Operating Instructions

PB610 Panel Builder 600 Programming Software for CP600 Control Panels

Power and productivity
for a better world™

Content

Before You Start 15

Safety Notices 15

Markups 15

Getting Started 16

Assumptions 16

Installing the Software 16

System Requirements 16
Installation 17

Installing multiple Versions of Panel Builder 600 on the same Computer 18

The HMI Runtime 21

Basic Unit Settings 22

Other Context Menu Options 24

Built-in SNTP Service 25

My first Project 26

Creating a New Project 26

The Workspace 28

Communication Protocols 30

Tags 31

Tag Editor 33
Data Types 36

Dictionaries 37

Designing a Page 38

Importing a Page 39

The Widget Gallery 41

“Attach To” and Dynamic Properties 43

The HMI Simulator 45

Launching the Simulator 45

Stopping the Simulator..45
Simulator Settings..46

Transferring the Project to Target..47

When Target Flash Memory is Low..50
Update Package..51
The Runtime Loader..54

My First AC500-CP600 Project..57

Pre-Conditions..57

Step 1: Basic Settings in the ABB Control Builder Plus..57

Step 2: CoDeSys Settings and a Small Test Project..59

Step 3: PB610 Panel Builder 600..63

Start a new Project from Control Builder Plus..63
Protocol Configuration..66
Tag Import – Work with Tags..68
Attach the Tags to Widgets..70
Download the Project to the Control Panel..72

Further Help..73

Communication Protocols..74

ABB Modbus TCP Driver..74

Protocol Editor Settings..74
Tag Import..75
Aliasing Tag Names in Network Configurations..77
Special Data Types..78
Communication Status..79
Implementation Details..79

ABB Modbus RTU Driver..80

Protocol Editor Settings..80
Tag Import..82
Aliasing Tag Names in Network Configurations..83
Special Data Types..84
Communication Status..85
Implementation Details..85

ABB CoDeSys Ethernet Driver..86

Protocol Editor Settings 86
CoDeSys Software Settings 89
Tag Import 90
Aliasing Tag Names in Network Configurations 91
Data Types 93
Standard Data Types 93
Special Data Types 93
Limitations 94
Communication Status 94

ABB CoDeSys Serial Driver 95

Protocol Editor Settings 95
CoDeSys Software Settings 98
Tag Import 99
Aliasing Tag Names in Network Configurations 100
Data Types 102
Standard Data Types 102
Limitations 102
Communication Status 102

ABB Mint Controller HCP Driver 103

Settings 103
Data Types 105
Communication Status 106

ABB Robot Controller IRC5 driver 107

Safety of personnel 107

ABB IRC5 Communication Driver 107

Protocol Editor Settings 108

Tag Import 109

Aliasing Tag Names in Network Configurations 110

Data Types 112

Communication Status 112

CoDeSys V2.3 Ethernet Driver 113

Protocol Editor Settings 113
CoDeSys Software Settings 116
Tag Import 116
Aliasing Tag Names in Network Configurations 118

Data Types 121
Standard Data Types 121
Special Data Types 122
Limitations 122
Communication Status 122

Modbus TCP Server Driver 123

Principle of Operation 123
Protocol Editor Settings 124
Communication Status 124
Implementation Details 125

Modbus RTU Server Driver 125

Principle of Operation 126
Protocol Editor Settings 127
Communication Status 128
Implementation Details 128

Modbus TCP Driver 129

Protocol Editor Settings 129
Tag Import 131
Aliasing Tag Names in Network Configurations 132
Special Data Types 133
Communication Status 134
Implementation Details 134

Modbus RTU Driver 135

Protocol Editor Settings 135
Tag Import 139
Aliasing Tag Names in Network Configurations 140
Special Data Types 141
Communication Status 142
Implementation Details 143

Variables 143

Programming Concepts 144

Attach To 144

Events 147

OnClick/OnMouseClick 148
OnHold/OnMouseHold 148

Project Properties / Project Widget 151
FreeType Font Rendering 153
System Variables 158
Alarms 159
Communication 160
Daylight Saving Time 161
Device 162
Dump Information 164
Network 165
Printing 166
SD Card 167
Time 168
USB Drive 169
User Management 170
Actions 171
Widget Actions 171
Show Widget 171
Trigger IP Camera 172
Slide Widget 173
Refresh Event 174
ContextMenu 174
ReplaceMedia 174
Keyboard Actions 174
Send Key 175
Send Key Widget 175
ShowKeyPad 177
Keyboard Macros 177
Page Actions 177
Load Page 178
Home Page 178
PrevPage 178
NextPage 178

LastVisitedPage ... 179
ShowDialogPage ... 179
CloseDialog ... 179
ShowMessage ... 179
LaunchApplication ... 180
Launch Browser ... 181
LaunchVNC ... 181

MultiLang Actions ... 181

SetLanguage ... 181

Tag Actions ... 183

Data Transfer ... 183
Toggle Bit ... 184
SetBit ... 185
ResetBit ... 186
WriteTag ... 186
StepTag ... 187
ActivateGroup ... 187
DeactivateGroup ... 187

Trend Actions ... 187

RefreshTrend ... 187
ScrollLeftTrend ... 188
ScrollRightTrend ... 188
PageLeftTrend ... 188
PageRightTrend ... 188
PageDurationTrend ... 188
ZoomInTrend ... 189
ZoomOutTrend ... 189
ZoomResetTrend ... 189
PauseTrend ... 189
ResumeTrend ... 189
ShowTrendCursor ... 189
ScrollTrendCursor ... 190
ScrollTrendToTime ... 190
Consumption MeterPageScroll ... 191

Alarm Actions ... 192

SelectAllAlarms ... 192
AckAlarm ... 192

ResetAlarm ... 192
EnableAlarms ... 193

Event Actions ... 193

ScrollEventsBackward ... 193
ScrollEventsForward ... 193

System Actions ... 193

Restart ... 193
Dump Trend ... 194
DeleteTrend ... 196
DumpEventArchive ... 197
DeleteEventArchive ... 199
ResetProtoErrCount ... 199
SafelyRemoveMedia ... 200

Recipe Actions ... 201

DownLoadRecipe ... 201
Upload Recipe ... 201
WriteCurrentRecipeSet ... 202
DownLoadCurrentRecipe ... 202
UploadCurrentRecipe ... 203
ResetRecipe ... 203
DumpRecipeData ... 204
RestoreRecipeData ... 204

User Management Actions ... 205

LogOut ... 205
SwitchUser ... 206
ResetPassword ... 206
AddUser ... 207
DeleteUser ... 208
EditUsers ... 208
DeleteUserManagementDynamicFile ... 209
ExportUsers ... 209
ImportUsers ... 210

Print Actions ... 211

PrintGraphicReport ... 211
PrintText ... 211
EmptyPrintQueue ... 211
PausePrinting ... 211

ResumePrinting 212
AbortPrinting 212

Using Windows Client 213

Time Zone Options for Windows Client 214

Using the Integrated FTP Server 215

Using ActiveX Client for Internet Explorer 216

Installing ActiveX 216

HTTP Access to ActiveX files 216

Internet Explorer Settings 217

Security Setting for Trusted Site Zone 218

Install Active X in Internet Explorer 219

Uninstalling Active X 219

ActiveX information 219

Using VNC for Remote Access 220

VNC Server 220

VNC Viewer 221

Alarms 222

Alarm Configuration Editor 222

Alarm's State Machine 226

Setting Events 227

Log Events 227
Actions 230

Active Alarms Widget 231

Alarms History Widget 233

Managing Alarms at Runtime 234

Enable / Disable Alarms at Runtime 234

Live Data in Alarms Widget 235

Exporting Alarm Buffers as CSV File 236

Recipes 237

Recipe Configuration Editor 238

Configuring Recipes Set on the Page 239

Defining Recipe Fields 240

Recipe Status 241

Configuring Recipe Widget for Runtime Execution 241

Configure Recipe Transfer Macros 242

Upload or Download of Recipes during Runtime 242

Recipe Download through Recipe Widget in Runtime 242
Recipe Download or Upload Through Recipe Transfer Macro in Runtime 243
Backup and Restore of Recipe Data 243

Trends 244

Real-Time Trend 244

History Trend 245

Configuring Trend Window for History Trends 249

Trend Window Properties (Advanced View) 250

Trend Cursor 251

Exporting Trend Buffer Data to CSV file 252

Scatter Diagram / XY Graph 253

Data Transfers 254

The Data Transfer Editor 254

Data Transfer Toolbar Buttons 254

Data Transfer Fields 255

Exporting Data to .csv Files 256

Data Transfer Limitations and Recommendations 257

Offline Node Management 258

Offline Node Management Process 258

Manual Offline Node Management Process 258

Manual Offline Configuration 259

Automatic Offline NodeDetection 261

Offline Management Toolbar Buttons 261

Offline Management Fields 261

Multi-Language 262

Add a Language to Project 263

Language Display Combo 263

Multi Language Widget 264

Multi Language for Static Text Widget 264

Multi Language for Message Widget 265

Multi Language for Alarm Messages 266

Multi Language for Pop-up Messages 266

Export and Import of Multilanguage Strings 267

Change Languages at Runtime 270

Scheduler 272

Configuring the Scheduler Engine 272

High Resolution Scheduler 274

Recurrence Scheduler 274

Configuring Location in PB610 Panel Builder 600 276

Configuring the Schedule Interface for Runtime Interaction 277

Schedule the Events during Runtime 278

User Management and Passwords 279

Configuring Security Options 280

Configuring Groups and Authorizations 280

Modifying the Access Permission of Groups 281

Widget Permissions 281

Action Permissions 283

FTP Authorizations 284

HTTP Authorizations 285

Miscellaneous 286

Access Priority 287

Configuring Users 287

Default User 288

Assigning Widget Permissions from Page View 288

Operation on Runtime 289

Force Remote Login 290

Audit Trails 291

Enable or Disable the Audit Trail 291

Configure Audit Events 292

Configure Tags in the Audit Trail 293

Configure Alarms in the Audit Trail 293

Configure Login or Logout Details in Audit Trail. 294

Viewing Audit Trails in Runtime 294

Exporting Audit Trail as CSV File 294

Reports 295

Adding a Report 295

Text Reports 296

Graphic Report 297

Page Body 298

Header and Footer 299

The Context Widget Gallery 299

Printer Configuration 300

Supported Printers 300

Print Events 301

Minimum Requirements 302

Screen Saver 303

Backup/Restore 305

Keypads 307

Creating and Using Custom Keypads 308

Deleting or Renaming Custom Keypads 310

Keypad Type 311

External Keyboards 312

Search and Filter 313

Shows 314

Clear Actions 314

Keyboard Layout 315

Enable Keyboard 315

Configure Macro Actions for Keys 316

Special Widgets 317

DateTime Widget 317

RSS Feed Widget 318

Control List Widget 319

State 320

Selection 321

Write on Select: 321

Write on Enter: 321

Read Only: 321

Variables Widgets: 322

Using Variables in JavaScript 324

IPCamera Widget 324

Multistate Image Widget 325

Combo Box Widget 326

Consumption Meter Widget 327

Custom Widgets 330

Creating Custom Widget 330

Adding the Properties 331

Editing Custom Properties 333

Sending E-Mail 334

Sending E-Mail Script 334

Configure E-Mail Server 334

Configure E-Mail Accounts 335

Sending Live Tag Data through E-Mail 335

Limitation 336

JavaScript 337

Execution 337

Events 337

Widget Events 338
Page Events 340
System Events 341

Objects 343

Widget 344
Page 347
Group 349
Project 350

Keywords 357

Global Functions 358

Limitations 358

Debugging of JavaScript 359

Remote JavaScript Debugger 360

System Settings Tool 361

User Mode 361

Updating System Components in HMI Panels 364

List of Upgradable Components 365

Update of System Components from Panel Builder 600 366

Update of the System Components via USB Flash Drive 368

Access Protection to HMI Devices 369

Firewalling 370

Factory Restore 371

PB610 Panel Builder 600 Functional Specifications and Compatibility 372

Table of Functions and Limits 372

Compatibility 373

Before You Start

Safety Notices

⚠ **DANGER!**

Indicates an imminent risk. It will lead to death or serious injury if not avoided.

⚠ **WARNING!**

Indicates a possible risk. It may lead to death or serious injury if not avoided.

⚠ **CAUTION!**

Indicates a possible risk. It may lead to light or slight injury or material damage if not avoided.

Markups

- Enumeration.

✓ Precondition for an operation instruction or a description.

→ Operation instruction with one step.

1. Operation instruction with several steps.

    ➤ Result of an operation.

NOTE

Helpful information with background information or an emphasized notice.

TIP

Application tips or other useful information and suggestions.

Getting Started

PB610 Panel Builder 600 is a software application that allows you to create graphical HMI pages. PB610 Panel Builder 600 uses a drag-and-drop system that makes it easy to create displays. The same features found in many popular Windows applications are also available in PB610 Panel Builder 600.

This document describes how to use the PB610 Panel Builder 600 application, and is divided into chapter that represent the key operations of PB610 Panel Builder 600. Each chapter is presented in a standalone manner, allowing you to jump from chapter to chapter, depending on the task you wish to perform.

Assumptions

We assume that those reading this manual are using PB610 Panel Builder 600 software to design Control Panel applications that run on ABB Control Panels and on a PC.

We also assume that you have a basic understanding of PCs, Microsoft Windows, and the type of network environment in which you will run the application.

Installing the Software

The Panel Builder 600 contains the following as part of the installation:

Panel Builder 600

Panel Builder 600 is an application for designing custom HMI projects in a user-friendly manner, along with a variety of options in its built-in library, the Widget Gallery.

Windows Client

Windows Client is a light-weight application that can be used on Windows computers to remotely view and manage an application running on an HMI Runtime.

HMIce Runtime

The HMIce Runtime is a standalone application that runs on the CP600 HMI panels. The HMI Runtime can be installed via Panel Builder 600 and is designed for working with WCE 6.0 OS.

System Requirements

PB610 Panel Builder 600 has the following system requirements:

- Windows XP (Service Pack 2 or 3), Windows Vista (Service Pack 1 or 2), Windows 7, Windows 8
- 500 MB of disk space
- Minimum of 512 MB RAM
- One Ethernet interface

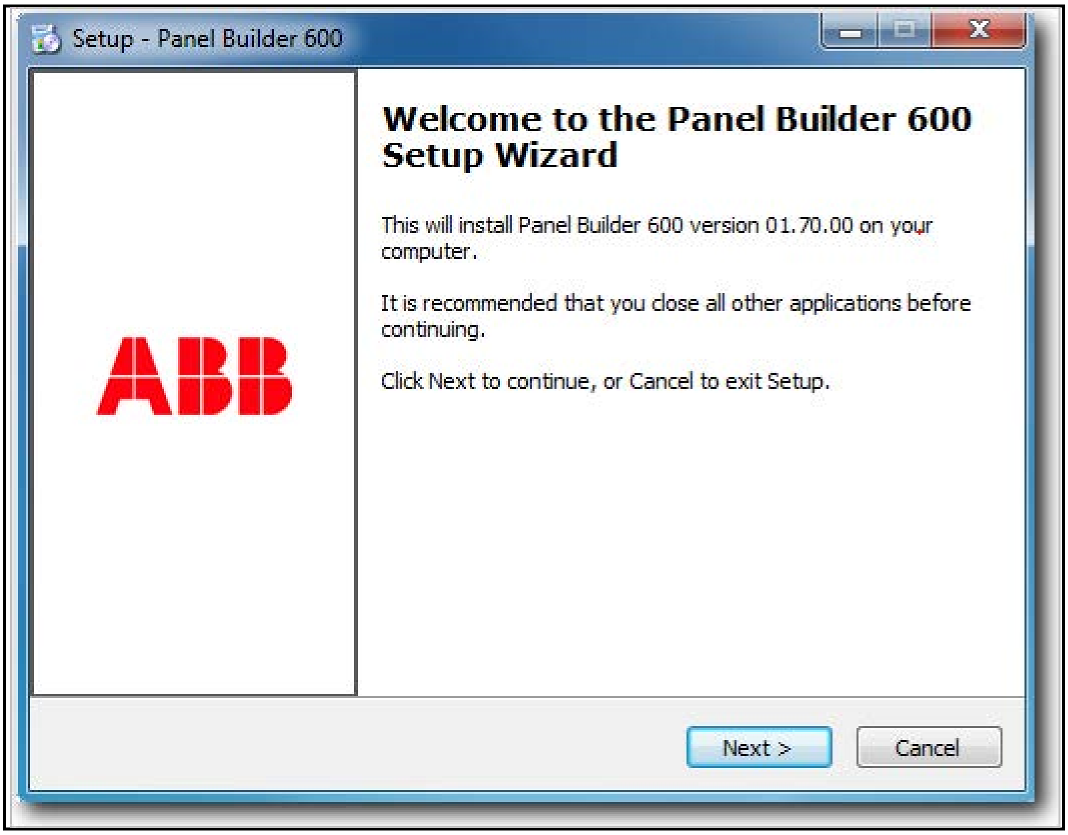
Installation

NOTE

When running Windows 7 operating system, the installation must be done from users with administration privileges.

Insert the USB ROM drive into the USB port. If your system has autorun enabled, the PB610 Panel Builder 600 installation will start automatically, otherwise run the PB610 Panel Builder 600 setup application.

1. Click on the **Start** button and select **Run** from the menu.
2. Type D:\setup in the box if you are running the installation from the USB drive (if your USB drive is not drive D, replace with the appropriate letter).
3. Click **Next**.



Setup - Panel Builder 600

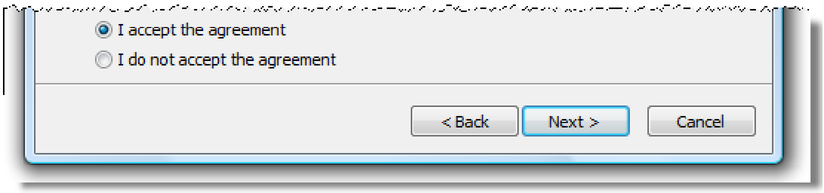
Welcome to the Panel Builder 600 Setup Wizard

This will install Panel Builder 600 version 01.70.00 on your computer.

It is recommended that you close all other applications before continuing.

Click Next to continue, or Cancel to exit Setup.

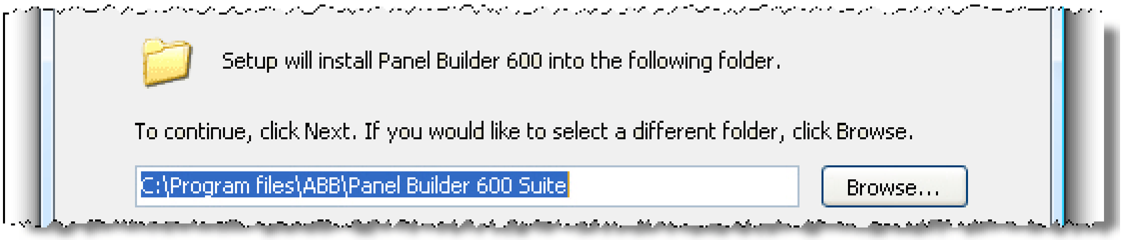
4. Read the PB610 Panel Builder 600 software license and accept the agreement.



● I accept the agreement

○ I do not accept the agreement

5. Follow the instructions on the screen. The default location for the PB610 Panel Builder 600 software is *C:\Program Files\ABB\Panel Builder 600 Suite*. Default installation path can be changed depending on need.



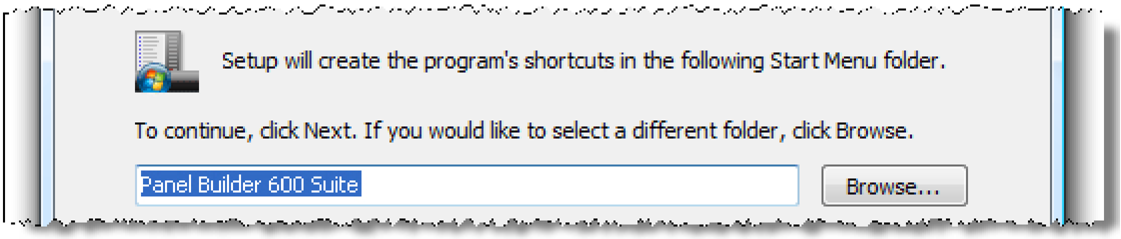
Setup will install Panel Builder 600 into the following folder.

To continue, click Next. If you would like to select a different folder, click Browse.

C:\Program files\ABB\Panel Builder 600 Suite

6. If the Select Components step is available, select the components you want to install.

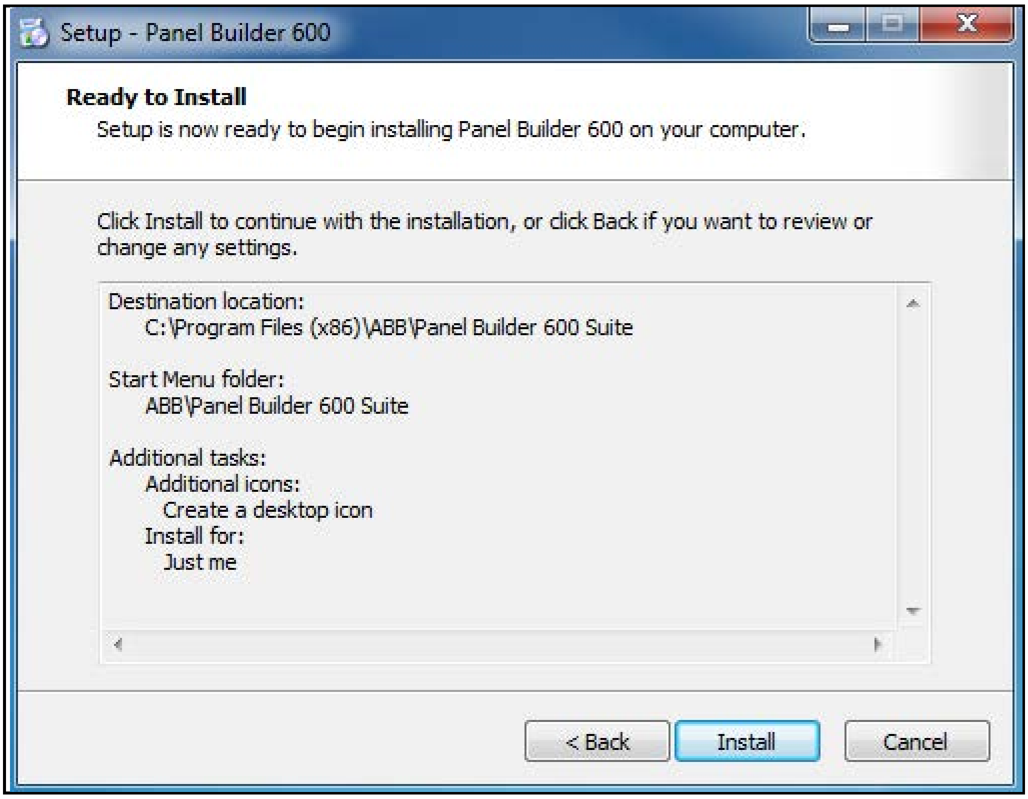
   ➤ The installation procedure will create a program group called *Panel Builder 600 Suite* in the start menu. A PB610 Panel Builder 600 icon can be added on the desktop.



Setup will create the program's shortcuts in the following Start Menu folder.

To continue, click Next. If you would like to select a different folder, click Browse.

Panel Builder 600 Suite



Setup - Panel Builder 600

Ready to Install

Setup is now ready to begin installing Panel Builder 600 on your computer.

Click Install to continue with the installation, or click Back if you want to review or change any settings.

```
Destination location:
    C:\Program Files (x86)\ABB\Panel Builder 600 Suite

Start Menu folder:
    ABB\Panel Builder 600 Suite

Additional tasks:
    Additional icons:
      Create a desktop icon
    Install for:
      Just me
```

➤ After installing the PB610 Panel Builder 600, you can run the PB610 Panel Builder 600 application by using the desktop icon or from *Start* > *All programs* > *Panel Builder 600 Suite*.

Installing multiple Versions of Panel Builder 600 on the same Computer

You may install different instances of Panel Builder 600 on the same computer. Each installation has its own settings and can be uninstalled individually.

During the installation process you may encounter three scenarios:

First installation of Panel Builder 600 in the system

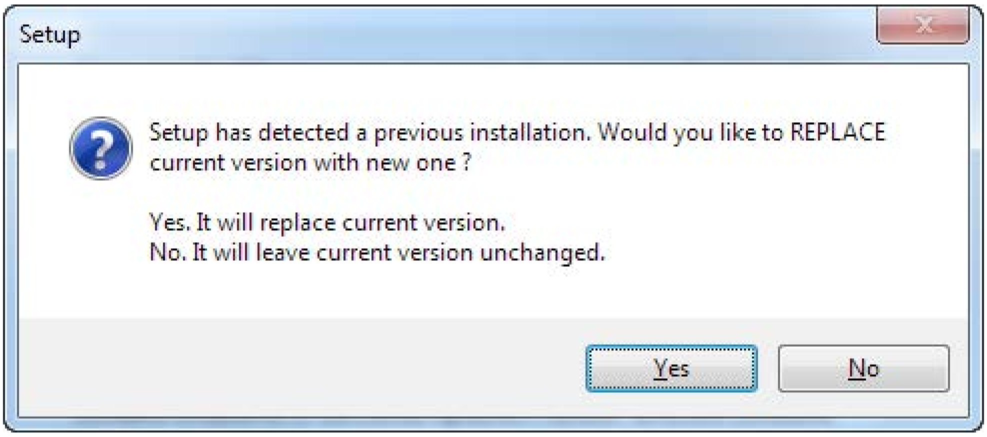
The installation procedure asks for the destination folder and installs that software in the specified folder.

System with only one instance of Panel Builder 600 already installed

The installer detects that one version is already available and it will ask if you want to replace the current version with the new one or if you want to install another instance.

System with multiple instances of Panel Builder 600 already installed

The installer detects that one or more installations of Panel Builder 600 are present and it will ask if you want to replace the last installed instance with the new one or if you want to install another instance.

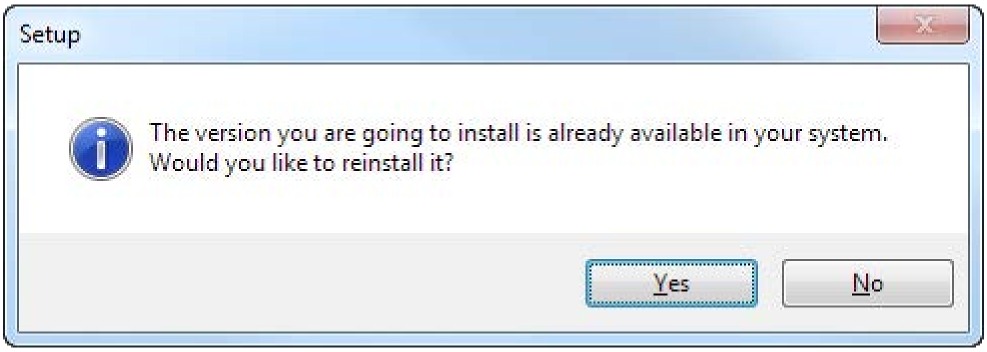


Setup

Setup has detected a previous installation. Would you like to REPLACE current version with new one ?

Yes. It will replace current version.
No. It will leave current version unchanged.

When the user tries to install a second instance of the same version of Panel Builder 600, the setup procedure will detect it and will show a warning message that the same version is already available in the computer. Setup will also provide the user with the reinstall option.



Setup

The version you are going to install is already available in your system.
Would you like to reinstall it?

Each installation goes to a sub-folder which has the version number as part of its name; for instance:

C:\Program Files\ABB\Panel Builder 600 1.90.

Each installed version provides its ID in the Control Panel. Add/remove program facility is provided in order to remove it individually from the computer.

Each installation identifies itself in the *Start Menu* with a program group name.

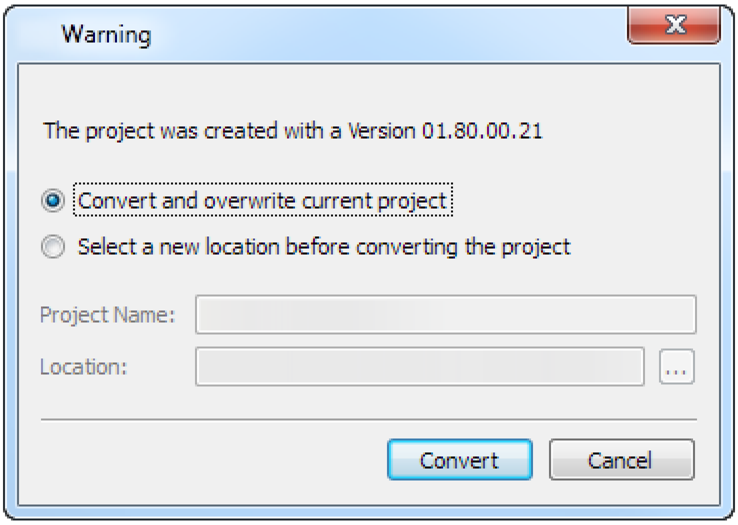
NOTE	Multiple Studio installations provide a common workspace folder for all instances of the Panel Builder 600 that have been installed.

Opening Projects created with older Version of Panel Builder 600

When a Panel Builder 600 project (file with .jpr extension) is opened, Panel Builder 600 checks for the match between the version ID stored in the jpr file and its version ID; if they match, the project will be opened normally; if they do not match, Panel Builder 600 shows a warning message to inform that the project has been created with a different version of Panel Builder 600 and report this version ID if it is available in jpr.

In this case Panel Builder 600 will offer two options to convert the project.

1. Convert and open the project from current path. The project will be converted without a backup copy of the original version.
2. Convert and save the project to a new location and Open. The older version is maintained as a backup copy.



Warning

The project was created with a Version 01.80.00.21

● Convert and overwrite current project

○ Select a new location before converting the project

Project Name:

Location:

WARNING!

Do not edit projects with a version of Panel Builder 600 older than the one used to create them. It can result in a damage of the project and to runtime instability.

The HMI Runtime

The HMI Runtime is designed to support different platforms and different operating systems.

All the panels are running today on the base of the Windows CE operating system (Version 6 R3).

The operating system and all its options are built around the minimum set of requirements of the HMI Runtime; there is no option to get direct access to the Operating system settings as all the needed components are managed via the runtime itself or via PB610 Panel Builder 600.

Later in this document you will find more information on how to install the HMI runtime and how to manage the update of other system components (firmware) on the units, but always with a dedicated interface which prevents a direct access to the operating system, often a source of complexity.

Runtime Modes

The HMI Runtime is composed of two logic units: the server and the client. The client unit is the part which is responsible for the visualization process: using the data collected at the server side to render it on the display as graphical information.

The server unit is responsible for handling the HMI services such as running the communication protocols, performing data acquisition, driving trend buffer sampling activities, monitoring alarms, and so on. The server unit of the HMI runtime can be in one of two operating modes.

Configuration mode: The server is idle; activity has not started: for example no project is loaded on the panel or system files are missing.

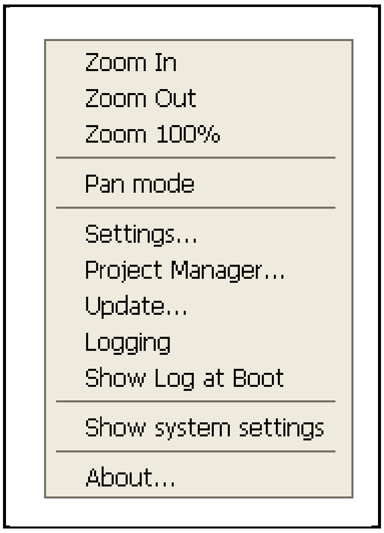
Operation mode: the server is active; it is operating according to the settings defined by the system files and by the application project.

The server operating mode is independent of the client side operating mode; you may have a visualization running but server activity stopped.

Basic Unit Settings

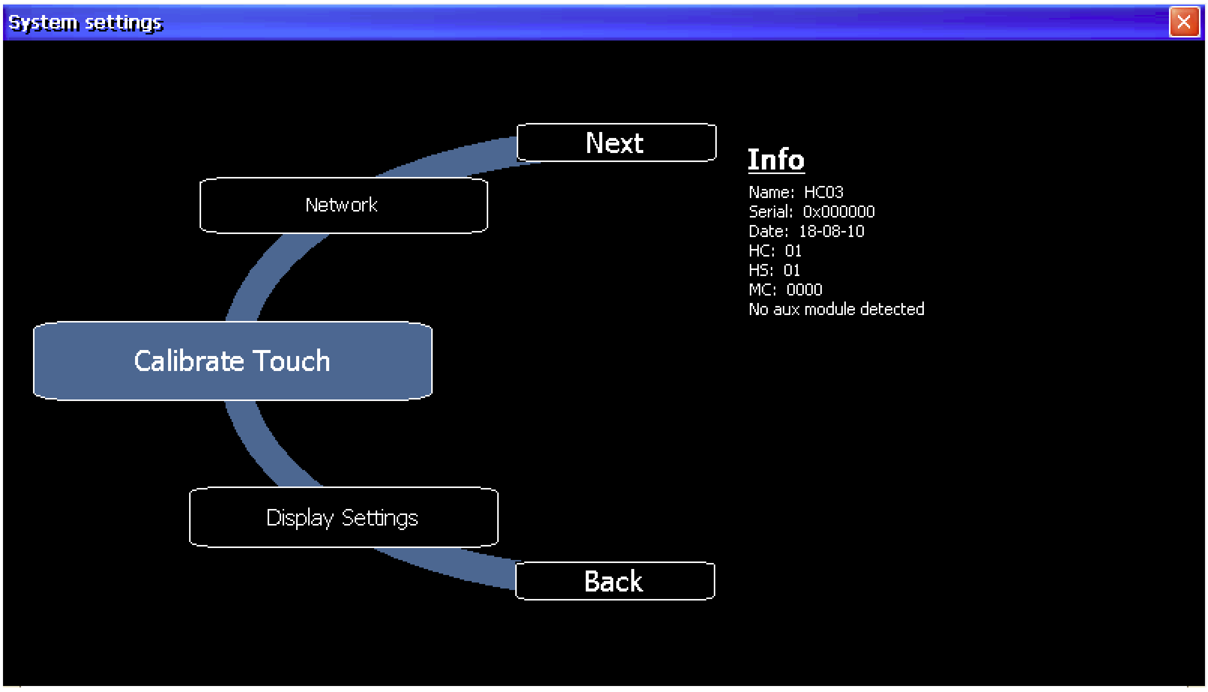
The settings of the device are available from the show system settings menu, which is accessible from the context menu , if the panel has the runtime already installed or by using the dedicated button on the unit, when it is in loader mode (see for this the chapter “The Runtime Loader” below in this document).

1. Press and hold your finger on the screen for a few seconds, until the context menu appears as shown in the figure below.



2. Select **Show system settings** to access the system settings tools.

3. The system settings tool is a rotating menu through which you can scroll using the **Next** and **Back** buttons.



Info

Name: HC03
Serial: 0x000000
Date: 18-08-10
HC: 01
HS: 01
MC: 0000
No aux module detected

The system settings tool includes the following entries:

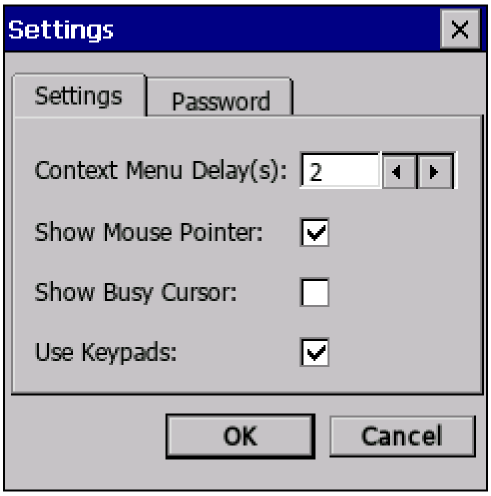
- **Calibrate Touch:** To calibrate the touch screen if needed
- **Display Settings:** Backlight and brightness control
- **Time:** Internal Real Time Clock (RTC) settings

- **BSP Settings:** Operating system version, unit operation timers: power up and activated backlight timers, Buzzer control, Battery LED control
- **Network:** IP address settings
- **Plugin List:** Provides a list of the plug-in modules installed and recognized by the system; this option may not be supported by all platforms and all versions.

Other Context Menu Options

The context menu has several other options:

- Zoom In/Out/100%: Select the view at runtime
- Pan Mode: Enables/disables the pan mode; works only when you have previously activated a zoom in
- Settings: Following runtime settings are available:



Context Menu Delay(s): 2

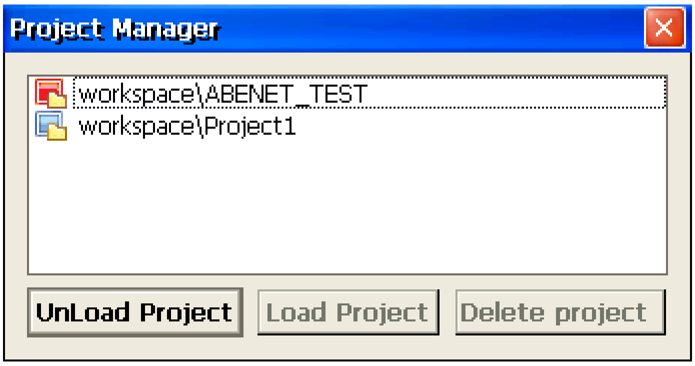
Show Mouse Pointer: ☑

Show Busy Cursor: □

Use Keypads: ☑

Setting	Explanation
Context Menu Delay (sec)	The context menu activation delay. Range is 1-60 seconds.
Show Busy Cursor	When enabled, shows an hourglass when the system is busy
Use keypads	When enabled, shows touch keypads when users touch/click on fields for data entry. When disabled, does not show any keypad on screen (useful when an external USB keyboard is connected to device).
Password	Change the password protecting operations such as: - Download Project/Runtime - Upload project - Board management (BSP Update) Refer to Remote access protection to HMI Panels for more information related to access protection.

- Project Manager: When activated, a dialog box will appear (see figure below) providing options to unload (de-activate) the current project, load (activate) another project present on the panel memory, or delete a project. Please note that projects can be deleted only after they are unloaded. If you click on a project name other than the active one, the option "Load project" will first unload the running application and then automatically activate the new one.



workspace\ABENET_TEST

workspace\Project1

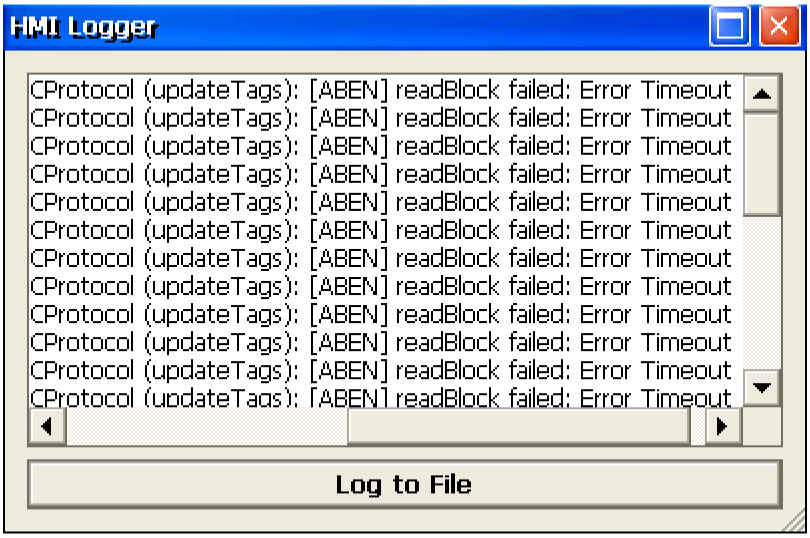
- Update: When activated, the panel verifies first the presence of an external USB memory stick inserted in the panel USB port, and later for the presence on its root folder of the update package. See the “Updating Runtime from USB Memory Stick“ chapter in this document for further information.

- Backup: Creates a backup copy of runtime and project.

- Logging: Enables you to display a trace of the system operation log; it may be very useful in case there is a need to debug a problem of any nature. The following figure shows a case in which the system reports a communication error; the decoding of the reported information may not be immediate, but you can always use the option “Log to file” to save the dialog context to a file that can be later provided to Technical Support for investigation. The log file is called “logger.txt” and it saved to the folder “..\var\log” on the panel flashdisk. The file can be retrieved from the panel using an FTP client.

NOTE

The “Log To File” Option is saved and retained after power cycles; when not needed any more it must be manually deactivated



```
CProtocol (updateTags): [ABEN] readBlock failed: Error Timeout
CProtocol (updateTags): [ABEN] readBlock failed: Error Timeout
CProtocol (updateTags): [ABEN] readBlock failed: Error Timeout
CProtocol (updateTags): [ABEN] readBlock failed: Error Timeout
CProtocol (updateTags): [ABEN] readBlock failed: Error Timeout
CProtocol (updateTags): [ABEN] readBlock failed: Error Timeout
CProtocol (updateTags): [ABEN] readBlock failed: Error Timeout
CProtocol (updateTags): [ABEN] readBlock failed: Error Timeout
CProtocol (updateTags): [ABEN] readBlock failed: Error Timeout
CProtocol (updateTags): [ABEN] readBlock failed: Error Timeout
CProtocol (updateTags): [ABEN] readBlock failed: Error Timeout
CProtocol (updateTags): [ABEN] readBlock failed: Error Timeout
```

- Show log at boot: Enables the logger at startup; if the “Log to file” option has been enabled, the files are saved, in this case, from the startup phase.

- Developer tools: It is a collection of utility functions useful for debugging problems at runtime.

- About: Shows information about the runtime version.

Built-in SNTP Service

The Control Panels operating system features an integrated SNTP (**S**imple **N**etwork **T**ime **P**rotocol) that synchronizes the internal RTC panel whenever the predefined server is available.

The server addresses are hard-coded and cannot be changed by the user. The system searches for the following servers:

- time.windows.com

- tock.usno.navy.mil

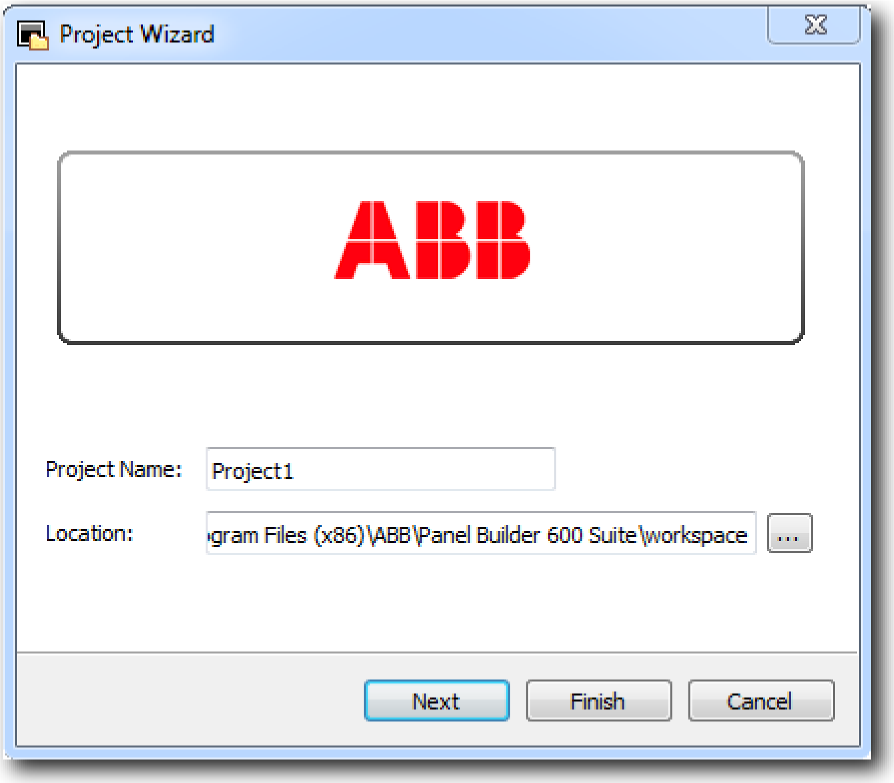
SNTP servers are checked at power up, or once per week if the panel is not powered off.

My first Project

This section describes the steps to create a simple PB610 Panel Builder 600 project.

Creating a New Project

1. To create a new project click on the *File* > *New Project* menu item.



Project Name: Project1

Location: ıgram Files (x86)\ABB\Panel Builder 600 Suite\workspace

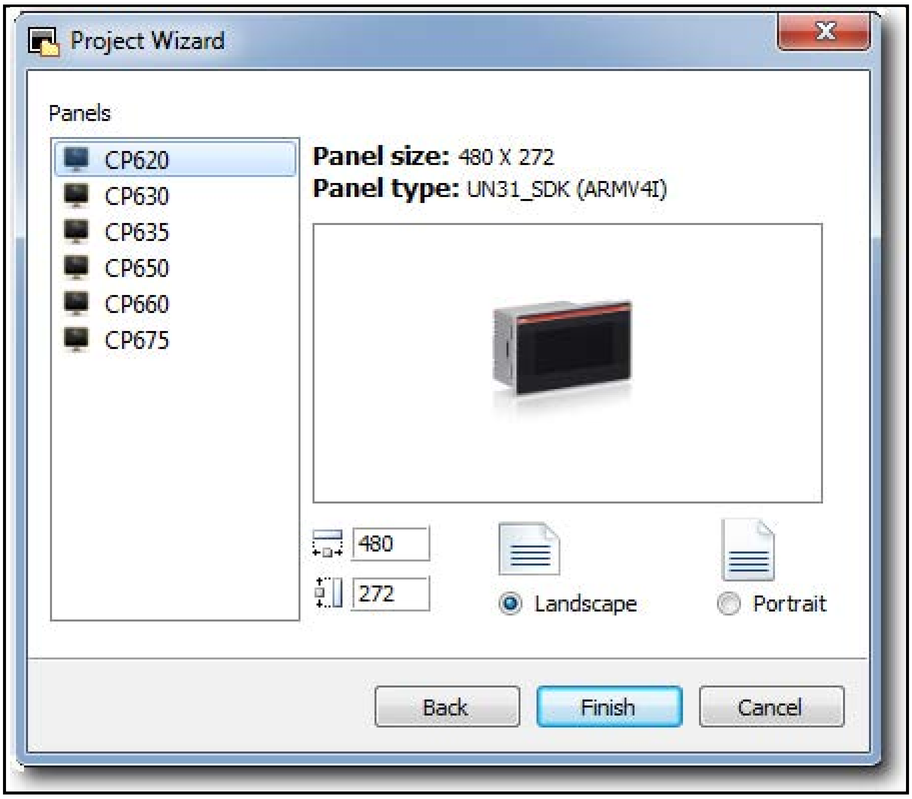
- ➤ The Project Wizard dialog will appear, asking for a project name and a path where the corresponding project folder will be stored.

