to “Project ➔ SVN ➔ Checkout”.
2. Enter user credentials and click OK.
3. Select the repository location, project folder and revision if any and click “OK”.

⇒ The project will checked out of the repository, saved in the selected location and opened as a primary project.

The tables below provide the descriptions of the options available in the check-out dialog.

<table>
<thead>
<tr>
<th>Checkout options</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Omit externals</td>
<td>Do not checkout external objects.</td>
</tr>
<tr>
<td>As library project</td>
<td>Saves the project as a CODESYS library file.</td>
</tr>
</tbody>
</table>
## 1.7.7.2.3 Examples

**Example 1**

If User1 modifies Panel_CP600 project, then the node indicator turns to orange with lock symbols. If User2 need to modify the same Panel_CP600 project, the Panel_CP600 project appears with a lock symbol.

To steal the lock of an affected object, proceed as follows:

1. Double-click “Panel_CP600” project.
   - Automatic lock failed dialog is displayed.
2. Enable “Try to steal the lock for the affected objects” and click “OK” to steal the lock.
3. In the Lock Message window, enter the reason to steal the lock and click “OK”.
4. User2 can modify and commit the project.

**Example 2**

If User1 adds a new object to the project and commit the changes, then User2 can update the project to see the latest modifications.
The user can revert to any of the available project revisions.
1. Right-click on object node and select “SVN ➔ Revert to Revision”.
2. Select or enter the revision number and click “OK”.
   ⇒ The revision command reverts local changes of this object back to the specific revision of the working copy.
3. Right-click on the object node and select “SVN ➔ Commit”.
4. In the commit window, enter the reason to change the project and click “OK” to make the changes.
   ⇒ The project node is updated with the latest changes.

SVN server allows to select the required revisions of Automation Builder project. You can checkout the project using “Project ➔ SVN ➔ Checkout” and then enter the credentials and click “OK”.

In the check-out dialog, do the following:
1. Select the project repository.
2. Activate “Revision” and select or enter the revision number and click “OK”.

The user can work on the selected revision. To commit the changes to the project, right-click on project and select “SVN ➔ Commit Project”.

1.7.8 Python

Further information on how to use Python can be found in the Python script examples: %Public %\Documents\AutomationBuilder\Examples\Python scripts.
1.8 Human machine interface
1.8.1 Panel Builder interface

This document describes HMI CP600 Control Panel configuration in Automation Builder and starting HMI configuration and programming software Panel Builder 600 from Automation Builder. The Panel Builder project created for the HMI CP600 is stored within the Automation Builder project.

1.8.1.1 Adding desired AC500 PLC to the project

Configuring the Symbol File

1. In the Automation Builder device tree, right-click the “Application” node and click “Add object”

2. Click on “Symbol Configuration” and click “Add object”
   ⇒ A “Symbol Configuration” object is added to the “Application” node.
3. Double-click on the “Symbol Configuration” object, then click on “Build”

![Symbol Configuration](image)

⇒ A list of all variables in the project is generated. Single variables or groups of variables can be selected by checking the corresponding item in the list.

4. After the symbols have been configured, download the project or click “Build Generate code” in the Automation Builder to create an .xml file containing all the variables read to be imported in the Tag Editor.

1.8.1.2 Creating a Panel Builder project

Adding a panel object

1. Right-click in the Automation Builder device tree and click “Add object → Panel-CP600”
2. Click on “CP600 Control Panel” and click “Add object”

⇒ A Control Panel object is added to the Automation Builder device tree.

Starting a Panel Builder project

1. In the device tree, double-click “Panel CP600” object to start Panel CP600 screen.

![Panel Builder](image)

2. Select the required PLC and enable the checkbox in the ‘Use Standard Connection Settings’ column to use it as a standard gateway connection.

You can set communication settings using the application program or by creating custom communication settings. Custom communication settings can be configured by clicking the button in the ‘Details’ column.
3. Enable the “Update Panel Builder project on launch” checkbox and click [Launch Panel Builder Editor].

If you update Automation Builder project with new variables and data types or if there are changes in existing Automation Builder project variables and data types (new, modified, deleted), recompile CODESYS application to refresh the symbol file, then launch Panel Builder editor.

4. Select “New” and click “Open” to create a new HMI project.
   ⇒ A project wizard is displayed.

If you want to import an already existing Panel Builder project file from the file system, select “Import existing project file” and proceed.

5. Select the required panel type and orientation and click “Finish”.
   ⇒ A new project wizard starts only if the Panel project is empty.

The panel projects can be compared in Automation Builder using the “Compare Objects” option.
Changing panel type

1. In the Panel project, double-click “Project properties” to change the panel type to the panel which is used.
   ⇒ The Properties dialog is displayed.

   ![Properties dialog]

2. In the Properties dialog, expand “Project” and click “Project Type”.
   ⇒ A project wizard dialog is displayed.

3. Select the desired panel type and click “Finish”.

Project information

The project information view provides an overview of the Panel Builder project without opening the project. To open the project information, double-click the “Panel_CP600” object.

The project information is updated every time the Panel Builder project is edited. You can rename the Panel Builder project via context menu.

The project name is internally used as a base for the Panel Builder project file name. Therefore, the project name has to comply with general file name restrictions.

---

Changing panel type

1. In the Panel project, double-click “Project properties” to change the panel type to the panel which is used.
   ⇒ The Properties dialog is displayed.

   ![Properties dialog]

2. In the Properties dialog, expand “Project” and click “Project Type”.
   ⇒ A project wizard dialog is displayed.

3. Select the desired panel type and click “Finish”.

Project information

The project information view provides an overview of the Panel Builder project without opening the project. To open the project information, double-click the “Panel_CP600” object.

The project information is updated every time the Panel Builder project is edited. You can rename the Panel Builder project via context menu.

The project name is internally used as a base for the Panel Builder project file name. Therefore, the project name has to comply with general file name restrictions.
The Panel Builder project information shows the list of PLCs added to the project.

1.8.1.3 Configuring Panel Builder

The user can configure a panel project manually in Panel Builder editor when there is a need to create individual panel projects. Otherwise, the configuration is updated in the panel project while launching Panel Builder editor in Automation Builder.

1. In the Panel Builder project structure, click “Config ➔ Protocols”.

2. Click to add the desired protocol.

3. Select the desired protocol and set the IP address, port, protocol type and PLC models. Click [OK].
**Importing tags**

1. In the Panel project view, click “Config ➔ Tags”.
2. Select the protocol from the drop-down list and click to import tags.
   
   *If the Panel Builder contains multiple tag importers, a dialog is displayed to select the required importer type.*

3. Select the symbol file which was exported to the file system.
4. In the lower part of the tag editor, mark the desired tags and click “Import Tag(s)” to import the tags to the Panel Builder project.

**Attaching tags to widgets**

1. In the project view, expand “Pages” and double-click Page1.
2. In the Panel Builder 600 main menu, select “View ➔ Toolbars and Docking Windows ➔ Widget Gallery”.
3. Drag-and-drop the desired widget to the page editor.
4. Right-click on the widget value and select “Attach To” to attach a tag to the widget.

5. Select the desired tag and select the desired option for the authorization “Read Only” or “Read/Write” or “Write Only”. Then, click [OK].

Downloading a project to panel

1. In the Panel Builder main menu, click “Run ➔ Download To Target”.
2. Select the CP600 project from the drop-down list and click “Download”.

Importing an existing Panel Builder project

1. In the Automation Builder device tree, right-click the Panel project and click “Import ➔ Panel Builder Project”.
   System prompts to overwrite the exiting project object data.
2. Click “Yes” to confirm.
3. Select the existing Panel Builder 600 project from the file system and click “Open”.
   The imported project is displayed.
1. In the Automation Builder device tree, right-click the Panel Builder 600 project and click “Export ➔ Panel Builder Project”.
2. Click “Browse” and select the desired location in the file system and save the project file.
   ⇨ A success message is displayed, if the project file exports successfully.

When you double-click the Panel Builder project node, the compressed information of the node is extracted into a temporary folder and then the external Panel Builder program is started. After the external Panel Builder program is closed, the corresponding Panel Builder files can be compressed back into the node and saved in the Automation Builder project.

We recommend to edit the Panel Builder project by starting Panel Builder through the Automation Builder. You can also export a Panel Builder project to the file system to edit the project by using the external Panel Builder. Then, reimport it to Automation Builder.

1.8.2 SCADA Integration

Overview
This document describes SCADA integration configuration in Automation Builder using zenon editor. The configured device network address information and variables are synchronized with zenon editor to avoid double entry.

The Automation Builder supports both standard and multi-user functionality.

1.8.2.1 Creating Workspace and Project
1. In the device tree, double-click “zenon_Project”.

⇨ To launch the zenon editor, click [Launch Zenon Editor].
To update the zenon project with latest changes of application program, click [Update zenon project].
2. Select the required PLC and select the “Use Standard Conn. Settings” option to use as a standard gateway connection. This enables the user to use the same communication settings that Automation Builder uses to communicate to the PLC.

![The configured gateway communication settings made in Automation Builder are displayed in the column 'Connection Type'.]

As an alternative you can create custom communication settings: Deselect the “Use Standard Conn. Settings” option and click the button in the 'Details' column.

3. Click [Launch Zenon Editor] to create a new workspace and project.

![Fig. 286: Connect to zenon project](image)

If Zenon Editor is already running, then select the “Use current workspace” option.

4. Select the “Create a new workspace” option and select the file location to create a new workspace.
5. Select the “Create new project” option to create a project.

⇒ ABB zenon editor is displayed.

![If you update or change an Automation Builder project with new variables or data types (new, modified, deleted), recompile the application to refresh the symbol file and click [Update zenon project].]

PLC Automation with V3 CPUs
Human machine interface > SCADA Integration

2020/12/10
After creating the project and workspace in Automation Builder, it is not required to set it again for the zenon object. A double-clicking on the zenon project shows the previously configured zenon project and the workspace.

### 1.8.2.2 Loading existing Workspace and Project

You can load an existing workspace and project to ABB zenon supervisor.

1. In the zenon_Project screen, click **Update zenon project**.
   - Connection to the zenon project dialog is displayed.
2. In the workspace area, enable “Load existing workspace” and select the location.
3. In the project area, enable “Select loaded project” and click **OK**.
   - Zenon editor loads the selected existing workspace and the project.

### 1.8.2.3 Checking the Gateway Settings in a Zenon Project

The gateway settings configured in Automation Builder can be checked in a zenon project. The IP address configured in Automation Builder are displayed in the zenon driver configuration.

In the Project Manager structure of the zenon editor, click “Variables ➔ Drivers” to configure the driver configuration.

The “Settings” tab shows all gateway settings based on the number of configured PLCs in Automation Builder. The IP address should be similar to the project gateway settings in Automation Builder.

---

*In the zenon project window, the Connect column should be checked to transfer the desired number of PLC connection settings to the zenon editor.*
1.8.2.4 Generating a Symbol File

Before generating the symbol file, define the variables in the CODESYS application.

1. In the CODESYS application main menu, click “Project ➔ Options”.
2. In the “Options” dialog, click “Symbol configuration”. Enable “Dump symbol entries” and “Dump XML symbol table” and click “Configure symbol file”.
   Set object attributes dialog is displayed.
3. Enable “Export variables of object”. If this option has a gray background, double-click on it to activate.
4. In the CODESYS application window, click at the bottom of the window and click “Resources ➔ Tools ➔ Target settings”.
5. In the target settings dialog, open the “General” tab and enable “Download symbol file”.
6. From the CODESYS application main menu, select “Project ➔ Build” to compile the project.

Precondition to generate a symbol file is to create the application and perform a PLC program build in CODESYS application.

The symbol file is generated after the build. The data exchange can be transferred to the zenon project by clicking [Update zenon project] in Automation Builder.

1.8.2.5 Updating Standard Data Types

The standard data types created in CODESYS application can be updated to the zenon project by clicking on “Update zenon project”.

Data types and variables can be updated from the desired number of PLCs configured in the zenon project of Automation Builder.

In the zenon project, double-click “Variables” and check the updated standard data type.
1.8.2.6 Creating Data Types

1. In the CODESYS application open the “Data types” tab. Right-click “Data types ➔ Add object” to create a new data type.
2. Enter the user defined data type name.
3. In “POUs” tab, add the user defined variable data type and compile.
   ⇒ The user defined data type is created and can be imported in the zenon editor.

   If you modify or delete the data types in CODESYS application, compile with “Rebuild all option”.

1.8.2.7 Importing Data Types in zenon Editor

1. In the zenon project, click [Update zenon project] to update the data types.
2. Click “Update” to update the variables and data types to the zenon project.
   ⇒ The user defined variables and data types are imported to the zenon project.

Fig. 287: User defined variables

1.9 Contact ABB

If you have questions on any ABB product you receive further contact information for your specific device in the ABB Global Contact Directory: [http://www.abb.com/contacts](http://www.abb.com/contacts).
If you have questions on Automation Builder software you receive further information on [http://www.abb.com/automationbuilder](http://www.abb.com/automationbuilder).

### 1.9.1 ABB contact for PLC and panel builder products

If you have questions on a PLC product you receive further information on [http://www.abb.com/plc](http://www.abb.com/plc).

#### Service - Control engineering

ABB operates a Technical Support Center to assist in the case of problems.

- Consultation by phone (Helpline)
- Failure analysis and removal at the machine/installation

In order to provide the customer with qualified assistance, customers are requested to provide the following:

- All required application documentation and information.
- Access (if required) to the system being examined.
- Assign key individuals to assist as required.

For inquiries and orders, please contact:

**ABB Automation Products GmbH**

Wallstader Straße 59
68526 Ladenburg
Germany

Phone sales: +49 6203 717-717
Phone Technical Support: +49 6221 701-1444
E-Mail: plc.support@de.abb.com

#### Repair services and replacement devices

The following services are covered by our activities:

- Delivery of replacement devices
- Repair of defective devices and systems

Please contact your responsible sales office for your inquiry.

#### Seminars/Workshops

ABB also offers seminars and workshops to support you in planning, commissioning and operation of AC500 devices. If desired, we offer seminars at the customer’s facility.

If necessary, it is also possible to arrange the seminar contents according to the customer’s wishes. Depending on the special requirements, the training can be held in Heidelberg or at any other location favored by the customer. We are pleased to prepare an individual offer for you according to your special requirements.
2 Index

1, 2, 3 ...