2. PB610 Panel Builder 600 projects are stored in a folder that has the same name as the project. This folder contains all the files of the project. To move, copy or backup a project, simply move or copy the project folder and all its contents to the desired location.

NOTE Do NOT rename the PB610 Panel Builder 600 Project folders manually. If you need to rename a project, use the *File* > *Save Project As* function. Depending on the size of the project, this could take some time. Please wait until the newly named project opens and all pages are visible in the project tree before closing or taking any actions.

3. Click **Next** to go the Control Panel selection dialog.

4. Scroll through a list of available units to select the control panel model you are working with.



Panel size: 480 X 272
Panel type: UN31_SDK (ARMV4I)

480
272

● Landscape
○ Portrait

Per each model two radio buttons are available to select the orientation: landscape (default) or portrait. In portrait mode the panel is rotated 90° clockwise.

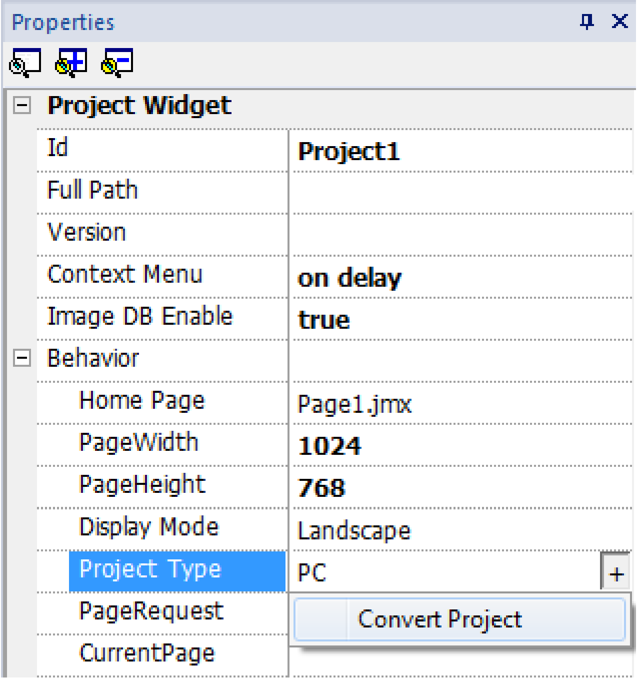
NOTE

Some software features are not rotated when portrait mode has been selected These features are listed below.

• WCE dialogs	all dialogs related to “System Settings”
• System Dialogs	ex. System Mode
• Context Menu and related dialogs	Project Manager, About, Settings, Logging, Backup
• Video	Analog Video Input, IP Camera, Media Player
• Java Script	Alert and Print function
• Dialog pages	“Title” of dialog pages
• Scheduler	Dialogs for data entry do not support portrait
• Macro	ShowMessage, LaunchApplication, LaunchBrowser
• External applications	PDF reader, VNC

5. Click **Finish** to complete the wizard.

6. Once the panel is chosen, you can convert the project to any other model, using the project properties portion of the screen, as shown below. This will not resize all widgets in the project to the correct size to fit a smaller or larger screen; it will simply change the model type and give a warning if some objects will be lost during the conversion.



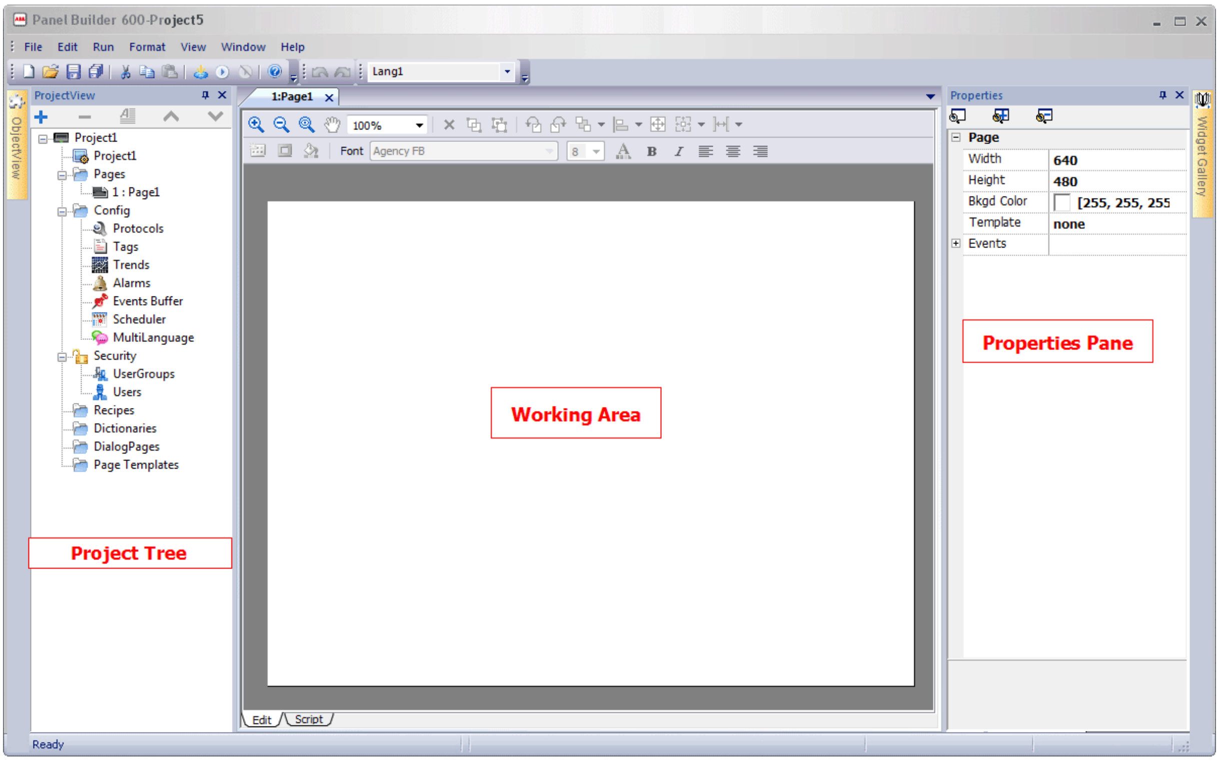
Properties

Project Widget	
Id	Project1
Full Path	
Version	
Context Menu	on delay
Image DB Enable	true
Behavior	
Home Page	Page1.jmx
PageWidth	1024
PageHeight	768
Display Mode	Landscape
Project Type	PC
PageRequest	Convert Project
CurrentPage	

The Workspace

The PB610 Panel Builder 600 workspace is divided into following main areas:

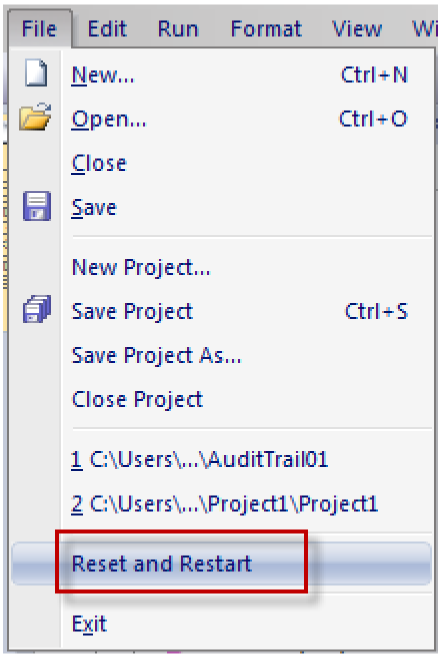
Main area	Explanation
Project View	Presents the elements of the project in the form of a hierarchical Project Tree.
Object View	Lists the Widgets with the corresponding ID's used in the page.
Working Area	Main working space where editors create the HMI pages. The current page or pages opened in the Editor View are indicated by a tab at the top of the center area. You can quickly switch between the different pages in the Editor View by clicking on the desired tab.
Properties	Properties for the selected object/widgets.
Widget Gallery	Large library of symbols and graphic objects.



Page

Width	640
Height	480
Bkgd Color	[255, 255, 255
Template	none
Events	

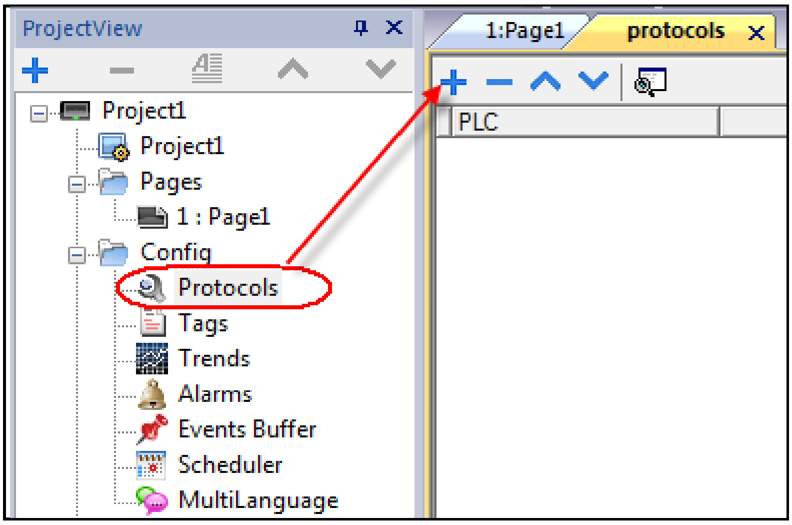
NOTE The workspace layout can be freely changed at any time; any change is saved and maintained among Studio activations. In case you need to reset the workspace to the original default layout, use the command called “Reset and Restart” from the “File” menu.



Communication Protocols

Device communication drivers are configured in the protocol editor, which is accessible from the project tree.

1. Double click on the **Protocols** icon in the project tree to open the protocol editor.
2. To add a driver, click on the **+** icon and select the driver from the list in the controller field.



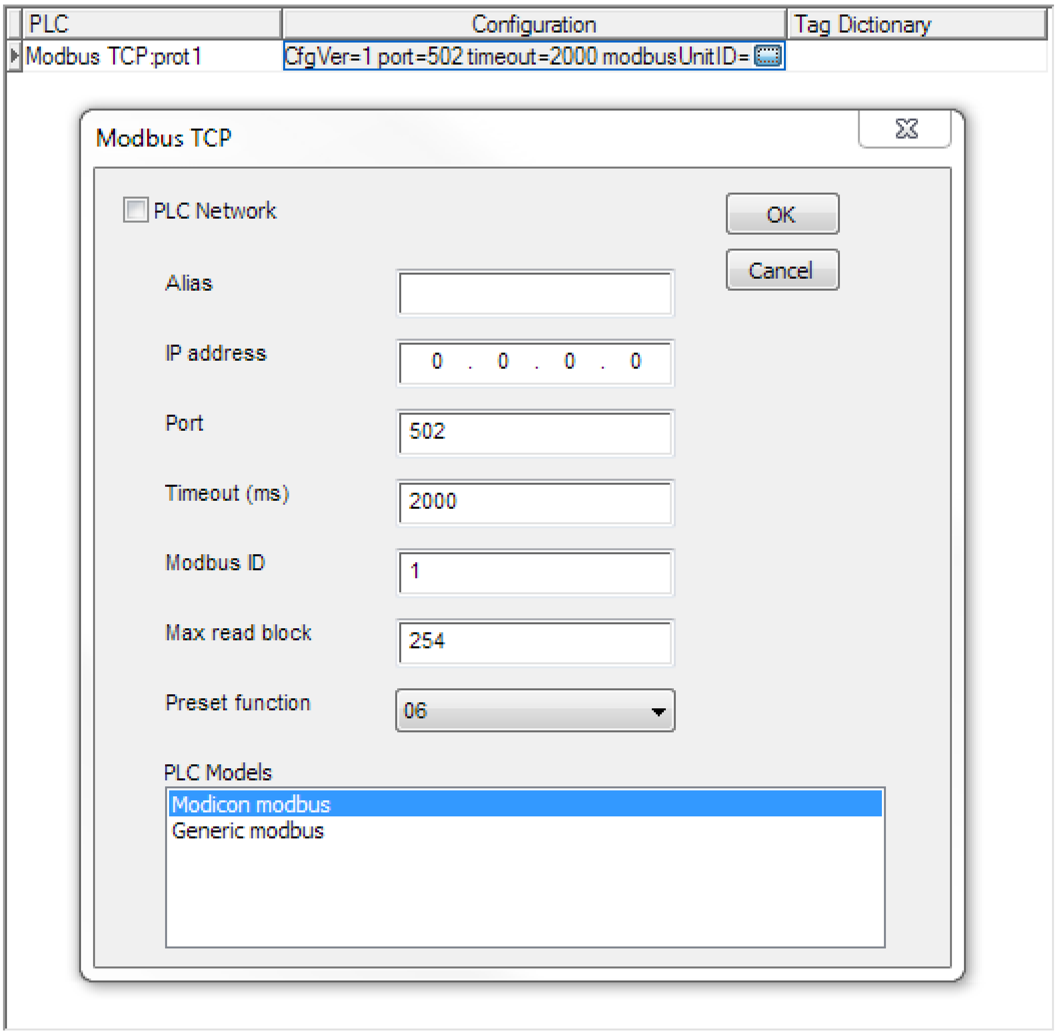
➤ The combo box shows the list of available drivers.

3. Once a driver is selected, configure the driver by clicking on the **Browse** button in the column *Configuration*. A configuration dialog will be displayed, allowing you to set the parameters of the driver.

NOTE Refer to CP600 operating instructions manual in case you need information about serial communication cables.

The combo box shows the list of available communication drivers. Once a driver is selected, configure the driver by clicking on the browse button in the Configuration field. A configuration dialog will be displayed, allowing you to set the parameters of the driver (as shown in the figure below).

4. To create a project for Modbus TCP, select the Modbus TCP driver.
5. Configure the communication parameters by selecting the **browse** button in the *Configuration* column.



PLC	Configuration	Tag Dictionary
Modbus TCP:prot1	CfgVer=1 port=502 timeout=2000 modbusUnitID=	

Modbus TCP

□ PLC Network

Alias:

IP address: 0 . 0 . 0 . 0

Port: 502

Timeout (ms): 2000

Modbus ID: 1

Max read block: 254

Preset function: 06

PLC Models: Modicon modbus (selected), Generic modbus

PB610 Panel Builder 600 supports configurations that include more than one communication protocol. By repeating the steps shown above, you can add up to four protocols in the protocol editor.

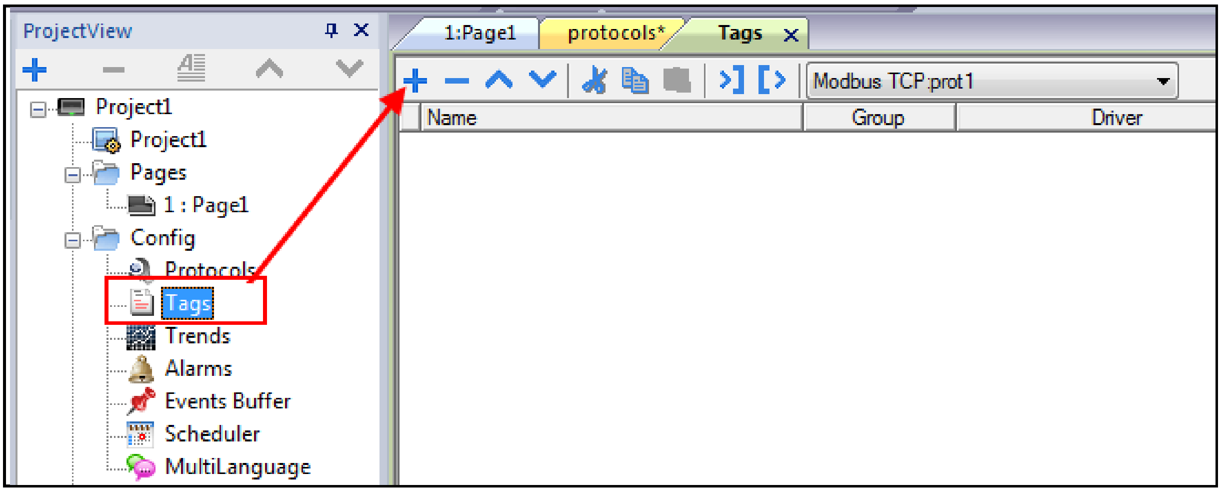
> **NOTE** While it is possible to run different Ethernet protocols over the same physical Ethernet port, you cannot run different serial protocols over a single serial port. Some serial protocols support access to multiple PLCs, but this is an option that has to be configured within the protocol that still counts one.

Other parameters in the protocol editor are:

- Tag Dictionary: Tags imported for a particular protocol. Refer to chapter Dictionaries for more details
- Enable Offline Algorithm / Offline Retry Timeout: Refer to chapter OFFline Node Management for more details.
- Version: Version of the protocol available in Panel Builder 600 for selected target. Version of the protocol is not read from hmi directly but from studio internal DLLs.

Tags

PB610 Panel Builder 600 uses tag names to access all device data. All fields and reference locations in the device need to be assigned a tag name to be used in the HMI. To assign tags, double click on the **Tags** icon in the Project View and the tag editor will be displayed.



1. To add a new tag, click on the **+** icon, and select the address from the *communication protocol address* dialog.

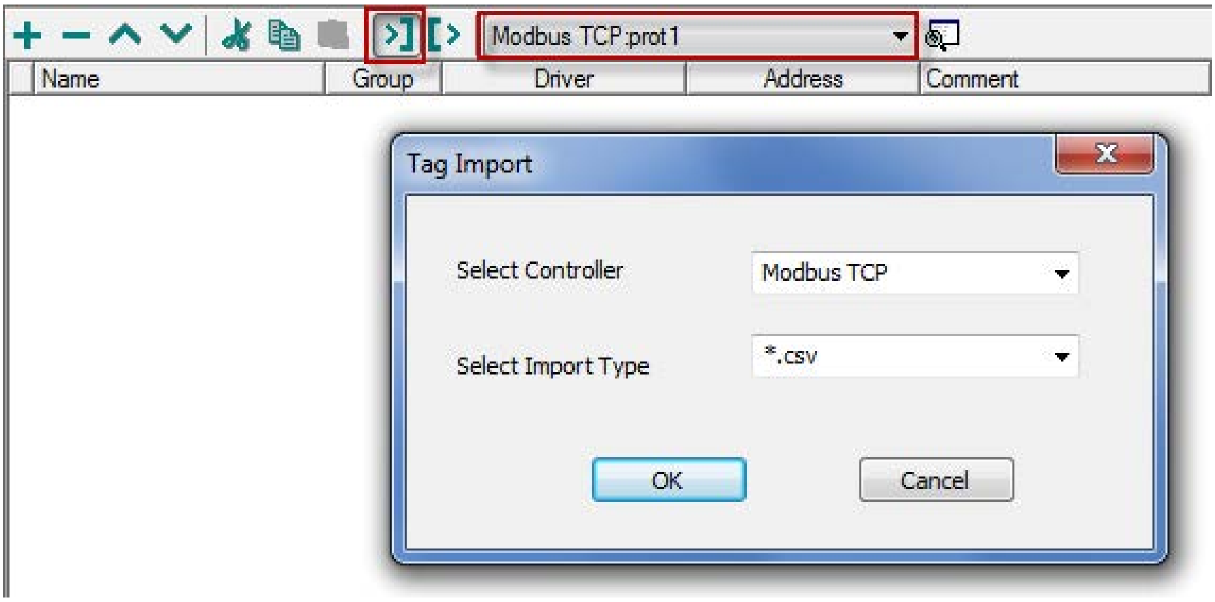
   ➤ When tags are initially added, these tags are named Tag1, Tag2 etc. by default.

2. To rename the tag click once on the tag name.

The tag editor provides a tag import feature, which is available based on the protocol selected.

Not all protocols support tag import.

3. If the protocol does support this feature (see specific protocol documentation), first select the protocol from the filter button.
4. Click on the **Import** button.



Select Controller: Modbus TCP

Select Import Type: *.csv

   ➤ You will see the dialog that corresponds to the selected protocol.

5. Browse for the symbol file. The symbol file is exported by the controller programming software.

Tag Editor

The tool in PB610 Panel Builder 600 designed to handle tags' creation and managing is called tag editor. Per each tag the tag editor allows you to specify several properties.

Name

This is the unique name at project level of the tag. This is the primary key used to identify the information in the internal runtime tag database.

> **NOTE**
>
> You cannot use the same tag name even if you are referring to different communication protocols.

Group

After the tags have been defined in the tag editor, they are used in the project screens by attaching them to the widgets' properties. See chapter “Attach To” for a complete explanation.

Per each screen the system is able to identify which tags are used in the specific page and identifies them as part of the “page group”. This allows an easy handling at runtime of the requests made by the communication protocol to the connected controller(s): Only the tags included in the displayed page are queued for retrieval from the controller memory.

This mechanism is fully automatic and there is no intervention required by the user.

The tag editor allows you to define groups of tags not belonging to a specific page but for instance grouped according to their logical meaning. These groups are called “users' group”. Users' groups have no meaning for the local visualization, but they are very useful when external software communicate with the Runtime requesting sets of data that must be independent from the currently displayed screen.

The Runtime web server publishes a set of communication interfaces that can be used from a 3rd party application to interface with the local tag database and read the tags according to their grouping.

The group column allows to define the users' groups and assign tags to them.

Driver

Specifies the communication protocol for which the tag is going to be defined.

Address

Shows the PLC controller memory address.

→ To edit the address, click on the right side of the column to get the dialog box where you can enter the address information.

Encoding

This is the encoding type for string data type (UTF-8, Latin1, UTF-2 and UTF-16)

Comment

Allows to add a description of the tag.

Rate (ms)

Define the refresh time for tag. Default is 500ms that means tag is update every 500ms.

R/W (Read/Write)

This option determines if the tag must be managed as Read only (R), Write only (W) or Read/Write (R/W). If a Tag is Write Only (W), the system never reads the tag value (& status) just writes it. When communication is not active, content of Write Only tags may not be available in widgets.

Active

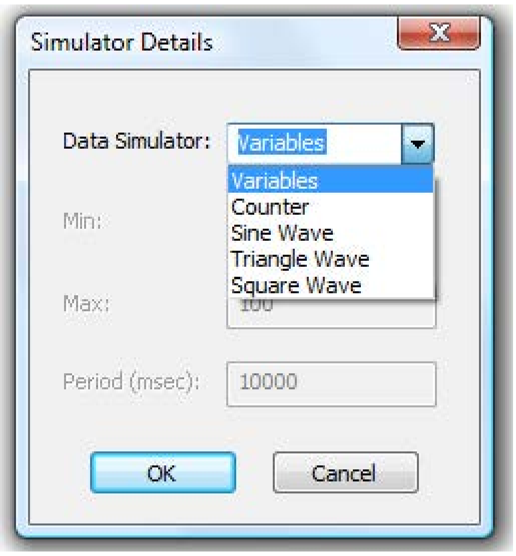
As explained above tags are grouped per page and if needed in users' groups. By default tags are not active. This means they are automatically activated by the runtime when the visualization requires them. You can force the system to continuously read a certain tag even if not present in current page by setting its active property to true.

TIP

We recommend that you leave this parameter to false to avoid unexpected results in terms of overall device performances.

Simulator

PB610 Panel Builder 600 provides on-line and off-line simulation. The behavior of each tag during off-line simulation mode can be specified by choosing between several profiles.



Simulator Details

Data Simulator: Variables

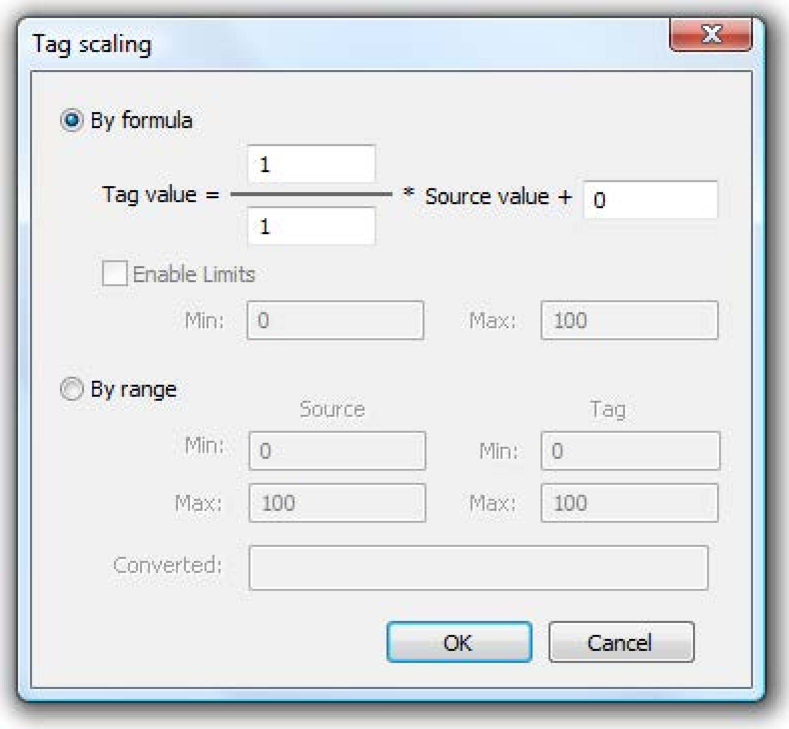
Min:

Max: 100

Period (msec): 10000

Scaling

Tags' values are normally transferred “as they are” from the protocol to the real time tag database. You can specifically apply scaling to the tag values before they are stored in the database. Scaling can be specified in terms of linear relationship as a formula or as range conversion.



Tag scaling

● By formula

Tag value = $\frac{1}{1}$ * Source value + 0

□ Enable Limits

Min: 0 Max: 100

○ By range

	Source		Tag
Min:	0	Min:	0
Max:	100	Max:	100

Converted:

Available scaling options.

The tag name must be always unique at the project level; often it may happen that the same tags, from the same symbol file have to be used for two different controllers. Since having tags with the same name is not supported, you can use the “Alias” feature to automatically add a prefix to the imported tag to make them unique at the project level.

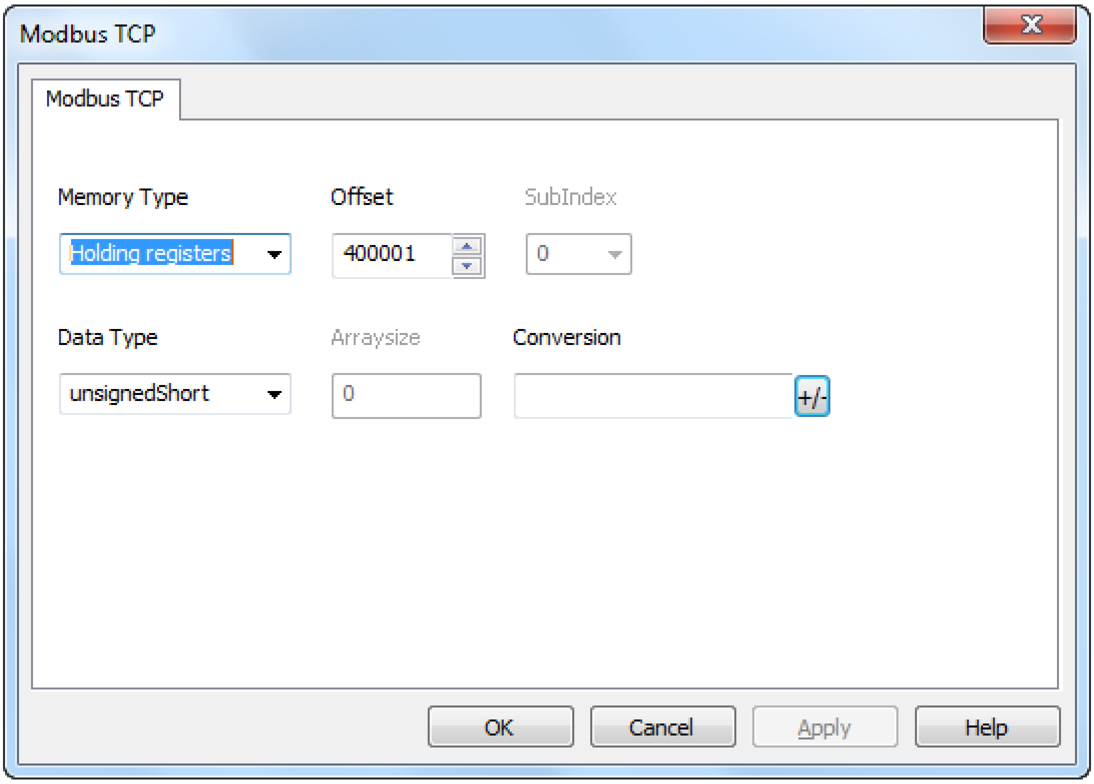
When importing tags for a protocol, the tag names may be prefixed by the name given in the “Alias” item of the protocol configuration dialog box. Please note that not all protocols support the “Alias” feature. See protocol documentation for specific information.

PLC Tag Name

This is the original name of the tag when imported from PLC. This field is managed automatically by Tag Importer and is available as R/W in advanced view just to allow users to change it if any problem during tag import operation. For tags not imported from external files usually this field is empty and can be ignored. PLC Tag Name are used so as link between tags used by HMI application (Tag Name) and tags exported from PLC.

Data Types

When creating a tag, PB610 Panel Builder 600 shows a dialog in which you need to specify the tag properties. The tag's memory types are specific for the selected protocol.



Memory Type: Holding registers
Offset: 400001
SubIndex: 0
Data Type: unsignedShort
Arraysize: 0
Conversion: +/-

The tag's data type must be selected from the list of available PB610 Panel Builder 600 data type, according to the PB610 Panel Builder 600 internal representation you need for the selected controller address. PB610 Panel Builder 600 data types are summarized in the following table.

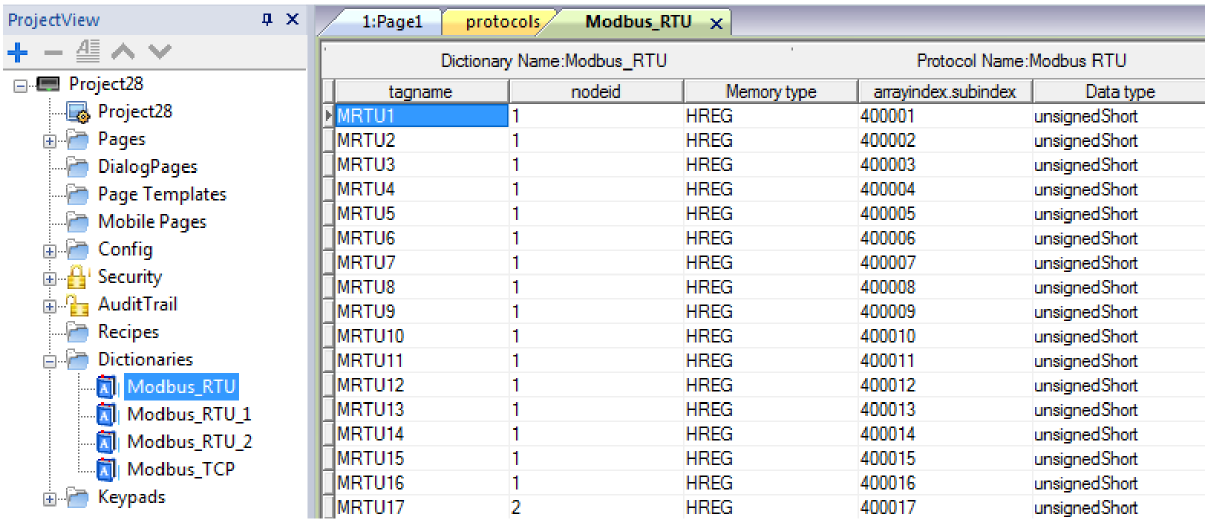
Data Type	**Description**
string	Character strings. The characters are coded in UTF-8 format.
boolean	Boolean is one bit data
float	Float corresponds to the IEEE single-precision 32-bit floating point type
double	Double corresponds to IEEE double-precision 64-bit floating point type
binary	Binary represents arbitrary binary data
int	Int is signed 32 bit data
short	Short is signed 16 bits data
byte	Byte is signed 8 bits data
unsignedInt	UnsignedInt is unsigned 32 bit data
unsignedShort	UnsignedShort is unsigned 16 bit data
unsignedByte	UnsignedByte is unsigned 8 bit data
time	Time data
boolean [ ]	Array of boolean
byte [ ]	Array of byte
short [ ]	Array of short

Data Type	Description
int [ ]	Array of int
unsignedbyte [ ]	Array of unsignedbyte
unsignedshort [ ]	Array of unsignedshort
unsignedint [ ]	Array of unsignedint
float [ ]	Array of float
double [ ]	Array of double
time [ ]	Array of time

Dictionaries

A dictionary is a list of tags imported in the tag editor for a specific protocol. Usually these files are generated by 3d party tools and are in .csv, .xml or other formats. Refer to **Tag Import** section of each protocol for details related to supported formats.

Dictionaries folder in ProjectView list all files imported in the tag editor for each protocol. Selecting a particular protocol, it is possible to delete or look at the imported dictionary files for the related protocol.



Dictionary Name:Modbus_RTU

Protocol Name:Modbus RTU

tagname	nodeid	Memory type	arrayindex.subindex	Data type
MRTU1	1	HREG	400001	unsignedShort
MRTU2	1	HREG	400002	unsignedShort
MRTU3	1	HREG	400003	unsignedShort
MRTU4	1	HREG	400004	unsignedShort
MRTU5	1	HREG	400005	unsignedShort
MRTU6	1	HREG	400006	unsignedShort
MRTU7	1	HREG	400007	unsignedShort
MRTU8	1	HREG	400008	unsignedShort
MRTU9	1	HREG	400009	unsignedShort
MRTU10	1	HREG	400010	unsignedShort
MRTU11	1	HREG	400011	unsignedShort
MRTU12	1	HREG	400012	unsignedShort
MRTU13	1	HREG	400013	unsignedShort
MRTU14	1	HREG	400014	unsignedShort
MRTU15	1	HREG	400015	unsignedShort
MRTU16	1	HREG	400016	unsignedShort
MRTU17	2	HREG	400017	unsignedShort

To import a new Dictionary proceed as follow in Tag Editor:

1. From top toolbar, select interested protocol
2. Click on **>]** button to call importer
3. Verify controller type and select format of file to import (.csv, .txt, .sym – protocol dependent)
4. Click **OK**
5. Select file to import

As result, a new dictionary file is added to the **Dictionaries** folder and a list of tags imported are available in the tag editor and shown at the bottom of the tag editor page. Tags shown in dictionary can be imported into the project using following:

- Import Tags (to add new tags to the project)
- Update Tags (to update tags already imported previously)

NOTE

When importing tags, the "." period is replaced with a "/" forward slash character. This is normal and the protocol will use the correct syntax when communicating to the PLC. The "." is a reserved character and cannot be used in a tag name.

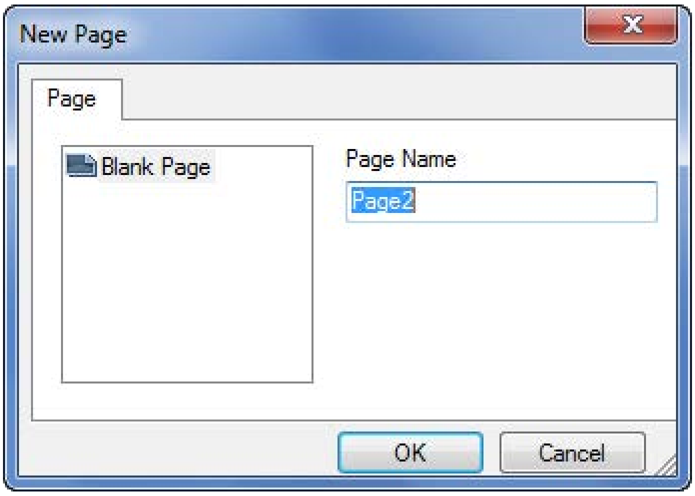
NOTE

The “&” ampersand character cannot be used in a tag name, as it can cause communication issues.

Designing a Page

When a project is created, a page is automatically added to the project and shown in the page editor.

1. To add objects to a page, simply drag and drop the objects from the widget gallery to the page.
2. To add a new page, right click on the page node from the project tree and select “Insert new page”.
   - ➤ A dialog box will appear asking for the name of the new page.



New Page

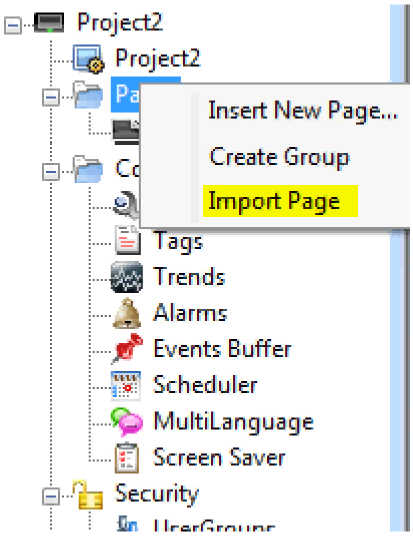
Page

Blank Page

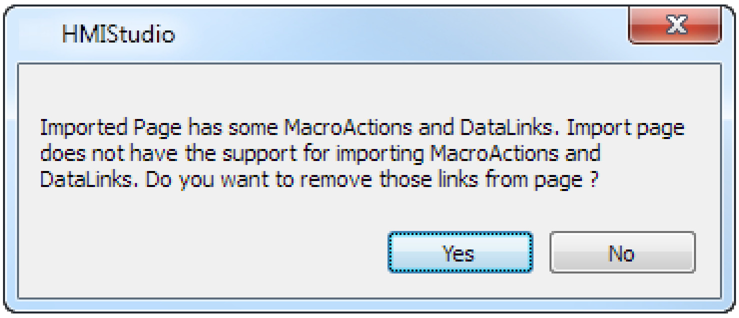
Page Name: Page2

Importing a Page

1. A page can also be imported from one project to another. By right clicking on the page folder in the project view, there is an option named “Import Page”.



2. After selecting a page to be imported from the desired project, when you click OK, you get a warning message in the editor as shown below.



Imported Page has some MacroActions and DataLinks. Import page does not have the support for importing MacroActions and DataLinks. Do you want to remove those links from page ?

Page import can support only import of page and the widgets in it, but not the macro actions and datalinks attached to the widget. By selecting “Yes” all the datalinks and the macro actions attached to the widgets will be removed. Only the widgets will remain. By selecting “No” the macro actions and the datalinks will remain attached to the widgets, but may not function properly during runtime unless the tags associated to the macros and datalinks are present or created in the new project.

NOTE The import page can be done between projects made in the same version of the software. If the versions are different then a warning message will pop up to save the project in the new version, then again try to import the page.

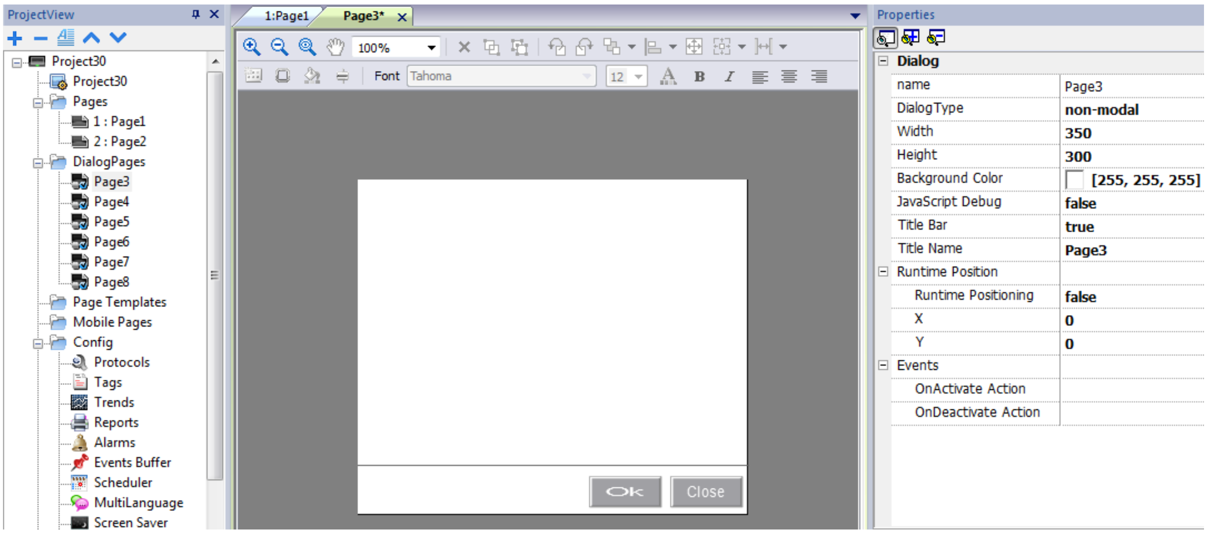
Dialog Pages

Dialog pages are windows opened at runtime on top of the current page when requested by the application.

Dialogs are used to inform user about something happening (ex. alarms/notifications/status errors) or to allow user to answer a question.

Dialog type can be defined in the property window of each dialog and can be:

- **Modals:** user cannot return to main project window/page until dialog is closed.
- **Non-Modal:** user can continue to use main project window (or other dialogs non- modal) while a dialog is shown on top of it.



Dialog	
name	Page3
DialogType	non-modal
Width	350
Height	300
Background Color	□ [255, 255, 255]
JavaScript Debug	false
Title Bar	true
Title Name	Page3
Runtime Position	
Runtime Positioning	false
X	0
Y	0
Events	
OnActivate Action	
OnDeactivate Action	

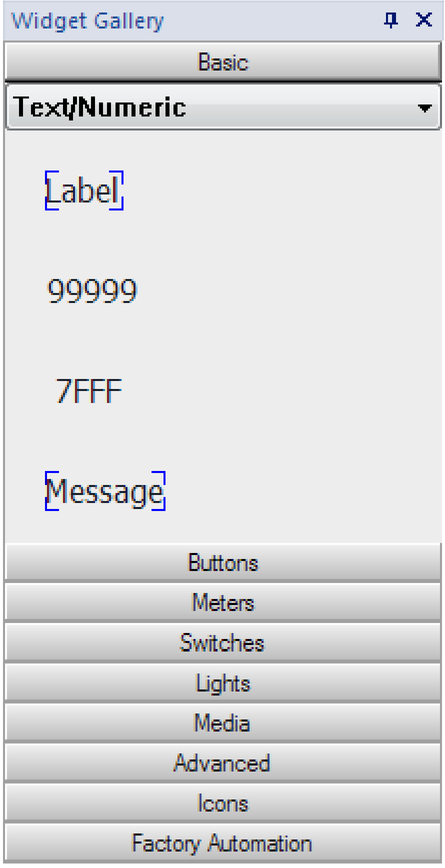
Max number of dialogs allowed is reported in **Table of functions and limits.** When max number of open dialog has been reached, the runtime will close automatically the oldest dialog open to open the new one.

A dialog can have a **Title Bar** on top of it. When Title Bar is enabled (**Title Bar = true),** a **Title Name** may be shown.

Runtime Position can be used to specify a fixed position for the dialog window.

The Widget Gallery

The gallery is next to the Property View panel and can be opened by clicking on the tab **Widget Gallery tab**.



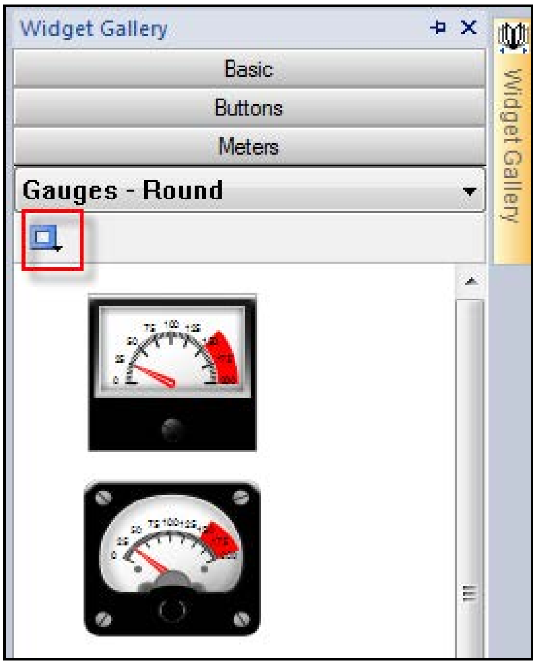
1. Select the desired object from the widget gallery and drag and drop it on to the page.
2. To change the appearance of the object, select the desired property from the property pane and change the property settings.