__AdrInst .................................. 652
__BitOffset .............................. 652
__Cast ................................ 652
__CATCH ................................ 546
__COMPARE_AND_SWAP .......... 552
__Copy ................................ 652
__CRC .................................. 652
__CURRENTTASK ........................ 551
__DELETE ................................ 538
__ENDTRY’ ................................ 546
__FCall ................................ 652
__FINALLY ................................ 546
__Init ................................... 652
__ISINVALIDREF .................. 541, 579
__Lazy .................................. 652
__Localoffset ....................... 652
__MaxOffset ......................... 652
__NEW .................................. 541
__PropertyInfo ..................... 652
__QUERYINTERFACE .................. 544
__QUERYPOINTER ...................... 545
__RefAdr ................................ 652
__RELOC ................................ 652
__System.ExceptionCode, data type 546
__SystemScope ....................... 652
__TEST_AND_SET ..................... 554
__TRY .................................. 546
__TypeOf ................................ 652
__uxint ................................ 577
__VarInfo ............................... 652
__VCADD ................................ 588
  vector operator .................. 588
__VCDIV ................................ 589
  vector operator .................. 589
__VCDOT ................................ 590
  vector operator .................. 590
__VCLOAD_LREAL ..................... 592
  vector operator .................. 592
__VCLOAD_REAL ....................... 592
  vector operator .................. 592
__VCMAX ................................ 591
  vector operator .................. 591
__VCMIN ................................ 591
  vector operator .................. 591
__VCMUL ................................ 589
  vector operator .................. 589
__VCSET_LREAL ....................... 592
  vector operator .................. 592
__VCSET_REAL ......................... 591
  vector operator .................. 591
__VCSQRT ................................ 590
  vector operator .................. 590
__VCSTORE ................................ 593
  vector operator .................. 593
__VCSUB ................................ 588
  vector operator .................. 588
__VECTOR ................................ 586
__Wait .................................. 652
__XADD ................................ 553
__xword ................................ 577
_cnt.library .............................. 372
__Itfs.library .............................. 372
//--compare, command line .......... 367
//--culture, command line .......... 366
//--profile, command line .......... 366
//--project, command line .......... 367
//--projectarchive, command line . 367
//--runscript, command line .......... 367
//--skipunlicensedplugins, command line . 371
% ........................................... 1546
format definition %c, %s for text variables . 1548
format definition in the output text, visualization . 1546
3rd party device diagnosis ............... 3519
1939 ECU
  RX signals . 3345
  TX signals . 3343
1939 manager
  general ................................ 3341
A
ABS ..................................... 534
AC500_Diag ............................. 3533, 3543
<table>
<thead>
<tr>
<th>Keyword</th>
<th>Page Numbers</th>
</tr>
</thead>
<tbody>
<tr>
<td>AC500_DiagTypes</td>
<td>3526, 3527</td>
</tr>
<tr>
<td>access</td>
<td></td>
</tr>
<tr>
<td>symbol set</td>
<td>275</td>
</tr>
<tr>
<td>access control</td>
<td></td>
</tr>
<tr>
<td>properties</td>
<td>1050</td>
</tr>
<tr>
<td>access protection</td>
<td></td>
</tr>
<tr>
<td>development system</td>
<td>423</td>
</tr>
<tr>
<td>general information</td>
<td>421</td>
</tr>
<tr>
<td>runtime system / PLC</td>
<td>423</td>
</tr>
<tr>
<td>user management</td>
<td>421</td>
</tr>
<tr>
<td>access right</td>
<td></td>
</tr>
<tr>
<td>device editor</td>
<td>762</td>
</tr>
<tr>
<td>object</td>
<td>141</td>
</tr>
<tr>
<td>visualization element</td>
<td>1583</td>
</tr>
<tr>
<td>accessor</td>
<td>791</td>
</tr>
<tr>
<td>access method</td>
<td>791</td>
</tr>
<tr>
<td>access method, interface</td>
<td>788</td>
</tr>
<tr>
<td>Accumulator</td>
<td>3461</td>
</tr>
<tr>
<td>Ack</td>
<td>3533, 3543</td>
</tr>
<tr>
<td>Acked_Transitions</td>
<td>3460</td>
</tr>
<tr>
<td>acknowledge</td>
<td></td>
</tr>
<tr>
<td>alarm</td>
<td>715</td>
</tr>
<tr>
<td>AcknowledgeAlarm</td>
<td>3454</td>
</tr>
<tr>
<td>ACOS</td>
<td>538</td>
</tr>
<tr>
<td>action</td>
<td>454</td>
</tr>
<tr>
<td>SFC, do not display embedded objects</td>
<td>976</td>
</tr>
<tr>
<td>SFC, duplication mode</td>
<td>970</td>
</tr>
<tr>
<td>SFC, remove duplication</td>
<td>975</td>
</tr>
<tr>
<td>SFC, set duplication</td>
<td>975</td>
</tr>
<tr>
<td>Action</td>
<td>794</td>
</tr>
<tr>
<td>ACTION</td>
<td>652</td>
</tr>
<tr>
<td>action association</td>
<td></td>
</tr>
<tr>
<td>insert, command</td>
<td>972</td>
</tr>
<tr>
<td>activate keyboard usage</td>
<td></td>
</tr>
<tr>
<td>command, visualization</td>
<td>1560</td>
</tr>
<tr>
<td>active callbacks number</td>
<td>2129</td>
</tr>
<tr>
<td>ActiveX element</td>
<td>1479, 1883</td>
</tr>
<tr>
<td>visualization element</td>
<td>1479, 1883</td>
</tr>
<tr>
<td>add</td>
<td></td>
</tr>
<tr>
<td>EN/ENO</td>
<td>978</td>
</tr>
<tr>
<td>file as text</td>
<td>866</td>
</tr>
<tr>
<td>input pin</td>
<td>987</td>
</tr>
<tr>
<td>language in a text list</td>
<td>1020</td>
</tr>
<tr>
<td>output pin</td>
<td>987</td>
</tr>
<tr>
<td>ADD</td>
<td>509</td>
</tr>
<tr>
<td>add all instance paths</td>
<td>1012</td>
</tr>
<tr>
<td>add file as text</td>
<td>866</td>
</tr>
<tr>
<td>add folder</td>
<td>897</td>
</tr>
<tr>
<td>add POU</td>
<td></td>
</tr>
<tr>
<td>dialog</td>
<td>776</td>
</tr>
<tr>
<td>add user</td>
<td></td>
</tr>
<tr>
<td>device user management</td>
<td>760</td>
</tr>
<tr>
<td>additional device-specific diagnosis</td>
<td>3519</td>
</tr>
<tr>
<td>AddListElement</td>
<td>3455</td>
</tr>
<tr>
<td>address</td>
<td>566</td>
</tr>
<tr>
<td>absolute</td>
<td>273</td>
</tr>
<tr>
<td>assign variable</td>
<td>228</td>
</tr>
<tr>
<td>broadcast</td>
<td>274</td>
</tr>
<tr>
<td>I/O</td>
<td>168</td>
</tr>
<tr>
<td>network</td>
<td>272</td>
</tr>
<tr>
<td>node</td>
<td>273</td>
</tr>
<tr>
<td>relative</td>
<td>273</td>
</tr>
<tr>
<td>specification</td>
<td>566</td>
</tr>
<tr>
<td>addressing</td>
<td>271</td>
</tr>
<tr>
<td>relative</td>
<td>271</td>
</tr>
<tr>
<td>ADR</td>
<td>524, 577</td>
</tr>
<tr>
<td>AI523</td>
<td>2600</td>
</tr>
<tr>
<td>AI531</td>
<td>2621</td>
</tr>
<tr>
<td>AI561</td>
<td>2543</td>
</tr>
<tr>
<td>AI562</td>
<td>2554</td>
</tr>
<tr>
<td>AI563</td>
<td>2565</td>
</tr>
<tr>
<td>alarm</td>
<td>258, 3519</td>
</tr>
<tr>
<td>acknowledgement</td>
<td>715</td>
</tr>
<tr>
<td>activation</td>
<td>718</td>
</tr>
<tr>
<td>archiving</td>
<td>718</td>
</tr>
<tr>
<td>confirmation</td>
<td>715</td>
</tr>
<tr>
<td>create objects for alarm management</td>
<td>258</td>
</tr>
<tr>
<td>definition</td>
<td>258</td>
</tr>
<tr>
<td>event</td>
<td>258</td>
</tr>
<tr>
<td>export</td>
<td>721</td>
</tr>
<tr>
<td>import</td>
<td>721</td>
</tr>
<tr>
<td>message</td>
<td>718</td>
</tr>
<tr>
<td>notification action</td>
<td>715</td>
</tr>
<tr>
<td>status</td>
<td>258</td>
</tr>
<tr>
<td>text</td>
<td>718</td>
</tr>
<tr>
<td>visualization</td>
<td>1146</td>
</tr>
<tr>
<td>alarm acknowledgement</td>
<td>1582</td>
</tr>
<tr>
<td>alarm banner</td>
<td>1402, 1806</td>
</tr>
<tr>
<td>visualization element</td>
<td>1402, 1806</td>
</tr>
<tr>
<td>alarm class</td>
<td>715</td>
</tr>
<tr>
<td>C sources</td>
<td></td>
</tr>
<tr>
<td>-----------</td>
<td>-------</td>
</tr>
<tr>
<td>export</td>
<td>921</td>
</tr>
<tr>
<td>update</td>
<td>920</td>
</tr>
<tr>
<td>C stub file for external library</td>
<td>917</td>
</tr>
<tr>
<td>C0001</td>
<td>660</td>
</tr>
<tr>
<td>C0002</td>
<td>661</td>
</tr>
<tr>
<td>C0004</td>
<td>661</td>
</tr>
<tr>
<td>C0005</td>
<td>662</td>
</tr>
<tr>
<td>C0006</td>
<td>662</td>
</tr>
<tr>
<td>C0007</td>
<td>662</td>
</tr>
<tr>
<td>C0008</td>
<td>663</td>
</tr>
<tr>
<td>C0009</td>
<td>663</td>
</tr>
<tr>
<td>C0010</td>
<td>663</td>
</tr>
<tr>
<td>C0011</td>
<td>664</td>
</tr>
<tr>
<td>C0013</td>
<td>664</td>
</tr>
<tr>
<td>C0015</td>
<td>664</td>
</tr>
<tr>
<td>C0018</td>
<td>665</td>
</tr>
<tr>
<td>C0020</td>
<td>665</td>
</tr>
<tr>
<td>C0022</td>
<td>666</td>
</tr>
<tr>
<td>C0023</td>
<td>666</td>
</tr>
<tr>
<td>C0026</td>
<td>666</td>
</tr>
<tr>
<td>C0027</td>
<td>667</td>
</tr>
<tr>
<td>C0030</td>
<td>667</td>
</tr>
<tr>
<td>C0031</td>
<td>667</td>
</tr>
<tr>
<td>C0032</td>
<td>668</td>
</tr>
<tr>
<td>C0035</td>
<td>668</td>
</tr>
<tr>
<td>C0037</td>
<td>669</td>
</tr>
<tr>
<td>C0038</td>
<td>669</td>
</tr>
<tr>
<td>C0039</td>
<td>669</td>
</tr>
<tr>
<td>C0040</td>
<td>670</td>
</tr>
<tr>
<td>C0041</td>
<td>670</td>
</tr>
<tr>
<td>C0043</td>
<td>671</td>
</tr>
<tr>
<td>C0044</td>
<td>671</td>
</tr>
<tr>
<td>C0045</td>
<td>672</td>
</tr>
<tr>
<td>C0046</td>
<td>672</td>
</tr>
<tr>
<td>C0047</td>
<td>672</td>
</tr>
<tr>
<td>C0048</td>
<td>673</td>
</tr>
<tr>
<td>C0049</td>
<td>673</td>
</tr>
<tr>
<td>C0050</td>
<td>673</td>
</tr>
<tr>
<td>C0061</td>
<td>674</td>
</tr>
<tr>
<td>C0062</td>
<td>674</td>
</tr>
<tr>
<td>C0064</td>
<td>675</td>
</tr>
<tr>
<td>C0066</td>
<td>675</td>
</tr>
<tr>
<td>C0068</td>
<td>676</td>
</tr>
<tr>
<td>C0069</td>
<td>676</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>C0070</td>
<td></td>
</tr>
<tr>
<td>C0072</td>
<td></td>
</tr>
<tr>
<td>C0074</td>
<td></td>
</tr>
<tr>
<td>C0075</td>
<td></td>
</tr>
<tr>
<td>C0076</td>
<td></td>
</tr>
<tr>
<td>C0077</td>
<td></td>
</tr>
<tr>
<td>C0080</td>
<td></td>
</tr>
<tr>
<td>C0081</td>
<td></td>
</tr>
<tr>
<td>C0082</td>
<td></td>
</tr>
<tr>
<td>C0084</td>
<td></td>
</tr>
<tr>
<td>C0086</td>
<td></td>
</tr>
<tr>
<td>C0087</td>
<td></td>
</tr>
<tr>
<td>C0089</td>
<td></td>
</tr>
<tr>
<td>C0090</td>
<td></td>
</tr>
<tr>
<td>C0091</td>
<td></td>
</tr>
<tr>
<td>C0094</td>
<td></td>
</tr>
<tr>
<td>C0096</td>
<td></td>
</tr>
<tr>
<td>C0097</td>
<td></td>
</tr>
<tr>
<td>C0098</td>
<td></td>
</tr>
<tr>
<td>C0101</td>
<td></td>
</tr>
<tr>
<td>C0102</td>
<td></td>
</tr>
<tr>
<td>C0104</td>
<td></td>
</tr>
<tr>
<td>C0114</td>
<td></td>
</tr>
<tr>
<td>C0116</td>
<td></td>
</tr>
<tr>
<td>C0117</td>
<td></td>
</tr>
<tr>
<td>C0118</td>
<td></td>
</tr>
<tr>
<td>C0119</td>
<td></td>
</tr>
<tr>
<td>C0120</td>
<td></td>
</tr>
<tr>
<td>C0122</td>
<td></td>
</tr>
<tr>
<td>C0124</td>
<td></td>
</tr>
<tr>
<td>C0125</td>
<td></td>
</tr>
<tr>
<td>C0126</td>
<td></td>
</tr>
<tr>
<td>C0129</td>
<td></td>
</tr>
<tr>
<td>C0130</td>
<td></td>
</tr>
<tr>
<td>C0131</td>
<td></td>
</tr>
<tr>
<td>C0132</td>
<td></td>
</tr>
<tr>
<td>C0136</td>
<td></td>
</tr>
<tr>
<td>C0139</td>
<td></td>
</tr>
<tr>
<td>C0140</td>
<td></td>
</tr>
<tr>
<td>C0141</td>
<td></td>
</tr>
<tr>
<td>C0142</td>
<td></td>
</tr>
<tr>
<td>C0143</td>
<td></td>
</tr>
<tr>
<td>C0144</td>
<td></td>
</tr>
<tr>
<td>C0145</td>
<td></td>
</tr>
<tr>
<td>C0149</td>
<td></td>
</tr>
<tr>
<td>C0161</td>
<td></td>
</tr>
</tbody>
</table>
CIA405.SDO_WRITE4 .......................... 2141, 2146
CIA405.TRANSITION_STATE .................. 2138
class
Python ...................................... 408
clean all ................................... 919
clean gaps .................................. 1011
clock
visualization element .................. 1536, 1935
close directory ............................ 2175
close file ................................... 2177
closed branch, LD ...................... 475
CM ....................................... 2246
CM579 ...................................... 2297, 2301
CM598 ....................................... 2290
CmpBACnet .................................. 3440
  BACnet component ...................... 3444
CmpBACnet library ...................... 3440, 3441
CmpBACnet.library ......................... 3462
CmpTraceMgr
  runtime system component ............ 335
CmpTraceMgr.library ....................... 335
code ....................................... 308
  analysis ................................ 3688
  analysis, pragmas ................... 3611
  analyze ................................ 230
  code coverage ......................... 3688
  collapse all .......................... 868
  create .................................. 917
  encrypt ................................ 139
  encryption ............................. 304
  exclude from the static analysis .... 231
  expand all ................................ 868
  go to matching bracket ............... 868
  profiling ................................ 3688
  runtime behavior ..................... 3688
tests ....................................... 3607
code analysis
  forbidden symbols ....................... 3609
  getting started ......................... 3601
  metrics .................................. 3609
  naming conventions .................... 3608
  naming conventions, extended ........ 3610
  rules ..................................... 3607
code check
  forbidden symbols ....................... 3609
metrics ..................................... 3609
rules ....................................... 3607
settings .................................. 3606
code coverage
  function block ......................... 3693
  function block instance ............... 3693
  print table ................................ 3703
  Profiler .................................. 3688, 3693
  Profiler, online ....................... 3701
code generation .......................... 917
CODESYS Memory Tools ..................... 3597
CODESYS Redundancy ....................... 362
CoDeSys V2.3 Converter
  option .................................... 1077
  coding guidelines ....................... 230
  coil ....................................... 474
    insert ................................ 996
    reset .................................. 996
    set .................................... 996
cold start ................................ 931
collapse all ................................ 868
color ....................................... 1152
  code as hexadecimal number .......... 1152
  visualization .......................... 1115
  color animation
    configure for visualization element .................................. 1117, 1153
  color definition ........................ 1152
    byte order ................................ 1152
  color gradient
    specify for visualization element .................. 1116
  color space ................................ 1152
  combo box array ......................... 1311, 1709
    visualization element .................. 1311, 1709
  combo box integer ....................... 1305, 1714
    visualization element .................. 1305, 1714
command ‘Go to instance’ ............... 876
  command icon ................................ 1090
    customize ................................ 105
  command line ................................ 366
    --compare ................................ 367
    --culture ............................... 366
    --ignorecomments ....................... 370
    --ignoreproperties ..................... 370
    --ignorewhitespace ...................... 370
    --profile ................................ 366
content operator ................................ 524
Continuous Function Chart .................. 188
Continuous Function Chart (CFC) ............ 188
Continuous Function Chart (CFC) - page-oriented ............... 189, 477
control panel ................................ 1503, 1907
control point
 CFC ................................ 486
create ................................ 988
remove ................................ 988
controller
 communication, certificate-encrypted ......... 139
 security ................................ 423
 symbol access ............................. 766
 unencrypted communication .................. 427
cross-reference ............................. 888
create localization template ................. 903
create stub implementation in C ............ 921
CpuLoad
 DeviceTrace ................................ 343
 cross-reference ............................ 888
 cross-reference list ........................ 886
 classic view ................................ 888
 collapse all ................................ 868
 limit ....................................... 1038
 occurrence location ......................... 232
 view ....................................... 886
CSV export of alarm storage
 command .................................. 722
 configuration ............................ 722
CurrentVisu
 Variable for visualization name ........... 1613
customize ................................... 1089
D
 D ............................................ 566, 572
 constant ................................ 560
 keyword ................................ 572
 literal .................................. 560
 size prefix ................................ 566
DA501 ..................................... 2719
DA502 ..................................... 2752
conversion .................................. 38, 2264, 3507
copy file .................................. 2179
core dump .................................. 349
close ....................................... 948
create ....................................... 947
load ......................................... 947
load device log ................................ 948
open memory view .......................... 891
COS ...................................... 537
CPU display
 diagnosis description ...................... 3520
CPU load .................................. 342, 839
cpuload
 multicore .................................. 242
trace ....................................... 335, 1032
CpuLoad
 DeviceTrace ................................ 343
 cross-reference ............................ 888
 cross-reference list ........................ 886
 classic view ................................ 888
 collapse all ................................ 868
 limit ....................................... 1038
 occurrence location ......................... 232
 view ....................................... 886
DA501 ..................................... 2719
DA502 ..................................... 2752
convert V2 project to V3 project ............. 38, 2264, 3507
convertion ...................... 38, 2264, 3507
copy file .................................. 2179
core dump .................................. 349
close ....................................... 948
create ....................................... 947
load ......................................... 947
load device log ................................ 948
open memory view .......................... 891
COS ...................................... 537
CPU display
 diagnosis description ...................... 3520
CPU load .................................. 342, 839
cpuload
 multicore .................................. 242
trace ....................................... 335, 1032
CpuLoad
 DeviceTrace ................................ 343
 cross-reference ............................ 888
 cross-reference list ........................ 886
 classic view ................................ 888
 collapse all ................................ 868
 limit ....................................... 1038
 occurrence location ......................... 232
 view ....................................... 886
DA501 ..................................... 2719
DA502 ..................................... 2752
convert V2 project to V3 project ............. 38, 2264, 3507

decimal number
  format definition %d, %i ............................... 1547
declaration ............................................ 3611
attribute, static analysis .......................... 3611
Auto Declare ........................................ 209
change order ........................................ 238
global variable ...................................... 176
go to .............................................. 234
refactoring .......................................... 238
variable .............................................. 172
declaration editor .................................. 173, 428edit declaration header ............................ 1009
option ............................................. 1078
show/hide ........................................... 964	
	
tabular/textual ...................................... 173
declare
  array .............................................. 175
automatic ........................................... 1082
short form feature .................................. 210
task-local global variable list .................. 177
variable, command ................................ 872
default keyboard shortcuts
  visualization manager ............................ 1617
default keyboard usage
  activate, visualization .......................... 1134
define, pragma ..................................... 637
DELETE ............................................. 538
delete callbacks ..................................... 2129
Delete Directory .................................. 2174
delete file .......................................... 2180
delete IL line ....................................... 999
DeleteObject ......................................... 3459
deleting callbacks handleknown ................... 2133
deleting callbacks handleunknown ................ 2133
dereferencing ...................................... 577
design attribute .................................... 627
devdesc.xml ......................................... 957
development system
  appearance and behavior .......................... 102
customize user interface ........................... 102
Development System
  Features .......................................... 100
device ............................................... 740
add ................................................ 898
application ........................................ 747
configuration ....................................... 158, 741
configuration mode .................................. 915
configure .......................................... 162
connect ............................................ 741
connection ......................................... 936
conversion ......................................... 1041
database ............................................ 957
device editor ........................................ 158
device tree .......................................... 155
devices view ........................................ 159, 881
disable .............................................. 913
encrypted communication ......................... 296, 741, 3597
favorite ............................................. 741
files ................................................ 750
function block instance ........................... 759
I/O mapping ......................................... 755
IEC objects ......................................... 759
insert .............................................. 912
install ............................................. 374, 957
interactive login ................................... 1058
log ................................................... 750
map I/Os ............................................ 164
online config mode ................................ 915
options ............................................. 1058
PLC settings ........................................ 751
PLC shell ............................................ 351
plug ................................................... 899
properties .......................................... 1058
scan ................................................ 160, 741, 3376
scan of the hardware .............................. 899, 3345, 3375
security ............................................. 296
send echo service ................................... 741
simulation mode .................................... 309, 937
status ............................................... 158
symbol access ....................................... 766
uninstall ............................................. 957
update ............................................... 901
user management ................................... 300, 760, 762
wink ................................................... 741
device application
  concept .......................................... 358
login to all applications ......................... 1036
logout all applications ........................... 1036
object .............................................. 712
persistent variable configuration ................ 714
Index

Persistent variables ........................................... 358
reset cold all applications ................................ 1036
reset warm all applications ................................ 1037
start all applications ....................................... 1037
stop all applications ........................................ 1037
symbol configuration ......................................... 358, 825
Task configuration .............................................. 358
device description
download ........................................................... 957
download, option ................................................. 1079
install .................................................................. 957
device diagnosis .................................................. 3526, 3540
device diagnostics .................................................. 1099
device editor .......................................................... 758
access rights .......................................................... 762
add user ................................................................. 761
applications ............................................................ 747
backup and restore ............................................... 748
communication ...................................................... 741
communication settings .......................................... 741
encrypted communication ........................................ 741
EtherCAT ................................................................. 3347
files .................................................................. 357, 750
generic .................................................................. 741
I/O mapping ............................................................ 755
IEC objects ................................................................ 759
information ................................................................ 768
KNX ......................................................................... 3436
log ........................................................................ 750
options ................................................................... 1081
parameters ............................................................... 746
PLC settings .............................................................. 751
PLC shell ................................................................. 753
PROFINET IO ............................................................. 3363
status .................................................................... 768
symbol rights ............................................................ 766
synchronized files .................................................. 749
task deployment ....................................................... 767
users and groups ...................................................... 760
device reader ............................................................. 960
device repository ...................................................... 957
renew .................................................................... 957
device state ............................................................. 3519, 3524, 3531, 3541
device tree .............................................................. 155
Python ................................................................... 411
device user
add .................................................................. 934
change password .................................................... 935
logout current user .................................................. 934
remove .................................................................. 935
device user management .......................................... 300
device version ......................................................... 1071
DEVICE_STATE ....................................................... 3541
DeviceCommunicationControl .................................. 3459
devices
view ..................................................................... 155
devices view ............................................................. 881
DeviceTrace .............................................................. 335
CPU load ................................................................. 342
display processor load ............................................. 242
download ................................................................. 1032
object .................................................................. 845
DlS54 .................................................................. 2490
DlS61 .................................................................. 2346
DlS62 .................................................................. 2352
DlS71 .................................................................. 2360
DlS72 .................................................................. 2369
Diag (function block) ................................................. 3533, 3543
DIAG_TXT_TYPE ....................................................... 3528
DIAGVAL_TYPE ........................................................ 3528
Diag.NumClass ........................................................ 3531
Diag.NumTotal ........................................................ 3531
DiagAck ................................................................. 3533, 3543
DiagGet... (method) .................................................. 3533, 3543
diagnosis
core dump .............................................................. 349
device diagnosis ....................................................... 3526
PLC shell ................................................................. 351
system diagnosis ..................................................... 3526, 3531
V3 ........................................................................ 3518
diagnosis description ................................................. 3519, 3520, 3525, 3527
diagnosis message
V3 ........................................................................ 3519
diagnosis messages list
CM579-PNIO .......................................................... 3582
CPU .......................................................... 3565
I/O bus ................................................................. 3567
S500 I/O module ...................................................... 3569
diagnosis system
V3 ........................................................................ 3518
diagnostics
fieldbus .................................. 1099
diagram .................................. 1092
autofit Y-trace axis ..................... 1025
in trace editor .......................... 1092
mouse zooming .......................... 1029
DiagValToTxt ............................. 3539, 3549
dialog .................................... 1194, 1199
calling in a visualization ............... 1194
close, input action ...................... 1587
implement with interface ............... 1199
open, input action ...................... 1588
opening globally in a visualization .... 1196
user management for visualization ..... 1615
dialog manager .......................... 1552
methods .................................. 1552
dictionary
Python ................................... 404
DINT ..................................... 570
dip switch ............................... 1457, 1861
visualization element .................. 1457, 1861
directory attributes .................... 2181, 2183
directory example ........................ 2187
disable breakpoint ...................... 940
disassembly file ......................... 922
disconnect
device ................................... 936
display variant .......................... 1210
configure .............................. 1210
executing as integrated ............... 1213
executing webvisu ...................... 1211
maximum number ...................... 1617
size of the paintbuffer ............... 1616
TargetVisu ................................ 1623
WebVisu .................................. 1624
displaymode, pragma .................... 607
DIV ....................................... 511
division by zero ........................ 801, 802, 803
DO ......................................... 437
DO524 .................................. 2500
DO526 .................................. 2509
DO561 .................................. 2378
DO562 .................................. 2386
DO571 .................................. 2396
DO572 .................................. 2406
DO573 .................................. 2416
document .................................. 905
donngle .................................. 144, 953
encryption ............................. 242
project settings ....................... 1064
dot product ............................. 590
download
device description from the server .. 1079
encrypt code ............................ 242
library from server .................... 1075
multiple ............................... 929
source code, project setting .......... 1063
trace ..................................... 1026
user data, visualization .............. 1621
download source code ................... 307
download to controller
multiple ............................... 929
DT ........................................ 572
constant ............................... 560
keyword ............................... 572
 literal ............................... 560
DTU.DateConcat ......................... 2163
DTU.DateSplit ........................... 2163
DTU.DTConcat ........................... 2164
DTU.DTSplit ............................. 2164
DTU.ERROR .............................. 2159
DTU.gc_tzTimeZoneCET ................. 2159
DTU.GetDateAndTime .................... 2160
 DTU.GetDayOfWeek ...................... 2162
 DTU.GetTimeZoneInformation .......... 2162
 DTU.SetDateAndTime ................... 2161
 DTU.SetTimeZoneInformation .......... 2161
DTU.TODConcat .......................... 2163
DTU.TODSplit ........................... 2163
dum file .................................. 760
duplication ............................. 970
remove ................................. 975
SFC ..................................... 970
SFC, set ............................... 975
duplication mode ....................... 970
DUT ...................................... 735
add ....................................... 735
WORD ................................... 570
DX522 .................................. 2520
DX531 .................................. 2532
DX561 ................................ 2428
DX571 ................................ 2440
dynamic memory allocation ................. 541