All the HMI objects required to make an application are available in the widget gallery. The widget gallery is accessible as a slide in pane from the right side of the workspace (as explained in the previous chapter).

The gallery is divided into several categories, each with collections of different types of objects.

3. Click on a category to display its sub-categories.

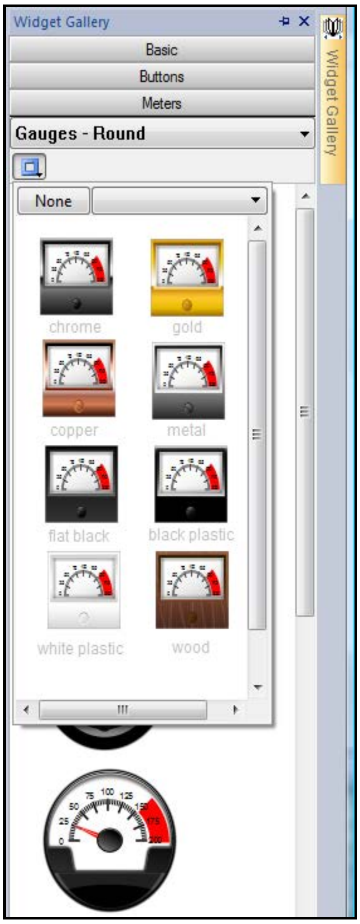
For each sub-category, the gallery offers the option of applying different styles to the objects within that category (when possible).



4. Click on the **style** button to display the available styles for the current object.
5. Select one of the available styles to apply it to the gallery objects.

   ➤ The object was inserted on the page, with the new applied style.

Once on the page, the object can still be subject to additional style changes using the page toolbar.



NOTE

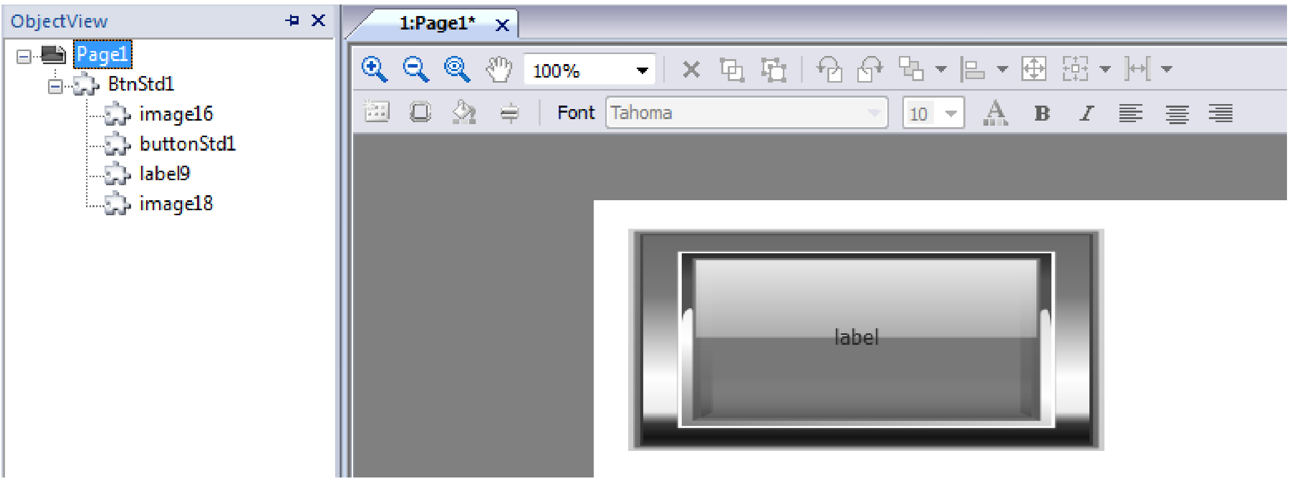
Style change may not be available for all the widgets.

Once on the page, the object can still be subject to additional style changes.
This is done using the Page Toolbar shown in the figure below.

Depending on the object selected, you can have options for the style, frame, fill color…along with the font type and size and other standard object properties

NOTE

Some widgets are composed of many sub widgets. For example a button is a complex widget composed by two Image widgets, a button widget and label. This is clearly visible in the **ObjectView** when the widget is selected. To select a sub widget like the label in a button, use ObjectView or **Shift + leftClick** of mouse. In this way sub widget can be changed without ungroup all widget.

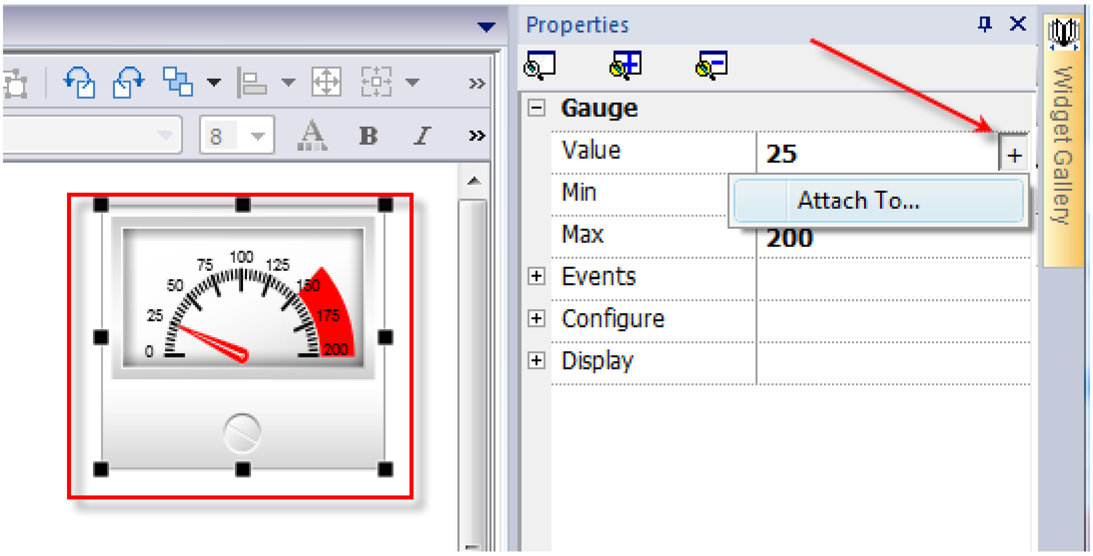


“Attach To” and Dynamic Properties

Panel Builder 600 allows for simple binding between Tags and Widget Properties. Many different Widget Properties can be attached to a Tag, which allows you to control the device and animate objects based on live data.

To attach a Tag to a property, click on the property in Property view. A **[+]** button will be displayed on the right side of the property. Click on this button and select the item **Attach To…** from the menu (as shown in the figure below).

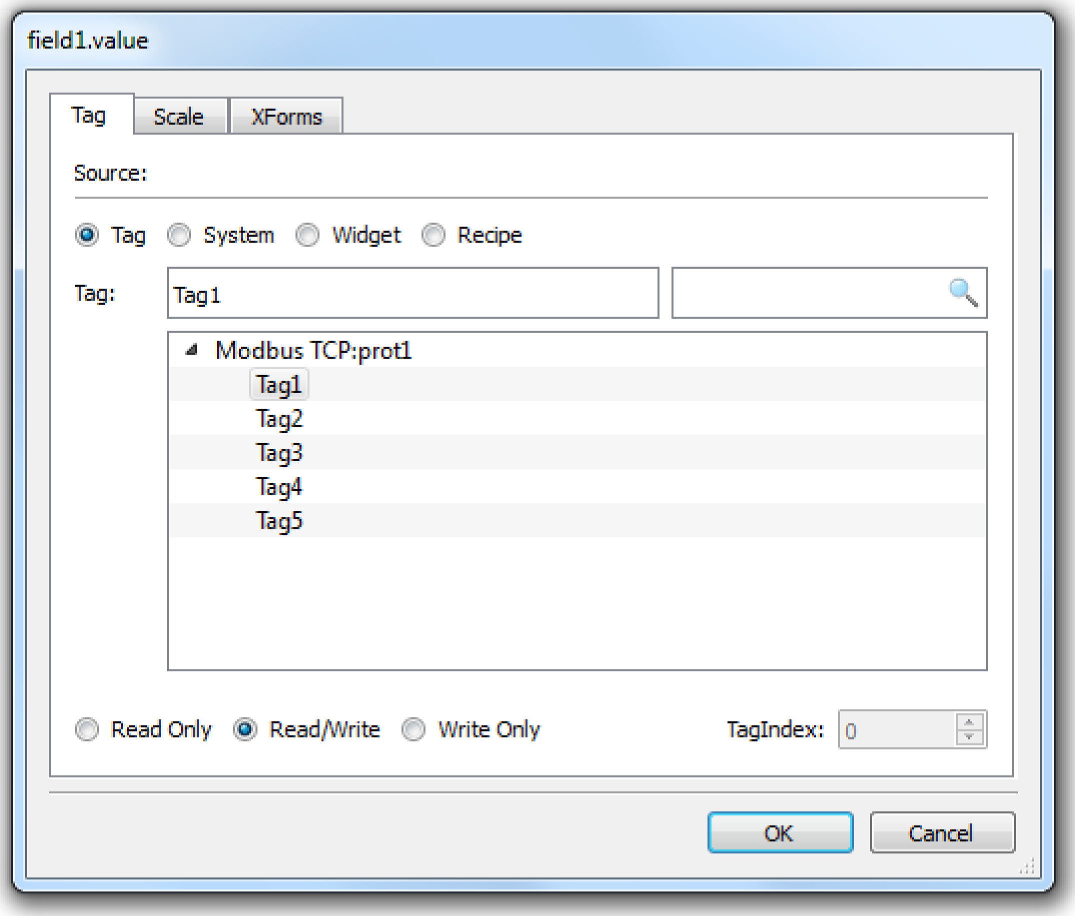
For example, when working with a gauge object, the most common action taken by the programmer is to attach a Tag to the needle, so that the value of the Tag referenced in the controller memory is represented by the needle movement.



Gauge

Value	25
Min	
Max	200
Events	
Configure	
Display	

To attach the Tag to the needle, single click on the object to display its properties in the Property view. Locate the **Value** property and click on the **[+]** button on the right part of the field as shown in above figure. Select the **Attach To…** menu item and a dialog will be displayed as shown in the figure below.



Source: ● Tag ○ System ○ Widget ○ Recipe

Tag: Tag1

- Modbus TCP:prot1
  - Tag1
  - Tag2
  - Tag3
  - Tag4
  - Tag5

○ Read Only ● Read/Write ○ Write Only TagIndex: 0

When attaching a Tag, you can attach four types of data sources:

- **Tags**: tag defined in the Tag editor
- **System**: predefined system tags (example date/time)
- **Widget**: connect to a widget property (example: value of a slider widget)
- **Recipe**: recipe data from recipe manager

Select the Tag from the Tag list and Click OK to confirm.

Tags can be attached to many different properties of the object. You can attach a Tag to a property by selecting the property in the Property view and clicking on the Attach To or you can right-click on the object and select the Attach To… menu item.

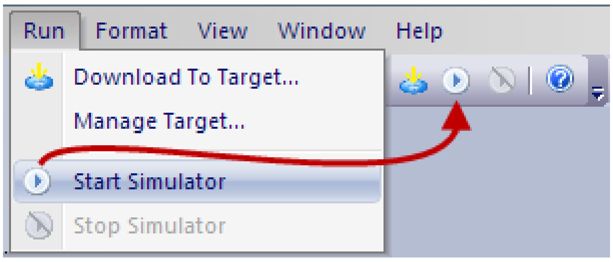
The HMI Simulator

The HMISimulator provides the facility to test the project functionality before downloading it to the panel. This feature is useful to test the project when no PLC or HMI hardware is available and to speed up the development and debugging projects

The HMI simulator supports online simulation in communication with real devices (PLC based on Ethernet or RS-232 based protocols) and offline simulation (where using Tag Editor -> Simulator field allows the configuration behavior of each tag in simulation mode).

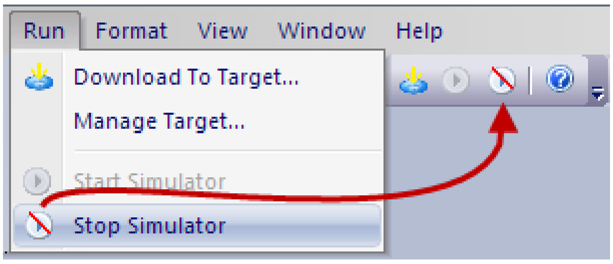
Launching the Simulator

The Simulator can be launched from Panel Builder 600 using *Run* > *Start Simulator* menu item to start it.



At this point, the simulator is running locally in the computer, similar to the way the server runs on a panel.

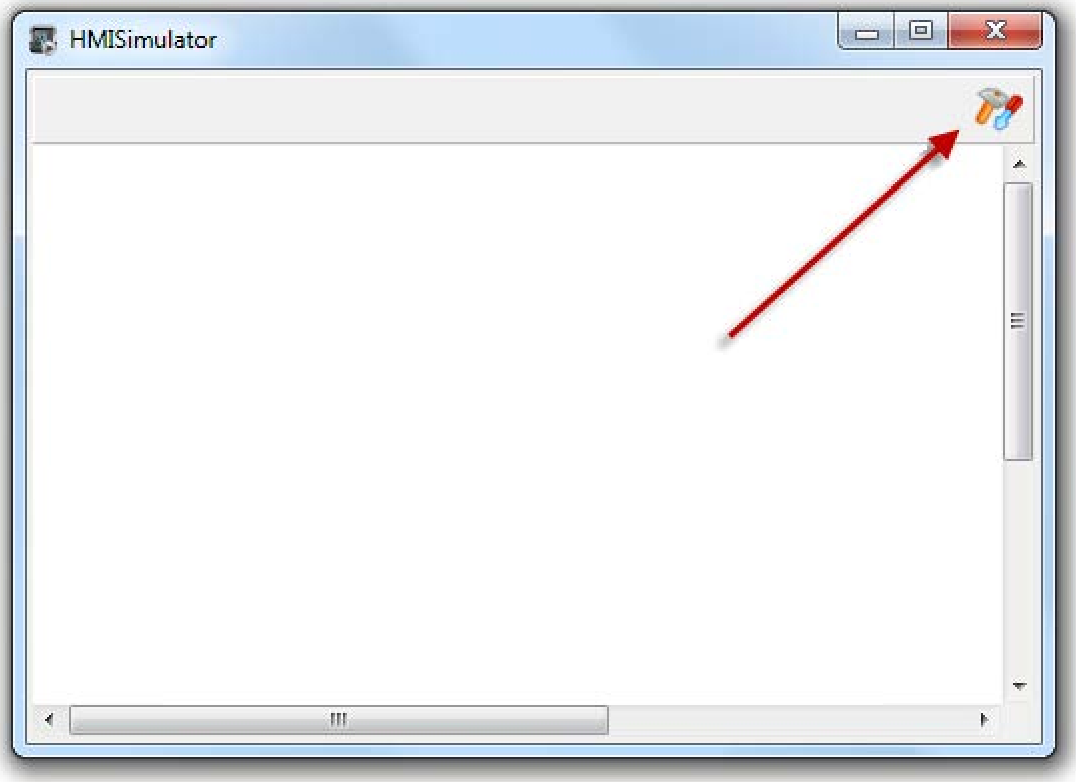
Stopping the Simulator



To stop the Simulator, select the Run >Stop Simulator menu item. You can also exit the simulator using close button of the Simulator or by using the Exit option from the contextual menu.

Simulator Settings

The Simulator can be used with real protocols and PLCs (Ethernet or RS-232 based protocols) or with simulated protocols.

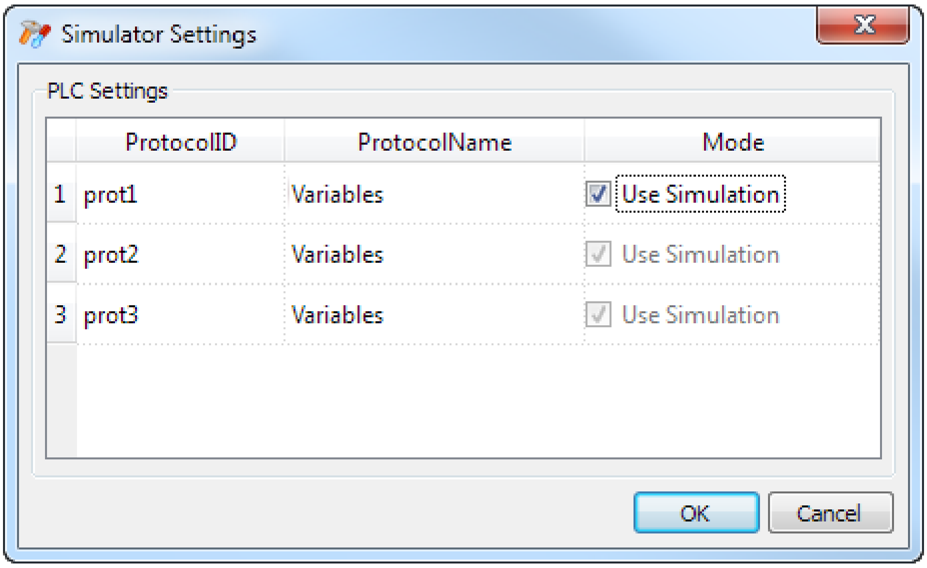


When we invoke the Simulator Settings button, a dialog showing the protocols used in the project will pop up. Users can select to use actual or simulated protocols by using the Use Simulation checkbox.

By default the Simulator uses simulated protocols defined in the Tag Editor Simulator column for each tag. Unchecking flag Mode, the simulator will communicate using real protocols with devices.

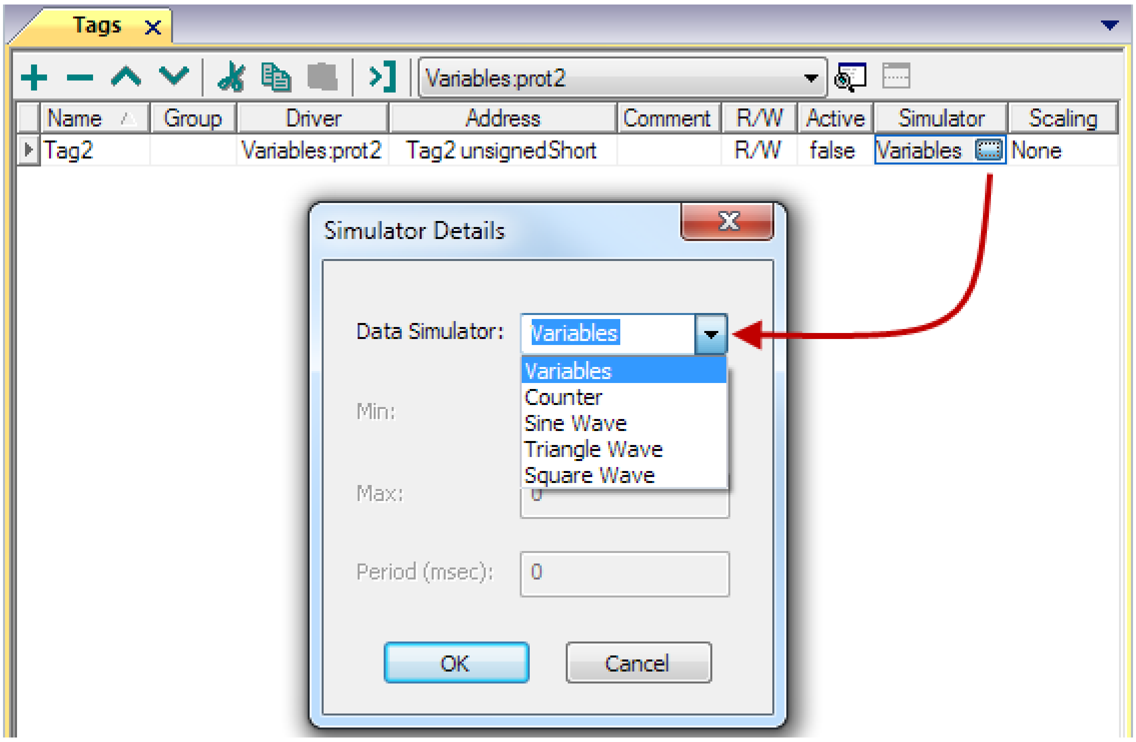
NOTE

Some protocols, for example the Variables protocol, don't support communication with devices (Win32). For these protocols this option remains disabled. Usually all protocols based on Ethernet or RS-232 can be simulated in Win32 platform or in general all protocols that do not require special hardware.



	ProtocolID	ProtocolName	Mode
1	prot1	Variables	☑ Use Simulation
2	prot2	Variables	☑ Use Simulation
3	prot3	Variables	☑ Use Simulation

When defining Tag values, the Tag Editor also includes a field to select a method for simulating the data as shown in the figure below. Tag values can be simulated in the following ways:



Name	Group	Driver	Address	Comment	R/W	Active	Simulator	Scaling
Tag2		Variables:prot2	Tag2 unsignedShort		R/W	false	Variables	None

Data Simulator: Variables

Min:

Max: 0

Period (msec): 0

Variables

The data is stored in a variable in the simulator. This variable holds the value of the Tag so the client can read and write to the Tag value.

Counters

A count value is incremented from 1 to 1000. When the counter reaches 1000, the value is reset to 0 and the counter restarts.

Sine Waves

A sine wave value is generated and written to the Tag value. The Min, Max and Period values of the Sine wave can be defined for each Tag.

Triangle Waves

A triangle wave value is generated and written to the Tag value. The Min, Max and Period values of the wave can be defined for each Tag.

Square Waves

A square wave value is generated and written to the Tag value. The Min, Max and Period values of the Sine wave can be defined for each Tag.

Transferring the Project to Target

The PB610 project can be transferred to the control panel in two ways:

1. Using the **Download to Target** item in the run menu
2. Using an **Update Package** via USB

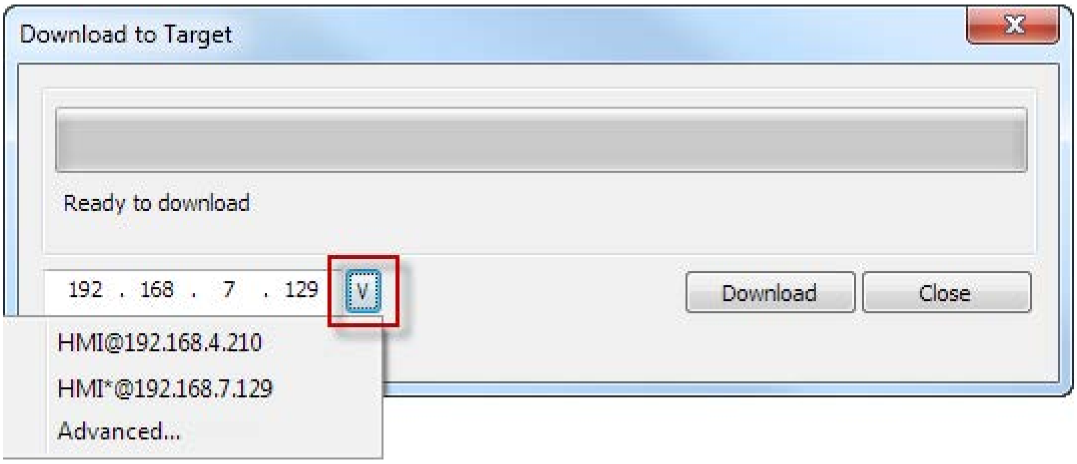
Download to Target

Run ➔ Download to Target can be used to transfer project and runtime via Ethernet from the Panel Builder 600 to the control panel.

NOTE

The panel must have assigned a valid IP address. Please see the chapter “Basic Unit Settings” for further information about how to assign an IP address to the panel.

Once the panel has a valid IP assigned, it will become discoverable on the local network. Click on the “discovery” button and select the HMI from the list of IP addresses.

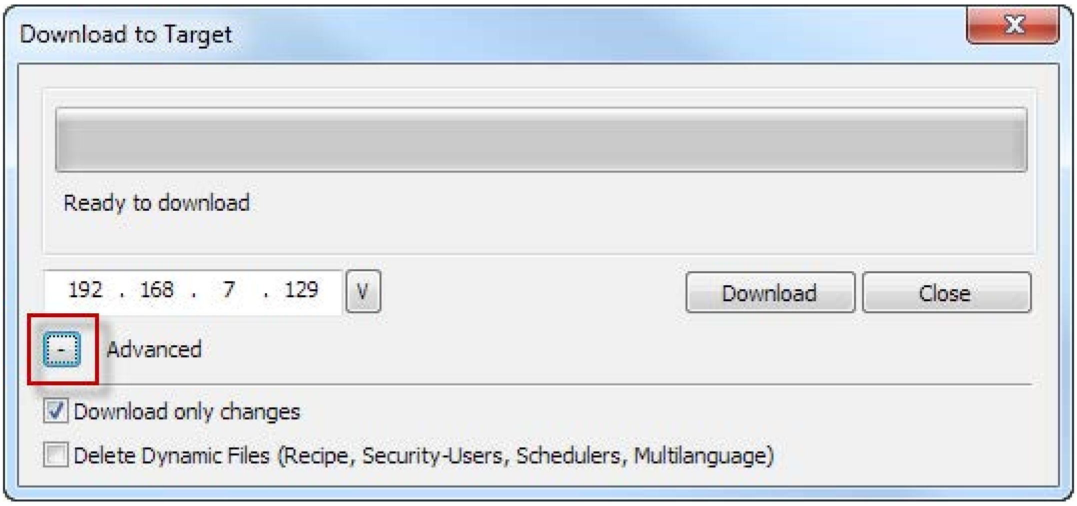


192 . 168 . 7 . 129

1. Click on the **Download** button to start the process.
   - The system will switch the target to configuration mode and transfer the files.
   - When the download operation is completed, the target is automatically switched to operation mode and the downloaded project is started.

Any time a project is changed, the modified files need to be transferred to the target device. When updating a target, PB610 Panel Builder 600 provides the option **Download only changes** to transfer only the modified files to the device.

The figure below shows this:



192 . 168 . 7 . 129

☑ Download only changes

□ Delete Dynamic Files (Recipe, Security-Users, Schedulers, Multilanguage)

Expanded Advanced options

The other option is **Delete Dynamic Files**. There are files that can be modified in the control panel at runtime, for example you can create new users at runtime or you can upload new value in the recipes. If the option to delete the files is selected, the edited configuration of the recipes, or users, or the schedulers will be deleted and overwritten by the project configuration.

NOTE

Dynamic files are not deleted using **Delete Dynamic Files** if storage type is external (example USB / SD Cards).

CAUTION!

Please make sure to check carefully before selecting this option as some data can be lost. Data cannot be restored after being deleted.

When transferring a project, Studio is using a combination of an HTTP and FTP connection. The HTTP connection is used to issue commands to the target device like “turn in transfer mode” or “unload running project”; the FTP session is instead used to transfer the files to the panel´s flash memory.

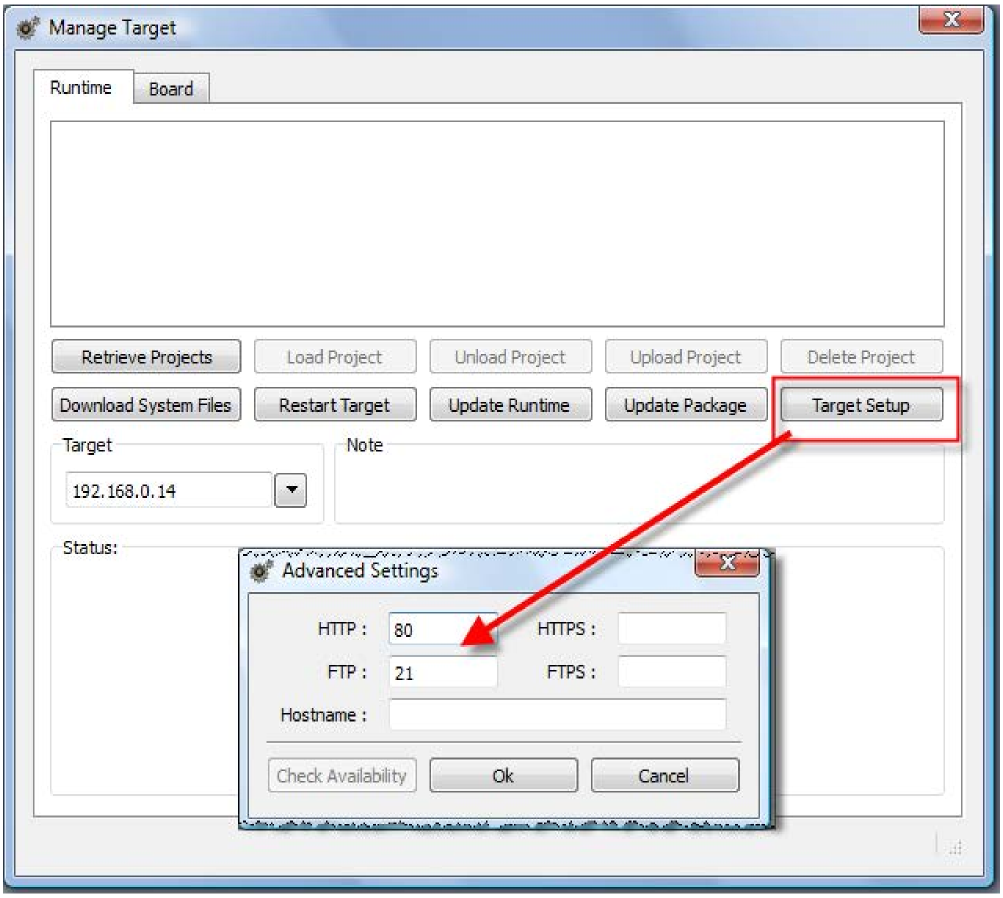
The default port for HTTP connections on the target is set as 80. However, the user can change the port number to a different value.

2. Click on the *Run* > *Manage Target* to set the port number from PB610 Panel Builder 600

3. Click on **Target Setup on the dialog:** the HTTP, FTP port or HTTPS, FTPS port can be set for the target.

The host name can be defined by the user, in the appropriate box in the target port pop-up. This will allow each panel to be easily identified on a network with multiple panels. The drop down box will no longer show HMI@10.2.0.6, but will show, for example, Machine1@10.2.0.6.

After renaming the host, it is necessary to download the system files to the target.



Target: 192.168.0.14

HTTP : 80

HTTPS :

FTP : 21

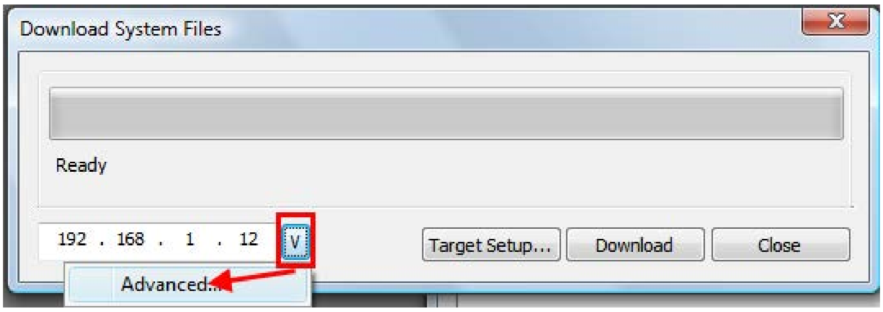
FTPS :

Hostname :

NOTE

Transferring a project after the above settings will result in a change of the default configuration. At the next download the new ports will be used on target and the new ports will have to be specified in Studio to match the new selection.

4. Click on **Advanced Menu** in the download dialog and set the port.



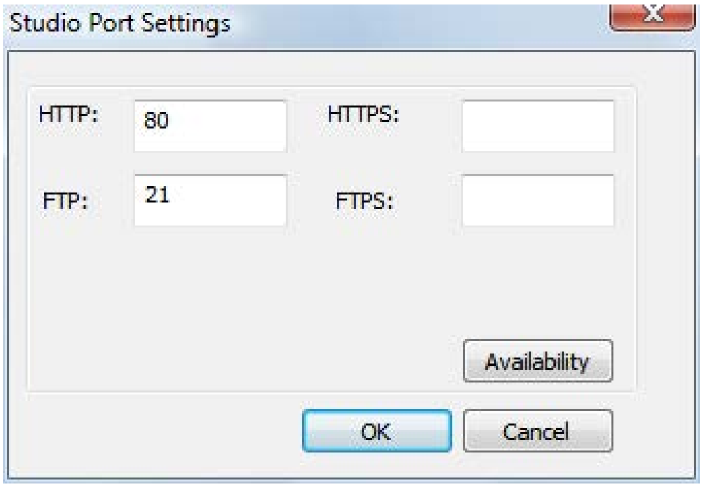
Download System Files

Ready

192 . 168 . 1 . 12

5. Set the HTTP/HTTPS port and FTP/FTPS port of the target.

They represent the port numbers Studio uses to connect to the FTP(S) and the HTTP(S) servers on target. This is useful whenever default ports are, for some reason, in use by other applications or services, or if the local network requires using different port settings.



Studio Port Settings

HTTP: 80

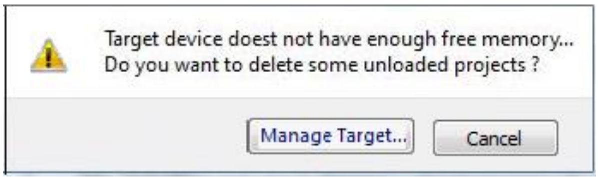
HTTPS:

FTP: 21

FTPS:

When Target Flash Memory is Low

While trying to download a project to the target if the project size is almost near or greater than the free space available in the flash memory, then it’s not possible to download the project directly. The difference of project size and available free memory should be at least 2MB.



Target device doest not have enough free memory...
Do you want to delete some unloaded projects ?

While clicking **Download**, a warning message will pop up mentioning that the target memory is low and whether you need to delete some projects as in the figure given above.

1. Click **Manage Target.**

   ➤ **Manage Target** window will open showing all the available projects in the target.

2. Delete the unwanted projects from the target to create more memory space

   ➤ Hence it is possible to download the current project.

   ➤ By pressing Cancel, the dialog will close, and the download operation is aborted.

NOTE

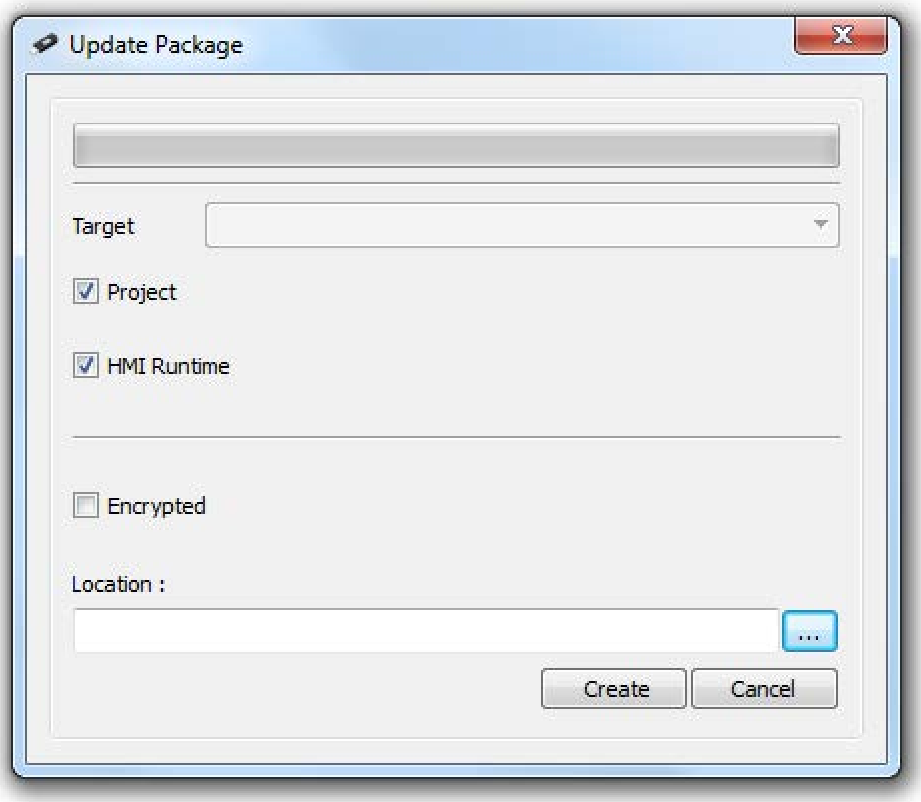
The automatic check for available space for project download is a feature present from Studio and runtime version 1.80.

Update Package

Both Runtime and project can be updated using a USB key package. To create an update package proceed as follows:

1. From the Run menu, select **Manage Target** -> **Update package**
2. Select the target and components you need to update
3. Specify the output directory - for update package (example. USB flash drive).
4. Click **Create** to generate package.

An optional flag is available to encrypt package. An encrypted package is a zip package; it cannot be read by any user and can be unzipped only by the HMI runtime.

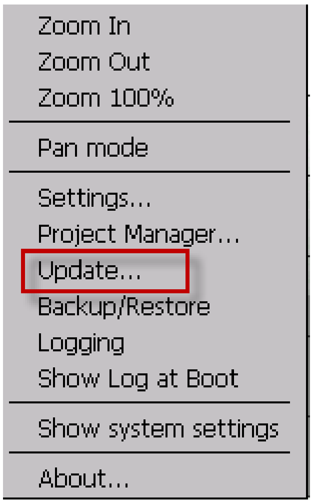


☑ Project

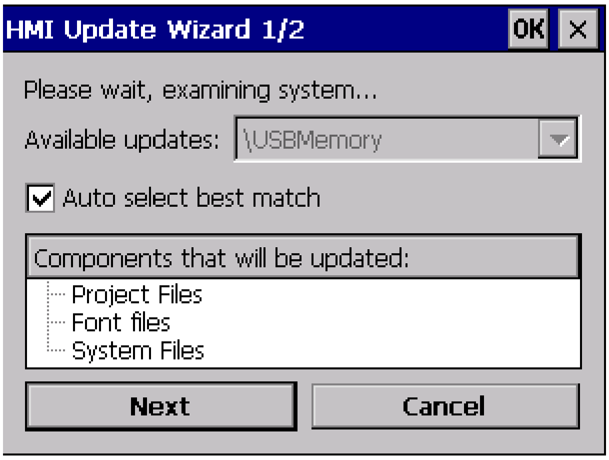
☑ HMI Runtime

□ Encrypted

Assuming you have stored the package in the root folder of a USB drive, remove the drive from the PC, plug it into the HMI, activate the context menu by holding your finger for a few seconds on the screen (see also "Basic Unit Settings")and select "Update…" as shown in the figure below.



The system will automatically check for the presence of the update package in the root of the USB drive and ask confirmation to proceed with the update according to the figure below.



Please wait, examining system...

Available updates: \USBMemory

☑ Auto select best match

Components that will be updated:

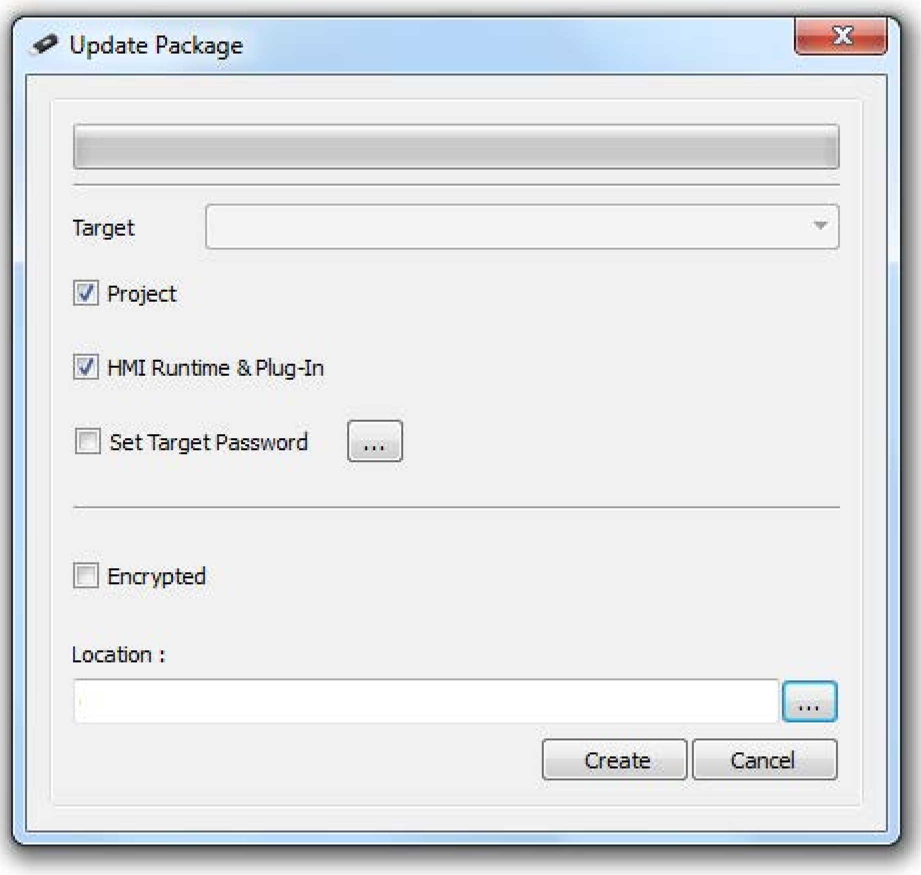
- Project Files
- Font files
- System Files

Mark the “Auto select best match” check box and click the “Next” button.

The rest is automatically done by the system.

NOTE

It’s always recommended to create update packages with both flags Project and HMI Runtime checked to keep both aligned. Use latest runtime with old projects not converted with Panel Builder 600 could create stability problems in existing projects.

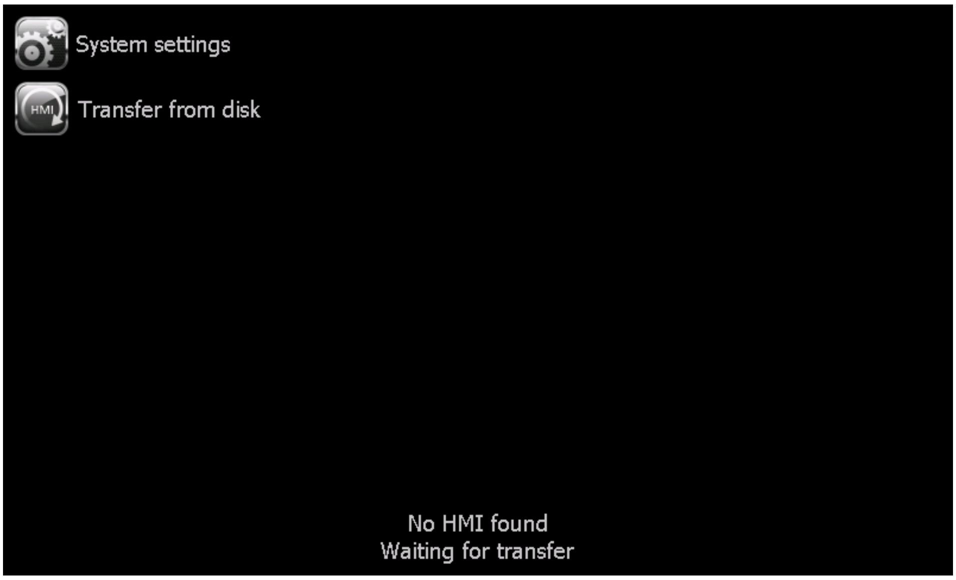


Menu entry	Explanation
Target	Target type. When a project is open the target type is selected automatically otherwise it is responsibility of the user to select the correct target type.
Project	Project opened in Panel Builder 600 is added to the update package.
HMI Runtime & Plug-In	HMI runtime is added to the update package. If a project is open in Panel Builder 600, also required plugins will be added to update package.
Set Target Password	Can be used to set password used by HMI runtime to protect operations like upload of projects, board management, download of projects, etc. Ref. to Remote access protection to HMI Panels for more information on to access protection.
Encrypted	Enable Encryption of update package; it cannot be read by any user and can be unzipped only by the HMI runtime.
Location	Path for saving the update package.

The Runtime Loader

The explanations provided in the previous chapters are valid when using a panel with runtime system already installed.

The HMI devices are delivered from the factory without the runtime. When you power up the unit for the first time, it starts with the **Runtime Loader** screen.



No HMI found
Waiting for transfer

NOTE

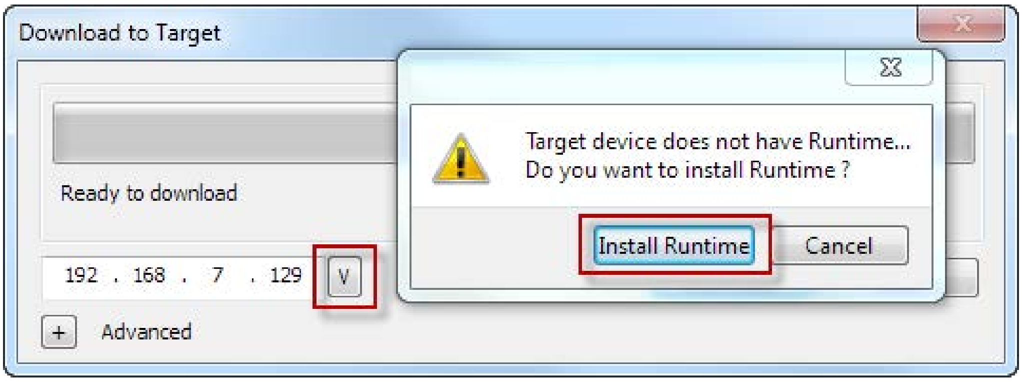
The **Runtime Loader** is a feature depending on the device operating system and may not be available on all the units. The description provided in this chapter assumes you are using Studio V1.80 or later. On MIPS based units (CP650 – CP675) the **Runtime Loader** is available from version V2.65; on ARM based units (CP620 – CP635) the **Runtime Loader** is supported from BSP version V1.52.