E
echo service ................................ 741
EDE export
  BACnet ................................ 3445
EDE file ................................ 3446
export, import ................................ 3446
EDE import
  BACnet ................................ 3445
editing mode
  graphical editor ........................ 429
editor
  close all .............................. 961
  close all (inactive applications) ..... 962
  close all other ......................... 965
  next ................................ 961
  previous ................................ 961
  visualization ........................ 1608
element properties ......................... 883
  SFC ................................ 459
element property
  visualization ........................ 1611
elementlist
  command, visualization ................ 1559
  tab, visualization .................... 1559
elements for alarms acknowledgement ...... 1582
ellipse .................................. 1224, 1628
  visualization element ................ 1224, 1628
ELSE .................................. 436
ELSIF .................................. 436
empty box
  insert ................................ 994
empty box with eno ......................... 994
EN/ENO
  add ................................ 978
  FBD/LD/IL .......................... 471
enable
  breakpoint ............................ 940
ENABLE ................................ 2143
enable_dynamic_creation, pragma .......... 637
encrypted communication
  device editor ................................ 741
encryption ................................ 421
application ................................ 242
boot application, certificate ............. 3598
certificate ................................ 139
certificate controller ...................... 3597
communication with PLC .................... 296
dialog, security screen .................... 891
dongle .................................. 242
download code ................................ 242
method ................................ 421
project ................................ 144
properties, application ................... 1047
Security Agent ........................... 3597
signature ................................ 421
  with certificate, instructions ........ 149
download code encryption .................. 349
end of file ................................ 2182
END_ACTION ................................ 652
END_CASE ................................ 438
END_FOR ................................ 437
END_FUNCTION .......................... 652
END_FUNCTION_BLOCK .................... 652
END_IF ................................ 436
END_PROGRAM ........................... 652
END_REPEAT ................................ 439
END_STRUCT ............................ 594
END_TYPE ................................ 595
END_UNION ................................ 489
END_VAR ................................ 439
END_WHILE ................................ 640
ENDTRY’ ................................ 546
enlarging/reducing a pin group ............ 626
entry action ................................ 455
  SFC, add ................................ 970
enumeration ................................ 595
  conversion TO_STRING .................. 634
data type ................................ 595
default value ............................. 595
initialization ............................. 595
namespace ................................ 556
  pragma strict ........................ 596
EQ .................................... 523
ERROR ................................ 2143, 3542
error analysis ............................ 349
error codes cia405 canopen kernel ........ 2139
online operation .................................. 465
view as function block diagram .......... 1003
view as instruction list .................... 1003
view as ladder logic ....................... 1003
FBD/LD/IL editor ............................. 461
line branch .................................. 1002
toolbar ........................................ 429

Features
Development System ........................ 100
fieldbus diagnostics ....................... 1099
file
add ........................................... 736
load by application download .......... 749
save .......................................... 150
save as ....................................... 150
to and from PLC ............................. 357
transfer, input action ...................... 1596

File - Modification .......................... 2190
File - Standard ................................ 2188
File – internal pointer ..................... 2191
File – Properties ............................ 2194
file attributes ................................ 2181, 2183
file internal pointer example .......... 2191
file modification example ............... 2190
file size ...................................... 2182
file standard example ..................... 2188
file transfer
configure mode .............................. 1215
visualization - PLC ......................... 1596

File transfer
on controller and visualization ........ 1616
FILE.ATTRIB ................................. 2170
FILE.Close .................................. 2177
FILE.Copy .................................... 2179
FILE.Delete .................................. 2180
FILE.DIR_ENTRY .............................. 2171
FILE.DirClose ................................ 2175
FILE.DirCreate ................................ 2173
FILE.DirList .................................. 2174
FILE.DirOpen ................................. 2173
FILE.DirRemove .............................. 2174
FILE.DirRename .............................. 2175
FILE.EOF ..................................... 2182
FILE.ERROR .................................. 2171
FILE.Flush .................................... 2180
select for I/O mapping ................................ 1040
Function block ...................................... 778
call ................................................... 778
function block diagram ........................... 182
function pointers .................................... 577
FUNCTION_BLOCK .................................. 778

G

gateway
  add ............................................. 741
  block driver ................................ 1013
  configuration ................................ 1013
  configuration file ........................... 1013
  manage ...................................... 741
gateway.cfg ..................................... 1013
GE .................................................. 523
general .......................................... 3341
  1939 manager ................................ 3341
  CANbus ....................................... 3347
  CANopen manager ............................ 3332
  CANopen remote device ...................... 3335
  EtherCAT slave ................................ 3350
  KNX ........................................... 3437, 3438
General
  EtherCAT master ................................ 3348
generate code
  application code ............................ 303
  messages ..................................... 304
gesture
  for operating a visualization ................ 1126
get
  access method, interface .................... 788
Get
  accessor method ................................ 791
Get... (method) .................................. 3533, 3543
GetAlarmSummary ................................. 3454
GetEnrollmentSummary ............................ 3454
GetEventInformation ............................ 3457
getting started
  display histogram ............................ 1958
  trend visualization ........................... 1165
global namespace operator ..................... 555
global network variable list .................... 277
global text list
  add language and translate text ............ 214
  check .......................................... 217
  compare and export differences ............. 216
  create ........................................ 1020
  create again with current IDs .............. 218
  enter text in visualization element ....... 217
  export ........................................ 214
  for static application ...................... 217
  import file ................................... 215
  object ........................................ 768
  remove text list entries .................... 218
  update ID .................................... 218
  update with replacement file .............. 218
global variable .................................. 495
  declare ...................................... 176
global variable list ............................ 769
declare task-local ................................ 177
task-local ...................................... 769
global variables list
  namespace .................................... 555
global_init_slot, pragma ........................ 612
GlobalImagePool .................................. 221
go to
  definition, how to ........................... 234
  line .......................................... 867
  matching bracket ............................. 868
  network ..................................... 1004
go to definition
  command ...................................... 875
go to source position ......................... 882
gradient editor
  visualization ................................ 1586
graphical editor toolbar ....................... 429
grid
  visualization ................................ 1600
group ............................................. 760
  CFC, create ................................... 989
  CFC, remove .................................. 989
configure, visualization ....................... 1140
  create for the first time, visualization .... 1139
in the visualization editor .................... 1564
user management ................................ 140
user management, visualization .............. 1618
group box ....................................... 1333, 1737
visualization element ........................... 1333, 1737
<table>
<thead>
<tr>
<th>Index</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>image</td>
<td>1272, 1676</td>
</tr>
<tr>
<td>visualization element</td>
<td>1272, 1676</td>
</tr>
<tr>
<td>image file</td>
<td></td>
</tr>
<tr>
<td>insert</td>
<td>1009</td>
</tr>
<tr>
<td>names and directories for visualization</td>
<td>1600</td>
</tr>
<tr>
<td>image pool</td>
<td>221, 770</td>
</tr>
<tr>
<td>create</td>
<td>222</td>
</tr>
<tr>
<td>download</td>
<td>1057</td>
</tr>
<tr>
<td>global</td>
<td>221</td>
</tr>
<tr>
<td>object</td>
<td>770</td>
</tr>
<tr>
<td>properties</td>
<td>1057</td>
</tr>
<tr>
<td>image selection</td>
<td>771</td>
</tr>
<tr>
<td>image switcher</td>
<td>1448, 1852</td>
</tr>
<tr>
<td>visualization element</td>
<td>1448, 1852</td>
</tr>
<tr>
<td>implement interfaces</td>
<td>1038</td>
</tr>
<tr>
<td>IMPLEMENTS</td>
<td>266</td>
</tr>
<tr>
<td>import</td>
<td></td>
</tr>
<tr>
<td>I/O mapping</td>
<td>914</td>
</tr>
<tr>
<td>PLCopenXML</td>
<td>911</td>
</tr>
<tr>
<td>project in SVN</td>
<td>3720</td>
</tr>
<tr>
<td>Import</td>
<td>910</td>
</tr>
<tr>
<td>import assistant</td>
<td></td>
</tr>
<tr>
<td>configuration programming system</td>
<td>1039</td>
</tr>
<tr>
<td>import users</td>
<td></td>
</tr>
<tr>
<td>device user management</td>
<td>760</td>
</tr>
<tr>
<td>index</td>
<td>966</td>
</tr>
<tr>
<td>INDEXOF</td>
<td>513</td>
</tr>
<tr>
<td>information</td>
<td>967</td>
</tr>
<tr>
<td>device editor</td>
<td>768</td>
</tr>
<tr>
<td>EtherCAT</td>
<td>3347</td>
</tr>
<tr>
<td>KNX</td>
<td>3436</td>
</tr>
<tr>
<td>PROFINET IO</td>
<td>3363</td>
</tr>
<tr>
<td>INI</td>
<td>556</td>
</tr>
<tr>
<td>init step, SFC</td>
<td>967</td>
</tr>
<tr>
<td>init_namespace, pragma attribute</td>
<td>615</td>
</tr>
<tr>
<td>init_on_onchange, pragma attribute</td>
<td>616</td>
</tr>
<tr>
<td>initialization</td>
<td>172</td>
</tr>
<tr>
<td>array</td>
<td>581</td>
</tr>
<tr>
<td>avoid, pragma</td>
<td>623</td>
</tr>
<tr>
<td>input variable, pragma</td>
<td>615</td>
</tr>
<tr>
<td>namespace, pragma</td>
<td>615</td>
</tr>
<tr>
<td>online change, download</td>
<td>616</td>
</tr>
<tr>
<td>sequence, pragma</td>
<td>612</td>
</tr>
<tr>
<td>vector</td>
<td>586</td>
</tr>
<tr>
<td>with FB_Init, FB_Reinit</td>
<td>653</td>
</tr>
<tr>
<td>initialize_on_call, pragma attribute</td>
<td>615</td>
</tr>
<tr>
<td>inline monitoring</td>
<td>324</td>
</tr>
<tr>
<td>enable</td>
<td>869</td>
</tr>
<tr>
<td>example</td>
<td>324</td>
</tr>
<tr>
<td>input</td>
<td></td>
</tr>
<tr>
<td>CFC</td>
<td>486</td>
</tr>
<tr>
<td>device</td>
<td>164</td>
</tr>
<tr>
<td>input action</td>
<td>1586</td>
</tr>
<tr>
<td>visualization</td>
<td>1586</td>
</tr>
<tr>
<td>input assistance</td>
<td>208</td>
</tr>
<tr>
<td>Auto Declare</td>
<td>209</td>
</tr>
<tr>
<td>behavior in visualization</td>
<td>1600</td>
</tr>
<tr>
<td>input assistant</td>
<td>209</td>
</tr>
<tr>
<td>List components</td>
<td>209</td>
</tr>
<tr>
<td>short form feature</td>
<td>210</td>
</tr>
<tr>
<td>input assistant</td>
<td></td>
</tr>
<tr>
<td>categories</td>
<td>874</td>
</tr>
<tr>
<td>dialog</td>
<td>874</td>
</tr>
<tr>
<td>text search</td>
<td>875</td>
</tr>
<tr>
<td>Input Assistant</td>
<td></td>
</tr>
<tr>
<td>options</td>
<td>1082</td>
</tr>
<tr>
<td>input configuration</td>
<td>1586</td>
</tr>
<tr>
<td>visualization</td>
<td>1586</td>
</tr>
<tr>
<td>input event</td>
<td></td>
</tr>
<tr>
<td>visualization element</td>
<td>1125</td>
</tr>
<tr>
<td>input memory</td>
<td>566</td>
</tr>
<tr>
<td>input pin order</td>
<td>627</td>
</tr>
<tr>
<td>input variable</td>
<td>490</td>
</tr>
<tr>
<td>refactoring</td>
<td>237</td>
</tr>
<tr>
<td>input/output variable</td>
<td>491</td>
</tr>
<tr>
<td>constant</td>
<td>493</td>
</tr>
<tr>
<td>VAR_IN_OUT</td>
<td>491</td>
</tr>
<tr>
<td>insert</td>
<td>1009</td>
</tr>
<tr>
<td>box parallel LD</td>
<td>994</td>
</tr>
<tr>
<td>branch above</td>
<td>1001</td>
</tr>
<tr>
<td>branch below</td>
<td>1001</td>
</tr>
<tr>
<td>branch, ld</td>
<td>996</td>
</tr>
<tr>
<td>contact</td>
<td>999</td>
</tr>
<tr>
<td>contact parallel above</td>
<td>997, 998</td>
</tr>
<tr>
<td>contact parallel below</td>
<td>997</td>
</tr>
<tr>
<td>contact right</td>
<td>997</td>
</tr>
<tr>
<td>contact, ld</td>
<td>996</td>
</tr>
<tr>
<td>contact, negated parallel</td>
<td>998</td>
</tr>
<tr>
<td>empty box</td>
<td>994</td>
</tr>
</tbody>
</table>

2020/12/10
3ADR010583, 1, en_US
3791
image file ........................................ 1009
instruction line ................................ 999
line branch parallel .......................... 1001
negated contact ............................... 998
set coil ......................................... 996
text in a text list ............................. 1021
variable declaration in the tabular editor ... 1009
insert action association
   insert, command ............................. 973
insert assignment, FBD/LD/IL .................. 993
insert contact
   parallel above .............................. 997, 998
   parallel below ................................ 997
insert input
   box, FBD/LD/IL ................................ 995
insert jump ..................................... 995
insert jump label ................................ 995
insert network .................................. 992
insert network below .......................... 993
Inserting Callbacks ............................ 2132
install
   device ........................................ 374, 957
   library ....................................... 951
   package .................................... 377
instance path .................................. 1012
   add in global persistent variable list .... 1012
   add in variables configuration .......... 1012
instance variables ............................ 496
instance-path, pragma .......................... 616
instruction list ................................ 183
INT ............................................... 570
integer data type ............................. 570
   convert ..................................... 528
interface ....................................... 783
   call visualization with interface ....... 1189
   command, visualization .................... 1556
   editor, visualization ...................... 1557
   extend ....................................... 267
   implement ................................... 266
   property ..................................... 788
   update data source programmatically ..... 292
   update, visualization ...................... 1584
INTERFACE .................................... 783
interface method ................................ 788
INTERNAL ...................................... 784
   method ....................................... 784
   property ..................................... 791
Intrinsic reporting ............................ 3460
invisible input .................................. 1377, 1781
   visualization element ..................... 1377, 1781
IO
   S500 ......................................... 2250
   S500-eCo ..................................... 2249
io_function_block
   pragma attribute ............................ 1040
io_function_block_mapping .................... 617
   pragma attribute ............................ 617, 1040
IoConfig.par .................................. 163
IOxS ............................................ 3369
IP address ....................................... 26
IronPython
   scripting ..................................... 396
   strings ....................................... 420
J
   J1939 ......................................... 3340
   bus cycle .................................... 3340
   general ....................................... 3341
   JMP ............................................ 440, 466
   JMPC .......................................... 466
   JMPCN ......................................... 466
   jump .......................................... 458
   add ............................................ 973
   CFC .......................................... 487
   insert after .................................. 973
   insert, FBD/LD/IL ............................ 995
jump label
   FBD/LD/IL .................................... 472
   insert, FBD/LD/IL ............................ 995
   ST ............................................. 440
K
   key
      certificate .................................. 422
      key combination ............................ 106
      key file ...................................... 134
   keyboard
      call for virtual input, visualization .... 1128
      keyboard configuration
         command, visualization .................. 1558
Index

2020/12/10

tab, visualization ........................................ 1558
keyboard shortcut ........................................ 1091
customize .................................................. 106
keyboard shortcuts ........................................ 106
configure for elements .................................. 1131
configure for visualization .............................. 1132
for default keyboard action, visualization ............ 1555

keypad
  visualization .............................................. 1614

keyword ................................................... 652
uppercase .................................................. 1082

KNX ......................................................... 3436
device editor ............................................. 3436
ET5 .......................................................... 3438
general .................................................... 3437, 3438
I/O mapping ............................................... 3436
IEC objects ................................................ 3436
information ................................................ 3436
parameters ............................................... 3436
status ....................................................... 3436

L

label .......................................................... 487, 1301, 1705
  CFC ...................................................... 487
  visualization element .................................. 1301, 1705
ladder diagram ........................................... 183
lamp .......................................................... 1453, 1857
  visualization element .................................. 1453, 1857
language
  current in the visualization ......................... 1614
  help ...................................................... 1083
  project localization .................................. 153
  switch input action ................................... 1589
  user interface, command line ......................... 366
  user interface, options ................................ 1083
  visualization ............................................ 1143
latch variable ............................................ 718
LD .......................................................... 183, 466
  closed branch ......................................... 475
  online operation ...................................... 465
  option .................................................... 1079
  programming in ........................................ 186
LDN .......................................................... 466
LE .......................................................... 522

legend ........................................................ 1475, 1879
visualization element .................................... 1475, 1879

library
  add ........................................................ 1004
  adding to the project .................................. 373
  check compatibility ................................... 920
  checks_in_libs ........................................... 796
  convert library reference ............................. 1040
  create .................................................... 1105
  download ............................................... 771
  download, option ....................................... 1075
  export ................................................... 374, 951, 1008
  install ................................................... 951
  integrate ................................................ 771
  integrate in project .................................... 773
  Library Manager ......................................... 771
  library types ............................................ 372
  location .................................................. 951
  mapping definition ..................................... 1075
  namespace ................................................ 556, 771, 772
  options ................................................... 1075
  outdated versions ........................................ 1070
  placeholder ............................................. 1008
  profile ................................................... 951
  properties ................................................. 771, 1006
  protected, signed ....................................... 372
  reload ....................................................... 1005
  save as compiled library ............................... 856
  signing ................................................... 856
  uninstall ................................................... 951
  use POUs ................................................... 213

library development
  information ................................................. 372

Library Development Summary .......................... 1105

library manager
  general ..................................................... 371

Library Manager .......................................... 771

library project
  category .................................................. 811
  compiled library ........................................ 814
  license .................................................... 133
  licensing .................................................. 813
  sign ....................................................... 133, 814

library reference
  conversion ............................................... 1040

3ADR010583, 1, en_US

3793
<p>| M | memory range prefix | 566 |
| M4 interface file for external library | 917 |
| macro | 458 |
| add | 974 |
| insert after | 974 |
| SFC | 458 |
| zoom into | 974 |
| zoom out of | 974 |
| magnification tool | 429 |
| main action | 455 |
| manage localizations | 904 |
| mapping | 163 |
| I/O mapping | 163 |
| mapping (see I/O mapping) | 164 |
| MAX | 520 |
| Max_pres_Val | 3461 |
| Maximum_Value | 3461 |
| MC502 | 3028, 3092 |
| meter | 1428, 1832 |
| 90°, visualization element | 1415, 1819 |
| 180°, visualization element | 1421, 1825 |
| visualization element | 1428, 1832 |
| method | 784 |
| call | 267, 785 |
| call recursively | 269 |
| call with external implementation | 208 |
| example of recursive call | 609 |
| factorial calculation | 609 |
| FB_Init, FB_Reinit, FB.Exit | 653 |
| interface | 788 |
| monitor | 328 |
| object-oriented programming | 784 |
| Python | 408 |
| reaction to type change | 605 |
| recursive call | 786 |
| virtual call | 267 |
| METHOD | 784 |
| Methode | 496 |
| metrics | code analysis | 3609 |
| static analysis | 3609 |
| migrate V2 project to V3 project | 38, 2264, 3507 |
| migration | 38, 2264, 3507 |
| MIN | 520 |
| Min_Pres_Val | 3461 |
| Minimum_Value | 3461 |
| MOD | 512 |
| Modbus | parameters | 746 |
| RTU protocol | 3087 |
| TCP/IP protocol | 3182 |
| modifier. IL | 466 |
| module | call tree | 871 |
| Python | 409 |
| monitoring | area for arrays | 429, 1046 |
| CFC editor | 480 |
| display mode | 948 |
| enable inline monitoring | 869 |
| function call | 619 |
| inline | 324 |
| interval | 1058 |</p>
<table>
<thead>
<tr>
<th>Term</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>options</td>
<td>1083</td>
</tr>
<tr>
<td>pragma</td>
<td>619</td>
</tr>
<tr>
<td>properties</td>
<td>1058</td>
</tr>
<tr>
<td>property</td>
<td>619</td>
</tr>
<tr>
<td>using pous for implicit checks</td>
<td>262</td>
</tr>
<tr>
<td>Monitoring</td>
<td></td>
</tr>
<tr>
<td>SFC</td>
<td>443</td>
</tr>
<tr>
<td>implicit</td>
<td>796</td>
</tr>
<tr>
<td>Motion control library</td>
<td>2024</td>
</tr>
<tr>
<td>MOVE</td>
<td>512</td>
</tr>
<tr>
<td>move down</td>
<td>1010</td>
</tr>
<tr>
<td>move up</td>
<td>1010</td>
</tr>
<tr>
<td>MQTT client library</td>
<td>2196</td>
</tr>
<tr>
<td>MqttConnectWithCertBuffer</td>
<td>2197</td>
</tr>
<tr>
<td>MqttConnectWithCertFile</td>
<td>2201</td>
</tr>
<tr>
<td>MqttDisconnect</td>
<td>2205</td>
</tr>
<tr>
<td>MqttGetReceivedPacket</td>
<td>2207</td>
</tr>
<tr>
<td>MqttPing</td>
<td>2209</td>
</tr>
<tr>
<td>MqttPublish</td>
<td>2211</td>
</tr>
<tr>
<td>MqttSubscribe</td>
<td>2213</td>
</tr>
<tr>
<td>MUL</td>
<td>510</td>
</tr>
<tr>
<td>multicore</td>
<td>241, 838</td>
</tr>
<tr>
<td>show CPU load</td>
<td>343</td>
</tr>
<tr>
<td>trace</td>
<td>242, 342</td>
</tr>
<tr>
<td>using a task-local global variable list</td>
<td>177</td>
</tr>
<tr>
<td>multicore operator</td>
<td></td>
</tr>
<tr>
<td>__COMPARE_AND_SWAP</td>
<td>552</td>
</tr>
<tr>
<td>__XADD</td>
<td>553</td>
</tr>
<tr>
<td>TEST_AND_SET</td>
<td>554</td>
</tr>
<tr>
<td>multitouch</td>
<td></td>
</tr>
<tr>
<td>for operating a visualization</td>
<td>1126</td>
</tr>
<tr>
<td>implement event handling</td>
<td>1128</td>
</tr>
<tr>
<td>visualization</td>
<td>1616</td>
</tr>
<tr>
<td>MUX</td>
<td>521</td>
</tr>
</tbody>
</table>

**N**

<table>
<thead>
<tr>
<th>Term</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td></td>
</tr>
<tr>
<td>convention, static analysis</td>
<td>3611</td>
</tr>
<tr>
<td>nameprefix</td>
<td>3611</td>
</tr>
<tr>
<td>attribute</td>
<td>3611</td>
</tr>
<tr>
<td>nameprefix, attribute</td>
<td>3613</td>
</tr>
<tr>
<td>namespace</td>
<td>644</td>
</tr>
<tr>
<td>automatic</td>
<td>1082</td>
</tr>
</tbody>
</table>

enumeration .................................... 556
GVL ........................................... 555
library ....................................... 556
of variables .................................. 644
naming ........................................ 3611
attribute ..................................... 3611
naming convention ................................ 644
naming conventions ................................ 3608
code analysis .................................. 3608
static analysis ................................ 3608
naming conventions (2) ........................ 3610
naming conventions, extended .................. 3610
code analysis .................................. 3610
static analysis ................................ 3610
naming, attribute ................................ 3612
NE ............................................ 523
Negate ......................................... 978
negated coil ................................... 474
negated contact ................................ 473
negation, FBD/LD/IL ............................. 1000
network ........................................ 270
address ........................................ 272
addressing ...................................... 271
comment out ..................................... 993
FBD/LD/IL ...................................... 470
FBD/LD/IL, insert ................................ 992
FBD/LD/IL, insert below .......................... 993
jump ........................................... 1004
scan ........................................... 741
settings ........................................ 1053
topology ........................................ 270
NETWORK ....................................... 2142
network variable list (receiver) .............. 775
add ............................................ 775
network variable list (sender) ............... 774
add ............................................ 774
network variables ................................ 277
properties ...................................... 1052
NEW ............................................ 541
next message .................................... 882
no_assign_warning, pragma ...................... 621
no_assign, pragma ................................ 621
no_check, pragma ................................ 622
no_copy, pragma attribute ....................... 623
no_fast_online_change ........................... 616
offline help option ................................ 1081
offset position ................................ 2181
online login to application ..................... 923
logout from application ......................... 926
multiple download ................................ 929
pointer reference ................................ 876
Profiler ....................................... 3697
online change ................................... 928
active application ................................ 928
attribute ........................................ 616
compiler definition ................................ 616
memory reserve ................................... 206
selected application ............................. 928
online config mode ............................. 161, 915
online help option ................................ 1081
online mode status of devices ..................... 158
task monitoring .................................. 837
OPC UA server .................................. 1100
open directory ................................... 2173
open file ........................................ 2173, 2176
open in the IDE ................................... 920
open POU Profiler, online ......................... 3702
open project option ................................ 1083
Operand .......................................... 570
operating mode debug ............................. 937
locked ............................................ 937
operational ....................................... 937
operational, operating mode ....................... 937
operator .......................................... 505
binding strength .................................. 431
IL ................................................. 466
precedence ....................................... 431
options ............................................ 959
development status ................................ 102
device editor ..................................... 1081
import assistant .................................. 1039
monitoring ........................................ 1083
SVN ............................................... 3742, 3744