1. Click on **System settings** and activate the System menu in User Mode, where you can set the IP address of the panel (see the chapter "System Settings Tool" for additional information on this tool).

Once the IP address is assigned and the panel is connected to a valid network, the easiest way to install the runtime is to download a project from Studio. See the chapter "Transferring the Project to Target" for additional information.

The normal download procedure in Studio is able to recognize the needs of transferring the runtime and the process is automatically started.

As soon as the panel IP is selected from the list of available units in the network, Studio will in fact recognize the need for transferring the runtime, providing the information as shown in the following figure.



Ready to download

192 . 168 . 7 . 129

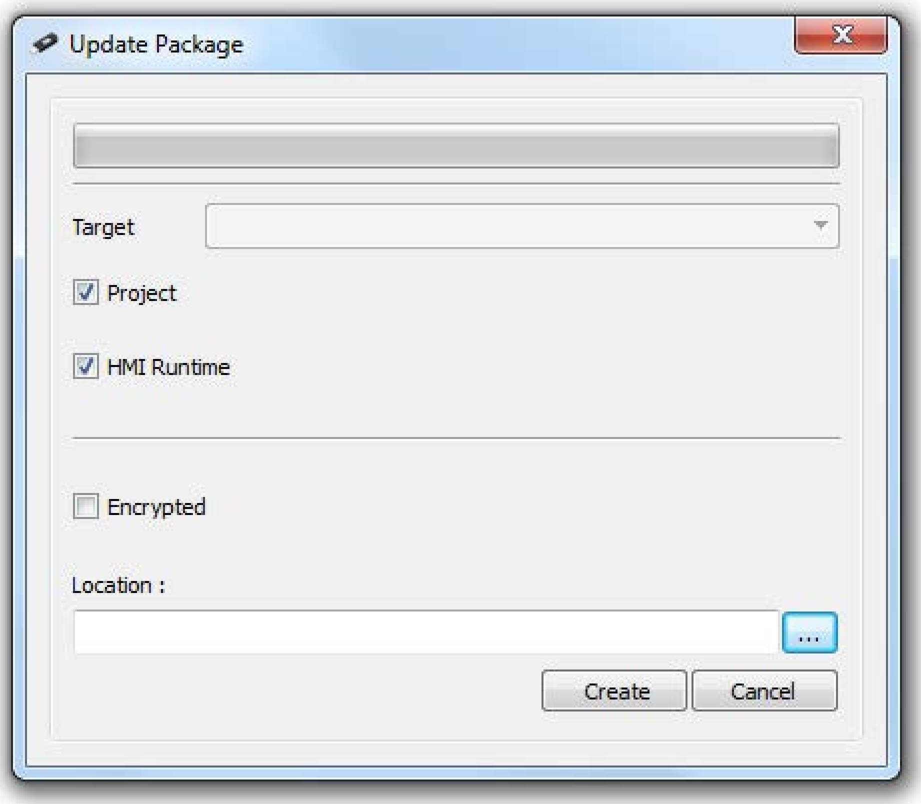
Target device does not have Runtime...
Do you want to install Runtime ?

2. Click on **Install Runtime** button to proceed.

   - The process will automatically go through the required steps, ending with the project download.

   - On an off-the-shelf unit the runtime can be installed also using an USB memory stick.

3. Prepare the **Update Package** according to the instructions provided in the chapter “Transferring the Project to Target“ and make sure to mark all the check boxes for the HMI Runtime as shown in the following figure.



Update Package

Target

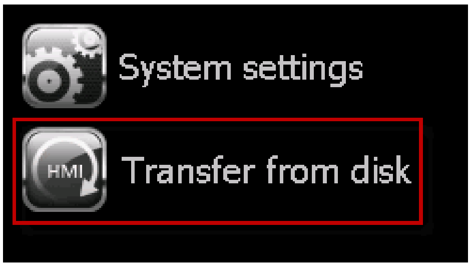
☑ Project

☑ HMI Runtime

□ Encrypted

Location :

4. Plug the USB memory stick into the panel and click on the **Transfer from disk** button.



5. Follow the instructions on the screen.

> **NOTE**
>
> The Runtime Loader on the panel does not support the automatic installation of the runtime with versions prior to 1.80; in case an older version of the runtime has to be used on a unit with the Runtime loader, please contact technical support for additional information.

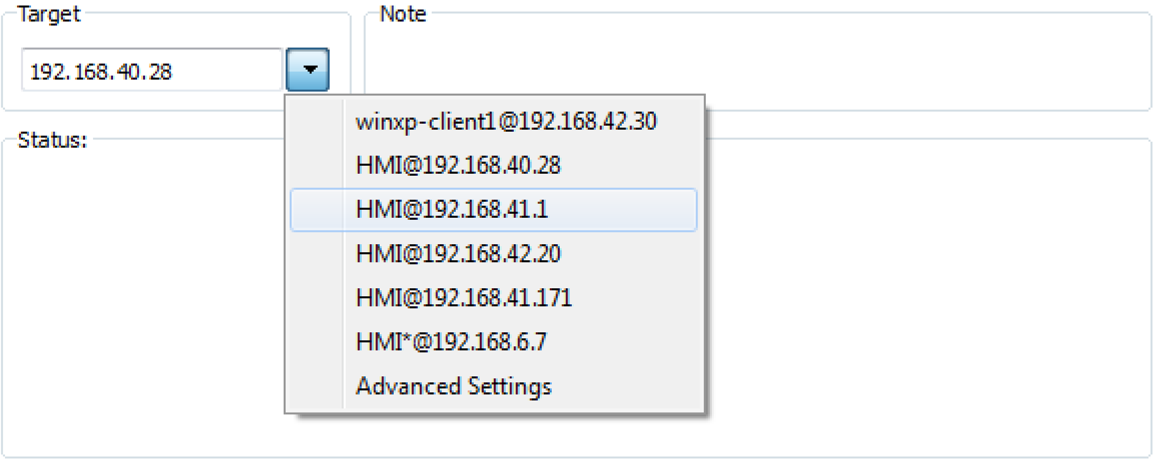
Upload Projects

You can retrieve a project from a target device using the command “Upload Project”. A copy of the project is transferred from runtime to the computer running Panel Builder 600.

To upload a project proceed as follows:

1. Run -> Manage Target.

2. In tab “Runtime”, Select IP of the device from “Target” menu.



3. Click on “Retrieve Projects” to list all projects available in the target device.
4. Select project to upload.
5. Click on “Upload Project”.
6. Enter password.
7. Upload process starts.

Once upload has completed, a copy of project is available in: C:\Users\username\Documents\Panel Builder 600\workspace\Uploaded\RuntimeIPAddress\workspace\ProjectName

Starting from Panel Builder 600 v1.90 upload is no longer based on User Management for access protection but is protected by a dedicated password scheme. Please refer to Remote access protection to HMI Panels for more information related to access protection.

My First AC500-CP600 Project

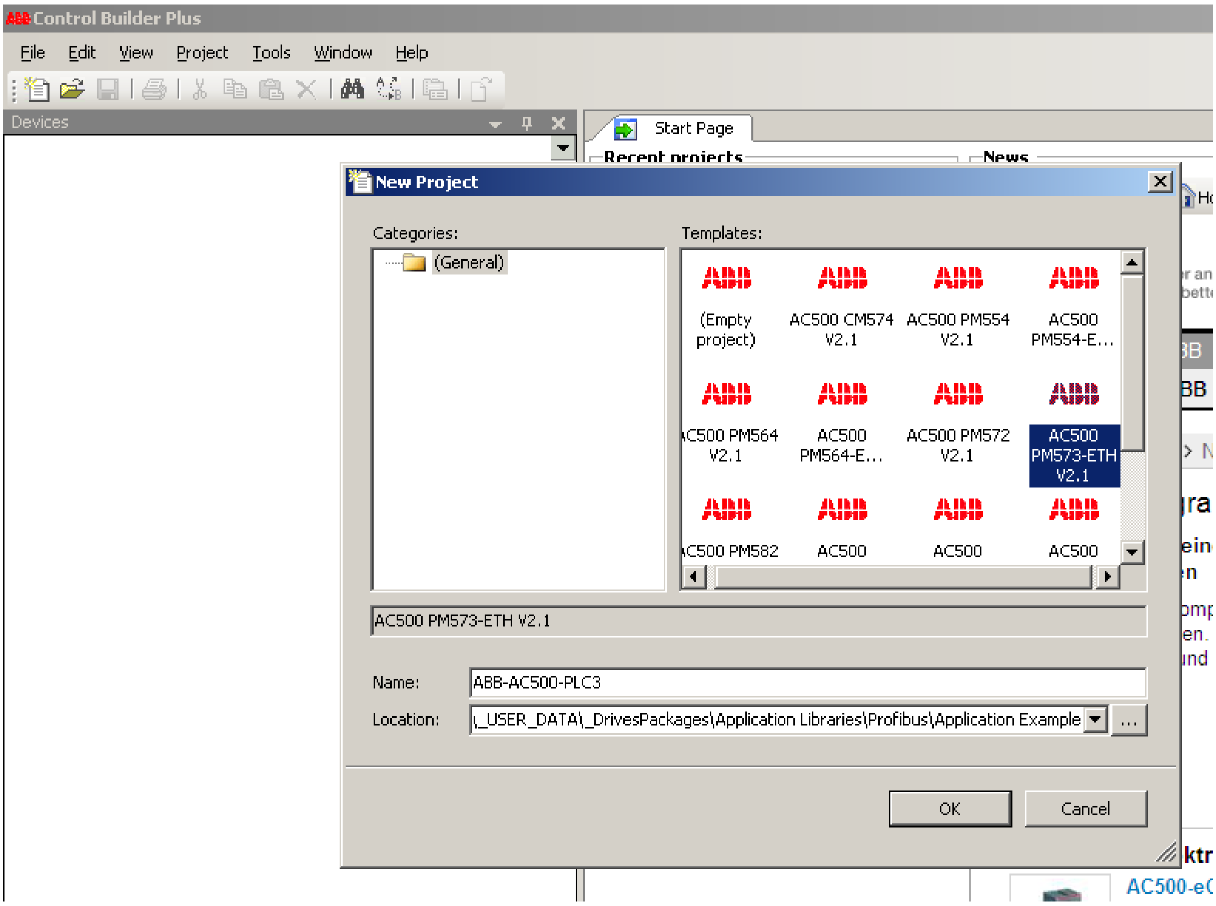
Pre-Conditions

The following pre-conditions have to be fulfilled for the communication between an AC500 PLC and a CP600 Control Panel via CoDeSys drivers:

- Hardware
  - CP600 series (not CP6xx-WEB)
  - ABB PM5xx-ETH as of firmware V2.x
- Software
  - PB610 Panel Builder 600 as of version V1.5
  - ABB Configurator PS501 as of version V2.0.0
  - AC500 Control Builder Plus as of version V2.x

Step 1: Basic Settings in the ABB Control Builder Plus

1. Start a new project.
2. Select the AC500 type you want to use, e.g. PM573-ETH V2.1.
3. Click **OK** to confirm your selection.



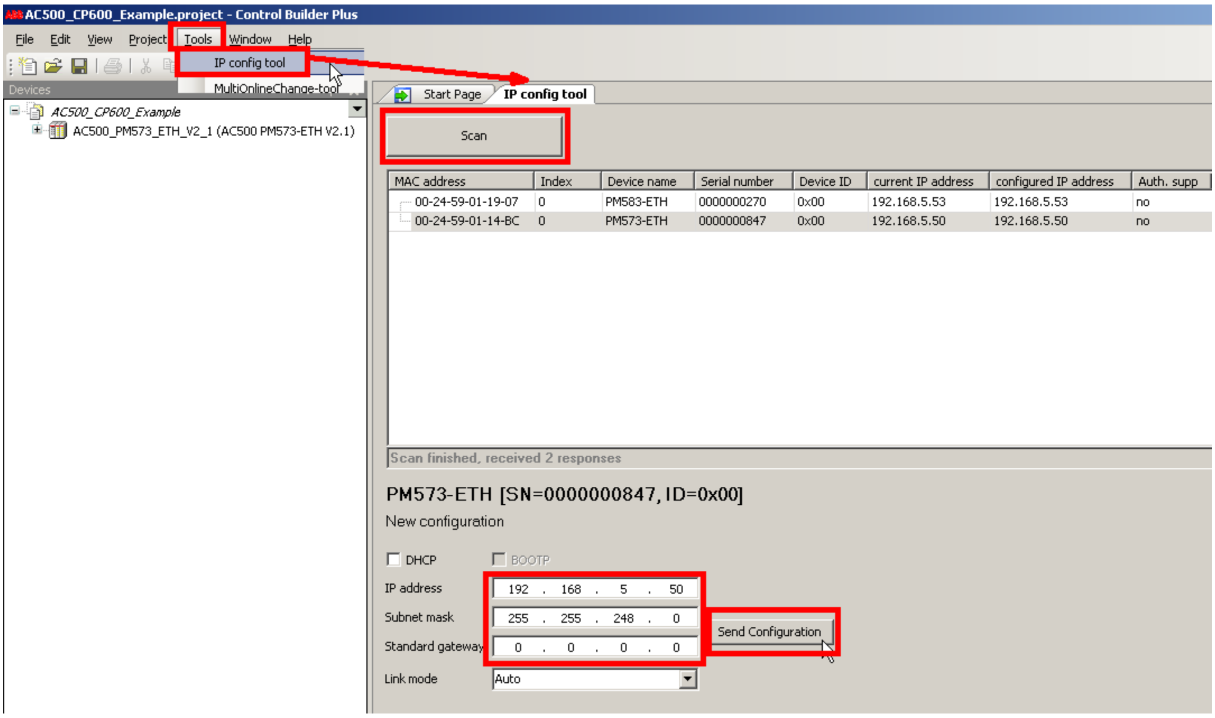
AC500 PM573-ETH V2.1

Name: ABB-AC500-PLC3

Location: _USER_DATA_DrivesPackages\Application Libraries\Profibus\Application Example

4. Start the ABB IP configuration tool.

5. Click **Scan** to start the scanning for the PLCs that are connected.



MAC address	Index	Device name	Serial number	Device ID	current IP address	configured IP address	Auth. supp
00-24-59-01-19-07	0	PM583-ETH	0000000270	0x00	192.168.5.53	192.168.5.53	no
00-24-59-01-14-BC	0	PM573-ETH	0000000847	0x00	192.168.5.50	192.168.5.50	no

Scan finished, received 2 responses

PM573-ETH [SN=0000000847, ID=0x00]

New configuration

□ DHCP □ BOOTP

IP address 192 . 168 . 5 . 50

Subnet mask 255 . 255 . 248 . 0

Standard gateway 0 . 0 . 0 . 0

Link mode Auto

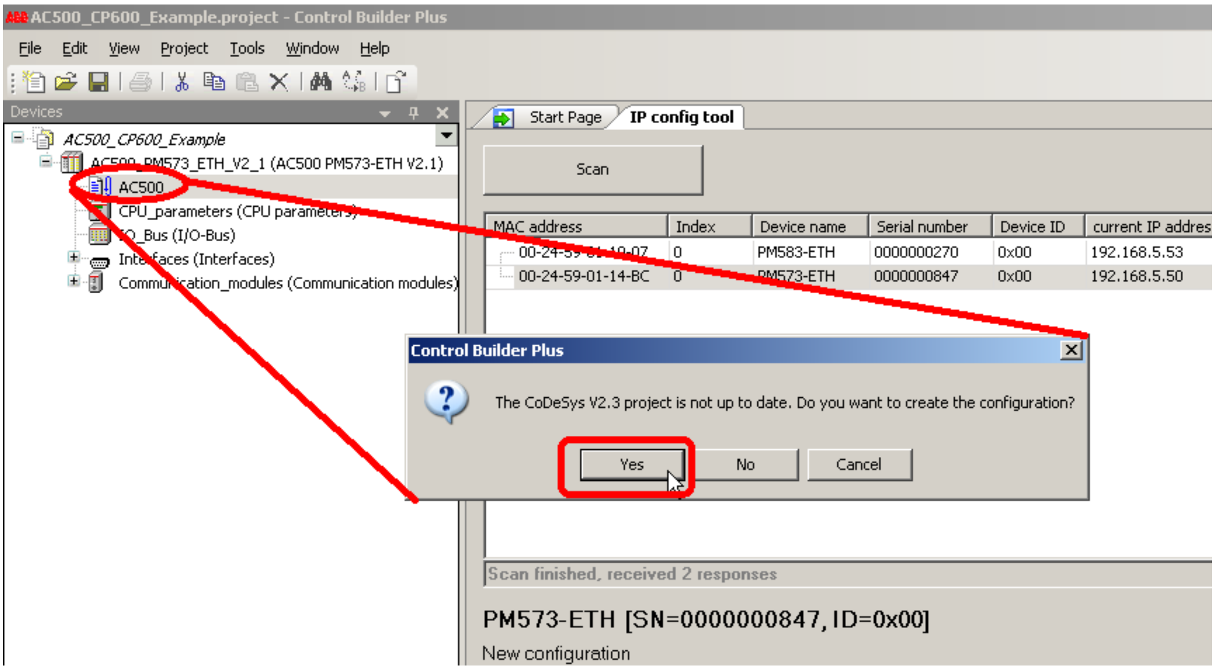
- ➤ Your PLCs will be displayed in the list. Click on the PLC you want to configure.

6. Set the IP address, the subnet mask and the IP address of your gateway.

7. Click the button **Send Configuration** to put the settings into your PLC.

- ➤ The PLC will restart automatically.

8. Double click on the menu item, marked by the red ellipse, to create the basic structure and the configuration files for the ABB *Control Builder Plus*


AC500_CP600_Example.project - Control Builder Plus
File
Edit
View
Project
Tools
Window
Help
Devices
AC500_CP600_Example
AC500_PM573_ETH_V2_1 (AC500 PM573-ETH V2.1)
AC500
CPU_parameters (CPU parameters)
IO_Bus (I/O-Bus)
Interfaces (Interfaces)
Communication_modules (Communication modules)
Start Page
IP config tool
Scan
Control Builder Plus
Yes
No
Cancel

MAC address	Index	Device name	Serial number	Device ID	current IP addres
00-24-59-01-19-07	0	PM583-ETH	0000000270	0x00	192.168.5.53
00-24-59-01-14-BC	0	PM573-ETH	0000000847	0x00	192.168.5.50

The CoDeSys V2.3 project is not up to date. Do you want to create the configuration?

Scan finished, received 2 responses

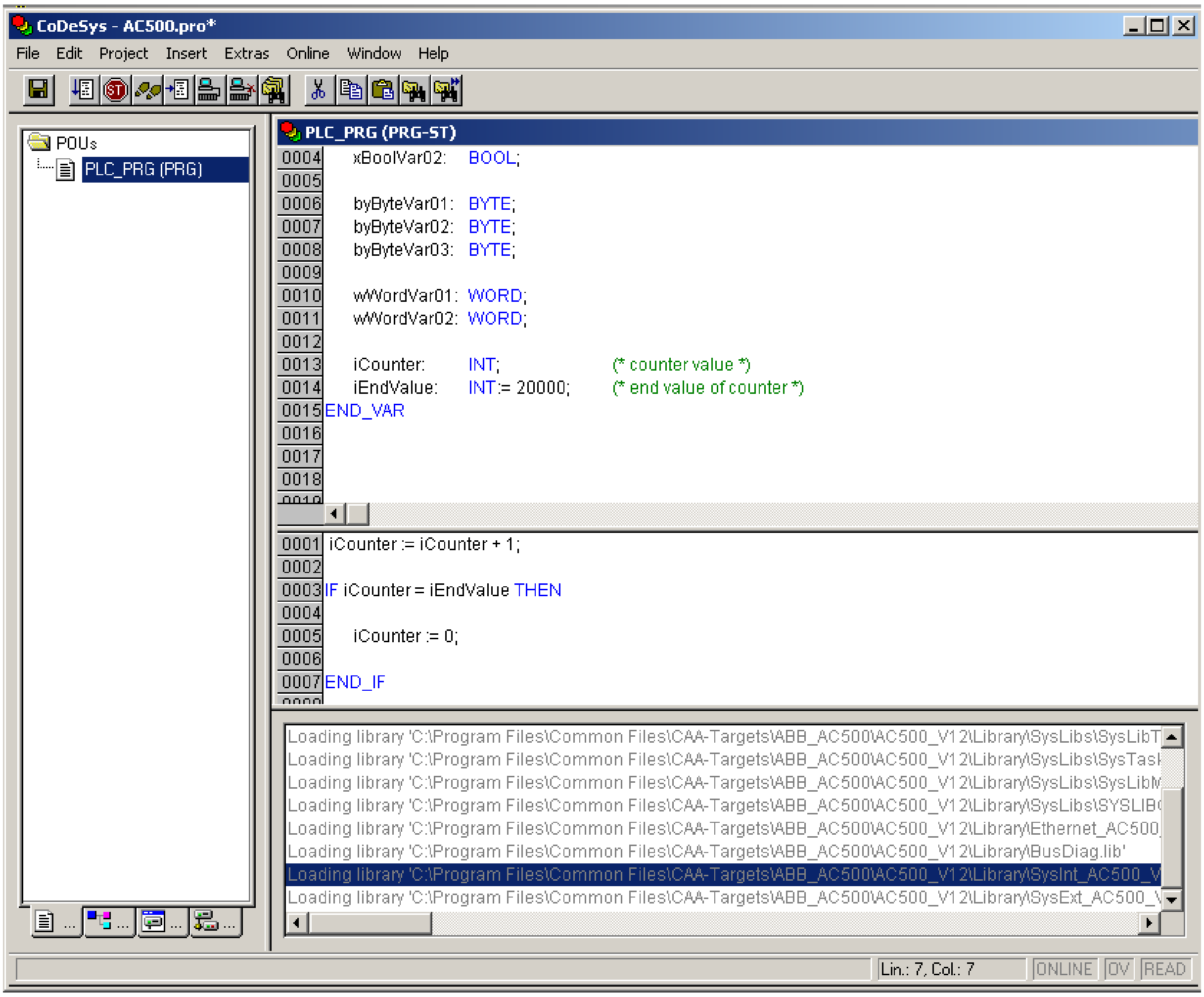
PM573-ETH [SN=0000000847, ID=0x00]

New configuration

- ➤ The CoDeSys programming pool starts automatically.

Step 2: CoDeSys Settings and a Small Test Project

The following example shows you some variables and a small test project. Also define some variables. A test project is not necessary for the communication start-up, but it demonstrates better changes of variables.

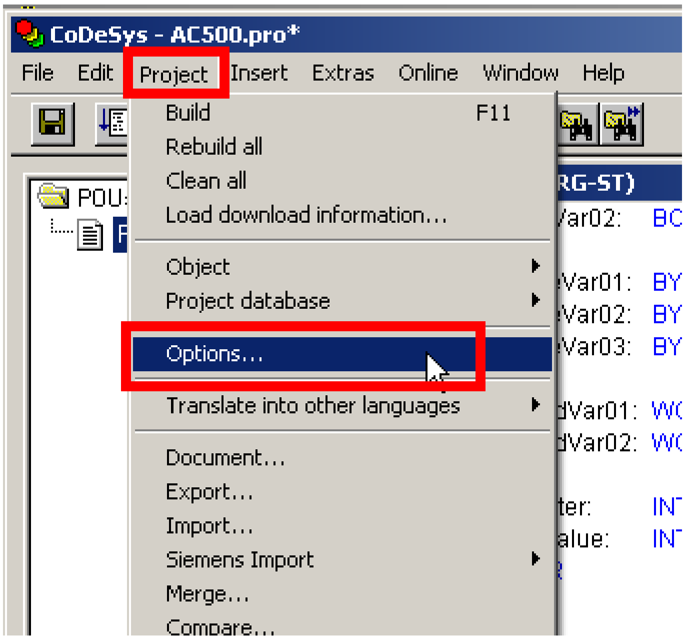


```
    xBoolVar02:   BOOL;

    byByteVar01:  BYTE;
    byByteVar02:  BYTE;
    byByteVar03:  BYTE;

    wWordVar01:  WORD;
    wWordVar02:  WORD;

    iCounter:     INT;           (* counter value *)
    iEndValue:    INT := 20000;  (* end value of counter *)
END_VAR
```

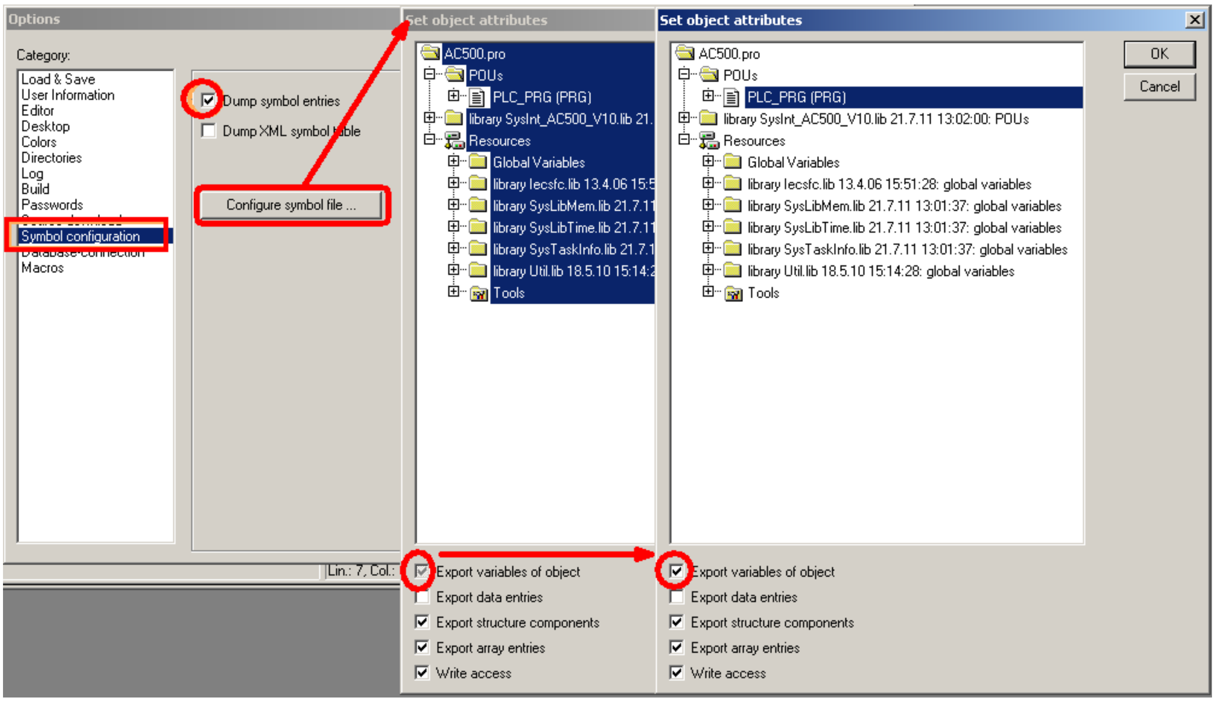
```
iCounter := iCounter + 1;

IF iCounter = iEndValue THEN

    iCounter := 0;

END_IF
```

```
Loading library 'C:\Program Files\Common Files\CAA-Targets\ABB_AC500\AC500_V1 2\Library\SysLibs\SysLibT
Loading library 'C:\Program Files\Common Files\CAA-Targets\ABB_AC500\AC500_V1 2\Library\SysLibs\SysTask
Loading library 'C:\Program Files\Common Files\CAA-Targets\ABB_AC500\AC500_V1 2\Library\SysLibs\SysLibM
Loading library 'C:\Program Files\Common Files\CAA-Targets\ABB_AC500\AC500_V1 2\Library\SysLibs\SYSLIBC
Loading library 'C:\Program Files\Common Files\CAA-Targets\ABB_AC500\AC500_V1 2\Library\Ethernet_AC500
Loading library 'C:\Program Files\Common Files\CAA-Targets\ABB_AC500\AC500_V1 2\Library\BusDiag.lib'
Loading library 'C:\Program Files\Common Files\CAA-Targets\ABB_AC500\AC500_V1 2\Library\SysInt_AC500_V
Loading library 'C:\Program Files\Common Files\CAA-Targets\ABB_AC500\AC500_V1 2\Library\SysExt_AC500_V
```

1. Open the menu *Project/Options*.



2. Click on **Symbol configuration**.


Options
Category:
Load & Save
User Information
Editor
Desktop
Colors
Directories
Log
Build
Passwords
Symbol configuration
Database-connection
Macros
Configure symbol file ...
Lin.: 7, Col.:
Set object attributes
AC500.pro
POUs
PLC_PRG (PRG)
library SysInt_AC500_V10.lib 21.7.11 13:02:00: POUs
Resources
Global Variables
library Iecsfc.lib 13.4.06 15:51:28: global variables
library SysLibMem.lib 21.7.11 13:01:37: global variables
library SysLibTime.lib 21.7.11 13:01:37: global variables
library SysTaskInfo.lib 21.7.11 13:01:37: global variables
library Util.lib 18.5.10 15:14:28: global variables
Tools
OK
Cancel

☑ Dump symbol entries
☐ Dump XML symbol table

☑ Export variables of object
☐ Export data entries
☑ Export structure components
☑ Export array entries
☑ Write access

3. Activate the checkbox **Dump symbol entries**.

4. Click the button **Configure symbol file** to open the menu for setting the object attributes.

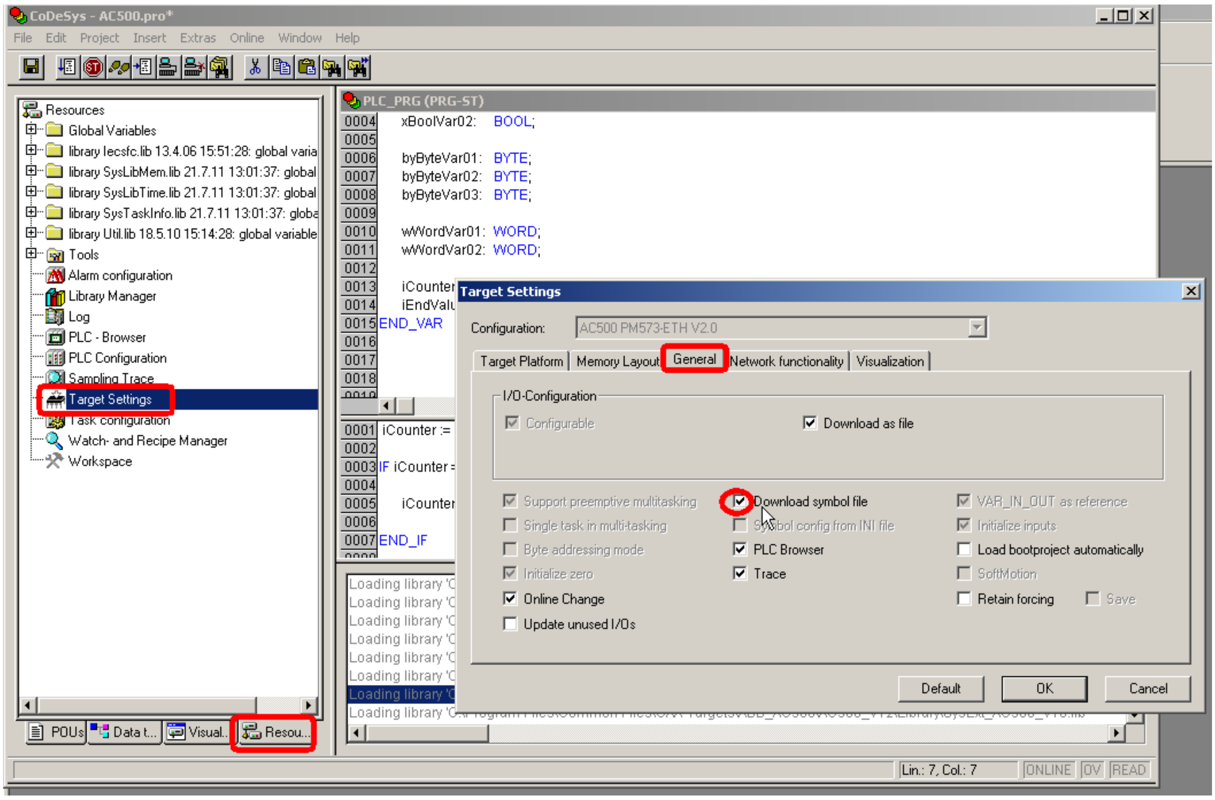
5. Select the program or variable list that should be accessible in the CP600 Control Panel.

6. The checkbox **Export variables of object** has a grey background. Double left click on the checkbox unit it is black on white background.

7. Click on **OK** to confirm your selection and leave the options.

8. Click on the tab *Resources* in the project manager.

9. Open the **Target settings**.



```
0004    xBoolVar02:   BOOL;
0005
0006    byByteVar01:  BYTE;
0007    byByteVar02:  BYTE;
0008    byByteVar03:  BYTE;
0009
0010    wWordVar01:  WORD;
0011    wWordVar02:  WORD;
0012
0013    iCounter
0014    iEndValu
0015 END_VAR
```

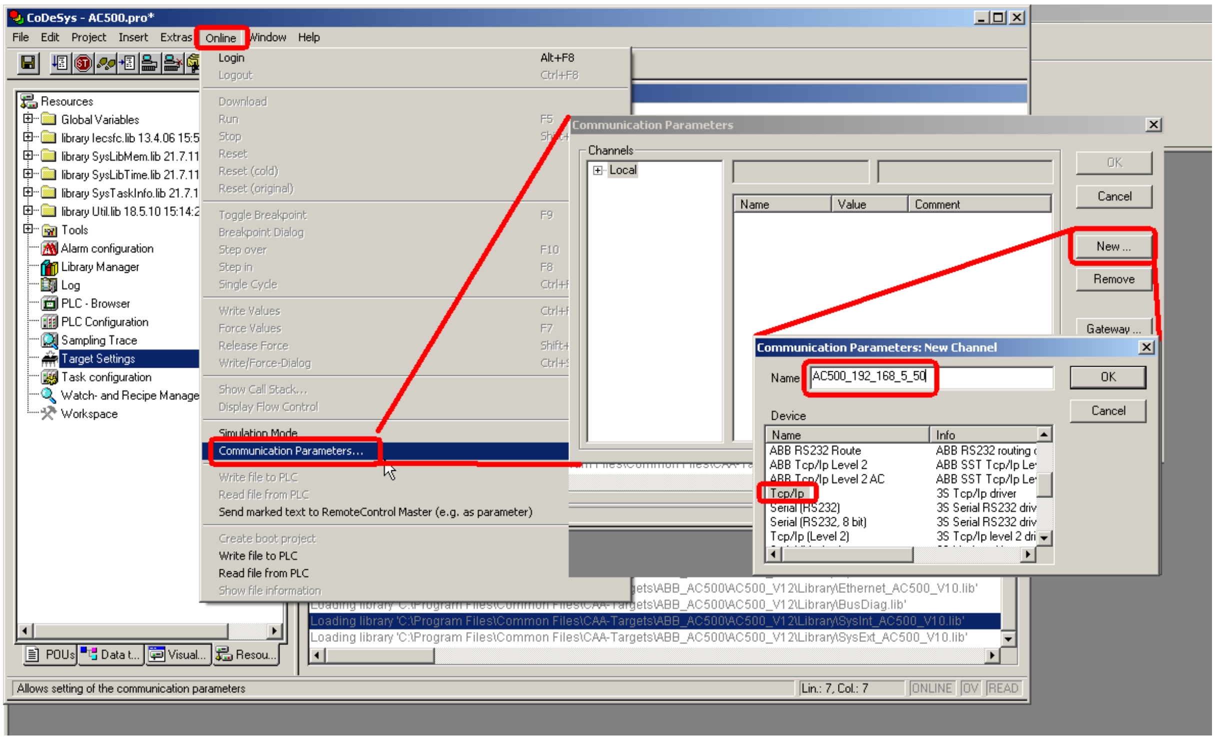
```
0001 iCounter :=
0002
0003 IF iCounter =
0004
0005    iCounter
0006
0007 END_IF
```

Configuration: AC500 PM573-ETH V2.0

I/O-Configuration
☑ Configurable
☑ Download as file

☑ Support preemptive multitasking
☐ Single task in multi-tasking
☐ Byte addressing mode
☑ Initialize zero
☑ Online Change
☐ Update unused I/Os
☑ Download symbol file
☐ Symbol config from INI file
☑ PLC Browser
☑ Trace
☑ VAR_IN_OUT as reference
☑ Initialize inputs
☐ Load bootproject automatically
☐ SoftMotion
☐ Retain forcing
☐ Save

10. Open the tab *General*.
11. Activate the checkbox **Download symbol file**.
12. Click **OK** to confirm your selection.
13. Open the menu *Online/Communication Parameters*.



Name: AC500_192_168_5_50

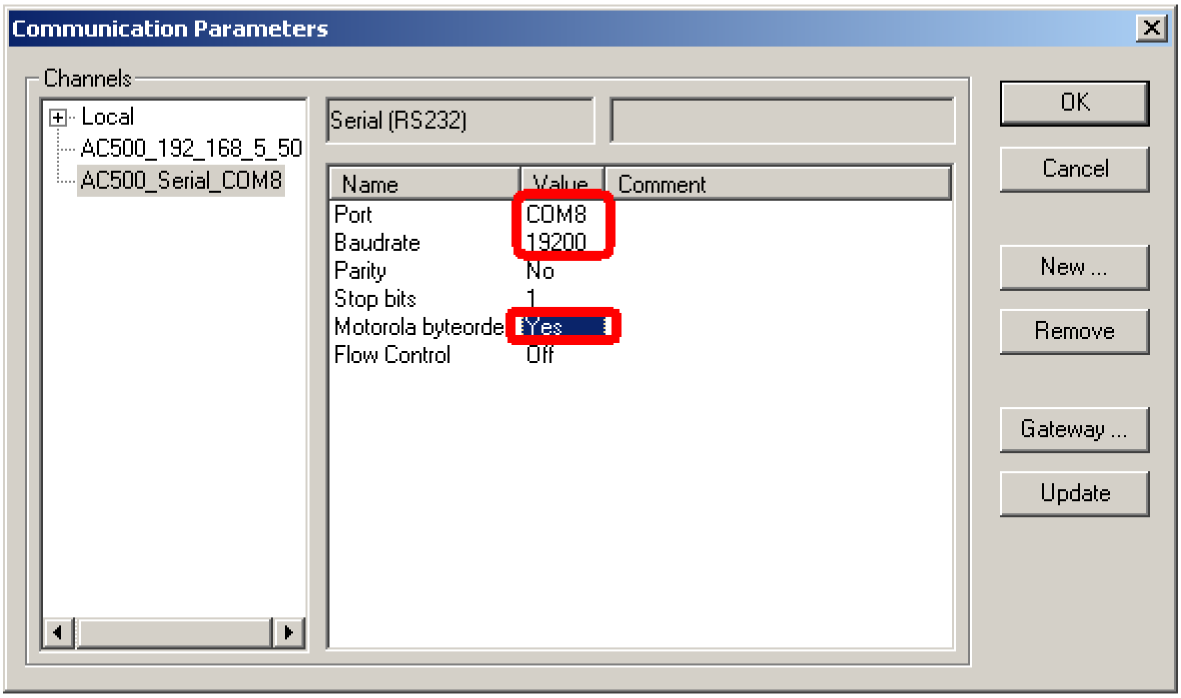
Device

Name	Info
ABB RS232 Route	ABB RS232 routing
ABB Tcp/Ip Level 2	ABB SST Tcp/Ip Le
ABB Tcp/Ip Level 2 AC	ABB SST Tcp/Ip Le
Tcp/Ip	3S Tcp/Ip driver
Serial (RS232)	3S Serial RS232 driv
Serial (RS232, 8 bit)	3S Serial RS232 driv
Tcp/Ip (Level 2)	3S Tcp/Ip level 2 dri

```
...ets\ABB_AC500\AC500_V12\Library\Ethernet_AC500_V10.lib'
Loading library 'C:\Program Files\Common Files\CAA-Targets\ABB_AC500\AC500_V12\Library\BusDiag.lib'
Loading library 'C:\Program Files\Common Files\CAA-Targets\ABB_AC500\AC500_V12\Library\SysInt_AC500_V10.lib'
Loading library 'C:\Program Files\Common Files\CAA-Targets\ABB_AC500\AC500_V12\Library\SysExt_AC500_V10.lib'
```

Set the communication parameters of the programming device to the PLC as follows.

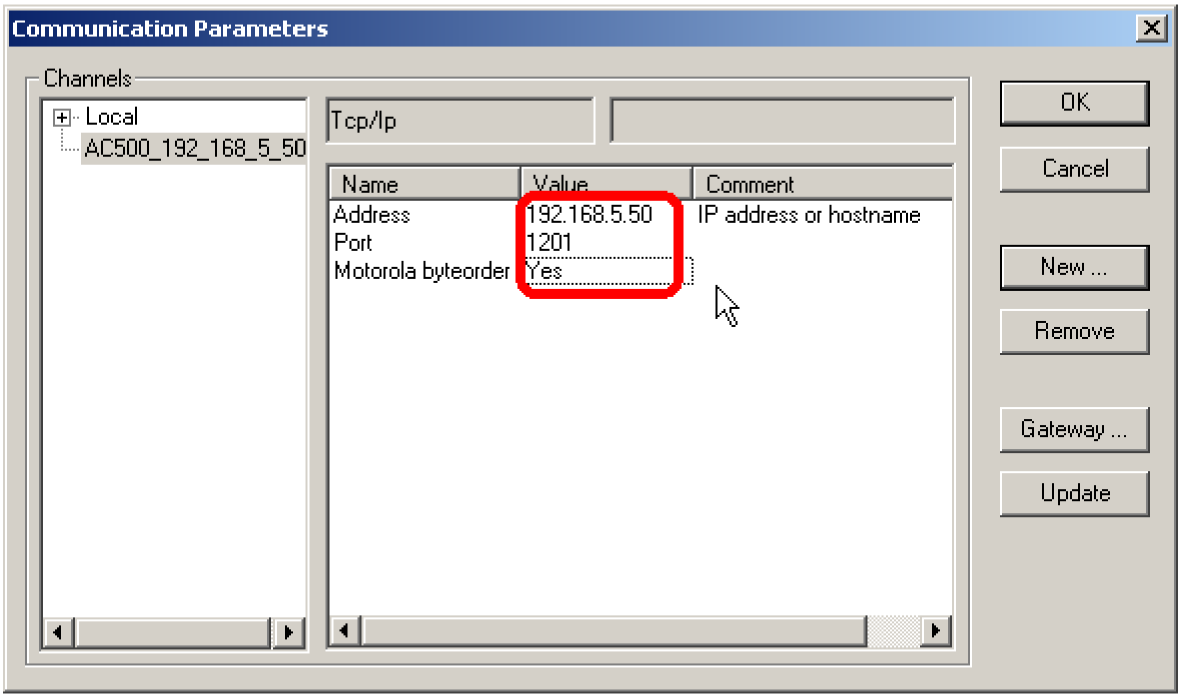
14. Click **New**.
15. Change the name of the communication and select the driver for the communication. Click **OK**.
16. To change the settings double click on the contents of the fields and change the value.



Serial (RS232)

Name	Value	Comment
Port	COM8	
Baudrate	19200	
Parity	No	
Stop bits	1	
Motorola byteorde	Yes	
Flow Control	Off	

Settings for serial communications.



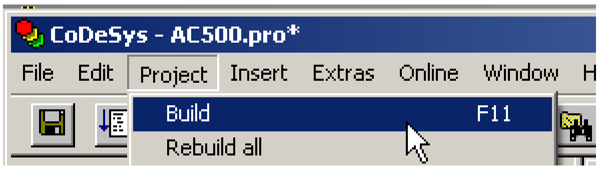
Local
AC500_192_168_5_50

Tcp/Ip

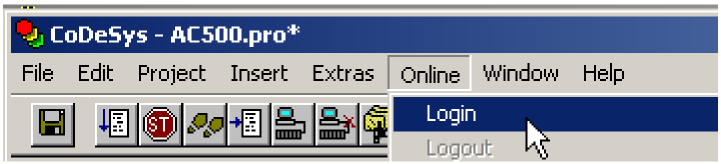
Name	Value	Comment
Address	192.168.5.50	IP address or hostname
Port	1201	
Motorola byteorder	Yes	

Settings for networks.

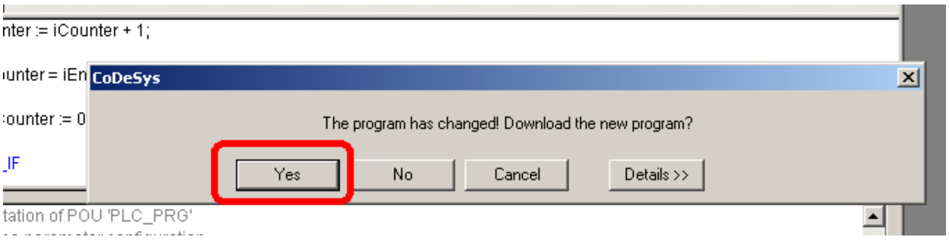
17. Compile your project:



18. Log in your PLC with the last communication parameters you have selected.



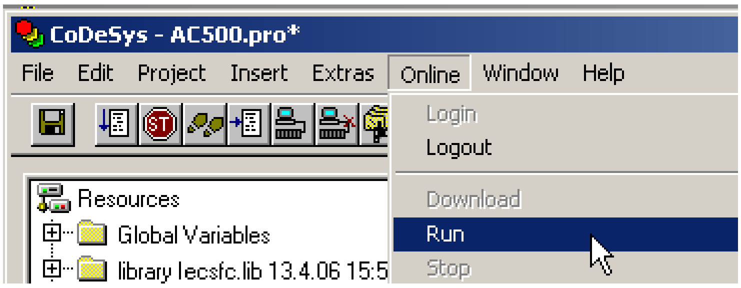
19. Upload your program to the PLC and confirm the following question with **Yes**.


```
nter := iCounter + 1;
ıunter = iEn
:ounter := 0
_IF
tation of POU 'PLC_PRG'
```

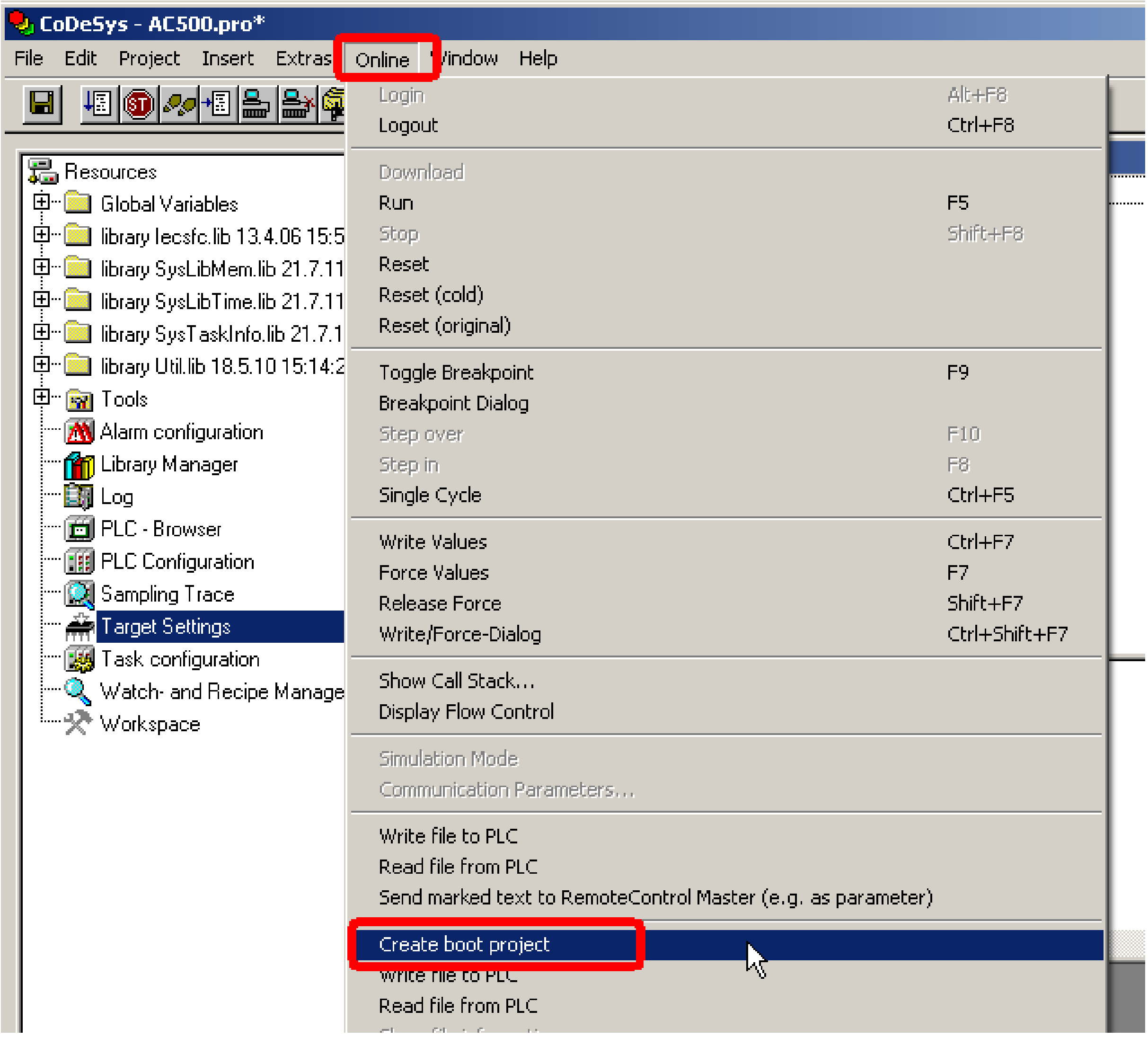
CoDeSys

The program has changed! Download the new program?

20. Start the project in your PLC.


CoDeSys - AC500.pro*
File Edit Project Insert Extras Online Window Help
Login
Logout
Download
Run
Stop
Resources
Global Variables
library Iecsfc.lib 13.4.06 15:5

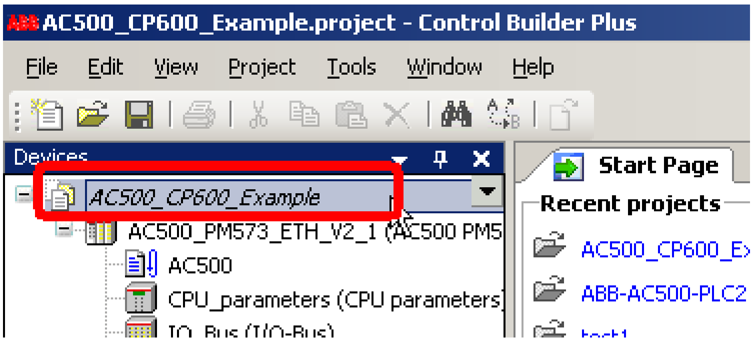
21. Create a boot project to permanently save the program in the PLC.



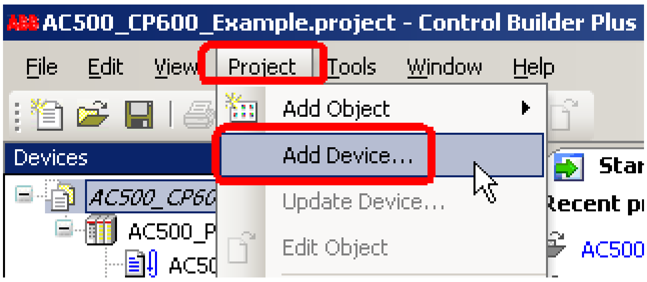
Step 3: PB610 Panel Builder 600

Start a new Project from Control Builder Plus

1. Start your Panel Builder 600 project in the Control Builder Plus.
2. Select the top level of your project.

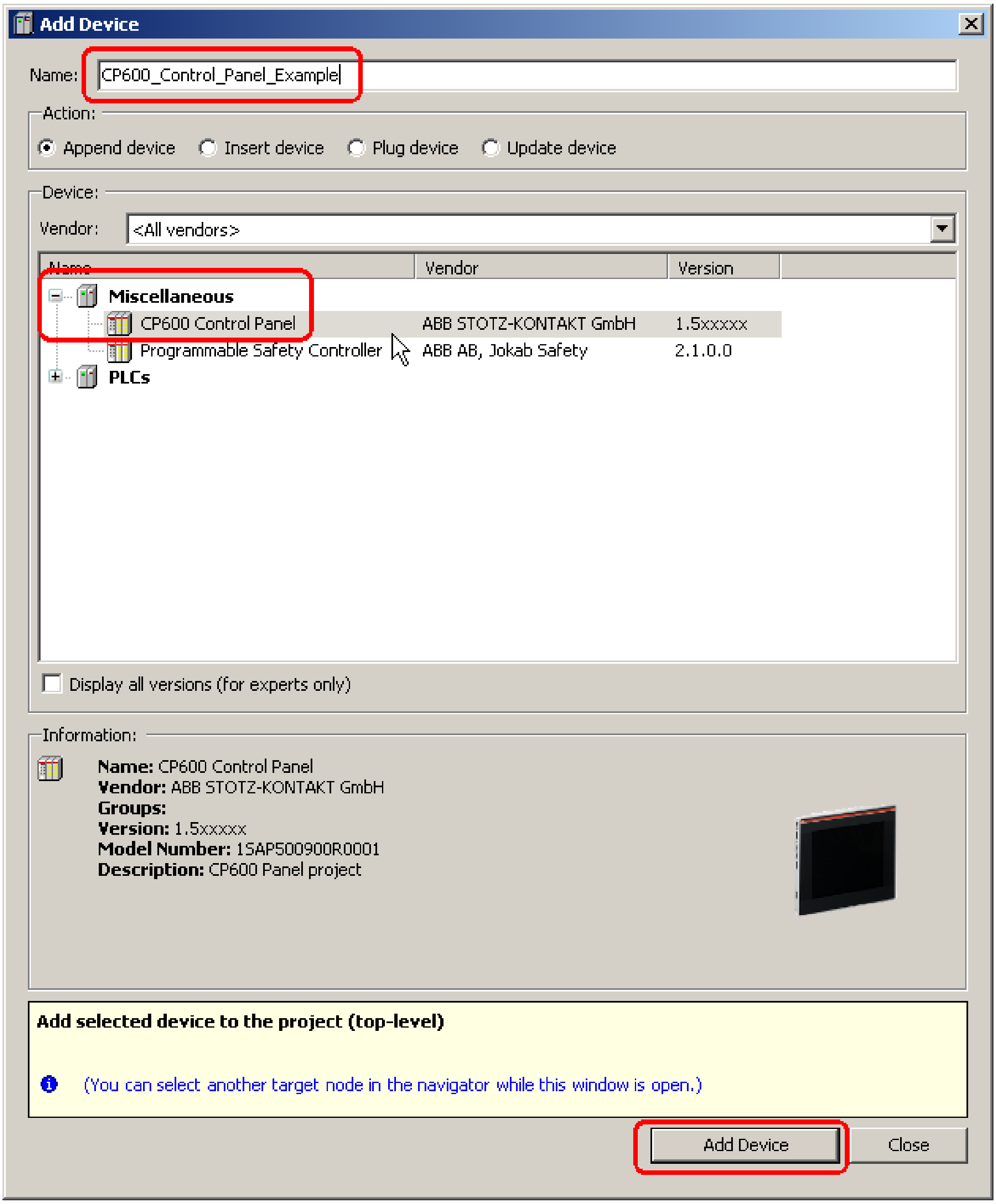

AC500_CP600_Example.project - Control Builder Plus
File Edit View Project Tools Window Help
Devices
AC500_CP600_Example
AC500_PM573_ETH_V2_1 (AC500 PM5
AC500
CPU_parameters (CPU parameters)
IO_Bus (I/O-Bus)
Start Page
Recent projects
AC500_CP600_E
ABB-AC500-PLC2
test1

3. Add device with a right click or select *Project/Add Device*.



4. Select *Miscellaneous/CP600 Control Panel* and adapt the name.

5. Click **Add Device** to insert the new CP600 and close the dialog by clicking **Close**.


Add Device

Name: CP600_Control_Panel_Example

Action:

● Append device ○ Insert device ○ Plug device ○ Update device

Device:

Vendor: <All vendors>

Name	Vendor	Version
Miscellaneous		
CP600 Control Panel	ABB STOTZ-KONTAKT GmbH	1.5xxxxx
Programmable Safety Controller	ABB AB, Jokab Safety	2.1.0.0
PLCs		

□ Display all versions (for experts only)

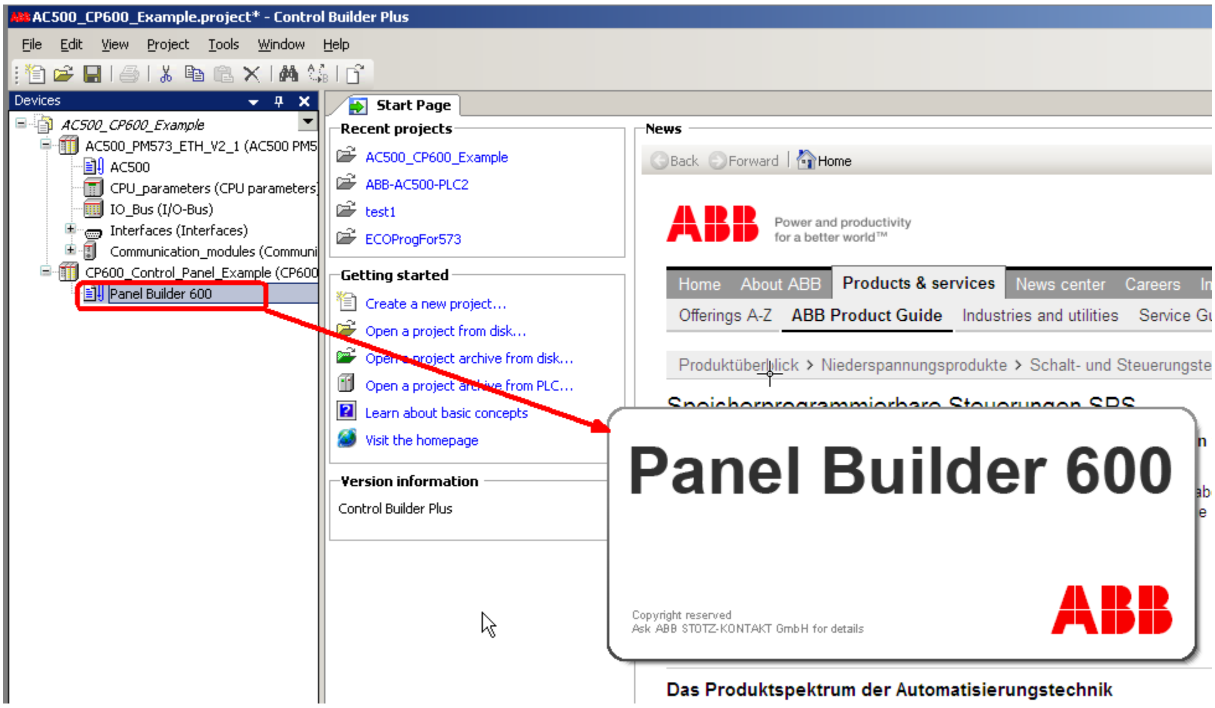
Information:

Name: CP600 Control Panel
Vendor: ABB STOTZ-KONTAKT GmbH
Groups:
Version: 1.5xxxxx
Model Number: 1SAP500900R0001
Description: CP600 Panel project

Add selected device to the project (top-level)

(You can select another target node in the navigator while this window is open.)

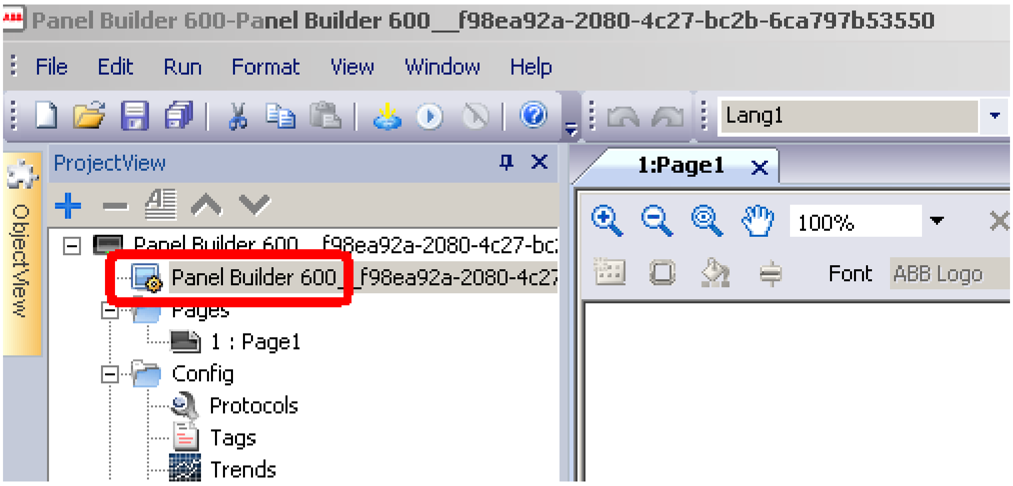
6. Double click on the Panel Builder 600 icon to start the Panel Builder 600 Suite.


AC500_CP600_Example.project* - Control Builder Plus
File Edit View Project Tools Window Help
Devices
AC500_CP600_Example
AC500_PM573_ETH_V2_1 (AC500 PM5
AC500
CPU_parameters (CPU parameters)
IO_Bus (I/O-Bus)
Interfaces (Interfaces)
Communication_modules (Communi
CP600_Control_Panel_Example (CP600
Panel Builder 600
Start Page
Recent projects
AC500_CP600_Example
ABB-AC500-PLC2
test1
ECOProgFor573
Getting started
Create a new project…
Open a project from disk…
Open a project archive from disk…
Open a project archive from PLC…
Learn about basic concepts
Visit the homepage
Version information
Control Builder Plus
News
Back Forward Home
Power and productivity for a better world™
Home About ABB Products & services News center Careers
Offerings A-Z ABB Product Guide Industries and utilities Service Gu
Produktüberblick > Niederspannungsprodukte > Schalt- und Steuerungste
Panel Builder 600
Copyright reserved
Ask ABB STOTZ-KONTAKT GmbH for details
ABB
Das Produktspektrum der Automatisierungstechnik

- ➤ The PB610 Panel Builder 600 opens.

Select your Control Panel type by the following steps.

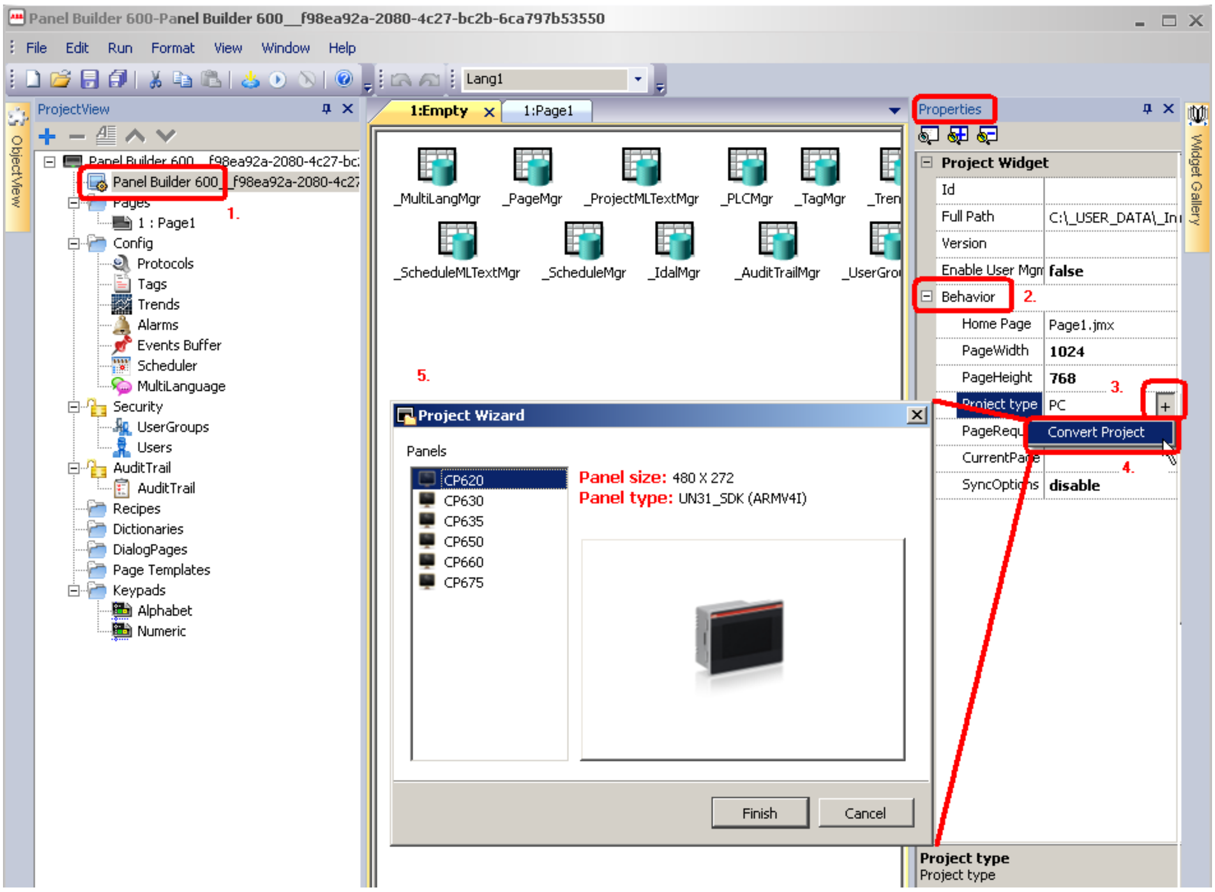
7. Double click on the Panel Builder 600 icon on the left side


Panel Builder 600-Panel Builder 600__f98ea92a-2080-4c27-bc2b-6ca797b53550
File Edit Run Format View Window Help
Lang1
ProjectView
ObjectView
Panel Builder 600__f98ea92a-2080-4c27-bc
Panel Builder 600__f98ea92a-2080-4c27
Pages
1 : Page1
Config
Protocols
Tags
Trends
1:Page1
100%
Font ABB Logo

- ➤ The Property View of the Panel Builder 600 opens on the right side.

8. Select **Behaviour** and open the **+** of the **Project type** property.

9. Click **Convert Project** to open the project wizard dialog and select your Control Panel type.

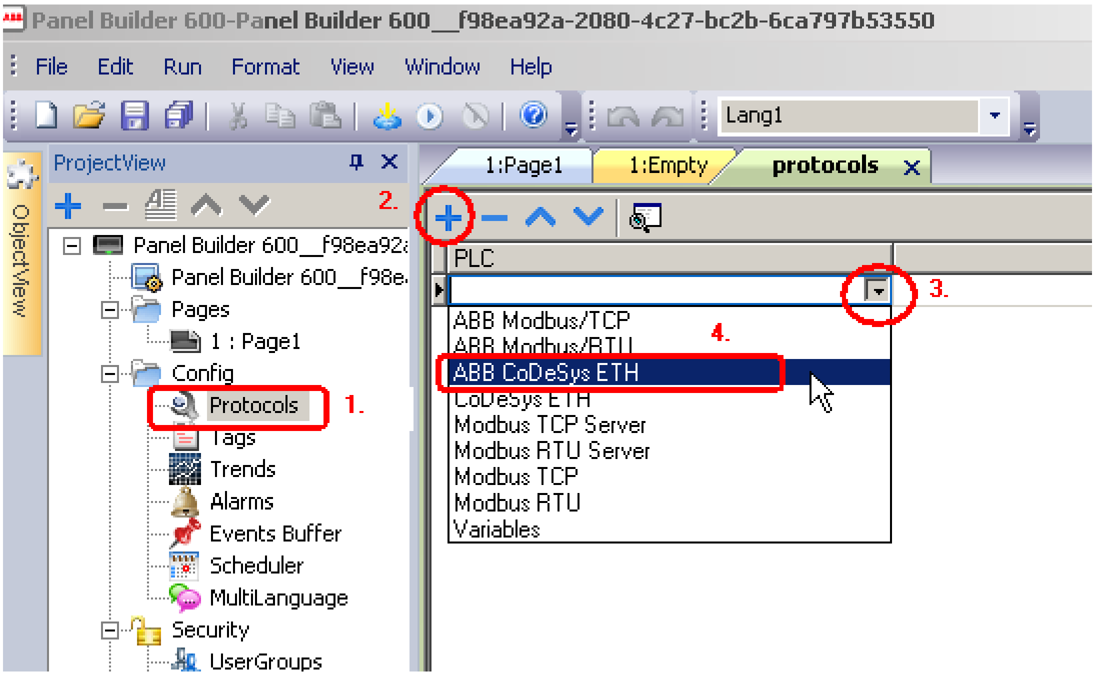


Protocol Configuration

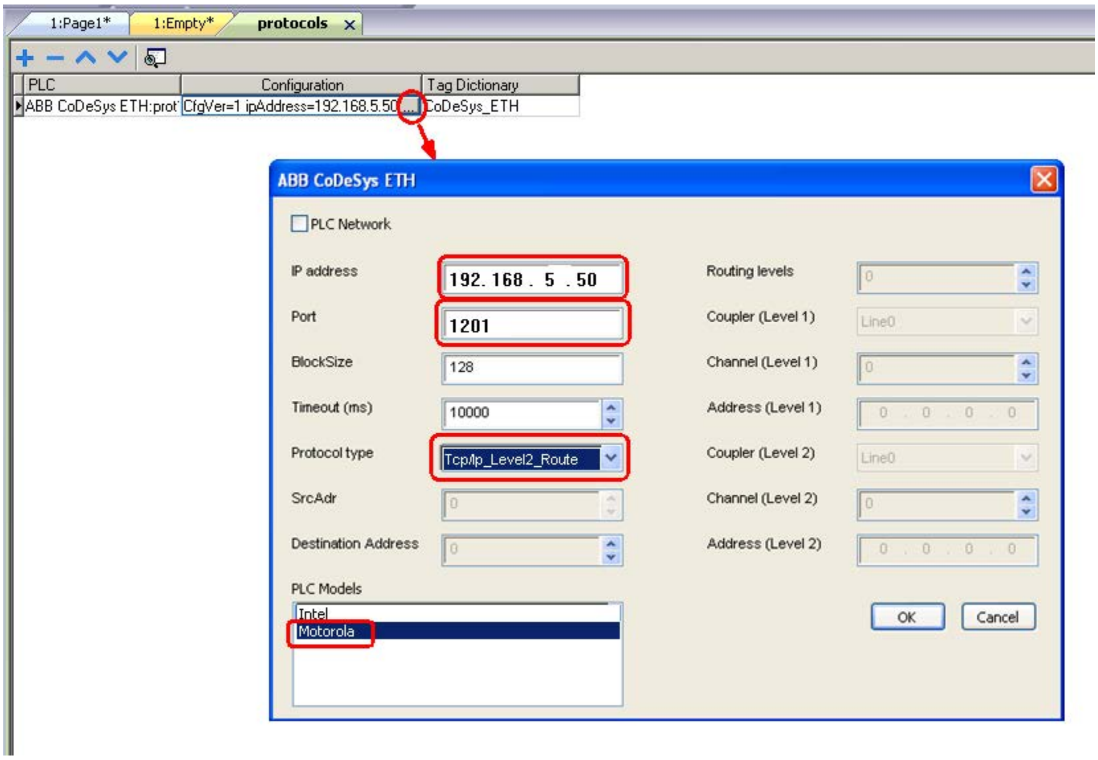
1. Via the Project View open the configuration of the protocols.

2. Click the **+** icon to insert a new protocol.

3. Select the protocol type in the pull down menu.



4. Adapt the protocol as shown in the picture below.



PLC	Configuration	Tag Dictionary
ABB CoDeSys ETH:prot	CfgVer=1 ipAddress=192.168.5.50	CoDeSys_ETH

ABB CoDeSys ETH

□ PLC Network

IP address: 192. 168 . 5 . 50

Port: 1201

BlockSize: 128

Timeout (ms): 10000

Protocol type: Tcp/Ip_Level2_Route

SrcAdr: 0

Destination Address: 0

PLC Models: Intel, Motorola

Routing levels: 0

Coupler (Level 1): Line0

Channel (Level 1): 0

Address (Level 1): 0 . 0 . 0 . 0

Coupler (Level 2): Line0

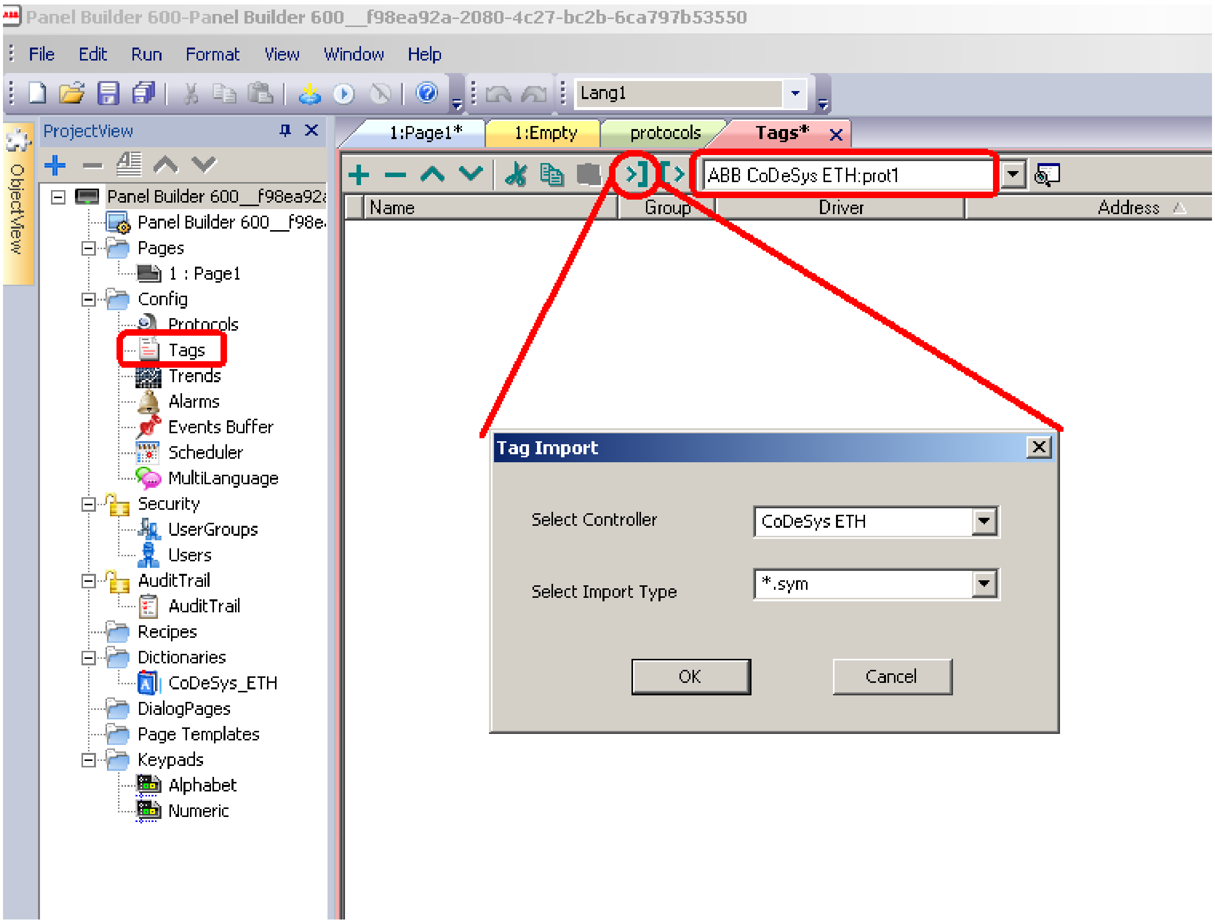
Channel (Level 2): 0

Address (Level 2): 0 . 0 . 0 . 0

5. Adapt the IP-Address of your target PLC.
6. Set the port to 1201.
7. Set the protocol type to Tcp/Ip_Level2_Route.
8. Select Motorola as PLC model.
9. To change the protocol properties later use the button on the right side in the column *Configuration*.
10. For detailed description refer to chapter “Communication Protocols”.

Tag Import – Work with Tags

1. Double click on the **Tags** icon in the Project View to open the tag configuration.
2. Press the marked button in the configuration page to import the tags and click **OK**.



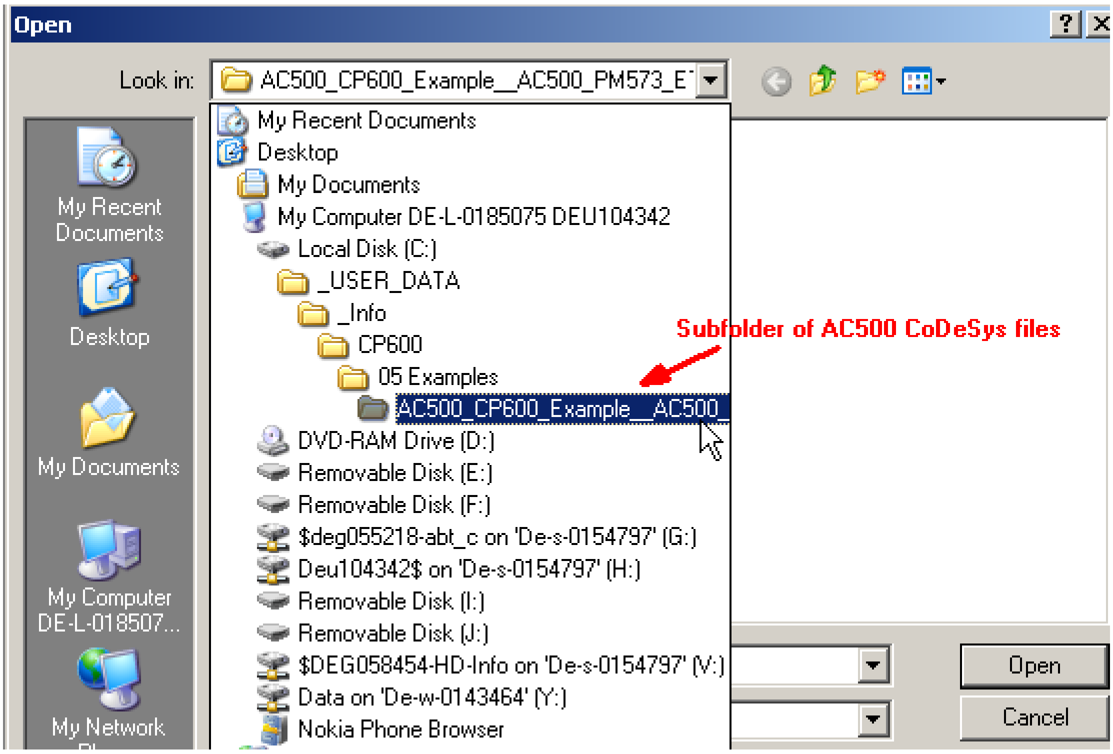
Tag Import

Select Controller: CoDeSys ETH

Select Import Type: *.sym

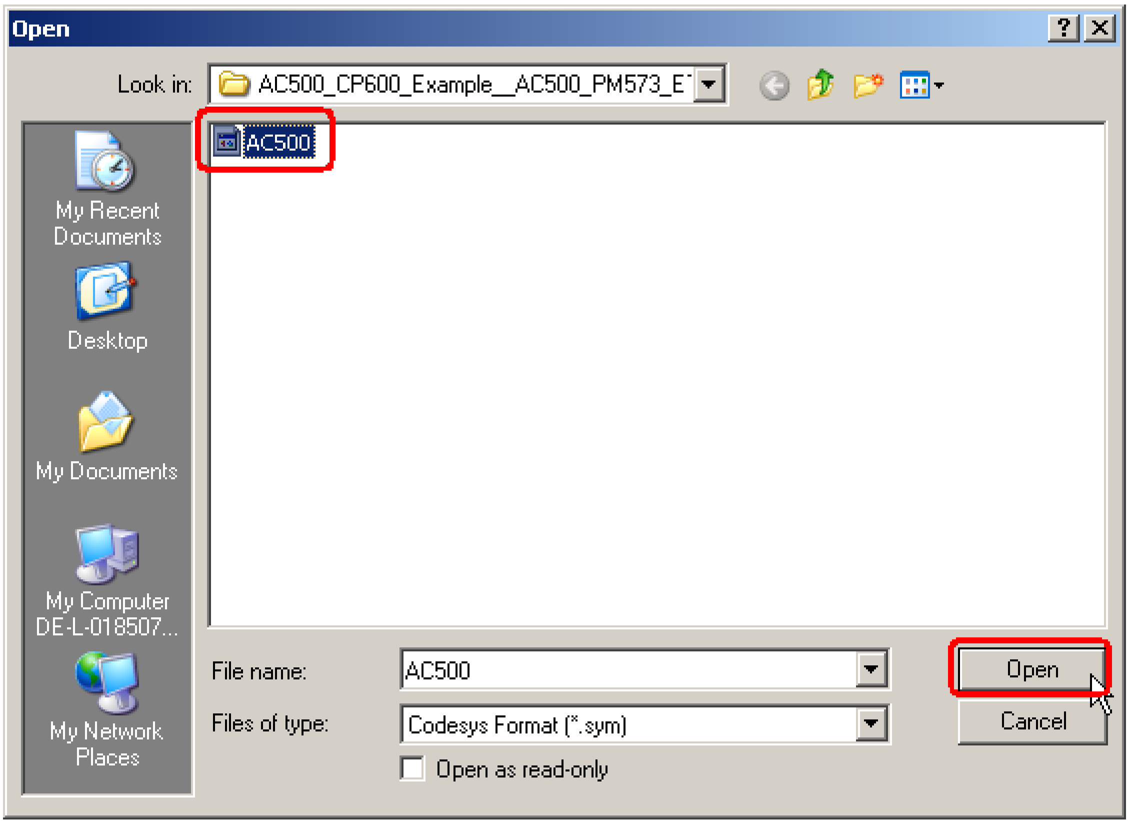
OK Cancel

3. The symbol file is placed in the subfolder for the AC500 CoDeSys files in the project folder of the *Control Builder Plus* project. This folder will be packed into the project file when the project is closed.


Open

Look in: AC500_CP600_Example__AC500_PM573_E

4. Select the symbol file and open it.



Look in: AC500_CP600_Example__AC500_PM573_E

File name: AC500

Files of type: Codesys Format (*.sym)

☐ Open as read-only

5. The accessible tags are shown in the lower part of the tag editor.

6. Mark the tags which you want to import to the Panel Builder project.

7. Click the marked symbol to import the tags.

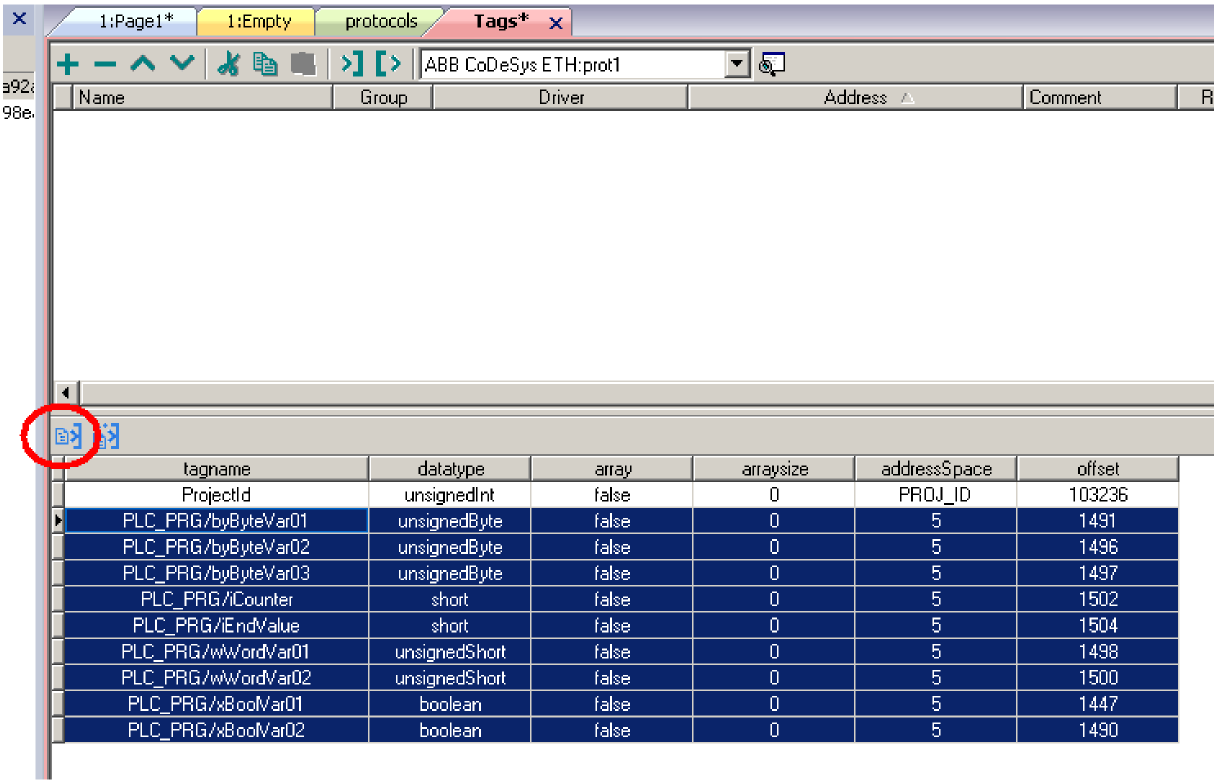
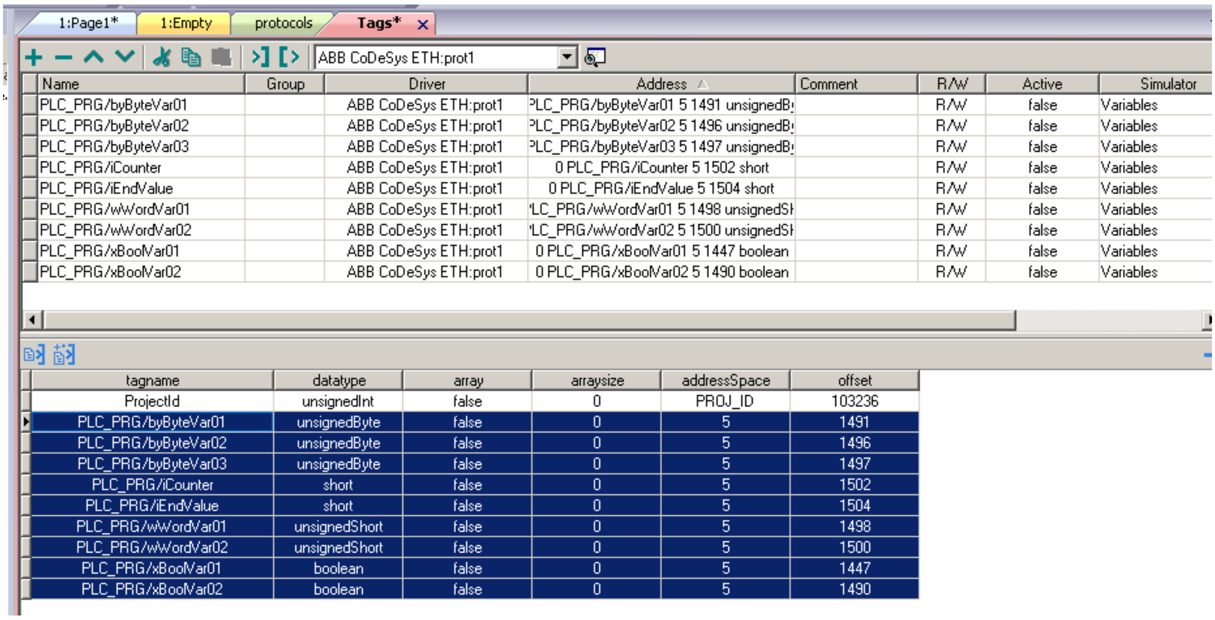


ABB CoDeSys ETH:prot1

tagname	datatype	array	arraysize	addressSpace	offset
ProjectId	unsignedInt	false	0	PROJ_ID	103236
PLC_PRG/byByteVar01	unsignedByte	false	0	5	1491
PLC_PRG/byByteVar02	unsignedByte	false	0	5	1496
PLC_PRG/byByteVar03	unsignedByte	false	0	5	1497
PLC_PRG/iCounter	short	false	0	5	1502
PLC_PRG/iEndValue	short	false	0	5	1504
PLC_PRG/wWordVar01	unsignedShort	false	0	5	1498
PLC_PRG/wWordVar02	unsignedShort	false	0	5	1500
PLC_PRG/xBoolVar01	boolean	false	0	5	1447
PLC_PRG/xBoolVar02	boolean	false	0	5	1490



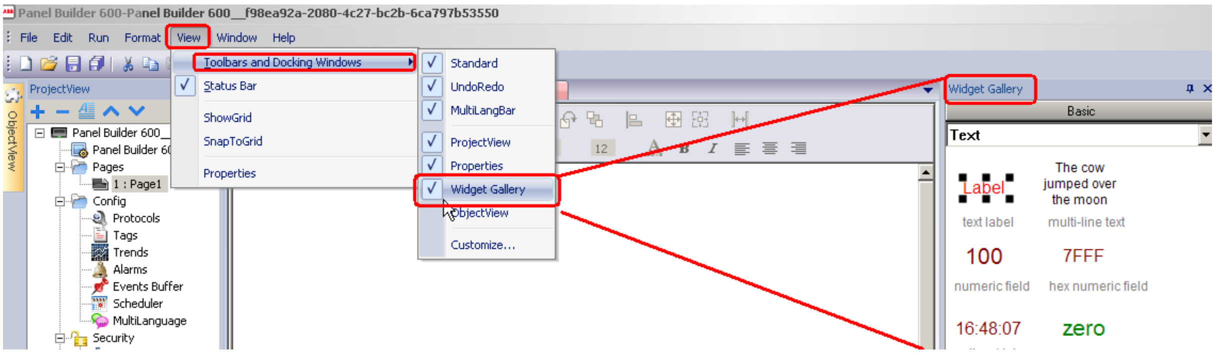
Name	Group	Driver	Address	Comment	R/W	Active	Simulator
PLC_PRG/byByteVar01		ABB CoDeSys ETH:prot1	ᴾLC_PRG/byByteVar01 5 1491 unsignedB		R/W	false	Variables
PLC_PRG/byByteVar02		ABB CoDeSys ETH:prot1	ᴾLC_PRG/byByteVar02 5 1496 unsignedB		R/W	false	Variables
PLC_PRG/byByteVar03		ABB CoDeSys ETH:prot1	ᴾLC_PRG/byByteVar03 5 1497 unsignedB		R/W	false	Variables
PLC_PRG/iCounter		ABB CoDeSys ETH:prot1	0 PLC_PRG/iCounter 5 1502 short		R/W	false	Variables
PLC_PRG/iEndValue		ABB CoDeSys ETH:prot1	0 PLC_PRG/iEndValue 5 1504 short		R/W	false	Variables
PLC_PRG/wWordVar01		ABB CoDeSys ETH:prot1	'LC_PRG/wWordVar01 5 1498 unsignedSh		R/W	false	Variables
PLC_PRG/wWordVar02		ABB CoDeSys ETH:prot1	'LC_PRG/wWordVar02 5 1500 unsignedSh		R/W	false	Variables
PLC_PRG/xBoolVar01		ABB CoDeSys ETH:prot1	0 PLC_PRG/xBoolVar01 5 1447 boolean		R/W	false	Variables
PLC_PRG/xBoolVar02		ABB CoDeSys ETH:prot1	0 PLC_PRG/xBoolVar02 5 1490 boolean		R/W	false	Variables

tagname	datatype	array	arraysize	addressSpace	offset
ProjectId	unsignedInt	false	0	PROJ_ID	103236
PLC_PRG/byByteVar01	unsignedByte	false	0	5	1491
PLC_PRG/byByteVar02	unsignedByte	false	0	5	1496
PLC_PRG/byByteVar03	unsignedByte	false	0	5	1497
PLC_PRG/iCounter	short	false	0	5	1502
PLC_PRG/iEndValue	short	false	0	5	1504
PLC_PRG/wWordVar01	unsignedShort	false	0	5	1498
PLC_PRG/wWordVar02	unsignedShort	false	0	5	1500
PLC_PRG/xBoolVar01	boolean	false	0	5	1447
PLC_PRG/xBoolVar02	boolean	false	0	5	1490

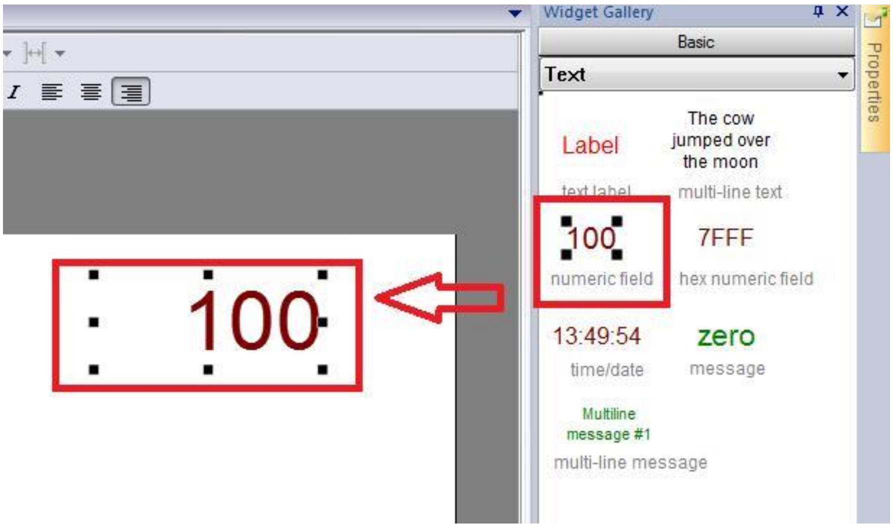
View of tag editor after selection and import of tags

Attach the Tags to Widgets

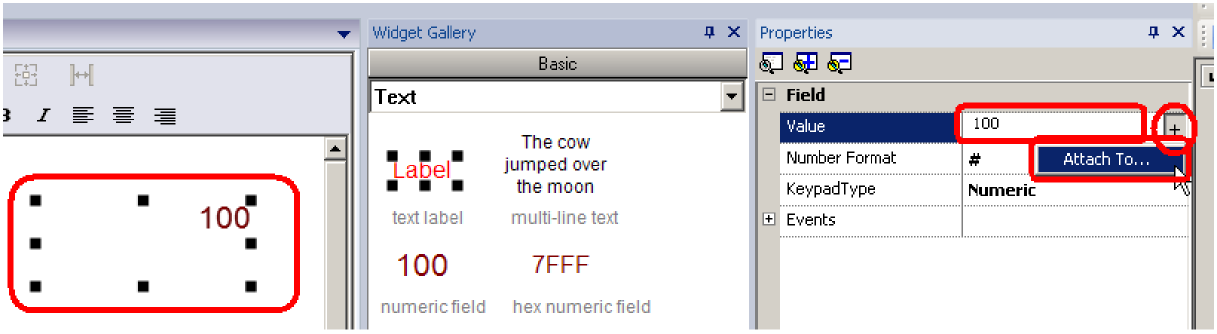
1. Double click on the page symbol to open a page of your project.
2. Open the widget gallery.