2020/12/10 3ADR010583, 1, en_US 3797
persistent variable configuration ............................. 714
device application ........................................... 714
persistent variable list ...................................... 770
pie .............................................................. 1260, 1664
visualization element ...................................... 1260, 1664
pin
reset ......................................................... 986
select ......................................................... 986
pin_presentation_order_inputs, pragma attribute .... 627
pin_presentation_order_outputs, pragma attribute .. 627
pingroup, pragma ........................................... 627
pins
remove ......................................................... 987
placeholder .................................................. 1008
with format definition in a character string, visualization .... 1546
PLC
read parameter file to configuration .................... 915
security ......................................................... 423
PLC load ....................................................... 342
PLC log ......................................................... 349
PLC parameter file ........................................... 163
PLC settings
device editor .................................................. 751
PLC shell ....................................................... 351
device editor .................................................. 753
redundancy editor ........................................... 817
PLC_PRG .......................................................... 49
plcload
DeviceTrace .................................................... 242
trace ........................................................... 335, 1032
PlcLoad
DeviceTrace .................................................... 343
PLCopenXML
export/import .................................................. 911
import ......................................................... 911
option .......................................................... 1084
PlcOperationControl .......................................... 352
PM5630 .......................................................... 2274
PM5650 .......................................................... 2274
PM5670 .......................................................... 2274
PM5675 .......................................................... 2274
PNIO device
IOxS ............................................................. 3369
po file ........................................................... 153
pointer
SUPER .......................................................... 501
THIS ............................................................. 502
Pointer
check function CheckPointer ............................... 809
POINTER TO ..................................................... 577
pointers ......................................................... 577
Go To Reference .................................................. 876
polygon ........................................................... 1248, 1652
code polygon, visualization element .................. 1248, 1652
polyline ........................................................... 1248, 1652
code polyline, visualization element ................ 1248, 1652
position
breakpoint ...................................................... 1045
pot file ........................................................... 153
potentiometer .................................................. 1435, 1839
visualization element .................................... 1435, 1839
POU .............................................................. 775
add ............................................................... 776
change type ..................................................... 1009
implicit checks ................................................ 796
monitor function call ....................................... 329
POUs view ....................................................... 882
POU locations .................................................. 713
POUs for implicit checks .................................... 262
POUs view ....................................................... 882
code check all pool objects ................................ 919
dataflow ......................................................... 536
operator ......................................................... 536
dataflow switch .............................................. 1457, 1861
code visualization element ............................. 1457, 1861
pragma .......................................................... 637
analysis:report-multiple-instance-calls .................. 3614
attribute ......................................................... 601
conditional ..................................................... 637
const_no_replaced ........................................... 606
const_replaced ................................................ 606
dataflow .......................................................... 607
define ............................................................. 637
dataflow enable_dynamic_creation ....................... 637
hasattribute ................................................... 637
hasconstantvalue ............................................. 637
hastype .......................................................... 637
if ................................................................. 637
<table>
<thead>
<tr>
<th>Term</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>message</td>
<td>600</td>
</tr>
<tr>
<td>no_copy</td>
<td>623</td>
</tr>
<tr>
<td>parameterstringof</td>
<td>1555</td>
</tr>
<tr>
<td>ProcessValue</td>
<td>637, 990</td>
</tr>
<tr>
<td>reflection</td>
<td>634</td>
</tr>
<tr>
<td>region</td>
<td>644</td>
</tr>
<tr>
<td>static analysis</td>
<td>3611</td>
</tr>
<tr>
<td>undefined</td>
<td>637</td>
</tr>
<tr>
<td>use</td>
<td>211</td>
</tr>
<tr>
<td>VAR_IN_OUT_AS_POINTER</td>
<td>1554</td>
</tr>
<tr>
<td>precedence, ST</td>
<td>431</td>
</tr>
<tr>
<td>prefix</td>
<td></td>
</tr>
<tr>
<td>convention, static analysis</td>
<td>3611</td>
</tr>
<tr>
<td>prepare value</td>
<td>1043</td>
</tr>
<tr>
<td>Present_Value</td>
<td>3460, 3461</td>
</tr>
<tr>
<td>previous message</td>
<td>882</td>
</tr>
<tr>
<td>print</td>
<td></td>
</tr>
<tr>
<td>input action</td>
<td>1589</td>
</tr>
<tr>
<td>page setup</td>
<td>1064</td>
</tr>
<tr>
<td>Python</td>
<td>398</td>
</tr>
<tr>
<td>Priority_Array</td>
<td>3460</td>
</tr>
<tr>
<td>PRIVATE</td>
<td>784</td>
</tr>
<tr>
<td>method</td>
<td>784</td>
</tr>
<tr>
<td>property</td>
<td>791</td>
</tr>
<tr>
<td>private key</td>
<td>422</td>
</tr>
<tr>
<td>process data</td>
<td></td>
</tr>
<tr>
<td>EtherCAT slave</td>
<td>3355</td>
</tr>
<tr>
<td>processing order in SFC</td>
<td>444</td>
</tr>
<tr>
<td>processing order, ST</td>
<td>431</td>
</tr>
<tr>
<td>processor load</td>
<td>242, 1032</td>
</tr>
<tr>
<td>trace</td>
<td>242</td>
</tr>
<tr>
<td>ProcessValue</td>
<td>637</td>
</tr>
<tr>
<td>products</td>
<td></td>
</tr>
<tr>
<td>activate</td>
<td>377</td>
</tr>
<tr>
<td>PROFIBUS DP</td>
<td></td>
</tr>
<tr>
<td>parameters</td>
<td>746</td>
</tr>
<tr>
<td>Profiler</td>
<td>3688</td>
</tr>
<tr>
<td>add</td>
<td>3695</td>
</tr>
<tr>
<td>copy table</td>
<td>3704</td>
</tr>
<tr>
<td>create snapshots</td>
<td>3694</td>
</tr>
<tr>
<td>display settings</td>
<td>3696</td>
</tr>
<tr>
<td>export</td>
<td>3704</td>
</tr>
<tr>
<td>list in clipboard</td>
<td>3704</td>
</tr>
<tr>
<td>measure code coverage</td>
<td>3693</td>
</tr>
<tr>
<td>measurement results</td>
<td>3697</td>
</tr>
<tr>
<td>object</td>
<td>3695</td>
</tr>
<tr>
<td>online</td>
<td>3697</td>
</tr>
<tr>
<td>online, backtraces</td>
<td>3697</td>
</tr>
<tr>
<td>online, call tree</td>
<td>3697</td>
</tr>
<tr>
<td>online, code coverage</td>
<td>3701</td>
</tr>
<tr>
<td>online, list</td>
<td>3697</td>
</tr>
<tr>
<td>online, overview</td>
<td>3697</td>
</tr>
<tr>
<td>online, sampling control</td>
<td>3697</td>
</tr>
<tr>
<td>open POU, online</td>
<td>3702</td>
</tr>
<tr>
<td>POU properties</td>
<td>3702</td>
</tr>
<tr>
<td>POU selection</td>
<td>3696</td>
</tr>
<tr>
<td>print table</td>
<td>3703</td>
</tr>
<tr>
<td>recording settings</td>
<td>3696</td>
</tr>
<tr>
<td>save list to file</td>
<td>3704</td>
</tr>
<tr>
<td>save snapshot</td>
<td>3697</td>
</tr>
<tr>
<td>settings</td>
<td>3695, 3696</td>
</tr>
<tr>
<td>snapshots</td>
<td>3701, 3703</td>
</tr>
<tr>
<td>watch list</td>
<td>3692</td>
</tr>
<tr>
<td>Profiler watch list</td>
<td></td>
</tr>
<tr>
<td>add POU</td>
<td>3706</td>
</tr>
<tr>
<td>Profiler Watch List</td>
<td>3692</td>
</tr>
<tr>
<td>profiling</td>
<td>3688</td>
</tr>
<tr>
<td>PROFINET IO</td>
<td></td>
</tr>
<tr>
<td>bus cycle</td>
<td>3367</td>
</tr>
<tr>
<td>device editor</td>
<td>3363</td>
</tr>
<tr>
<td>I/O mapping</td>
<td>3363</td>
</tr>
<tr>
<td>IEC objects</td>
<td>3363</td>
</tr>
<tr>
<td>information</td>
<td>746, 3363</td>
</tr>
<tr>
<td>parameters</td>
<td>3363</td>
</tr>
<tr>
<td>status</td>
<td>3363</td>
</tr>
<tr>
<td>troubleshooting</td>
<td>3378</td>
</tr>
<tr>
<td>PROFINET IO controller</td>
<td></td>
</tr>
<tr>
<td>general</td>
<td>3364</td>
</tr>
<tr>
<td>overview</td>
<td>3365</td>
</tr>
<tr>
<td>PROFINET IO device</td>
<td></td>
</tr>
<tr>
<td>general</td>
<td>3367</td>
</tr>
<tr>
<td>options</td>
<td>3369, 3370</td>
</tr>
<tr>
<td>PROFINET IO module</td>
<td></td>
</tr>
<tr>
<td>general</td>
<td>3370</td>
</tr>
<tr>
<td>options</td>
<td>3371</td>
</tr>
<tr>
<td>PROFINET IO slave device</td>
<td></td>
</tr>
<tr>
<td>general</td>
<td>3372, 3374</td>
</tr>
<tr>
<td>Linux</td>
<td>3373</td>
</tr>
<tr>
<td>VxWorks</td>
<td>3373</td>
</tr>
<tr>
<td>Windows</td>
<td>3373</td>
</tr>
</tbody>
</table>
Windows CE ........................................ 3374
program ........................................ 777
execute on client, input action ............ 1589
execute on controller, input action ....... 1589
property ........................................ 791
Program ........................................... 3462
PROGRAM ......................................... 777
programming
reference, visualization .................... 1223
progress bar .................................... 1382, 1791
visualization element ...................... 1382, 1791
project ............................................ 34, 124, 3228
access protection ............................ 139
add folder ...................................... 897
commit accepted changes .................... 910
compare ......................................... 137, 905, 3237
comparison ..................................... 137, 3237
create ............................................. 124
create property with key .................... 132
document ........................................ 905
donble ............................................ 144
encryption ...................................... 139, 144
encryption, instructions .................... 149
export ............................................. 135
export/import ................................... 135
file information ............................... 811
filing ............................................. 142
functions for accessing properties .......... 132
include with source code management ...... 152
information ...................................... 811
install in the library repository .......... 856
key for meta-information .................... 812
last used ......................................... 861
localization .................................... 903
login data ....................................... 147
manage localizations ....................... 904
meta-information ................................ 811
new ................................................. 852
object statistics ................................ 813
open ............................................... 853
open by command line ....................... 367
open V2.3 ......................................... 128
password ........................................ 143
password protection ......................... 139
POUs for keys .................................. 812
project settings ................................ 134
protection ........................................ 139
query information ............................ 132
released ......................................... 143
rights management ............................ 141
save .............................................. 150
save as .......................................... 150
save as, compiled library .................. 856
save as, command .............................. 855
saving in project archive .................... 152
security .......................................... 139
template ......................................... 852
toggle localization ............................ 905
transfer .......................................... 135, 136
user management .............................. 145
VisuSymbolLibrary key ...................... 812
write protection .............................. 139, 142
Project
close ............................................. 854
project archive ............................... 152
extract by command line .................... 367
project compare
configuration .................................... 906
detail ............................................. 908
differences ..................................... 906
project documentation print ............... 905
project environment
symbol library in visualization ............ 1073
visualization profile ......................... 1071, 1601
visualization style ........................... 1072, 1601
visualization symbol library .............. 1601
project localization .......................... 153
create template ................................ 903
localization template ....................... 153
manage .......................................... 904
toggle ............................................ 905
project setting
command ........................................ 902
make ............................................. 134
user management .............................. 145
users and groups .............................. 1060
visualization ................................. 1068, 1602
project settings .............................. 810
object ............................................. 810
rules, static analysis ....................... 3607
SFC ................................ 1059
SVN ............................... 3744
Prop_Pulse_Value_Source ................. 3461
properties
access control . ........................ 1050
BACnet ............................. 3447
bitmap .............................. 1051
boot application ........................ 1047
build ................................ 1048
build, C-integration .................... 1049
cam ................................ 1055
common ............................ 1047
device ................................ 1058
encryption ........................... 1047
external file .......................... 1050
image pool .......................... 1057
link to file ........................... 1055
monitoring ........................... 1058
network settings ...................... 1053
network variables ..................... 1052
POU in Profiler ....................... 3702
SFC ................................ 1055
task configuration ...................... 835
text list .............................. 1057
Properties
BACnet ................................ 3460
of an object .......................... 1046
property
CFC execution order .................... 1054
monitor ....................... 326
object ................................ 791
object-oriented programming .......... 791
SFC, do not display embedded objects .. 976
PROPERTY ............................. 791
object ................................ 791
PROTECTED ............................ 784
method .............................. 784
property .............................. 791
protection
data security ........................ 300
project ......................... 139
Protocols
Modbus RTU .......................... 3087
Modbus TCP/IP ........................ 3182
proxy
access data .......................... 1084
server option ........................ 1084
server, setting ........................ 1084
PUBLIC ................................. 784
method ................................ 784
property .............................. 791
public key ............................ 422
push switch ........................... 1457, 1861
visualization element .............. 1457, 1861
push switch LED ....................... 1457, 1861
visualization element .............. 1457, 1861
Python
.NET API ............................. 397
basic syntax .......................... 398
Boolean types ........................ 400
class ................................. 408
cPython .............................. 420
create POU ........................... 415
data types ............................ 400
dictionary ............................ 404
edit POU .............................. 415
floating-point types ........................ 400
function ............................. 408
IF/ELSE .............................. 408
in CODESYS .......................... 397
integer types ......................... 400
interaction with user .................... 416
IronPython ........................ 396, 420
list ..................... 403
loop .................................. 406
method .............................. 408
module .............................. 409
print ................................. 398
project device tree ...................... 411
Python2 .............................. 420
Python3 .............................. 420
read variables ........................ 412
standard library ........................ 409
strings ............................. 402, 420
tuple .................................. 403
variables ............................. 400

Q
memory range prefix ........................................... 566
qualified_only, pragma .................................. 634
qualifiers for SFC actions .......................... 445
QUERYINTERFACE .............................................. 544
QUERYPOINTER .................................................. 545

R
R= ................................................................. 434
reset assignment ........................................ 434
radio button .................................................. 1390, 1794
visualization element .................................. 1390, 1794
read directory properties .......................... 2174
read file .............................................................. 2178
read file properties .................................. 2174
READ_ONLY .............................................. 652
READ_WRITE ............................................. 652
ReadProperty ........................................... 3455
ReadPropertyMultiple .................................. 3456
ReadRange ...................................................... 3456
REAL .............................................................. 570
constant ...................................................... 562
literal ............................................................ 562
real time clock reading .......................... 2160
real time clock setting .......................... 2160
REAL_TO ...................................................... 529
Real-Time Clock ......................................... 3115
Realization with centralized PLC based Motion Control ................................................. 2046
rearrange list .................................................. 1011
recent projects ............................................. 861
recipe ............................................................... 331, 823
add .............................................................. 1015
create ............................................................. 332
create, input action .................................. 1589
delete, input action .................................. 1589
insert variable ........................................ 1015
load .............................................................. 1016
load and write ........................................ 1017
load from device ...................................... 1019
load from file ........................................... 333
load, input action .................................. 1589
read .............................................................. 1017
read and save ........................................ 1018
read, input action ........................................... 1589
remanent variable ....................................... 255
remove .......................................................... 1016
remove variables ........................................ 1018
save .............................................................. 1016
save, input action ........................................... 1589
visualization ............................................... 1176
write .............................................................. 1017
write, input action ........................................... 1589

recipe definition ........................................ 823
recipe file
  download ...................................................... 333
recipe management ...................................... 331
memory usage .................................................. 333
recipe manager ................................................ 820
recording, see data recording .......................... 1093
rectangle ....................................................... 1224, 1628
visualization element .................................. 1224, 1628
redundancy ..................................................... 362
applications, editor ........................................ 817
connection settings, editor ................................ 817
files, editor ..................................................... 817
general settings, editor .................................. 818
log, editor ......................................................... 817
PLC shell, editor ............................................. 817
registered areas, editor ................................... 819
status, editor ..................................................... 816