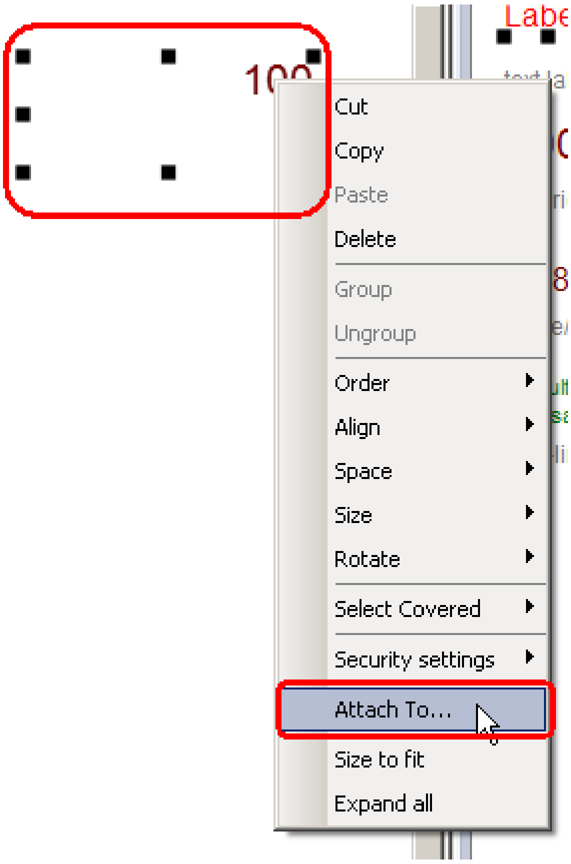
3. Drag and drop a widget to your page. In this example take a numeric field.



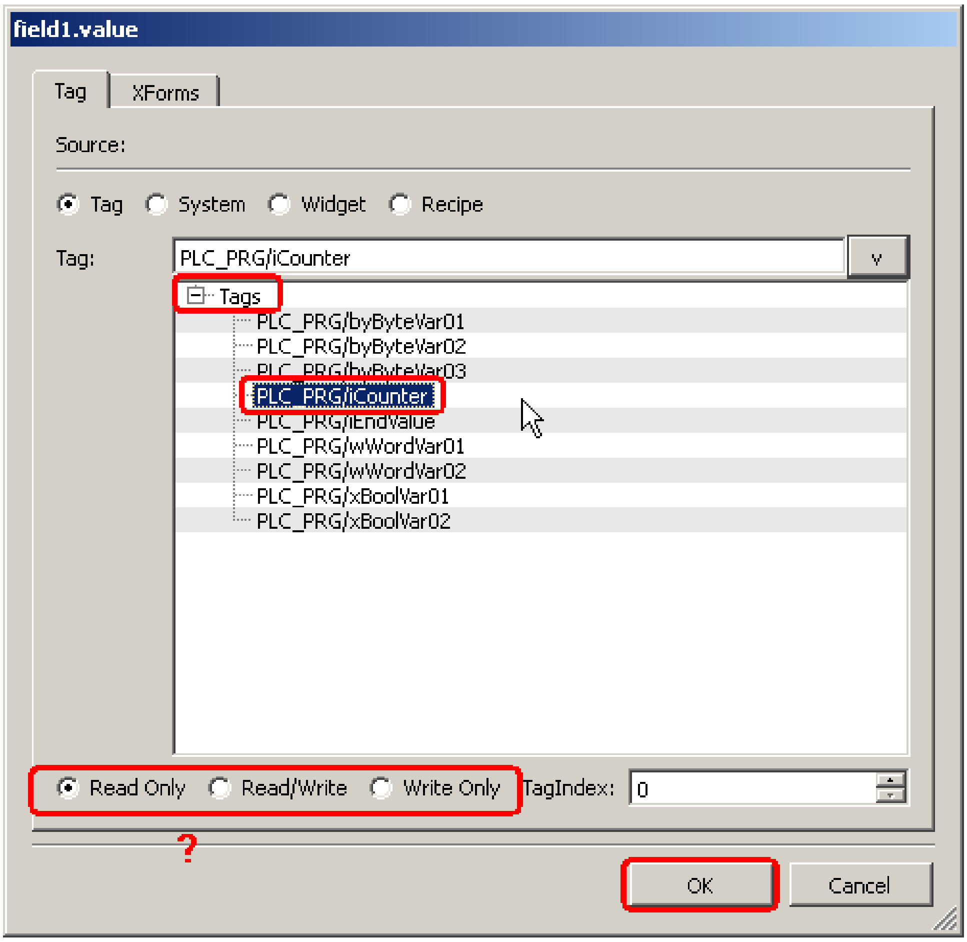
4. Use the property *Value* to attach a tag to the widget. Click in the field, press the **+** button and select **Attach To**.



5. Or right click on the tag and use the context menu.



6. Select the desired tag from the dialog box and select the authorization for read or read/write access of the tag.



field1.value

Tag: PLC_PRG/iCounter

Source: ● Tag ○ System ○ Widget ○ Recipe

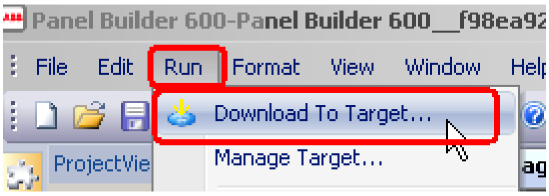
● Read Only ○ Read/Write ○ Write Only

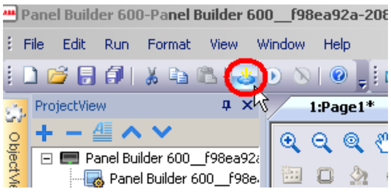
TagIndex: 0

7. Continue in that way for all other tags you want to use.

Download the Project to the Control Panel

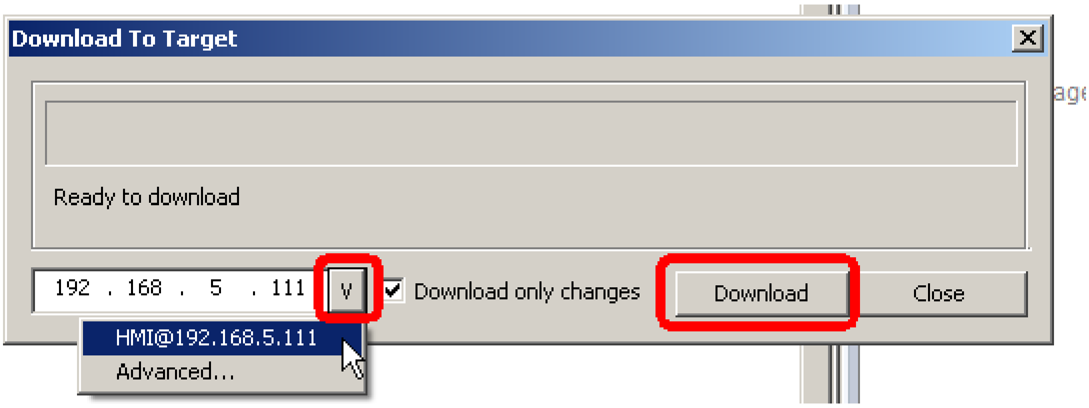
1. Select *Run/Download* or use the marked symbol to start the download dialog.





Panel Builder 600-Panel Builder 600__f98ea92a-208
File
Edit
Run
Format
View
Window
Help
ProjectView
1:Page1*
Panel Builder 600__f98ea92a
Panel Builder 600__f98e

2. Select your CP600 Control Panel from the pull down list and download the project.



Download To Target

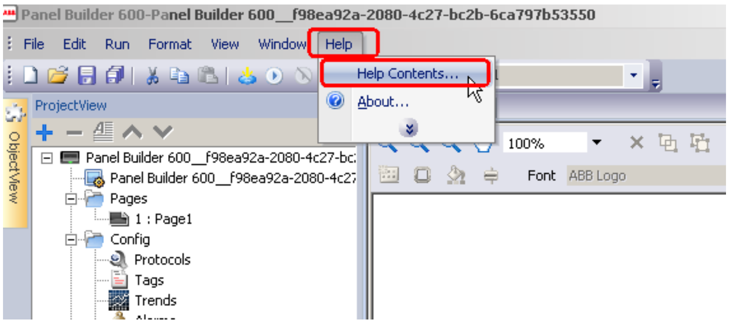
Ready to download

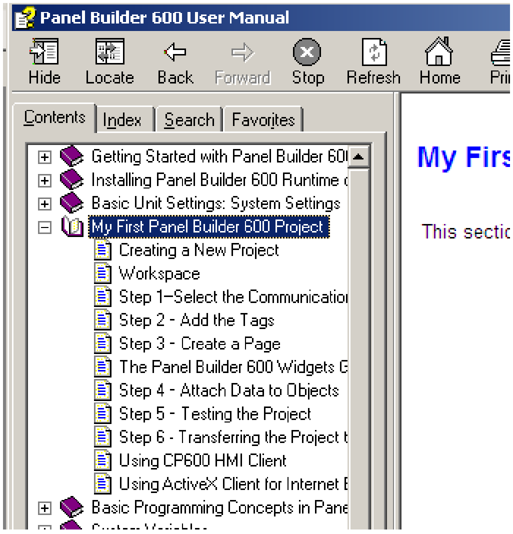
192 . 168 . 5 . 111

☑ Download only changes

Further Help

→ Select *Help/Help Contents*.


Panel Builder 600-Panel Builder 600__f98ea92a-2080-4c27-bc2b-6ca797b53550
File
Edit
Run
Format
View
Window
Help
Help Contents...
About...
ProjectView
ObjectView
Panel Builder 600__f98ea92a-2080-4c27-bc
Panel Builder 600__f98ea92a-2080-4c27
Pages
1 : Page1
Config
Protocols
Tags
Trends
100%
Font
ABB Logo


Panel Builder 600 User Manual
Hide
Locate
Back
Forward
Stop
Refresh
Home
Pri
Contents
Index
Search
Favorites
Getting Started with Panel Builder 60
Installing Panel Builder 600 Runtime
Basic Unit Settings: System Settings
My First Panel Builder 600 Project
Creating a New Project
Workspace
Step 1–Select the Communicatio
Step 2 - Add the Tags
Step 3 - Create a Page
The Panel Builder 600 Widgets G
Step 4 - Attach Data to Objects
Step 5 - Testing the Project
Step 6 - Transferring the Project t
Using CP600 HMI Client
Using ActiveX Client for Internet E
Basic Programming Concepts in Pane
My Firs
This sectio

Communication Protocols

ABB Modbus TCP Driver

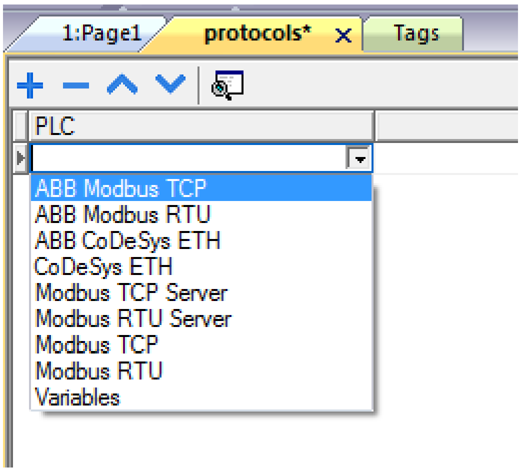
Various Modbus TCP-capable devices can be connected to the operator panel. To set-up your Modbus TCP device, please refer to the documentation you have received with the device.

This specific implementation of the Modbus TCP driver provides easy handling of the connection to the ABB controllers providing specific supports for PLC models and tag import facilities.

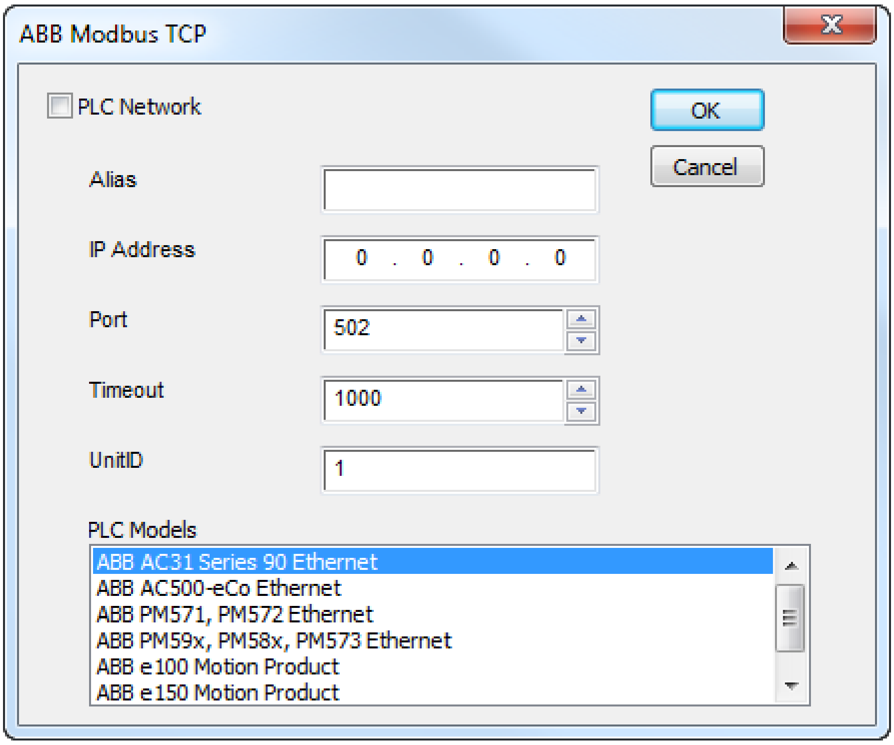
The implementation of the protocol operates as a Modbus TCP client only.

Protocol Editor Settings

→ Add with the button + a driver in the protocol editor and select the protocol called ABB Modbus TCP from the list of available protocols.



The driver configuration dialog is showed in the following figure.



☐ PLC Network

Alias

IP Address 0 . 0 . 0 . 0

Port 502

Timeout 1000

UnitID 1

PLC Models

ABB AC31 Series 90 Ethernet
ABB AC500-eCo Ethernet
ABB PM571, PM572 Ethernet
ABB PM59x, PM58x, PM573 Ethernet
ABB e100 Motion Product
ABB e150 Motion Product

Parameter	Description
Alias	Name to be used to identify nodes in network configurations. The name will be added as a prefix to each tag name imported for each network node
IP address	Ethernet IP address of the controller
Port	Allows to change the default port number used by the Modbus TCP driver; it could be useful whenever the communication passes through Routers or Internet gateways where the default port number is already in use
Timeout (ms)	Defines the time inserted by the protocol between two retries of the same message in case of missing response from the server device. It is expressed in milliseconds.
Unit ID	The Unit ID is rarely used and in most cases can be left zero. The value of the Modbus ID is simply copied into the Unit Identifier field of the Modbus TCP communication frame. Usually it is used when communicating over Ethernet-to-Serial bridges and is then representing the Slave ID.
PLC Models	The list allows selecting the PLC model you are going to connect to. The selection will influence the data range offset per each data type according to the specific PLC memory resources.
PLC Network	The protocol allows the connection of multiple controllers to one operator panel. To set-up multiple connections, check **PLC network** checkbox and enter IP address for all controllers.

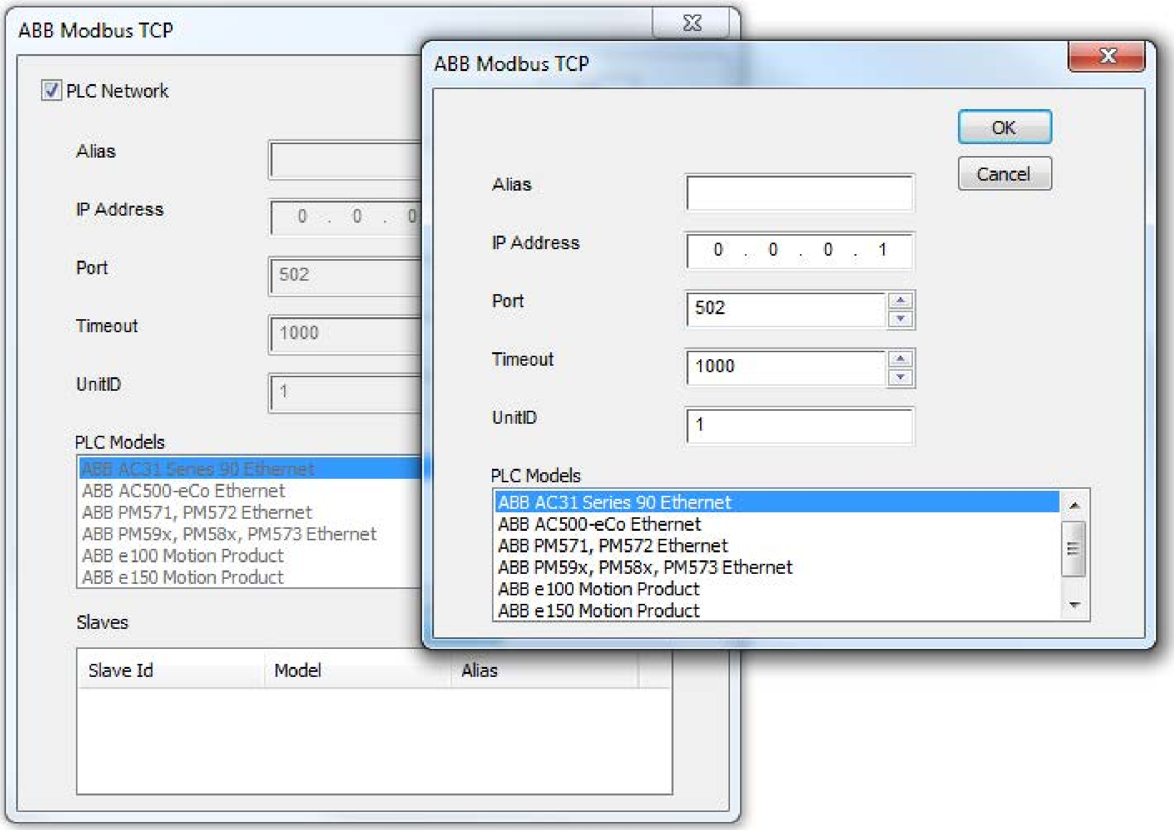


ABB Modbus TCP

☑ PLC Network

Alias:

IP Address: 0 . 0 . 0

Port: 502

Timeout: 1000

UnitID: 1

PLC Models:
- ABB AC31 Series 90 Ethernet
- ABB AC500-eCo Ethernet
- ABB PM571, PM572 Ethernet
- ABB PM59x, PM58x, PM573 Ethernet
- ABB e100 Motion Product
- ABB e150 Motion Product

Slaves

Slave Id	Model	Alias

ABB Modbus TCP

Alias:

IP Address: 0 . 0 . 0 . 1

Port: 502

Timeout: 1000

UnitID: 1

PLC Models:
- ABB AC31 Series 90 Ethernet
- ABB AC500-eCo Ethernet
- ABB PM571, PM572 Ethernet
- ABB PM59x, PM58x, PM573 Ethernet
- ABB e100 Motion Product
- ABB e150 Motion Product

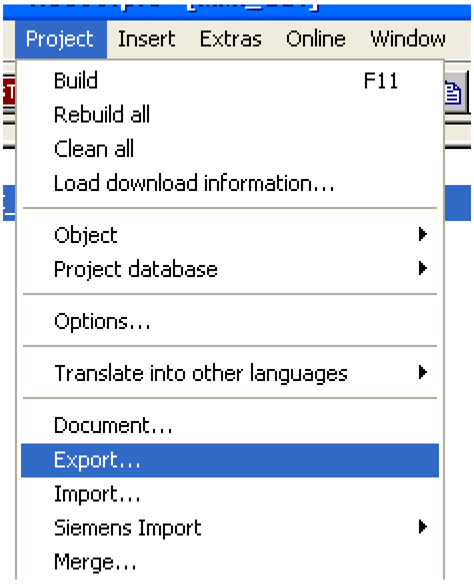
Tag Import

This special Modbus driver supports tag import.

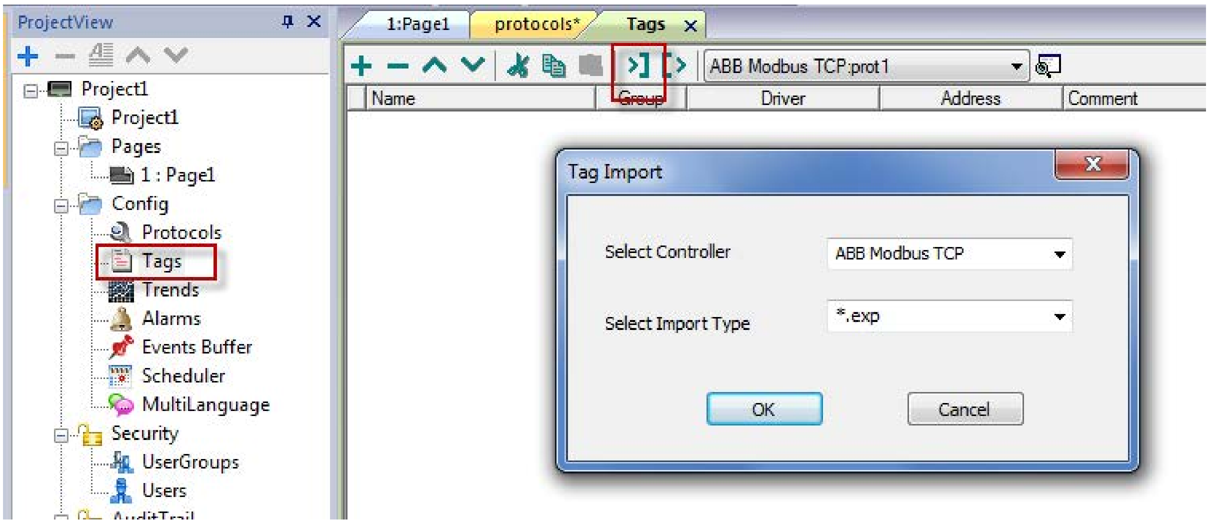
The ABB controllers are programmable with a programming tool called Control Builder Plus which is based on the CoDeSys V2.3 soft PLC.

The tag importer supports the CoDeSys export file in “.exp” format.

In the CoDeSys programming software the export command is available under the Project menu as shown in the following figure.



1. Select the driver in the PB610 Panel Builder 600 Studio tag editor and click on the **Import tag** button to start the importer.

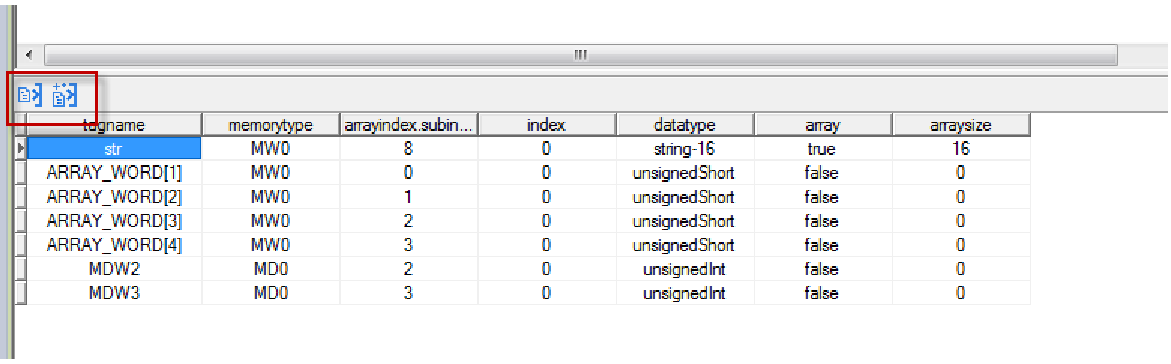


Select Controller: ABB Modbus TCP

Select Import Type: *.exp

2. Locate the “exp” file and confirm with **OK**.

The tags present in the exported document are listed in the tag dictionary from where they can be directly added to the project using the **add tags** button as shown in the following figure.

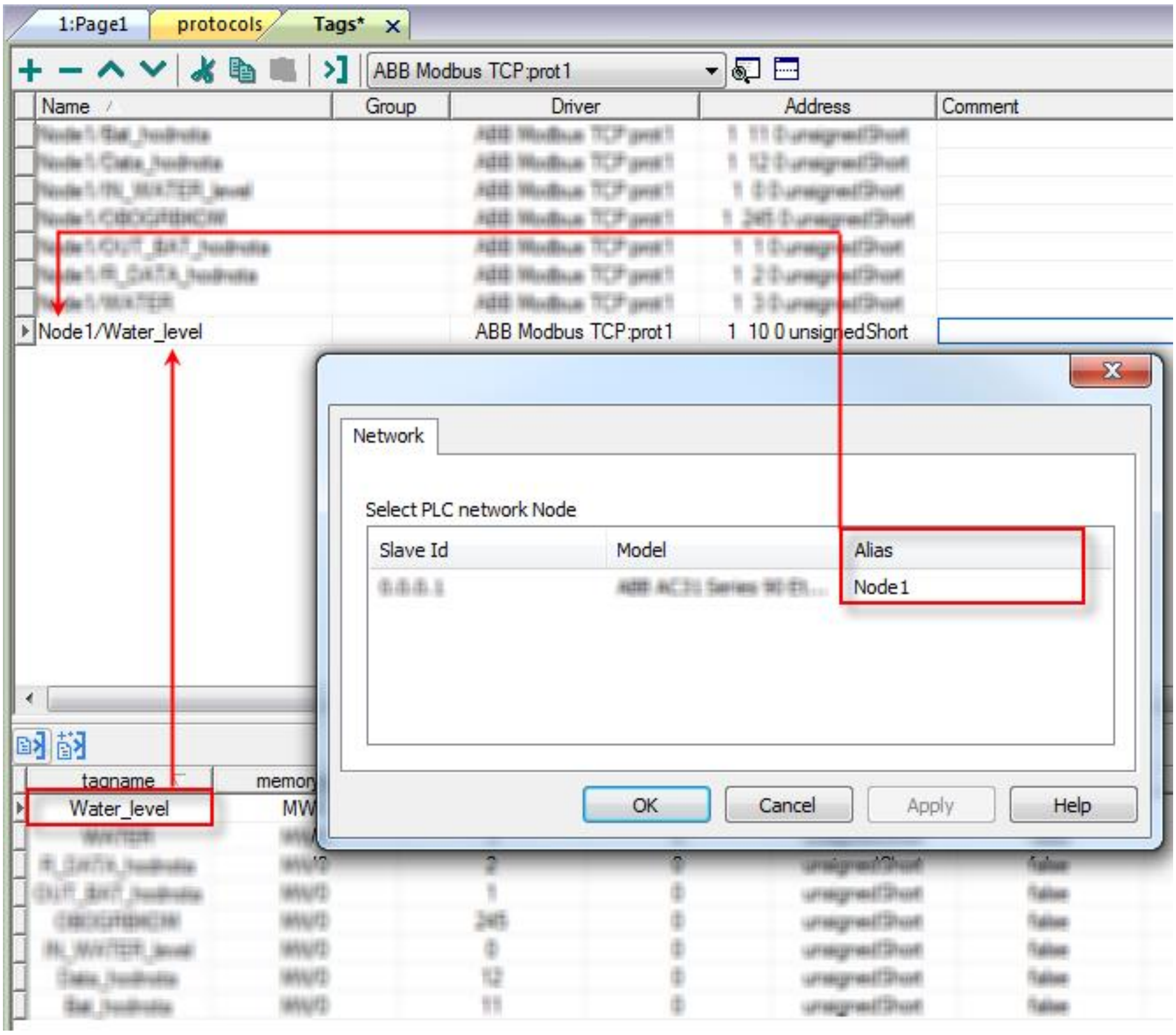


tagname	memorytype	arrayindex.subin...	index	datatype	array	arraysize
str	MW0	8	0	string-16	true	16
ARRAY_WORD[1]	MW0	0	0	unsignedShort	false	0
ARRAY_WORD[2]	MW0	1	0	unsignedShort	false	0
ARRAY_WORD[3]	MW0	2	0	unsignedShort	false	0
ARRAY_WORD[4]	MW0	3	0	unsignedShort	false	0
MDW2	MD0	2	0	unsignedInt	false	0
MDW3	MD0	3	0	unsignedInt	false	0

Aliasing Tag Names in Network Configurations

Tag names must be unique at project level; it often happens that the same tag names are to be used for different controller nodes (for example when the HMI is connected to two devices that are running the same application). Since tags include also the identification of the node and tag editor does not support duplicate tag names, the import facility in tag editor has an aliasing feature that can automatically add a prefix to imported tags. With this feature tag names can be done unique at project level.

The feature works when importing tags for a specific protocol. Each tag name will be prefixed with the string specified by the “Alias”. As shown in the figure below, the connection to a certain controller is assigned the name “Node1”. When tags are imported for this node, all tag names will have the prefix “Node1” making each of them unique at the network/project level.



Name	Group	Driver	Address	Comment
Node1/Water_level		ABB Modbus TCP:prot1	1 10 0 unsignedShort	

Select PLC network Node

Slave Id	Model	Alias
		Node1

tagname	memory
Water_level	MW

Aliasing tag names is only available when tags can be imported. Tags which are added manually in the tag editor do not need to have the Alias prefix in the tag name.
The Alias string is attached to the tag name only at the moment the tags are imported using tag editor. If you modify the Alias string after the tag import has been completed, there will be no effect on the names already present in the dictionary. When the Alias string is changed and tags are imported again, all tags will be imported again with the new prefix string.

NOTE

Special Data Types

The ABB Modbus TCP driver provides one special data type called "IP Override".

The IP Override allows changing at runtime the IP address of the target controller you want to connect. This memory type is an array of 4 unsigned bytes, one per each byte of the IP address.

The Node Override IP is initialized with the value of the controller IP specified in the project at programming time.

If the IP Override is set to 0.0.0.0, all the communication with the node is stopped, no request frames are generated anymore.

If the IP Override has a value different from 0.0.0.0, it is interpreted as node IP Override and the target IP address is replaced runtime with the new value.

In case the panel has been configured to access to a network of controllers, each node has its own override variable.

NOTE The IP Override values assigned at runtime are not retained among power cycles.

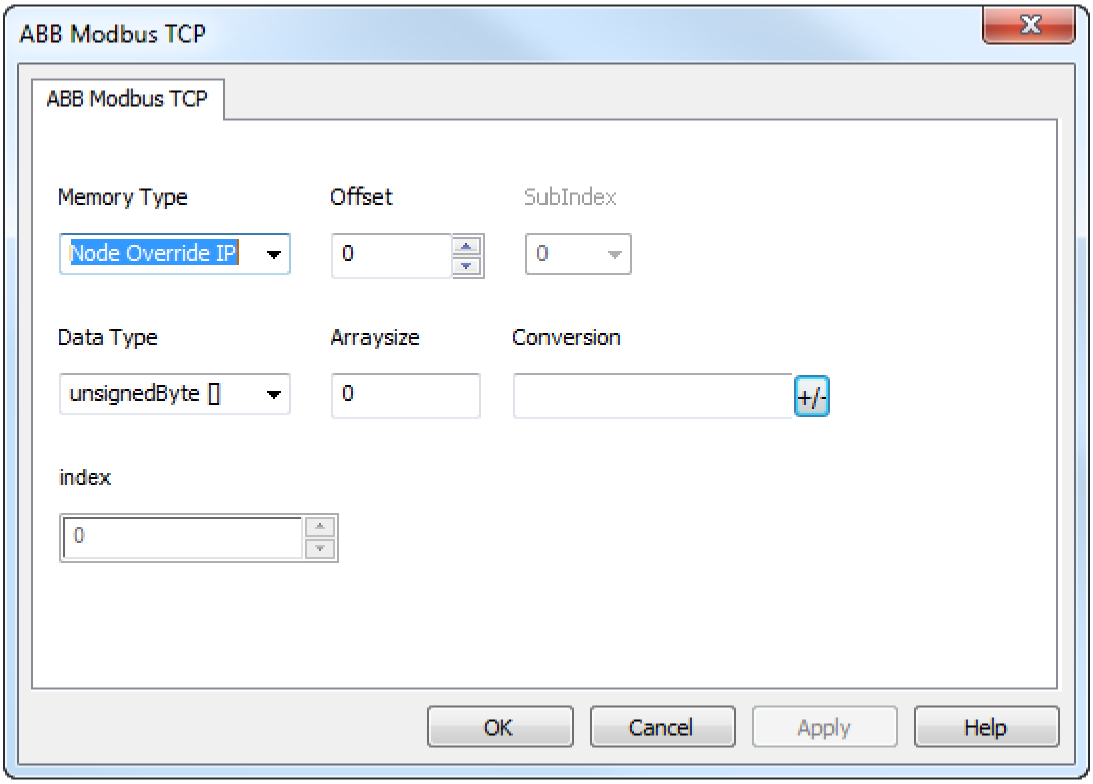


ABB Modbus TCP

Memory Type: Node Override IP

Offset: 0

SubIndex: 0

Data Type: unsignedByte []

Arraysize: 0

Conversion:

index: 0

Communication Status

The current communication status can be displayed using the dedicated system variables. Please refer to the chapter “system variables” about available types and their use.

The codes supported for this communication driver are:

Error	Notes
NAK	Returned in case the controller replies with a not acknowledge.
Timeout	Returned when a request is not replied within the specified timeout period; ensure the controller is connected and properly configured to get network access.
Invalid response	The panel did receive from the controller a response, but its format or its contents is not as expected; ensure the data programmed in the project are consistent with the controller resources.
General Error	Error cannot be identified; should never be reported; contact technical support.

Implementation Details

This Modbus TCP implementation supports only a subset of the standard Modbus TCP function codes.

The supported function codes are listed in the table below.

Code	Function	Description
01	Read Coil Status	Read multiple bits in the CP600 coil area.
02	Read Input Status	Read the ON/OFF status of the discrete inputs (1x reference) in the slave.
03	Read Holding Registers	Read multiple registers.
04	Read Input Registers	Read the binary contents of input registers (3x reference) in the slave.
05	Force Single Coil	Force a single coil to either ON or OFF.
06	Preset Single Register	Preset a value in a register.
16	Preset Multiple Registers	Preset value in multiple registers.

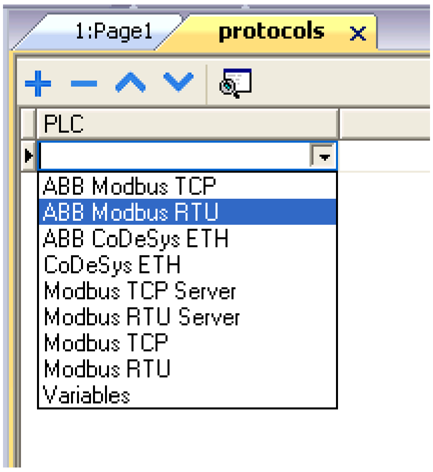
ABB Modbus RTU Driver

The operator panels can be connected to a Modbus network as the network master using this generic driver.

This specific implementation of the Modbus RTU driver provides easy handling of the connections to the ABB controllers providing specific support for PLC models and tag import facilities.

Protocol Editor Settings

→ Add with the button + a driver in the protocol editor and select the protocol called ABB Modbus RTU from the list of available protocols.



The driver configuration dialog is shown in the following figure.

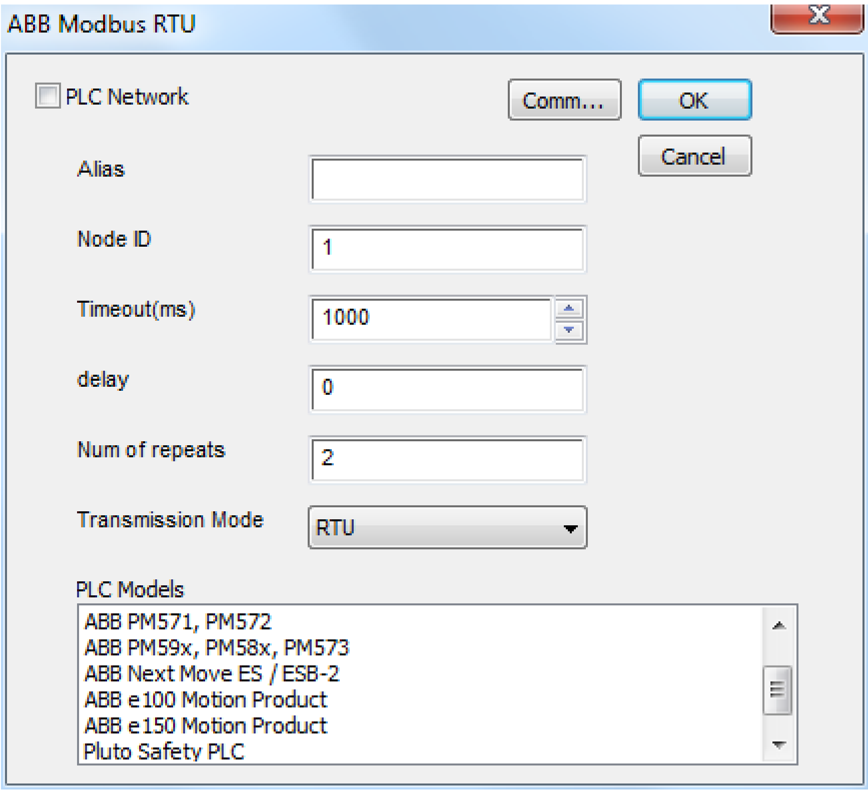


ABB Modbus RTU

□ PLC Network

Alias

Node ID: 1

Timeout(ms): 1000

delay: 0

Num of repeats: 2

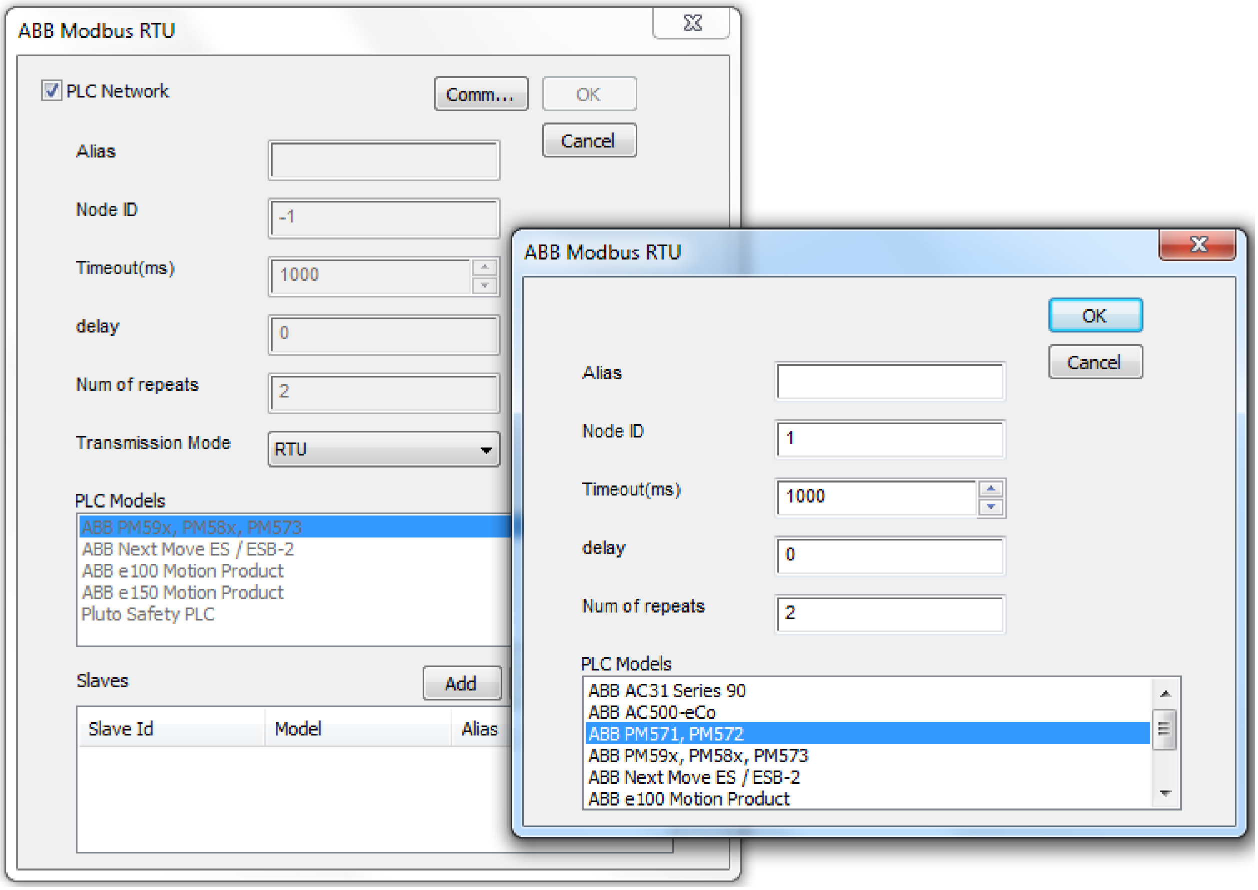
Transmission Mode: RTU

PLC Models

ABB PM571, PM572
ABB PM59x, PM58x, PM573
ABB Next Move ES / ESB-2
ABB e100 Motion Product
ABB e150 Motion Product
Pluto Safety PLC

Driver configuration dialog.