REF ............................................................. 435, 579
refactoring ..................................................... 236
add variable ..................................................... 878
add/remove variable ..................................... 237
declaration order of variables .......................... 238
option ............................................................. 1085
remove variable ............................................... 879
rename ............................................................. 876
rename variable ............................................. 236
reorder variables ........................................... 880
update referenced pins .................................. 877
reference ......................................................... 579
__ISVALIDREF ............................................... 579
data type ......................................................... 579
test operator .................................................... 579
REFERENCE .................................................... 579
REFERENCE TO ................................................ 579
reflection, pragma attribute .......................... 634
refresh
  structured variables ............................................ 1019
ReinitializeDevice ............................................. 3459
released .......................................................... 143
remanent ........................................................... 498
  recipe .................................................................. 255
    remanent variable of the Persistence Manager ............ 255
  variable ................................................................ 498
remote data
table data .......................................................... 293
remove
  IL line ................................................................... 999
  reset ...................................................................... 979
  set ........................................................................ 979
  unused parameters, FBD/LD .................................. 1002
remove force list ..................................................... 944
remove unused FB call parameters ................................ 1002
RemoveListElement .................................................... 3455
rename
table data .......................................................... 293
refactoring ............................................................ 876
rename directory ....................................................... 2175
rename file ............................................................ 2179
repair
  box ................................................................... 1002
  REPEAT ............................................................... 439
replace
table data .......................................................... 293
command ............................................................... 864
repository ............................................................. 951, 3711
  browse SVN repository ......................................... 3716
  library .................................................................. 951
  SVN ..................................................................... 3711
  visualization element .......................................... 1578
reserve memory
table data .......................................................... 293
online change ........................................................ 894
reset
table data .......................................................... 293
  application .......................................................... 318
  application (reset cold) ........................................ 931
  application (reset origin) ....................................... 932
  application (reset warm) ........................................ 931
  assignment ST ......................................................... 434
  cold ..................................................................... 931
device to origin ....................................................... 933
  origin ................................................................. 932
  origin device ......................................................... 933
output ........................................................................ 979
remove ................................................................. 979
SVN ....................................................................... 3733
warm ................................................................. 931
reset coil ............................................................... 474
  insert .................................................................... 996
reset cold all applications
  device application .................................................. 1036
  origin .................................................................... 932
  application .......................................................... 932
reset warm all applications
  device application .................................................. 1037
Resolution ............................................................ 3461
restore ................................................................. 353
restore values from recipe ........................................ 1011
RET ........................................................................ 466
RETAI N ................................................................. 500
  remanent variable .................................................. 254
  variable ............................................................... 500
RETC ................................................................. 466
RETCN ................................................................. 466
return
  CFC ............................................................... 487
  FBD/LD/IL ............................................................ 472
insert, FBD/LD/IL .................................................... 995
RETURN .................................................................. 440
revision
table data .......................................................... 293
copy to branch/tag .................................................. 3740
select in SVN .......................................................... 3745
RGB ................................................................. 1152
color space ........................................................... 1152
RGBA ................................................................. 1152
color space extended with alpha channel ................. 1152
rights management .................................................. 140
project ................................................................. 141
rising edge detection ............................................... 1000
rocker switch ........................................................ 1457, 1861
  visualization element .......................................... 1457, 1861
ROL ................................................................. 517
root ............................................................... 2185
root example ........................................................ 2185
ROR ................................................................. 518
rotary switch
  visualization element .......................................... 1461, 1865
INDEX

3ADR010583, 1

round rectangle ........................................ 1224, 1628
visualization element ................................... 1224, 1628
routing .................................................... 271
run stepping ............................................. 313
to cursor .................................................. 942
using step out ............................................ 941
run static analysis ...................................... 3605
runtime system security .................................. 423
runtime system files generate .......................... 917
runtime system service disable ........................ 352
RX signals 1939 ECU ...................................... 3345

S
S= .......................................................... 433
set assignment ............................................. 433
SA0001 ..................................................... 3617
SA0002 ..................................................... 3617
SA0003 ..................................................... 3618
SA0004 ..................................................... 3618
SA0005 ..................................................... 3646
SA0006 ..................................................... 3619
SA0007 ..................................................... 3620
SA0008 ..................................................... 3620
SA0009 ..................................................... 3621
SA0010 ..................................................... 3622
SA0011 ..................................................... 3622
SA0012 ..................................................... 3623
SA0013 ..................................................... 3623
SA0014 ..................................................... 3624
SA0015 ..................................................... 3625
SA0016 ..................................................... 3625
SA0017 ..................................................... 3626
SA0018 ..................................................... 3626
SA0019 ..................................................... 3641
SA0020 ..................................................... 3627
SA0021 ..................................................... 3628
SA0022 ..................................................... 3628
SA0023 ..................................................... 3629
SA0024 ..................................................... 3629
SA0025 ..................................................... 3630
SA0026 ..................................................... 3630
SA0027 ..................................................... 3631
SA0028 ..................................................... 3631
SA0029 ..................................................... 3632
SA0031 ..................................................... 3633
SA0032 ..................................................... 3633
SA0033 ..................................................... 3633
SA0034 ..................................................... 3635
SA0035 ..................................................... 3634
SA0036 ..................................................... 3634
SA0037 ..................................................... 3636
SA0038 ..................................................... 3636
SA0040 ..................................................... 3637, 3679
SA0041 ..................................................... 3638
SA0042 ..................................................... 3639
SA0043 ..................................................... 3639
SA0044 ..................................................... 3640
SA0046 ..................................................... 3680
SA0047 ..................................................... 3647
SA0048 ..................................................... 3647
SA0051 ..................................................... 3648
SA0052 ..................................................... 3649
SA0053 ..................................................... 3649
SA0054 ..................................................... 3650
SA0055 ..................................................... 3651
SA0056 ..................................................... 3651
SA0057 ..................................................... 3652
SA0058 ..................................................... 3652
SA0059 ..................................................... 3654
SA0060 ..................................................... 3654
SA0061 ..................................................... 3654
SA0062 ..................................................... 3655
SA0063 ..................................................... 3655
SA0064 ..................................................... 3656
SA0065 ..................................................... 3656
SA0066 ..................................................... 3657
SA0072 ..................................................... 3659
SA0073 ..................................................... 3659
SA0075 ..................................................... 3661
SA0076 ..................................................... 3662
SA0077 ..................................................... 3663
SA0078 ..................................................... 3663
SA0080 ..................................................... 3659
SA0081 ..................................................... 3660
SA0090 ..................................................... 3664

2020/12/10
Index

SA0095 .................................. 3664
SA0100 .................................. 3665
SA0101 .................................. 3666
SA0102 .................................. 3666
SA0103 .................................. 3667
SA0105 .................................. 3668
SA0106 .................................. 3669
SA0107 .................................. 3670
SA0111 .................................. 3672
SA0112 .................................. 3672
SA0113 .................................. 3672
SA0114 .................................. 3673
SA0115 .................................. 3673
SA0117 .................................. 3673
SA0118 .................................. 3678
SA0119 .................................. 3674
SA0120 .................................. 3674
SA0121 .................................. 3675
SA0122 .................................. 3676
SA0123 .................................. 3676
SA0124 .................................. 3678
SA0125 .................................. 3678
SA0130 .................................. 3642
SA0132 .................................. 3644
SA0133 .................................. 3644
SA0134 .................................. 3645
SA0140 .................................. 3679
SA0145 .................................. 3681
SA0147 .................................. 3676
SA0148 .................................. 3677
SA0150 .................................. 3682
SA0160 .................................. 3682
SA0161 .................................. 3683
SA0162 .................................. 3684
SA0163 .................................. 3685
SA0164 .................................. 3686
SA0165 .................................. 3686
SA0166 .................................. 3687
SA0167 .................................. 3687
sample
  show in the trace editor .................. 335
samples
  save in trace file ...................... 336
  sample project .......................... 108
  samples
    save in trace file .................... 336
    save current values to recipe .......... 1011
    save, project .......................... 1083
    saving project ........................ 142
    scalar product .......................... 590
    scan devices .......................... 160
    for devices ........................... 899, 3345, 3375
    SCE in LD ............................ 475
    scope ................................. 489
    script
      execute ............................. 959
    Script
      enable tracing ........................ 958
      script file
        execute by command line ............. 367
    ScriptEngine
      Python ............................... 397
    scripting
      call via custom toolbar ............... 392
      call via menu command ............... 390
      command line ........................ 391
      execute ............................. 958
      execute script file .................. 958
      IronPython .......................... 396
      Python ................................ 397
      Python .NET API ....................... 397
    Scripting
      enable script tracing ................ 958
      scripting language ................... 388
      scrollbar ............................ 1356, 1760
      visualization element ................ 1356, 1760
      SD memory card ....................... 3028, 3092, 3513

3806 3ADR010583, 1, en_US 2020/12/10
sdcard ................................ 3513
sdcard.ini .............................. 3513
SDO
  CANopen remote device ................ 3338
search ................................. 236, 865, 866
  object ............................... 881
search order
  Identifiers ............................ 650
  variable name ......................... 650
security ............................... 421
  add device user ........................ 934
  certificate ............................ 422
  certificate via PLC shell ............... 425
  certificates .......................... 139, 891
client ................................. 425
  communication with controller ........... 3597
data security .......................... 300
development system ..................... 423
device .................................. 296
disable user management .................. 426
encrypt boot application ................. 3598
encrypted communication ................. 741
encryption, signing, certificates ......... 139
general information ..................... 421
password device user .................... 935
  project encryption .................... 139
project settings ........................ 1064
remove device user ...................... 935
runtime system / PLC .................... 423
  Security Agent ........................ 3597
unencrypted communication ............... 427
Security Agent .......................... 3597
certificate ............................. 425
security functions
  certificate ............................ 422
  development system .................... 423
general information ..................... 421
runtime system / PLC .................... 423
security screen .......................... 891
SEL ................................... 519
select matching bracket .................. 868
select none
  of the selected visualization elements . . 1581
selection
  alarm class ........................... 1604
alarm group ........................... 1605
selector
  CFC ................................ 488
  semi-transparency
  visualization ......................... 1616
sequence editor
  option ............................... 1086
  setting ................................ 1086
sercos
  generate xml .......................... 913
Service hook
  BACnet .......................... 3454, 3459
  Implementation example ............... 3462
set
  access method, interface ............... 788
  output ................................ 979
  remove ................................ 979
Set
  accessor method ......................... 791
  assignment ST .......................... 433
set coil ............................... 474
insert .................................. 996
set output connection, FBD/LD ............ 1000
set/reset, FBD/LD/IL ..................... 1000
settings
  code check ............................ 3606
  Profiler ............................. 3695, 3696
  static analysis ........................ 3606
SFC ................................... 202
  action ................................ 454
  action qualifiers ....................... 445
  branch ................................ 457
  build .................................. 1055
  code generation ....................... 1055, 1060
  copy implementation .................... 970
  copy reference ........................ 970
  do not display embedded objects ........ 976
  duplication mode ....................... 970
  element properties ..................... 459
  implicit variables ...................... 446
  init step .............................. 967
  jump .................................. 458
  library ................................ 1060
  macro .................................. 458
  processing order ........................ 444
<table>
<thead>
<tr>
<th>Index</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>programming</td>
<td>202</td>
</tr>
<tr>
<td>project settings</td>
<td>1059</td>
</tr>
<tr>
<td>properties</td>
<td>1055</td>
</tr>
<tr>
<td>step</td>
<td>452</td>
</tr>
<tr>
<td>step time</td>
<td>446</td>
</tr>
<tr>
<td>tip mode</td>
<td>447</td>
</tr>
<tr>
<td>transition</td>
<td>452</td>
</tr>
<tr>
<td>SFC editor</td>
<td>442</td>
</tr>
<tr>
<td>character set</td>
<td>1074</td>
</tr>
<tr>
<td>layout</td>
<td>1074</td>
</tr>
<tr>
<td>online mode</td>
<td>443</td>
</tr>
<tr>
<td>online, step time</td>
<td>1074</td>
</tr>
<tr>
<td>options</td>
<td>1074</td>
</tr>
<tr>
<td>properties, visibility</td>
<td>1074</td>
</tr>
<tr>
<td>settings</td>
<td>1074</td>
</tr>
<tr>
<td>step actions, options</td>
<td>1074</td>
</tr>
<tr>
<td>toolbar</td>
<td>429</td>
</tr>
<tr>
<td>SFC flag</td>
<td>447</td>
</tr>
<tr>
<td>shadowing</td>
<td>650</td>
</tr>
<tr>
<td>shadowing rules</td>
<td>650</td>
</tr>
<tr>
<td>SHL</td>
<td>516</td>
</tr>
<tr>
<td>short form feature</td>
<td>210</td>
</tr>
<tr>
<td>short-circuit evaluation</td>
<td>475</td>
</tr>
<tr>
<td>show source position</td>
<td>888</td>
</tr>
<tr>
<td>show windows</td>
<td>107</td>
</tr>
<tr>
<td>show/hide implementation view</td>
<td>964</td>
</tr>
<tr>
<td>SHR</td>
<td>516</td>
</tr>
<tr>
<td>signature</td>
<td></td>
</tr>
<tr>
<td>encryption</td>
<td>421</td>
</tr>
<tr>
<td>signing</td>
<td></td>
</tr>
<tr>
<td>boot application</td>
<td>242</td>
</tr>
<tr>
<td>certificate</td>
<td>139</td>
</tr>
<tr>
<td>library project</td>
<td>814</td>
</tr>
<tr>
<td>with certificate, instructions</td>
<td>149</td>
</tr>
<tr>
<td>simulation</td>
<td>309</td>
</tr>
<tr>
<td>command</td>
<td>937</td>
</tr>
<tr>
<td>for testing</td>
<td>309</td>
</tr>
<tr>
<td>SIN</td>
<td>536</td>
</tr>
<tr>
<td>single cycle</td>
<td>939</td>
</tr>
<tr>
<td>SINT</td>
<td>570</td>
</tr>
<tr>
<td>SIZEOF</td>
<td>513</td>
</tr>
<tr>
<td>slider</td>
<td>1365</td>
</tr>
<tr>
<td>visualization element</td>
<td>1365</td>
</tr>
<tr>
<td>smart tag</td>
<td>211</td>
</tr>
<tr>
<td>SmartCoding</td>
<td>208</td>
</tr>
<tr>
<td>SmartCoding, options</td>
<td>1082</td>
</tr>
<tr>
<td>snapshot</td>
<td></td>
</tr>
<tr>
<td>snapshot</td>
<td></td>
</tr>
<tr>
<td>Profiler</td>
<td>3701</td>
</tr>
<tr>
<td>snapshots</td>
<td></td>
</tr>
<tr>
<td>create</td>
<td>3694</td>
</tr>
<tr>
<td>delete</td>
<td>3703</td>
</tr>
<tr>
<td>edit</td>
<td>3703</td>
</tr>
<tr>
<td>Profiler</td>
<td>3701</td>
</tr>
<tr>
<td>source code</td>
<td>307</td>
</tr>
<tr>
<td>download from controller</td>
<td>859</td>
</tr>
<tr>
<td>download to controller</td>
<td>860</td>
</tr>
<tr>
<td>download, project setting</td>
<td>1063</td>
</tr>
<tr>
<td>management</td>
<td>152</td>
</tr>
<tr>
<td>write, to connected device</td>
<td>929</td>
</tr>
<tr>
<td>space</td>
<td></td>
</tr>
<tr>
<td>in text editor</td>
<td>1087</td>
</tr>
<tr>
<td>SpinControl</td>
<td>1370</td>
</tr>
<tr>
<td>visualization element</td>
<td>1370</td>
</tr>
<tr>
<td>SORT</td>
<td>534</td>
</tr>
<tr>
<td>ST</td>
<td>200</td>
</tr>
<tr>
<td>assignment</td>
<td>432</td>
</tr>
<tr>
<td>expression</td>
<td>431</td>
</tr>
<tr>
<td>extended</td>
<td>201</td>
</tr>
<tr>
<td>programming in</td>
<td>201</td>
</tr>
<tr>
<td>R=</td>
<td>434</td>
</tr>
<tr>
<td>reset assignment</td>
<td>434</td>
</tr>
<tr>
<td>S=</td>
<td>433</td>
</tr>
<tr>
<td>set assignment</td>
<td>433</td>
</tr>
<tr>
<td>ST code</td>
<td></td>
</tr>
<tr>
<td>execute, input action</td>
<td>1596</td>
</tr>
<tr>
<td>ST code in FBD, LD</td>
<td>473</td>
</tr>
<tr>
<td>ST editor</td>
<td>430</td>
</tr>
<tr>
<td>find</td>
<td>430</td>
</tr>
<tr>
<td>online operation</td>
<td>431</td>
</tr>
<tr>
<td>option</td>
<td>1086</td>
</tr>
<tr>
<td>stack checking of recursive methods</td>
<td>608</td>
</tr>
<tr>
<td>standard commands</td>
<td>862</td>
</tr>
<tr>
<td>standard data types</td>
<td>569</td>
</tr>
<tr>
<td>standard library</td>
<td></td>
</tr>
<tr>
<td>Python</td>
<td>409</td>
</tr>
<tr>
<td>standard metrics</td>
<td>3605</td>
</tr>
<tr>
<td>standard project</td>
<td>124</td>
</tr>
<tr>
<td>start all applications</td>
<td></td>
</tr>
<tr>
<td>device application</td>
<td>1037</td>
</tr>
<tr>
<td>start page</td>
<td>895</td>
</tr>
</tbody>
</table>
start parameters
  EtherCAT module ................................ 3358
  EtherCAT slave .................................. 3356