Parameter	Description
Alias	Name to be used to identify nodes in network configurations. The name will be added as a prefix to each tag name imported for each network node.
Node ID	Modbus node of the slave device.
Timeout (ms)	Defines the time inserted by the protocol between two retries of the same message in case of missing response from the server device. It is expressed in milliseconds.
Delay (ms)	This parameter defines a fixed time delay in the communication between the end of the last received frame and the starting of a new request; when set to 0, the new request will be issued as soon as the internal system is able to reschedule it.
Num of repeats	This parameter defines the number of times a certain message will be sent to the controller before reporting the communication error status. A value of 1 for the parameter “No of repeats” means that the Control Panel will eventually report the communication error status if the response to the first request packet is not correct.
Transmission Mode	RTU is standard transmission mode for most PLC models. Some PLC models, e.g. Pluto Safety PLC, require the transmission mode ASCII.
PLC Models	The list allows to select the PLC model you are going to connect to. The selection will influence the data range offset per each data type according to the specific PLC memory resources.
PLC Network	The protocol allows the connection of multiple controllers to one operator panel. To set-up multiple connections, check **PLC network** checkbox and enter the node ID per each slave you need to access.



☑ PLC Network

Alias	
Node ID	-1
Timeout(ms)	1000
delay	0
Num of repeats	2
Transmission Mode	RTU

PLC Models
- ABB PM59x, PM58x, PM573
- ABB Next Move ES / ESB-2
- ABB e100 Motion Product
- ABB e150 Motion Product
- Pluto Safety PLC

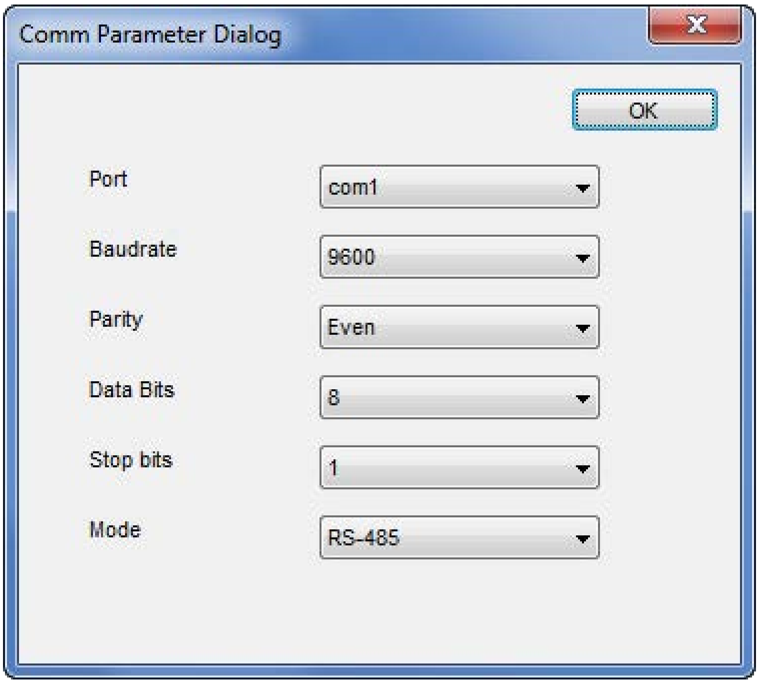
Slaves

Slave Id	Model	Alias

Alias	
Node ID	1
Timeout(ms)	1000
delay	0
Num of repeats	2

PLC Models
- ABB AC31 Series 90
- ABB AC500-eCo
- ABB PM571, PM572
- ABB PM59x, PM58x, PM573
- ABB Next Move ES / ESB-2
- ABB e100 Motion Product

The **Comm... button** displays the communication parameters setup dialog.



Port com1

Baudrate 9600

Parity Even

Data Bits 8

Stop bits 1

Mode RS-485

Parameter	**Description**
Port	Serial port selection. COM1 is the CP600 PLC port, COM2 is the PC/printer port.
Baud rate, Parity, Data bits, Stop bits	Communication parameters for the serial line.
Mode	Serial port mode. Can be selected between: RS-232, RS-485 (2 wires), RS-422 (4 wires).

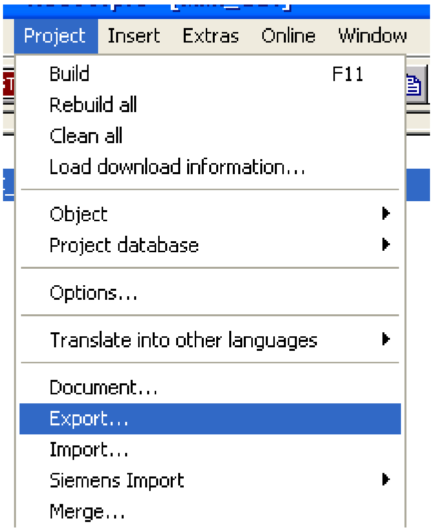
Tag Import

This special Modbus driver supports tag import.

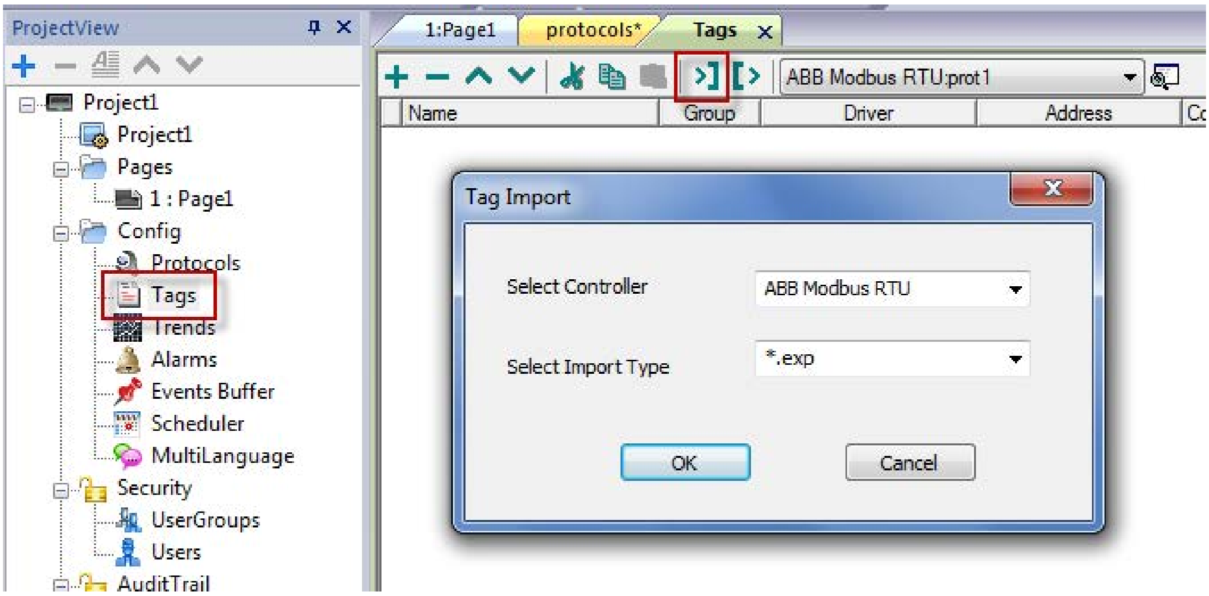
The ABB controllers are programmable with a programming tool called *Control Builder Plus* which is based on the CoDeSys V2.3 soft PLC.

The tag importer supports the CoDeSys export file in “exp” format.

In the CoDeSys programming software the *Export* command is available under the *Project* menu as shown in the following figure.



1. In the tag editor select the driver and click on the **Import tag** button to start the importer.

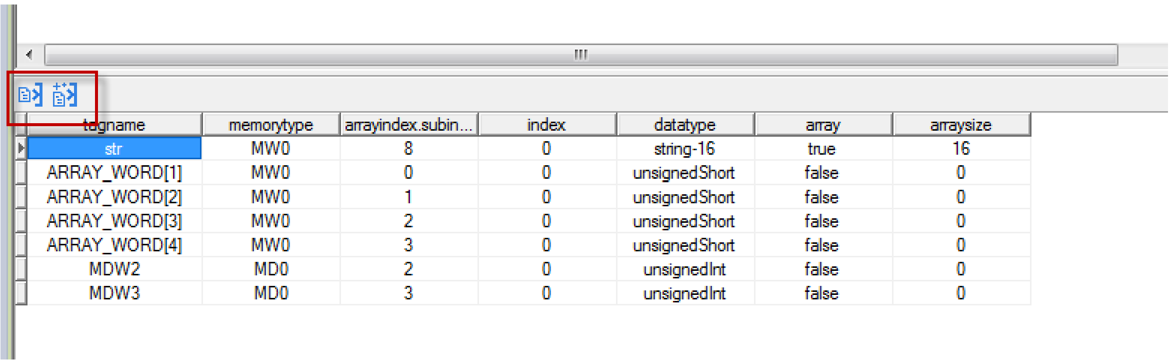


Select Controller: ABB Modbus RTU

Select Import Type: *.exp

2. Locate the “exp” file and confirm with **OK**.

The tags present in the exported document are listed in the tag dictionary from where they can be directly added to the project using the **add tags** button as shown in the following figure.

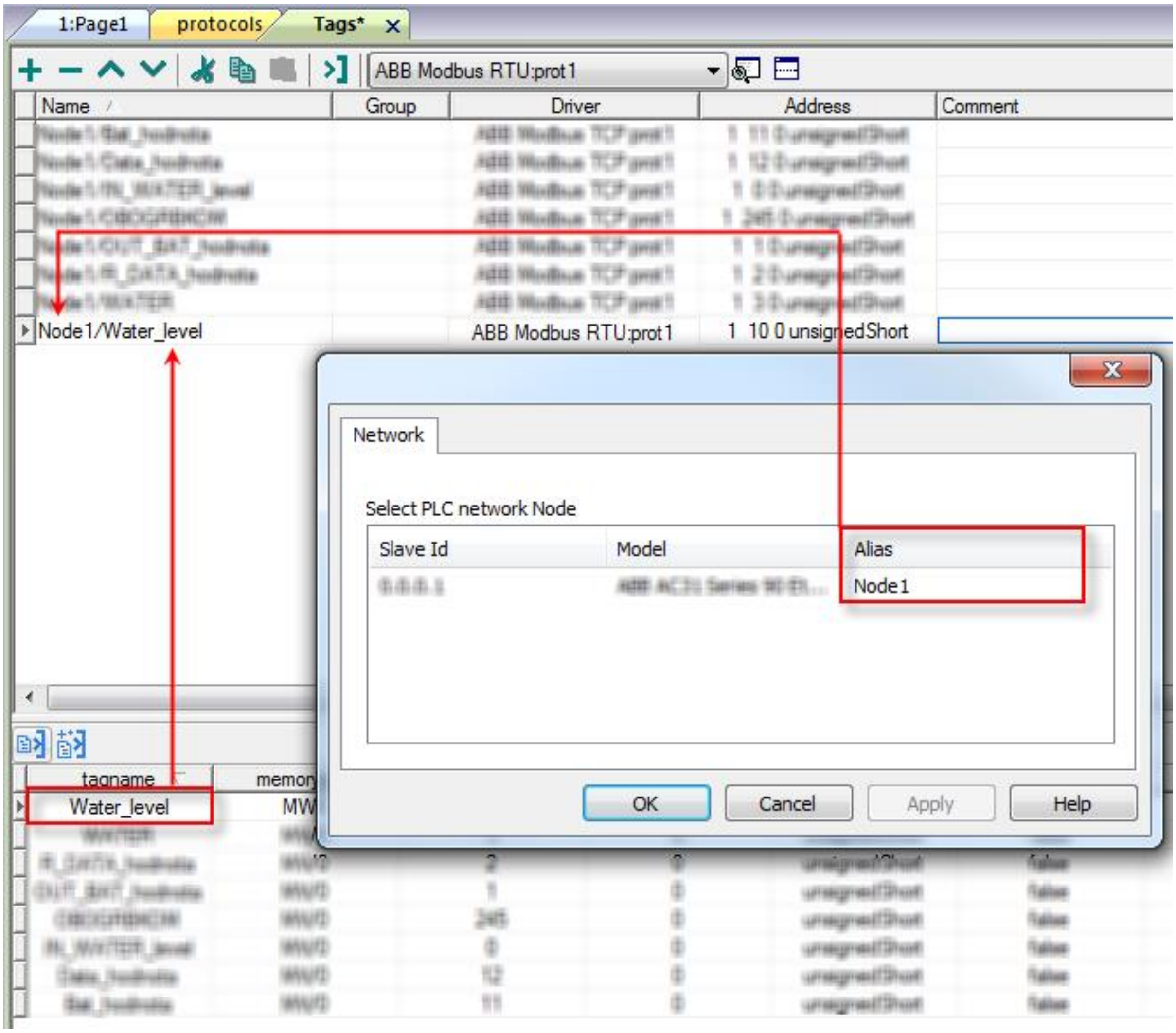


tagname	memorytype	arrayindex.subin...	index	datatype	array	arraysize
str	MW0	8	0	string-16	true	16
ARRAY_WORD[1]	MW0	0	0	unsignedShort	false	0
ARRAY_WORD[2]	MW0	1	0	unsignedShort	false	0
ARRAY_WORD[3]	MW0	2	0	unsignedShort	false	0
ARRAY_WORD[4]	MW0	3	0	unsignedShort	false	0
MDW2	MD0	2	0	unsignedInt	false	0
MDW3	MD0	3	0	unsignedInt	false	0

Aliasing Tag Names in Network Configurations

Tag names must be unique at project level; it often happens that the same tag names are to be used for different controller nodes (for example when the HMI is connected to two devices that are running the same application). Since tags include also the identification of the node and tag editor does not support duplicate tag names, the import facility in tag editor has an aliasing feature that can automatically add a prefix to imported tags. With this feature tag names can be done unique at project level.

The feature works when importing tags for a specific protocol. Each tag name will be prefixed with the string specified by the “Alias”. As shown in the figure below, the connection to a certain controller is assigned the name “Node1”. When tags are imported for this node, all tag names will have the prefix “Node1” making each of them unique at the network/project level.



Name	Group	Driver	Address	Comment
Node1/Water_level		ABB Modbus RTU:prot1	1 10 0 unsignedShort	

Select PLC network Node

Slave Id	Model	Alias
[illegible]	[illegible]	Node1

tagname	memory
Water_level	MW

NOTE

Aliasing tag names is only available when tags can be imported. Tags which are added manually in the tag editor do not need to have the Alias prefix in the tag name.

The Alias string is attached to the tag name only at the moment the tags are imported using tag editor. If you modify the Alias string after the tag import has been completed, there will be no effect on the names already present in the dictionary. When the Alias string is changed and tags are imported again, all tags will be imported again with the new prefix string.

Special Data Types

The ABB Modbus RTU driver provides one special data type called "Node Override ID".

The Node Override allows changing at runtime the target controller ID. This memory type is an unsigned byte.

The Node Override ID is initialized with the value of the controller ID specified in the project at programming time.

If the Node Override is set to 0, all the communication with the slave is stopped, no request frames are generated anymore.

If the Node Override has a value different from 0, it is interpreted as Node ID Override and the target ID address is replaced runtime with the new value.

In case the panel has been configured to access to a network of controllers, each node has its own override variable.

NOTE

The Node Override values assigned at runtime are not retained among power cycles.

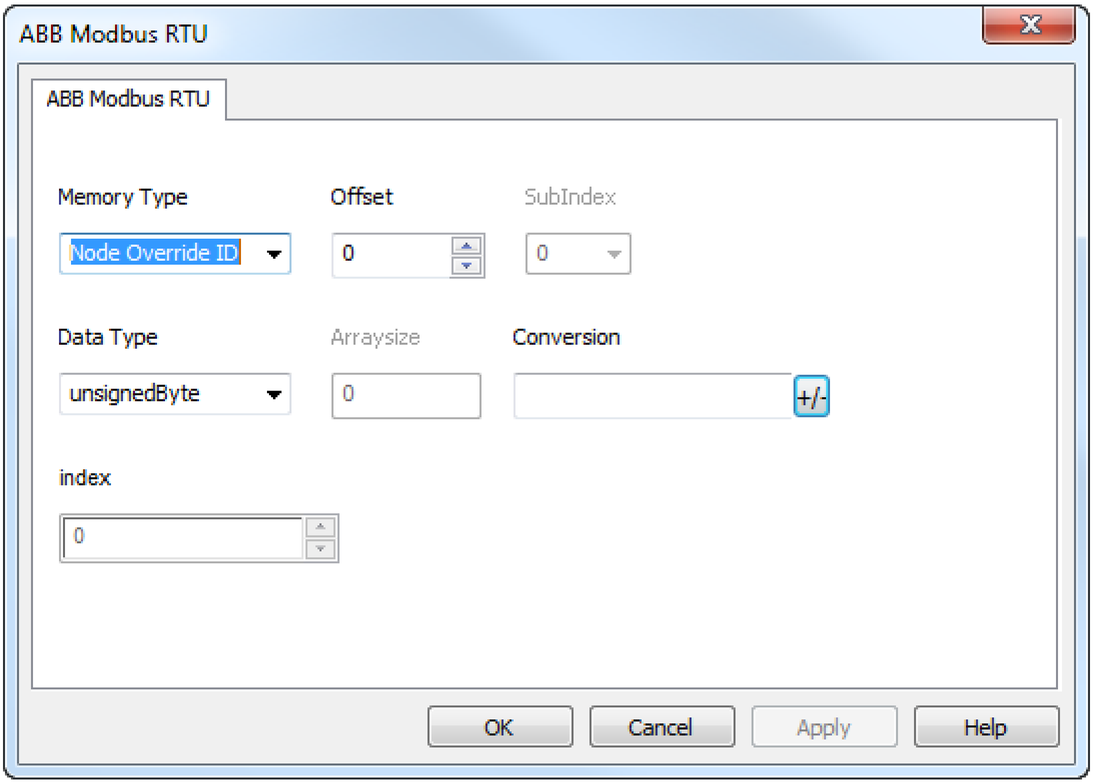


ABB Modbus RTU

Memory Type: Node Override ID

Offset: 0

SubIndex: 0

Data Type: unsignedByte

Arraysize: 0

Conversion:

index: 0

Communication Status

The current communication status can be displayed using the dedicated system variables. Please refer to the chapter System Variables about available types and their use.

The codes supported for this communication driver are:

Error	Notes
NAK	Returned in case the controller replies with a not acknowledge.
Timeout	Returned when a request is not replied within the specified timeout period; ensure the controller is connected and properly configured to get network access.
Line Error	Returned when an error on the communication parameter setup is detected (parity, baud rate, data bits, stop bits). Ensure the communication parameter settings of the controller is compatible with panel communication setup.
Invalid response	The Control Panel did receive from the controller a response, but its format or its contents is not as expected; ensure the data programmed in the project are consistent with the controller resources.
General Error	Error cannot be identified. Should never be reported. Contact technical support.

Implementation Details

This Modbus RTU implementation supports only a subset of the standard Modbus function codes.

The supported function codes are listed in the table below.

Code	Function	Description
01	Read Coil Status	Read multiple bits.
02	Read Input Status	Read the ON/OFF status of the discrete inputs (1 x reference) in the slave.
03	Read Holding Registers	Read multiple registers.
04	Read Input Registers	Read the binary contents of input registers (3 x reference) in the slave.
05	Force Single Coil	Force a single coil to either ON or OFF.
06	Preset Single Register	Preset a value in a register.
16	Preset Multiple Registers	Preset value in multiple registers.

Communication speed with controllers is supported up to 57600 baud.

Floating point data format is compliant to the IEEE standard.

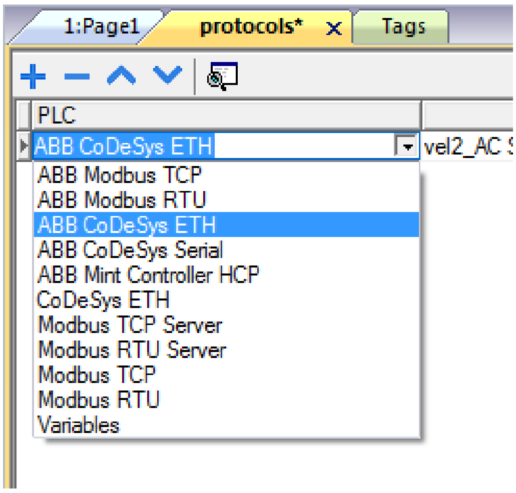
ABB CoDeSys Ethernet Driver

The ABB CoDeSys ETH communication driver for Ethernet has been specifically designed to support communication with ABB controllers series AC500 designed for standardized IEC 61131-3 programming, based on the CoDeSys V2.3 system.

NOTE Note that changes in the controller protocol or hardware, which may interfere with the functionality of this driver, may have occurred since this documentation was created. Therefore, always test and verify the functionality of the application. To accommodate developments in the controller protocol and hardware, drivers are continuously updated. Accordingly, always ensure that the latest driver is used in the application.

Protocol Editor Settings

→ Add with the button + a driver in the protocol editor and select the protocol called ABB CoDeSys ETH from the list of available protocols.



The ABB CoDeSys ETH driver supports three different protocol types:

- Tcp/Ip Level2 Route
- ABB Tcp/Ip Level2 Route AC
- Tcp/Ip

→ Select the protocol type from the dedicated combo box in the dialog.

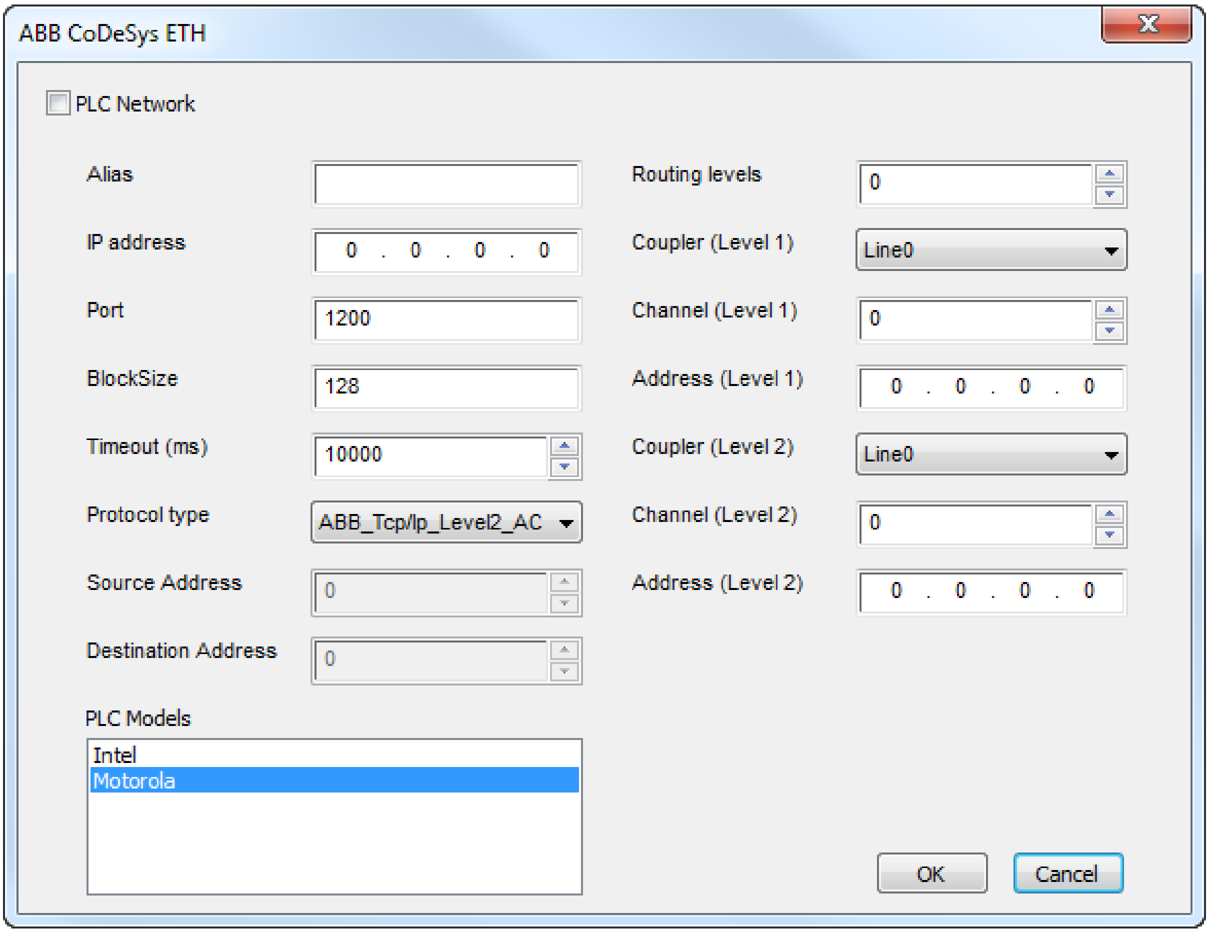


ABB CoDeSys ETH

□ PLC Network

Alias:

IP address: 0 . 0 . 0 . 0

Port: 1200

BlockSize: 128

Timeout (ms): 10000

Protocol type: ABB_Tcp/Ip_Level2_AC

Source Address: 0

Destination Address: 0

PLC Models: Intel [✓ Motorola]

Routing levels: 0

Coupler (Level 1): Line0

Channel (Level 1): 0

Address (Level 1): 0 . 0 . 0 . 0

Coupler (Level 2): Line0

Channel (Level 2): 0

Address (Level 2): 0 . 0 . 0 . 0

Some of the parameters of the dialog are common to the different protocol types, some others are specific.

The common parameters are the following:

Parameter	**Description**
Alias	Name to be used to identify nodes in network configurations. The name will be added as a prefix to each tag name imported for each network node.
IP address	Ethernet IP address of the controller.
Port	This parameter allows changing the port number used for the communication. Default value for this parameter is set to 1200 for ABB drivers. For AC500 and 3S drivers select port 1201.
Block Size	Enter the max. block size supported by your controller.
Timeout	The number of milliseconds between retries when communication fails.
PLC Model	Defines the byte order that will be used by the communication driver when sending communication frames to the PLC; Intel is also commonly referred as “little-endian”, Motorola as “big-endian”.
Protocol type	Shows a list with the available protocol variants. Please make sure you check what protocol variant is supported by the CoDeSys runtime you want to connect.
PLC Network	The protocol allows the connection of multiple controllers to one operator panel. To set-up multiple connections, check **Access Multiple PLC’s** checkbox and enter IP address for all controllers.

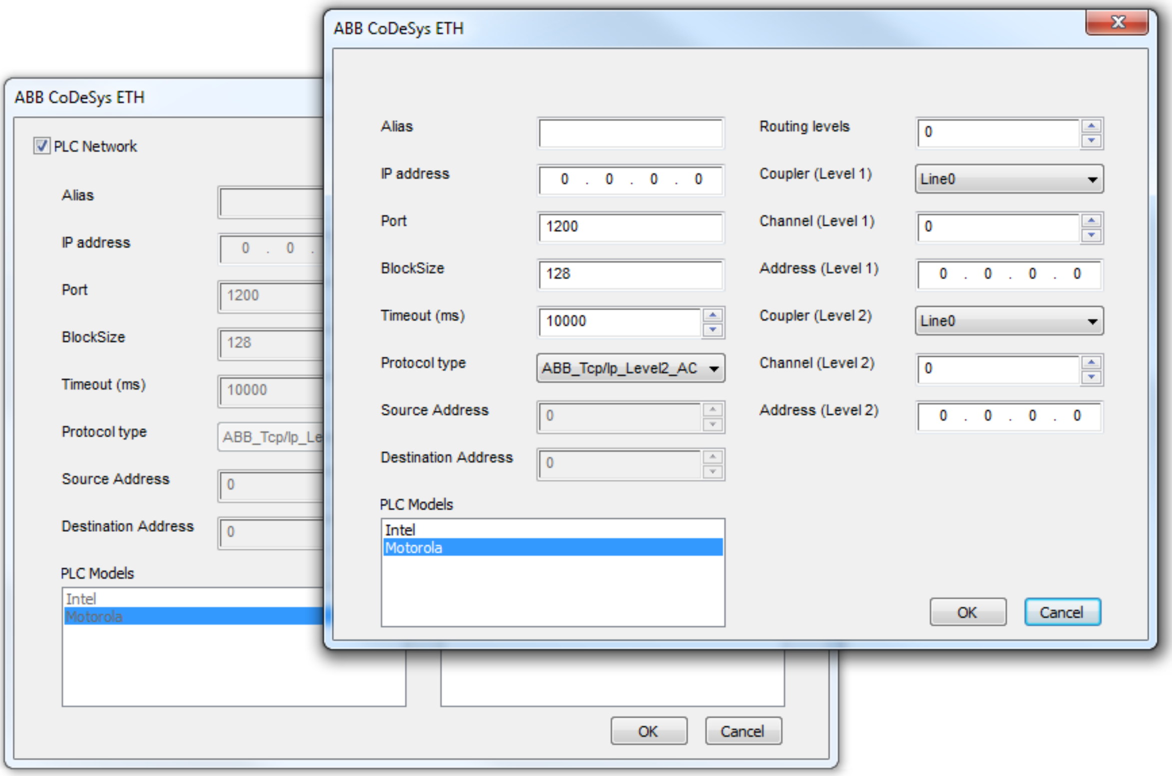


ABB CoDeSys ETH

Alias		Routing levels	0
IP address	0 . 0 . 0 . 0	Coupler (Level 1)	Line0
Port	1200	Channel (Level 1)	0
BlockSize	128	Address (Level 1)	0 . 0 . 0 . 0
Timeout (ms)	10000	Coupler (Level 2)	Line0
Protocol type	ABB_Tcp/Ip_Level2_AC	Channel (Level 2)	0
Source Address	0	Address (Level 2)	0 . 0 . 0 . 0
Destination Address	0		

PLC Models: Intel, Motorola (selected)

The **Tcp/Ip** protocol type corresponds to the 3S Level 4 driver. It does require additional setup except for the common parameters.

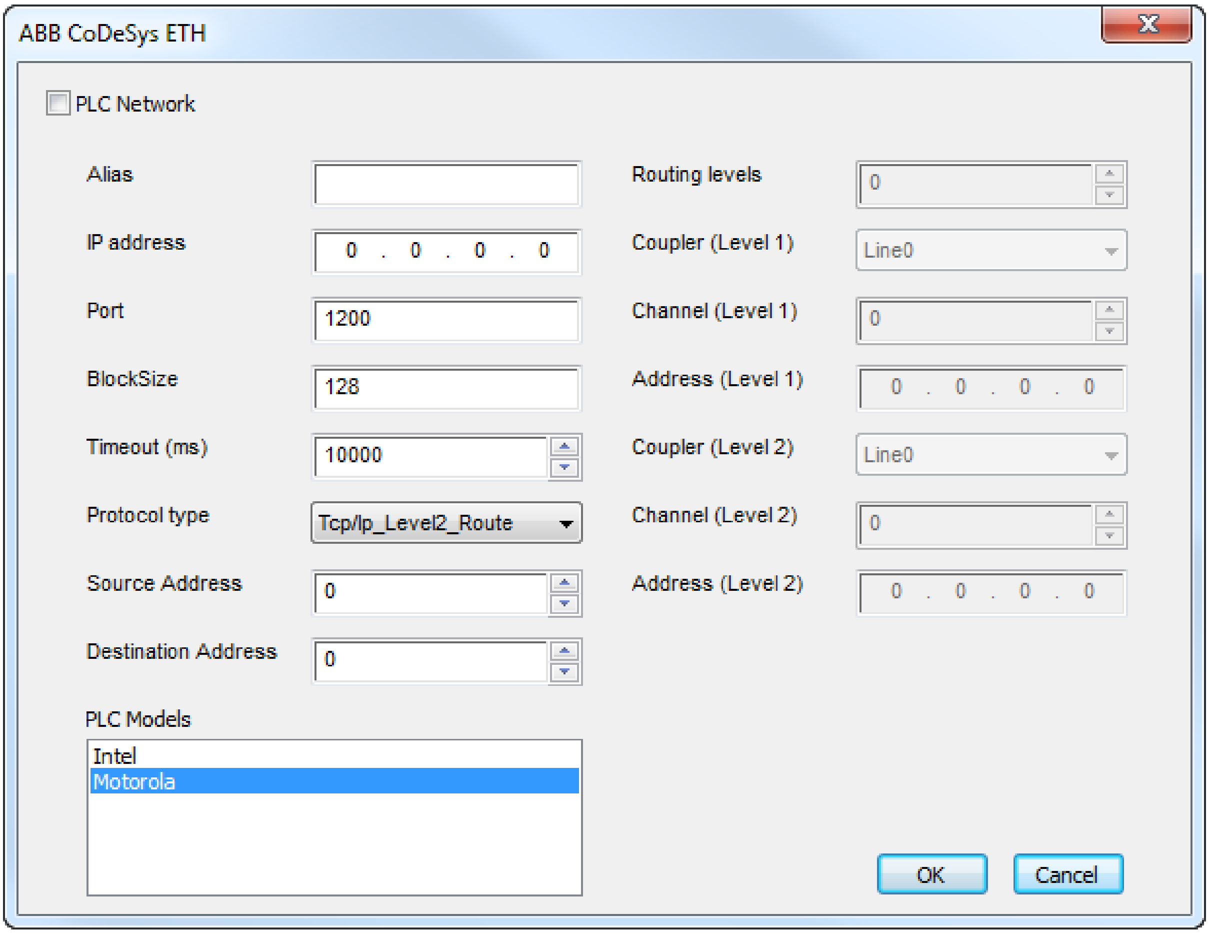


ABB CoDeSys ETH

☐ PLC Network

Alias		Routing levels	0
IP address	0 . 0 . 0 . 0	Coupler (Level 1)	Line0
Port	1200	Channel (Level 1)	0
BlockSize	128	Address (Level 1)	0 . 0 . 0 . 0
Timeout (ms)	10000	Coupler (Level 2)	Line0
Protocol type	Tcp/Ip_Level2_Route	Channel (Level 2)	0
Source Address	0	Address (Level 2)	0 . 0 . 0 . 0
Destination Address	0		

PLC Models: Intel, Motorola (selected)

The **Tcp/IP Level2 Route** protocol type corresponds to the standard 3S Level 2 route driver. For this protocol type two additional parameters are requested:

Parameter	Description
Source address (SrcAdr) & Destination address	The Destination is the node of the PLC and allows the protocol to read variables in a sub-network. The address is used to read variables when multiple PLCs are connected in a sub-network (serial network) but only one of it has the Ethernet interface. This is currently not applicable for AC500 PLCs.

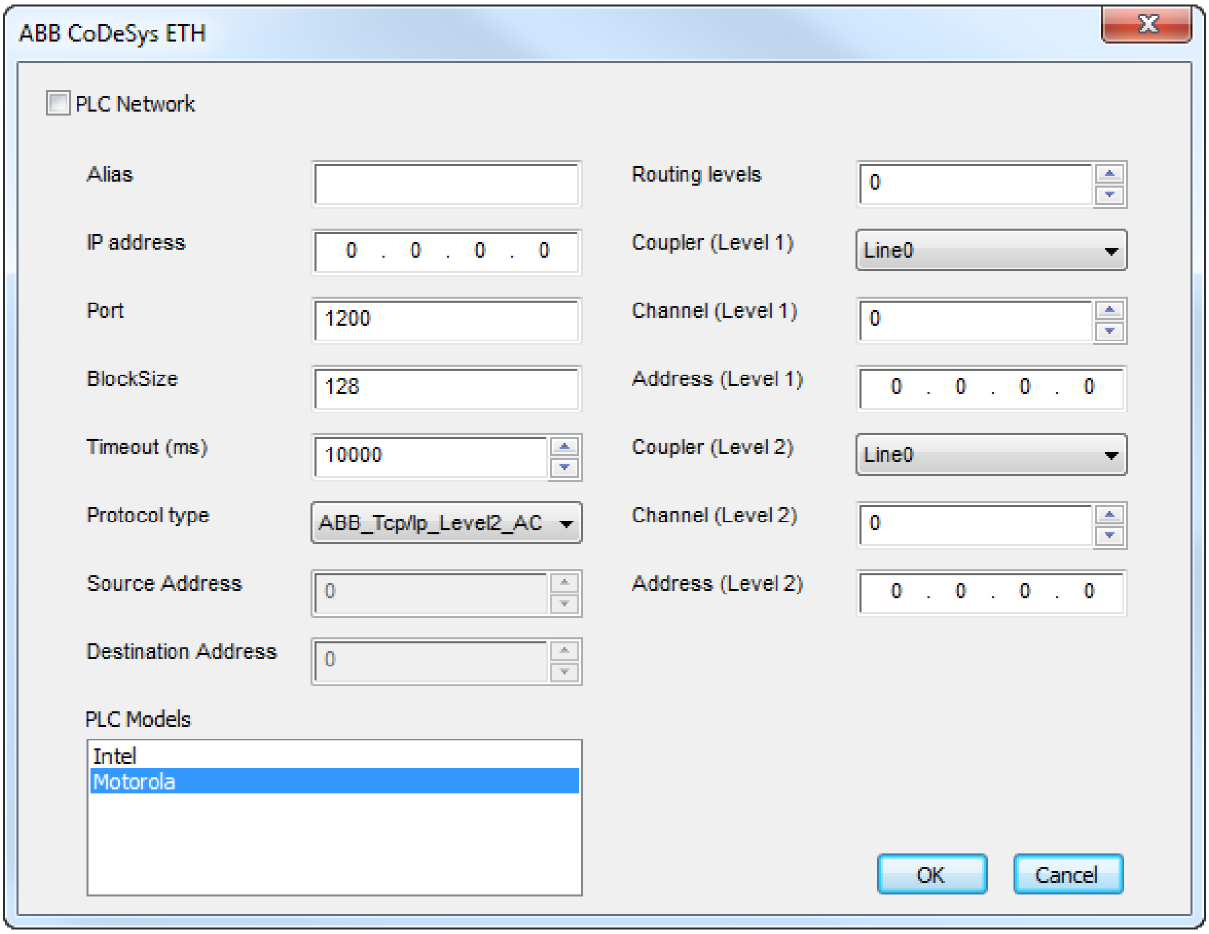


ABB CoDeSys ETH

□ PLC Network

Alias		Routing levels	0
IP address	0 . 0 . 0 . 0	Coupler (Level 1)	Line0
Port	1200	Channel (Level 1)	0
BlockSize	128	Address (Level 1)	0 . 0 . 0 . 0
Timeout (ms)	10000	Coupler (Level 2)	Line0
Protocol type	ABB_Tcp/Ip_Level2_AC	Channel (Level 2)	0
Source Address	0	Address (Level 2)	0 . 0 . 0 . 0
Destination Address	0		

PLC Models

Intel
Motorola

The **ABB Tcp/Ip Level2 AC** protocol type implements a specific variation of the standard Level 2 protocol with the additional use of a routing driver; this protocol type is normally used to connect to PLCs via other PLCs acting as gateways.

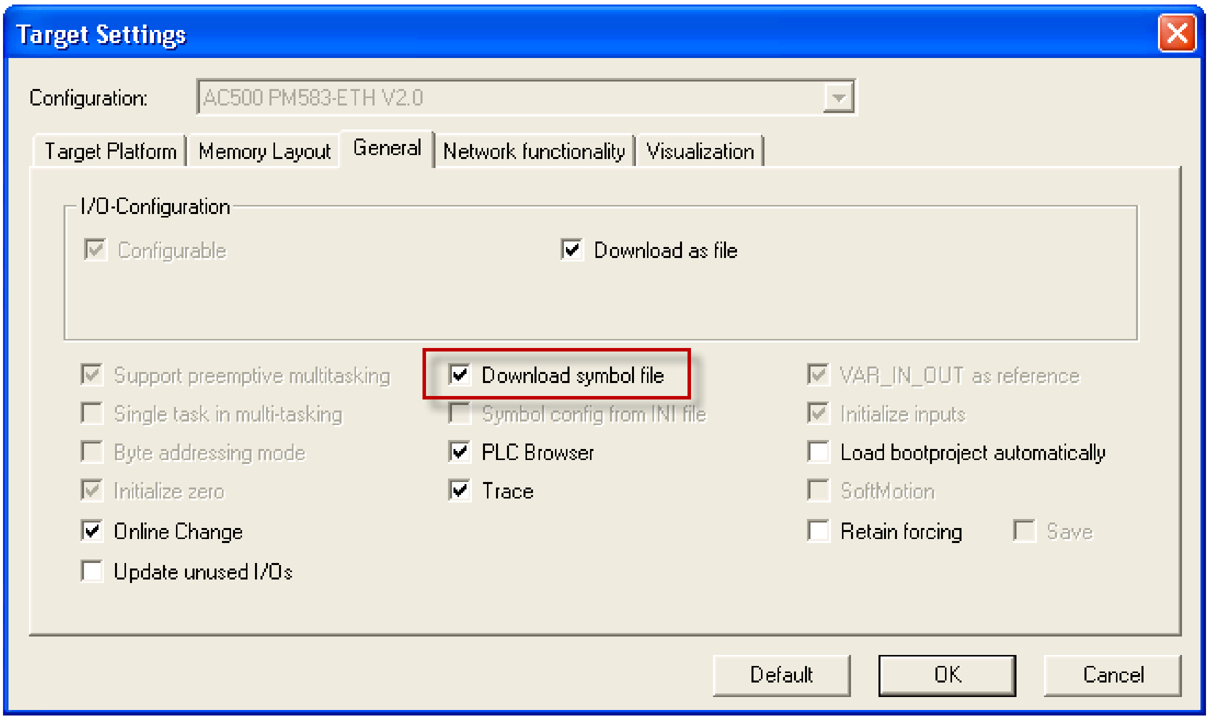
This protocol type requires the proper settings of the following additional parameters:

- Routing Levels
- Coupler (Level 1)
- Channel (Level 1)
- Address (Level 1)
- Coupler (Level 2)
- Channel (Level 2)
- Address (Level 2)

→ For detailed information refer to the AC500 and *Control Builder Plus* documentation, chapter “Programming Interfaces to the AC500 used by the *Control Builder Plus*”.

CoDeSys Software Settings

When creating the project in CoDeSys, the checkbox **Download symbol file** (*Target/Settings/General*) must be activated.



Target Settings

Configuration: AC500 PM583-ETH V2.0

I/O-Configuration

☑ Configurable

☑ Download as file

☑ Support preemptive multitasking

☑ Download symbol file

☑ VAR_IN_OUT as reference

□ Single task in multi-tasking

□ Symbol config from INI file

☑ Initialize inputs

□ Byte addressing mode

☑ PLC Browser

□ Load bootproject automatically

☑ Initialize zero

☑ Trace

□ SoftMotion

☑ Online Change

□ Retain forcing

□ Save

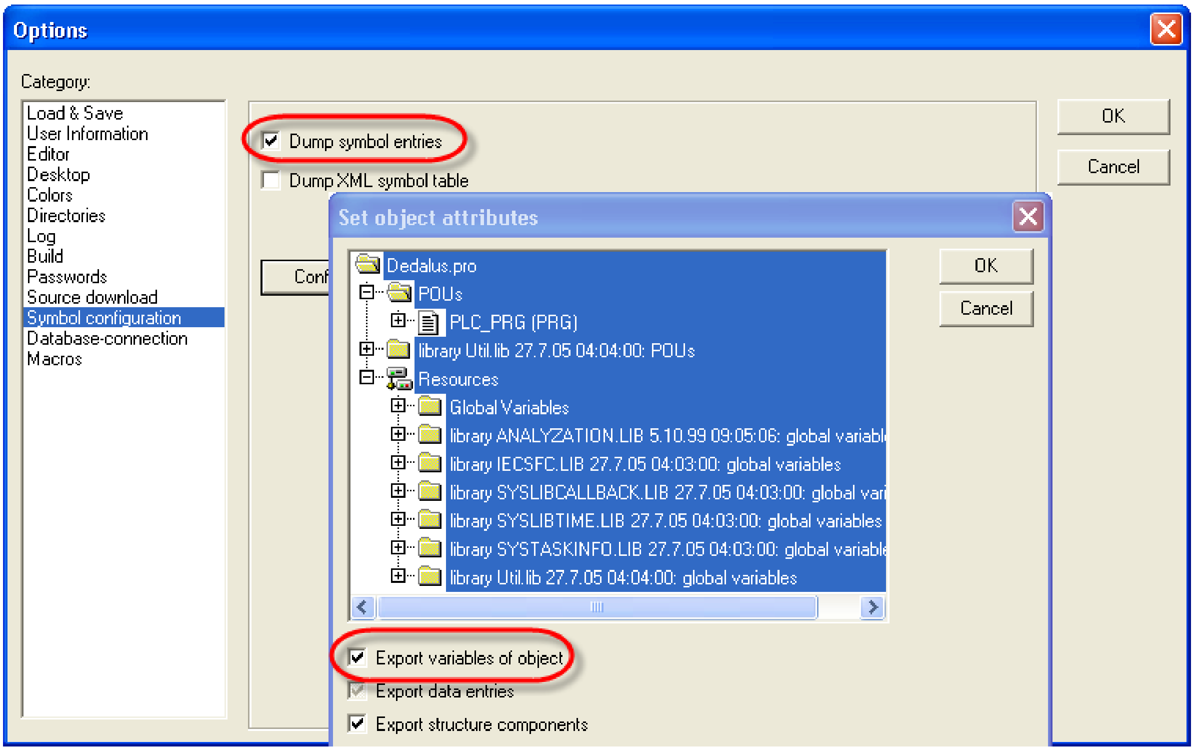
□ Update unused I/Os

NOTE ABB CoDeSys Ethernet driver supports the automatic symbol file (SDB) upload from the controller; any change in the tag offset due to new compilation on PLC software side does not require a symbol file re-import. The tag file has to be re-imported only in case of tag renaming or addition of new tags.