state
  device state .................................... 3519

statement ........................................ 436
  IF ............................................ 436
  set next ...................................... 942
  show next ..................................... 942

static
  code analysis ................................... 230

static analysis
  getting started .................................. 3601
  pragmas ........................................ 3611
  run ............................................ 3605
  Static Analysis Light ............................ 230
  project settings ................................ 1066

static variable .................................. 495

status
  device editor .................................... 768
  device state .................................... 3519
  EtherCAT ........................................ 3347
  KNX ............................................. 3436
  PROFINET IO ..................................... 3363
  redundancy configuration .......................... 816
  SFC actions ..................................... 446
  SFC steps ....................................... 446
  Status_Flags .................................... 3460

step ................................................ 452
  add exit action, SFC ............................ 970
  insert .......................................... 968
  insert after .................................... 968
  SFC, add entry action ............................ 970
  SFC, duplication mode ............................. 970

step action
  SFC ............................................. 454
  step into ...................................... 941
  run ............................................. 941
  step over ...................................... 941
  run ............................................. 941
  step status ..................................... 446

step-transition
  add ............................................. 969
  insert after .................................... 969
  STN ............................................. 466

stop all applications
  device application ............................... 1037

storage
  alarm ........................................... 721

strict
  pragma for enumeration .......................... 596

string
  Python ......................................... 402
  STRING ......................................... 571
    Data type ..................................... 571
  string constants ................................ 563
  STRING_TO ...................................... 532
  STRUCT .......................................... 594
    structure ..................................... 594
    access ........................................ 564
    data type ..................................... 594
  Structured-View .................................. 3462

style color
  select, visualization element ................. 1116
  visualization .................................. 1115

style property
  assign to a visualization element ............ 1216
  localize ....................................... 1221
  SUB ............................................. 510
  subdirectory ................................... 2173
  subrange types .................................. 599
  SubscribeCOV .................................... 3455
  SubscribeCOVProperty ............................. 3457

subsequent, pragma ............................... 635

Subversion
  source management ................................ 152
  SVN ............................................. 3710
    _VERSION_INFO ................................ 3749
      checkout .................................. 3721
      info ........................................ 3729
      overlay icon ................................ 3714
      project settings ................................ 3744
      repository ................................... 3711
      repository browser ............................ 3716
      version control ................................ 3711
      version Info ................................ 3749

switch
  visualization element ........................... 1457, 1461, 1861, 1865
symbol
  access rights .................................. 275
  access to controller .................................. 766
  overlay ........................................... 3714
symbol configuration .................................. 275, 824
  access rights .................................. 825
  add ............................................. 824
  comments and attributes .............................. 829
  data layout ....................................... 824
  editor ............................................. 825
  OPC UA ............................................. 824
  symbol set ........................................ 275
  task synchronization .................................. 830
symbol file ............................................. 824
symbol library
  project environment of visualization ................. 1073
  update .......................................... 1073, 1601
  visualization project environment ....................... 1601
symbol rights
  device editor ....................................... 766
  symbol access ..................................... 275
symbol set
  symbol configuration .................................. 275
symbol, pragma .......................................... 635
sync unit assignment
  EtherCAT master .................................. 3349
synchronize
  file by application download .......................... 749
  syntax check ....................................... 919
system diagnosis ..................................... 3526, 3531
system event ......................................... 835
  function call ...................................... 835
system time
  printed in a visualization .................................. 1549
system variable ...................................... 352
  operation control .................................. 352
SYSTEM.VAR_INFO ...................................... 548
data structure ........................................ 548

T
T
  constant ......................................... 558
  literal .......................................... 558
TA521 ............................................. 3030, 3088
TA523 ............................................. 3035
TA524 ............................................. 3034, 3094
TA525 ............................................. 3037
TA526 ............................................. 3035, 3038, 3098
TA535 ............................................. 3039
tab control ........................................... 1316, 1720
  reference visualizations .................................. 1185
  visualization element .................................. 1316, 1720
tab group
  new horizontal ....................................... 962
  new vertical ....................................... 962
table ............................................. 1337, 1741
  display structured variables .......................... 1155, 1960
  visualization element .................................. 1337, 1741
  visualizing data arrays .................................. 1155
table of contents ....................................... 966
tag
  configuring in visualization tools ......................... 1585
TAN .................................................. 537
TargetVisu
  object ............................................. 1623
task ............................................... 839
  check ............................................. 322
  configuration ...................................... 839
  cycle times ....................................... 349
  monitor ........................................... 349
  monitoring ........................................ 837
  processing order .................................... 239
  statistics ......................................... 349
task configuration ..................................... 239
  basic settings ...................................... 835
  basics ............................................ 239
  create ............................................ 240
  device application .................................. 358
  object ............................................. 834
  properties ........................................ 835
task deployment
  check ............................................. 322
  device editor ..................................... 767
task groups .......................................... 241, 838
task monitoring, online .................................. 837
task-local
  declare GVL ....................................... 177
  GVL ................................................. 769
TB .................................................. 2244
TB5600 ............................................. 2265
display text dynamically ........................................... 220
download .............................................................. 1057
DUT ................................................................. 1024
export ................................................................. 214
export as unicode text ............................................ 1021
export everything as text ......................................... 1020
export/import ....................................................... 1021
for dynamic application .......................................... 220
for input assistance ................................................ 215
import file ........................................................... 215
insert text ........................................................... 1021
names and directories for visualization ......................... 1600
object ................................................................. 823
properties ........................................................... 1057
remove language .................................................... 1022
remove unused entries ............................................ 1023
rename language .................................................... 1022
update ID ............................................................. 1023
visualization ......................................................... 1143
text list support
add ................................................................. 1024
DUT ................................................................. 1024
remove ............................................................... 1024
THEN ................................................................. 436
THIS ................................................................. 502
time ................................................................. 558
custom ............................................................... 558
duration .............................................................. 558
literal ................................................................. 558
TIME ................................................................. 572
custom ............................................................... 572
keyword ............................................................... 572
literal ................................................................. 572
time data types ...................................................... 572
TIME function ....................................................... 568
time of day .......................................................... 561
custom ............................................................... 561
literal ................................................................. 561
TIME OF DAY ....................................................... 2163
time picker .......................................................... 1525, 1924
visualization element .............................................. 1525, 1924
TIME OF DAY ....................................................... 572
keyword ............................................................... 572
TIME_OF_DAY ..................................................... 530
TIME_OF_DAY_TO .................................................. 530
TIME_OF_DAY# ....................................................... 561
transfer parameters
update ........................................ 1189
transition .................................... 452, 796
insert ......................................... 969
insert after .................................. 968
SFC, do not display embedded objects .... 976
trend
basis, visualization .......................... 1165
configure display settings, command ..... 1570, 1576
configure recording, visualization ...... 846
configure variables, visualization ...... 847
configure, visualization .................. 1577
visualization element ...................... 1471, 1875
Trend
insert elements for control, command ... 1577
trend configuration
add parameters .............................. 348
add variable .................................. 347
assign tasks .................................. 346
configure additional buffering .......... 348
configure data buffering on RTS ...... 348
delete variable ................................ 347
start conditional ........................... 347
trend recording .............................. 344, 846
configure ................................... 346
getting started .............................. 345
set additional buffer ....................... 1097
storage configuration ...................... 1097
trend recording task ....................... 849
trend recording manager ................... 845
trend visualization
edit ........................................... 1168
getting started .............................. 1165
programming .................................. 1168
sine-shaped curve of IEC variable, example 1166
Trend-Log ..................................... 3467
Trend-Log-Multiple ......................... 3467
trigger
activate in trace configuration ......... 340
reset trace configuration .................. 1032
trigger event .................................. 2126
triggering callbacks ....................... 2134
troubleshooting
PROFINET .................................... 3378
TRUE .......................................... 558
TRUNC ........................................ 533
TRUNC_INT ................................ 534
TRY ........................................... 546
TU ............................................. 2247
TU507 .......................................... 2307
TU508 .......................................... 2307
TU515 .......................................... 2311
TU516 .......................................... 2311
TU517 .......................................... 2317
TU518 .......................................... 2317
TU531 .......................................... 2320
TU532 .......................................... 2320
TU541 .......................................... 2311
TU542 .......................................... 2311
tuple
Python ........................................ 403
tutorial ....................................... 108
Tutorial
Visualisierung ................................ 1950
TX signals
1939 ECU ..................................... 3343
TYPE ........................................ 595
typed literals .................................. 564
U
UDINT ......................................... 570
UINT .......................................... 570
ULINT ......................................... 570
uncomment ................................... 869
UnconfirmedCOVNotification ............... 3459
UnconfirmedEventNotification .............. 3459
undefine, pragma ........................... 637
underflow data type ....................... 505
unforce ........................................ 944
unforce values ................................ 944
unicode
text list ...................................... 1021
Unicode
character string in visualization ........ 1613
uninstall
device ........................................ 957
library ........................................ 951
UNION ......................................... 572
unit conversion ............................. 246
apply ......................................... 247
Visualization ............................ 1106
Assignment of the visualizations to the display variants ................... 1617
visualization editor
configuration .................................. 1599
visualization element .................. 1224
add via command ..................... 1581
animation .................................. 1150
configuration with interface property .................. 1602
configuration with interface property activated .................. 1069, 1062
configure size and position ........... 1113
configuring an offset .................. 1150
configuring while rotating ............ 1150
design in color .......................... 1115
elementlist .................................. 1559
enter static text .......................... 217
in toolbox view ..................... 1609
many of the same type .................. 1156
multiple template .................. 1156
repository .................................. 1578
selecting in tools .................... 1112
user input .................................. 1125
viewing assignment in tools ........... 1585
visualization manager .................. 1613
activate multitouch .................. 1616
settings .................................. 1613
user management .................... 1615
visualization styles editor .......... 1614
visualization profile .................. 1601
project setting .................. 1069, 1603
repository .................................. 1578
version .................................. 1071, 1601
visualization style .................. 1216
copy .................................. 1947
create .................................. 1220, 1946
determine the appearance ........... 1216
edit .................................. 1217
install .................................. 1222
manage in repository ............. 1221
manage repository .............. 1222
option .................................. 1597
preview in the visualization manager .......... 1614
preview of installed styles ........... 1580
repository and contents .......... 1580
selection in the visualization manager .......... 1614
switch .................................. 1217
uninstall .................................. 1222
update version ..................... 1217
version .................................. 1072, 1601
visualization style editor ........... 1947
open .................................. 1219
open from development system .......... 1219
visualization user management .......... 1139
VisualizationManager .............. 64
VUM_ChangePassword
user management dialog, visualization .......... 1615
VUM_Login
user management dialog, visualization .......... 1615
VUM_UserManagement
user management dialog, visualization .......... 1615
W
W .................................. 566
size prefix .................................. 566
waiting symbol .................. 1487, 1891
cube, visualization element .......... 1487, 1891
flower, visualization element .......... 1490, 1894
warm start .................................. 931
warning disable, pragma ........... 636
warning restore, pragma ........... 636
watch
add all forces to watchlist .......... 884
open view .................................. 883
watch all forces ............. 317, 883
watch list .................................. 330
Profiler .................................. 3692
watchlist ................................ 883
add variable ............................... 1037
web browser
  visualization, example .................. 1961
webbrowser ............................... 1483, 1887
  visualization element .................. 1483, 1887
webvisu .................................. 1211
calling a page in the browser .......... 1212
execute .................................. 1211
WebVisu
  object .................................. 1624
WHILE .................................... 439
whitespace ................................. 866
  show in text editor ...................... 866
Who-Has .................................. 3458
Who-Is .................................. 3458
window
  auto hide ................................ 963
dock ....................................... 963
float ....................................... 963
reset layout ............................. 962
window <n> ................................ 965
windows .................................... 964
  hide ..................................... 107
  layout .................................. 106
  move ..................................... 106
  resize ................................... 107
  show ..................................... 107
toggle ..................................... 108
Windows Certificate Store ............... 139
Wink ....................................... 937
word
  addressing mode ........................ 566
WORD ....................................... 570
worksheet .................................. 977
write ....................................... 315
write file .................................. 2177
write protection .......................... 143
  project .................................. 142
WriteProperty ............................ 3456, 3460
WritePropertyMultiple ................. 3456, 3460
writing values
  command .................................. 944
  prepare value ............................ 1043
WSTRING .................................... 577

X
X ............................................. 566
  size prefix .............................. 566
XADD ....................................... 553
XOR ......................................... 514
XORN ....................................... 466
Z
zoom
  factor .................................... 429
  graphical editor ........................ 429
zoom in/out
  graphical editor ........................ 429