Tag Import

When configuring the PLC using the manufacturer's configuration software, make sure to enable *symbol file creation* (file with .SYM extension). It can be done under the CoDeSys programming software as follows:

1. Select *Project/Option/Symbol configuration* and activate the check box **Dump symbol entries.**



Options

Category:

Load & Save
User Information
Editor
Desktop
Colors
Directories
Log
Build
Passwords
Source download
Symbol configuration
Database-connection
Macros

☑ Dump symbol entries

□ Dump XML symbol table

Set object attributes

Dedalus.pro
POUs
PLC_PRG (PRG)
library Util.lib 27.7.05 04:04:00: POUs
Resources
Global Variables
library ANALYZATION.LIB 5.10.99 09:05:06: global variabl
library IECSFC.LIB 27.7.05 04:03:00: global variables
library SYSLIBCALLBACK.LIB 27.7.05 04:03:00: global vari
library SYSLIBTIME.LIB 27.7.05 04:03:00: global variables
library SYSTASKINFO.LIB 27.7.05 04:03:00: global variable
library Util.lib 27.7.05 04:04:00: global variables

☑ Export variables of object

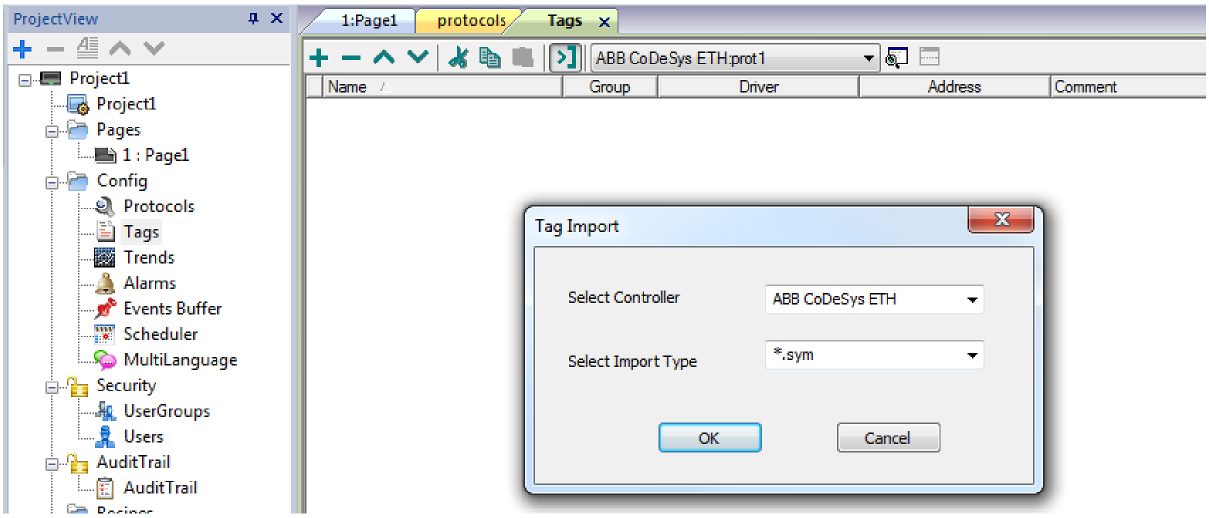
☑ Export data entries

☑ Export structure components

2. Click on the button **Configure symbol file**.

3. Make sure the **Export variables of object** check box is marked. We recommend to un-check the check box and mark it again to be sure about the proper settings.

4. Select the driver in the PB610 Panel Builder 600 Studio tag editor and click on the **Import tag** button to start the importer.

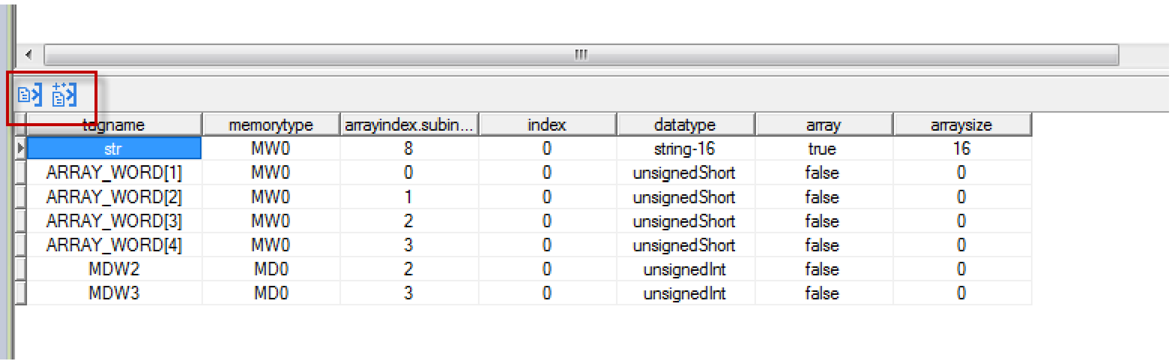


Select Controller: ABB CoDeSys ETH

Select Import Type: *.sym

5. Locate the “sym” file and confirm with **OK** button.

The tags present in the exported document are listed in the tag dictionary from where they can be directly added to the project using the add tags button as shown in the following figure.

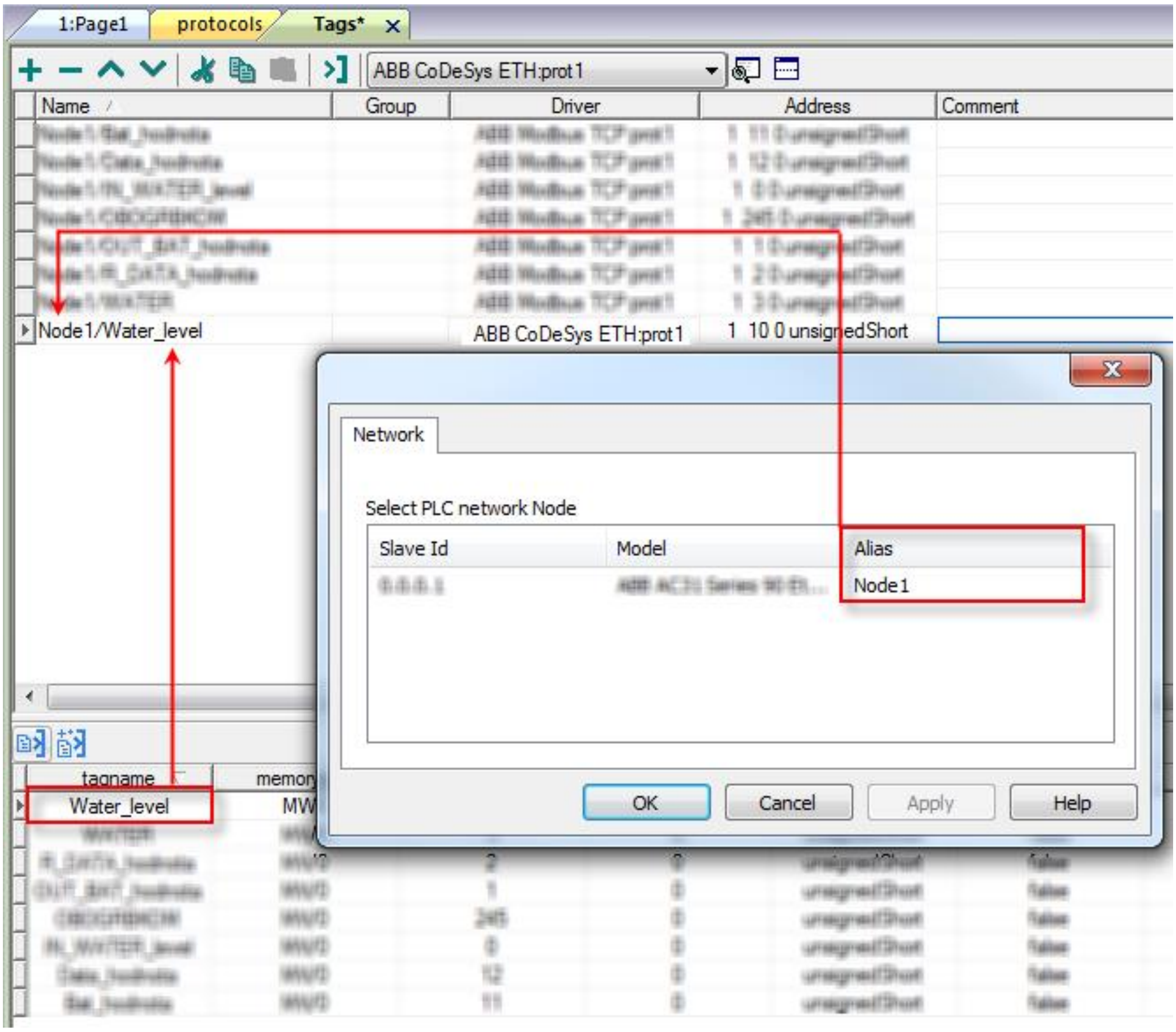


tagname	memorytype	arrayindex.subin...	index	datatype	array	arraysize
str	MW0	8	0	string-16	true	16
ARRAY_WORD[1]	MW0	0	0	unsignedShort	false	0
ARRAY_WORD[2]	MW0	1	0	unsignedShort	false	0
ARRAY_WORD[3]	MW0	2	0	unsignedShort	false	0
ARRAY_WORD[4]	MW0	3	0	unsignedShort	false	0
MDW2	MD0	2	0	unsignedInt	false	0
MDW3	MD0	3	0	unsignedInt	false	0

Aliasing Tag Names in Network Configurations

Tag names must be unique at project level; it often happens that the same tag names are to be used for different controller nodes (for example when the HMI is connected to two devices that are running the same application). Since tags include also the identification of the node and tag editor does not support duplicate tag names, the import facility in tag editor has an aliasing feature that can automatically add a prefix to imported tags. With this feature tag names can be done unique at project level.

The feature works when importing tags for a specific protocol. Each tag name will be prefixed with the string specified by the “Alias”. As shown in the figure below, the connection to a certain controller is assigned the name “Node1”. When tags are imported for this node, all tag names will have the prefix “Node1” making each of them unique at the network/project level.



Name	Group	Driver	Address	Comment
Node1/Water_level		ABB CoDeSys ETH:prot1	1 10 0 unsignedShort	

Select PLC network Node

Slave Id	Model	Alias
[illegible]	[illegible]	Node1

tagname	memory
Water_level	MW

Aliasing tag names is only available when tags can be imported. Tags which are added manually in the tag editor do not need to have the Alias prefix in the tag name.

NOTE

The Alias string is attached to the tag name only at the moment the tags are imported using tag editor. If you modify the Alias string after the tag import has been completed, there will be no effect on the names already present in the dictionary. When the Alias string is changed and tags are imported again, all tags will be imported again with the new prefix string.

Data Types

The import module supports variables of standard data types and user defined data types.

Standard Data Types

The following data types in the CoDeSys programming tool are considered as standard data types by the import module:

- BOOL
- WORD
- DWORD
- INT
- UINT
- UDINT
- DINT
- STRING
- REAL
- TIME
- DATE & TIME

and a

- 1-dimensional ARRAY of the types above.

The 64-bit data types LWORD, LINT and LREAL are not supported.

The string length of a STRING variable in the PLC should be max. 80 characters. Declare a STRING variable either with a specified size (str: STRING(35) or default size (str: STRING) which is 80 characters.

Special Data Types

The ABB CoDeSys Ethernet driver provides one special data type called "Node Override IP".

The Node Override IP allows changing at runtime the IP address of the target controller you want to connect. This memory type is an array of 4 unsigned bytes, one per each byte of the IP address.

The Node Override IP is initialized with the value of the controller IP specified in the project at programming time.

If the Node Override is set to 0.0.0.0, the communication with the slave is stopped, no request frames are generated anymore.

If the Node Override has a value different from 0.0.0.0, it is interpreted as node IP override and the target IP address is replaced runtime with the new value.

In case the panel has been configured to access to a network of controllers, each node has its own override variable.

NOTE The Node Override values assigned at runtime are not retained through power cycles

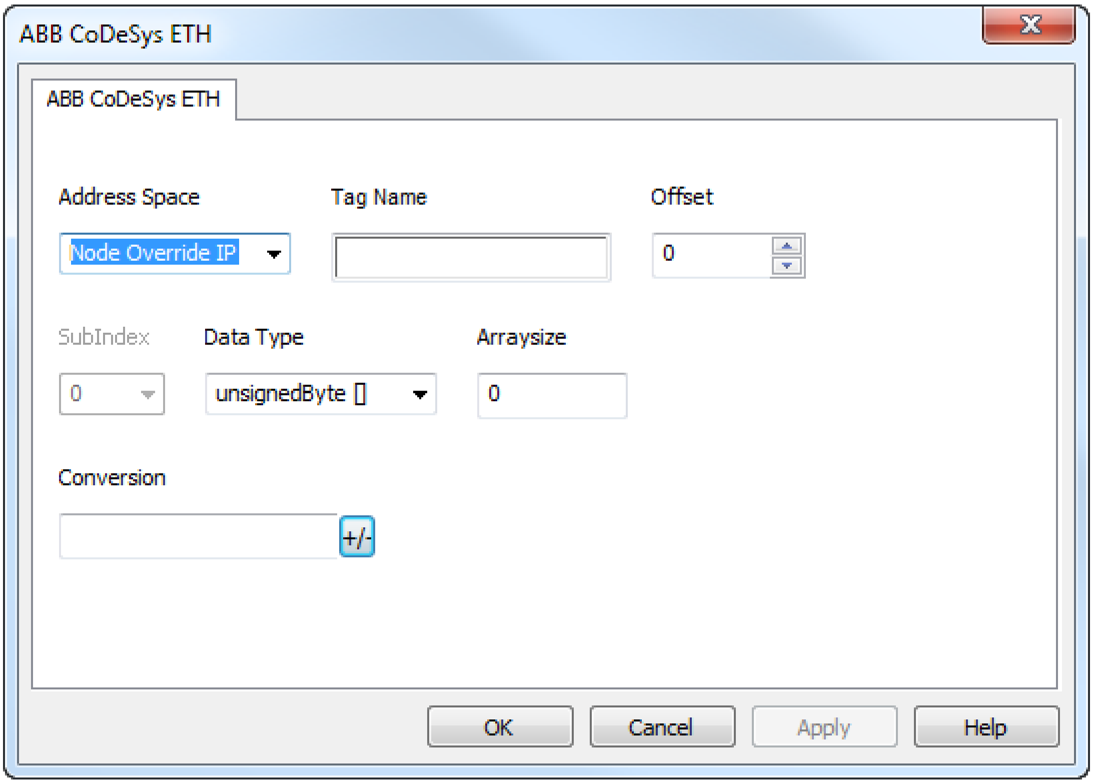


ABB CoDeSys ETH

Address Space: Node Override IP

Tag Name

Offset: 0

SubIndex: 0

Data Type: unsignedByte []

Arraysize: 0

Conversion:

Limitations

CoDeSys Level 4 is not supported. The max. block size is 1024.

Communication Status

The current communication status can be displayed using the dedicated system variables. Please refer to the chapter System Variables about available types and their use.

The codes supported for this communication driver are:

Error	Notes
Symbol file not present	Check symbol file and download again the PLC program.
“tag” not present in symbol file	Check if the tag is present in the PLC project.
Time out on Acknowledge	Controller didn’t send acknowledge.
Time out on last Acknowledge	Controller didn’t send last acknowledge.
Time out on data receiving	Controlled does not reply with data.
Connection timeout	Device not connected.

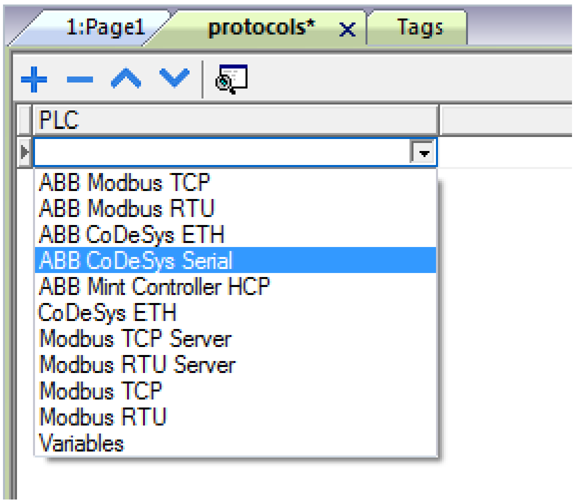
ABB CoDeSys Serial Driver

The ABB CoDeSys Serial communication driver has been specifically designed to support communication with ABB controllers series AC500 designed for standardized IEC 61131-3 programming, based on the CoDeSys V2.3 system.

NOTE Note that changes in the controller protocol or hardware, which may interfere with the functionality of this driver, may have occurred since this documentation was created. Therefore, always test and verify the functionality of the application. To accommodate developments in the controller protocol and hardware, drivers are continuously updated. Accordingly, always ensure that the latest driver is used in the application.

Protocol Editor Settings

→ Add with the button + a driver in the protocol editor and select the protocol called ABB CoDeSys Serial from the list of available protocols.



The ABB CoDeSys Serial driver supports two different protocol types:

- Serial_RS232
- ABB_RS232_AC

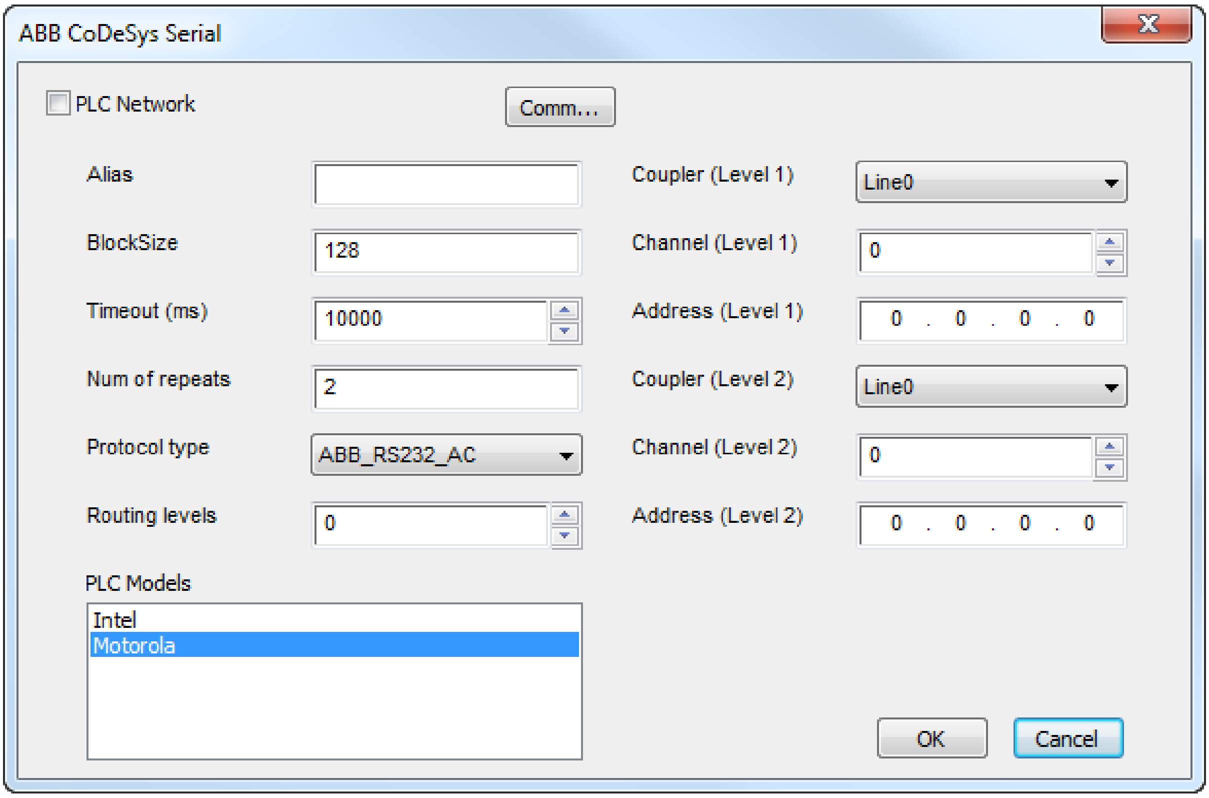


ABB CoDeSys Serial

□ PLC Network

Alias:

BlockSize: 128

Timeout (ms): 10000

Num of repeats: 2

Protocol type: ABB_RS232_AC

Routing levels: 0

PLC Models: Intel, Motorola

Coupler (Level 1): Line0

Channel (Level 1): 0

Address (Level 1): 0 . 0 . 0 . 0

Coupler (Level 2): Line0

Channel (Level 2): 0

Address (Level 2): 0 . 0 . 0 . 0

Driver configuration dialog

Parameter	**Description**
Alias	Name to be used to identify nodes in network configurations. The name will be added as a prefix to each tag name imported for each network node.
Block Size	Modbus node of the slave device.
Timeout (ms)	Defines the time inserted by the protocol between two retries of the same message in case of missing response from the server device. It is expressed in milliseconds.
Delay (ms)	This parameter defines a fixed time delay in the communication between the end of the last received frame and the starting of a new request; when set to 0, the new request will be issued as soon as the internal system is able to reschedule it.
Num of repeats	This parameter defines the number of times a certain message will be sent to the controller before reporting the communication error status. A value of 1 for the parameter “No of repeats” means that the Control Panel will eventually report the communication error status if the response to the first request packet is not correct.
PLC Models	The list allows to select the PLC model you are going to connect to. The selection will influence the data range offset per each data type according to the specific PLC memory resources.
Comm…	Gives access to the serial port configuration parameters as shown in the figure below where you can see the *Comm Parameter Dialog*.
PLC Network	The protocol allows the connection of multiple controllers to one operator panel. To set-up multiple connections, check **PLC network** checkbox and enter the node ID per each slave you need to access.

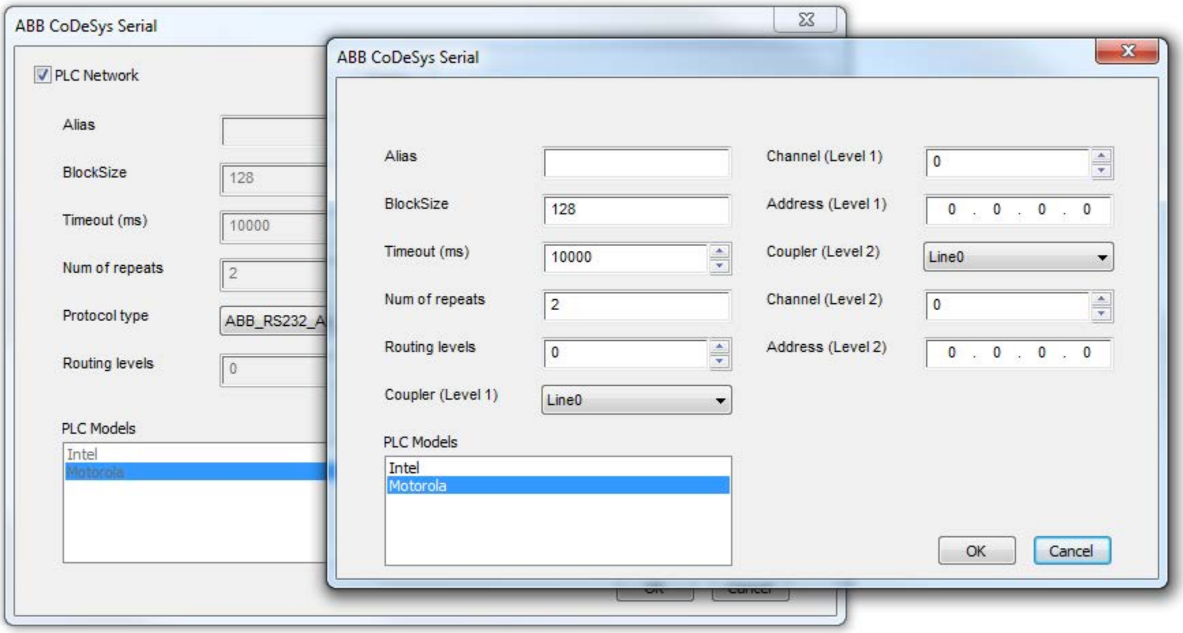


ABB CoDeSys Serial

☑ PLC Network

Alias:

BlockSize: 128

Timeout (ms): 10000

Num of repeats: 2

Protocol type: ABB_RS232_A

Routing levels: 0

PLC Models: Intel, Motorola

ABB CoDeSys Serial

Alias:

BlockSize: 128

Timeout (ms): 10000

Num of repeats: 2

Routing levels: 0

Coupler (Level 1): Line0

Channel (Level 1): 0

Address (Level 1): 0 . 0 . 0 . 0

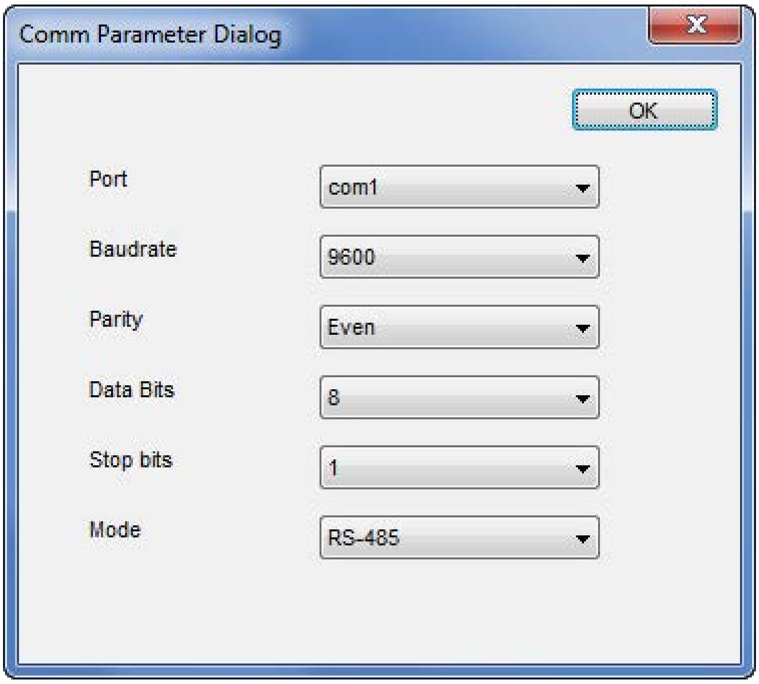
Coupler (Level 2): Line0

Channel (Level 2): 0

Address (Level 2): 0 . 0 . 0 . 0

PLC Models: Intel, Motorola

The Comm... button displays the communication parameters setup dialog.



Comm Parameter Dialog

Port: com1

Baudrate: 9600

Parity: Even

Data Bits: 8

Stop bits: 1

Mode: RS-485

Parameter	**Description**
Port	Serial port selection. COM1 is the CP600 PLC port, COM2 is the PC/printer port.
Baud rate, Parity, Data bits, Stop bits	Communication parameters for the serial line.
Mode	Serial port mode. Can be selected between: RS-232, RS-485 (2 wires), RS-422 (4 wires).

NOTE The parameter Parity has to be set to “none” for AC500!

The **Serial_RS232** protocol type corresponds to the standard 3S driver.

The **ABB_RS232_AC** protocol type implements a specific variation of the standard Level 2 protocol with the additional use of a routing driver; this protocol type is normally used to connect to PLCs via other PLCs acting as gateways.

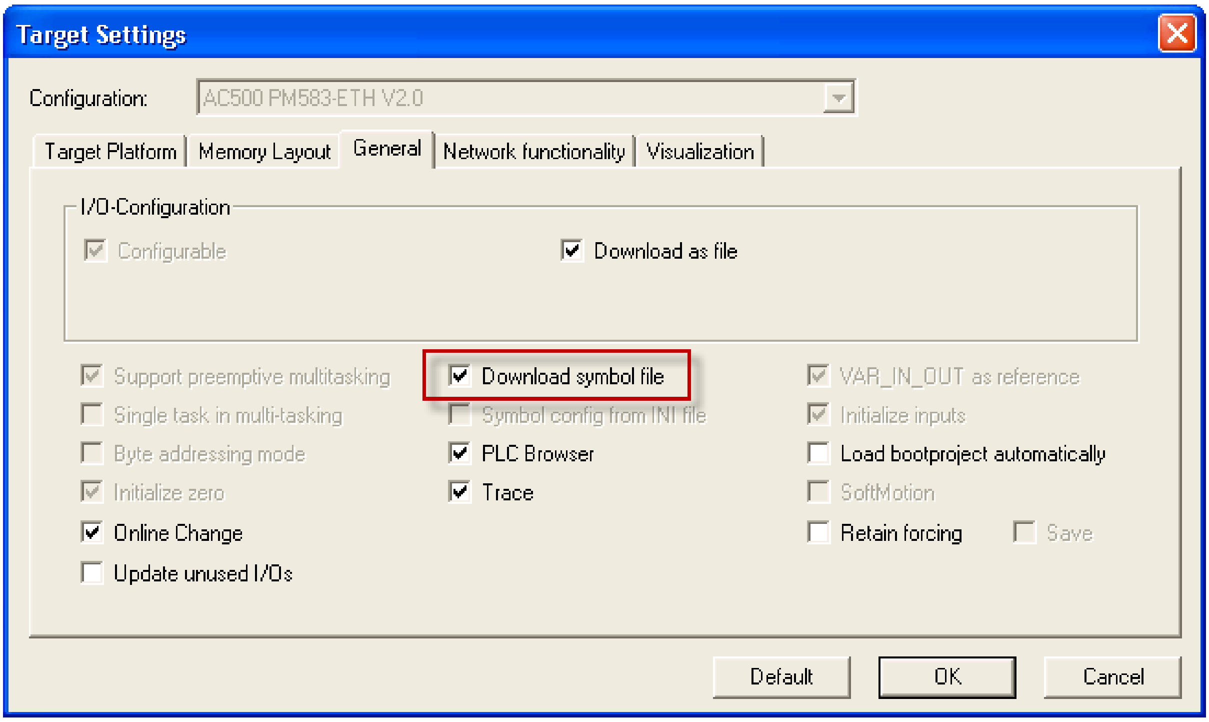
This protocol type requires the proper settings of the following additional parameters:

- Routing Levels
- Coupler (Level 1)
- Channel (Level 1)
- Address (Level 1)
- Coupler (Level 2)
- Channel (Level 2)
- Address (Level 2)

For detailed information we refer to the AC500 and *Control Builder Plus* documentation, chapter “Programming Interfaces to the AC500 used by the *Control Builder Plus*”.

CoDeSys Software Settings

When creating the project in CoDeSys, the option **Download symbol file** (under *Target/Settings/General*) must be checked.



Target Settings

Configuration: AC500 PM583-ETH V2.0

Target Platform | Memory Layout | General | Network functionality | Visualization

I/O-Configuration

☑ Configurable ☑ Download as file

☑ Support preemptive multitasking
□ Single task in multi-tasking
□ Byte addressing mode
☑ Initialize zero
☑ Online Change
□ Update unused I/Os

☑ Download symbol file
□ Symbol config from INI file
☑ PLC Browser
☑ Trace

☑ VAR_IN_OUT as reference
☑ Initialize inputs
□ Load bootproject automatically
□ SoftMotion
□ Retain forcing □ Save

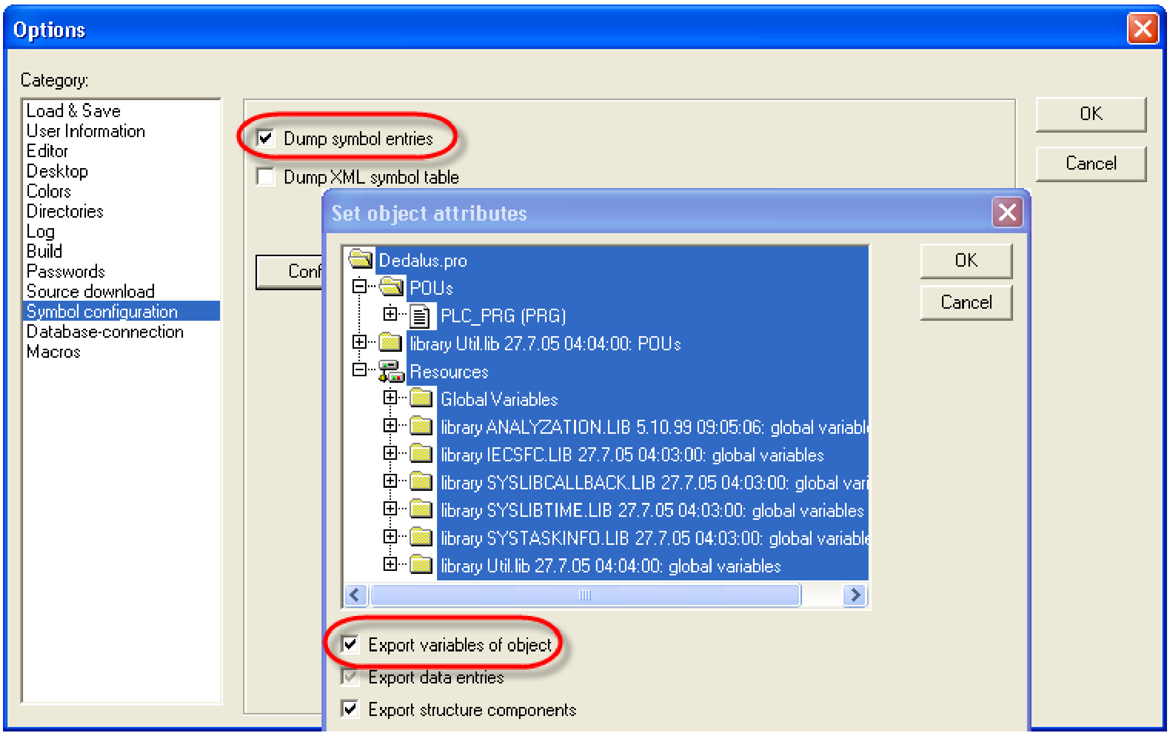
NOTE

ABB CoDeSys Serial driver supports the automatic symbol file (SDB) upload from the controller; any change in the tag offset due to new compilation on PLC software side does not require a symbol file re-import. The tag file has to be re-imported only in case of tag renaming or addition of new tags.

Tag Import

When configuring the PLC using the manufacturer’s configuration software, make sure to enable *symbol file creation* (file with .SYM extension). It can be done under the CoDeSys programming software as follows:

1. Select *Project/Option/Symbol configuration* and activate the check box **Dump symbol entries.**



Options

Category: Symbol configuration

☑ Dump symbol entries

□ Dump XML symbol table

Set object attributes

☑ Export variables of object

☑ Export data entries

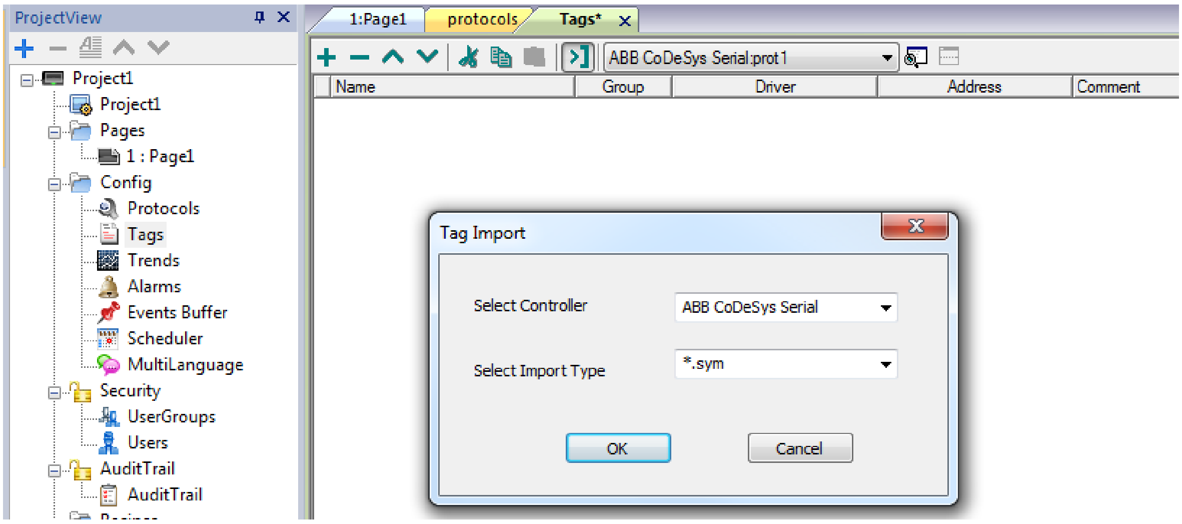
☑ Export structure components

Load & Save
User Information
Editor
Desktop
Colors
Directories
Log
Build
Passwords
Source download
Symbol configuration
Database-connection
Macros
OK
Cancel
Conf
Dedalus.pro
POUs
PLC_PRG (PRG)
library Util.lib 27.7.05 04:04:00: POUs
Resources
Global Variables
library ANALYZATION.LIB 5.10.99 09:05:06: global variabl
library IECSFC.LIB 27.7.05 04:03:00: global variables
library SYSLIBCALLBACK.LIB 27.7.05 04:03:00: global vari
library SYSLIBTIME.LIB 27.7.05 04:03:00: global variables
library SYSTASKINFO.LIB 27.7.05 04:03:00: global variabl
library Util.lib 27.7.05 04:04:00: global variables
OK
Cancel

2. Click on button **Configure symbol file.**

3. Make sure the Export variables of object check box is marked. We recommend to un-check the check box and mark it again to be sure about the proper settings.

4. Select the driver in the PB610 Panel Builder 600 Studio tag editor and click on the **Import tag** button to start the importer.



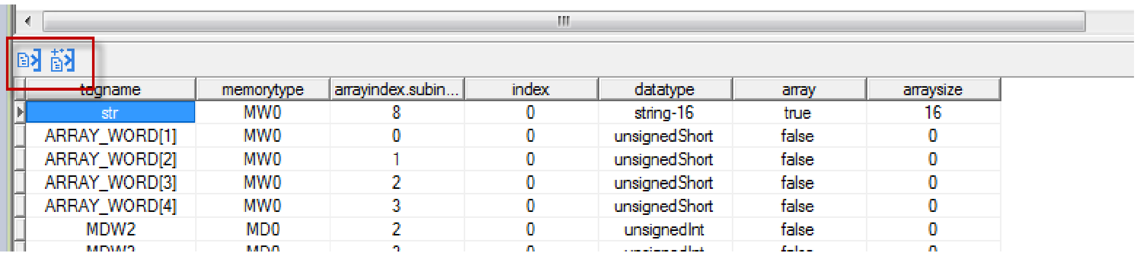
Tag Import

Select Controller: ABB CoDeSys Serial

Select Import Type: *.sym

5. Locate the “sym” file and confirm with **OK** button.

The tags present in the exported document are listed in the tag dictionary from where they can be directly added to the project using the add tags button as shown in the following figure.

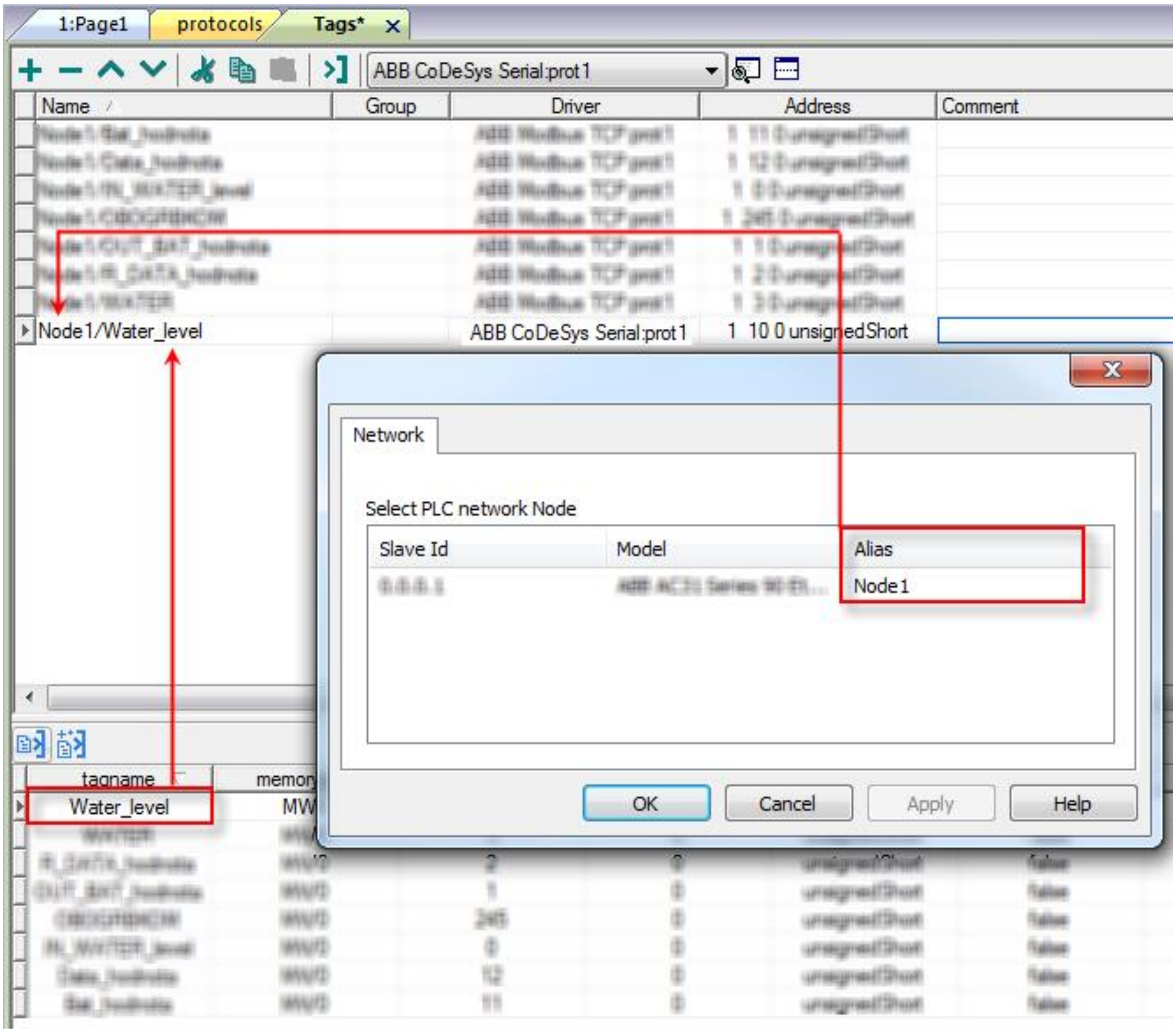


tagname	memorytype	arrayindex.subin...	index	datatype	array	arraysize
str	MW0	8	0	string-16	true	16
ARRAY_WORD[1]	MW0	0	0	unsignedShort	false	0
ARRAY_WORD[2]	MW0	1	0	unsignedShort	false	0
ARRAY_WORD[3]	MW0	2	0	unsignedShort	false	0
ARRAY_WORD[4]	MW0	3	0	unsignedShort	false	0
MDW2	MD0	2	0	unsignedInt	false	0

Aliasing Tag Names in Network Configurations

Tag names must be unique at project level; it often happens that the same tag names are to be used for different controller nodes (for example when the HMI is connected to two devices that are running the same application). Since tags include also the identification of the node and tag editor does not support duplicate tag names, the import facility in tag editor has an aliasing feature that can automatically add a prefix to imported tags. With this feature tag names can be done unique at project level.

The feature works when importing tags for a specific protocol. Each tag name will be prefixed with the string specified by the “Alias”. As shown in the figure below, the connection to a certain controller is assigned the name “Node1”. When tags are imported for this node, all tag names will have the prefix “Node1” making each of them unique at the network/project level.



Name	Group	Driver	Address	Comment
Node1/Water_level		ABB CoDeSys Serial:prot1	1 10 0 unsignedShort	

Select PLC network Node

Slave Id	Model	Alias
[illegible]	[illegible]	Node1

tagname	memory...
Water_level	MW...

NOTE

Aliasing tag names is only available when tags can be imported. Tags which are added manually in the tag editor do not need to have the Alias prefix in the tag name.

The Alias string is attached to the tag name only at the moment the tags are imported using tag editor. If you modify the Alias string after the tag import has been completed, there will be no effect on the names already present in the dictionary. When the Alias string is changed and tags are imported again, all tags will be imported again with the new prefix string.

Data Types

The import module supports variables of standard data types and user defined data types.

Standard Data Types

The following data types in the CoDeSys programming tool are considered as standard data types by the import module:

- BOOL
- WORD
- DWORD
- INT
- UINT
- UDINT
- DINT
- STRING
- REAL
- TIME
- DATE & TIME

and a

- 1-dimensional ARRAY of the types above.

The 64-bit data types LWORD, LINT and LREAL are not supported.

The string length of a STRING variable in the PLC should be max. 80 characters. Declare a STRING variable either with a specified size (str: STRING(35)) or default size (str: STRING) which is 80 characters.

Limitations

NOTE

This protocol does not support AC500 FW version earlier than V2.0.

Communication Status

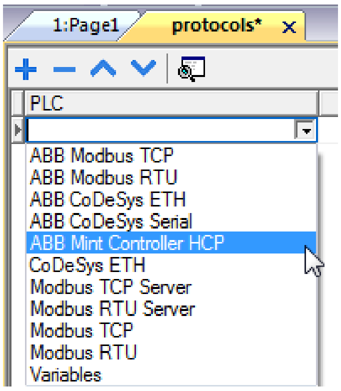
The current communication status can be displayed using the dedicated system variables. Please refer to the chapter System Variables about available types and their use.

The codes supported for this communication driver are:

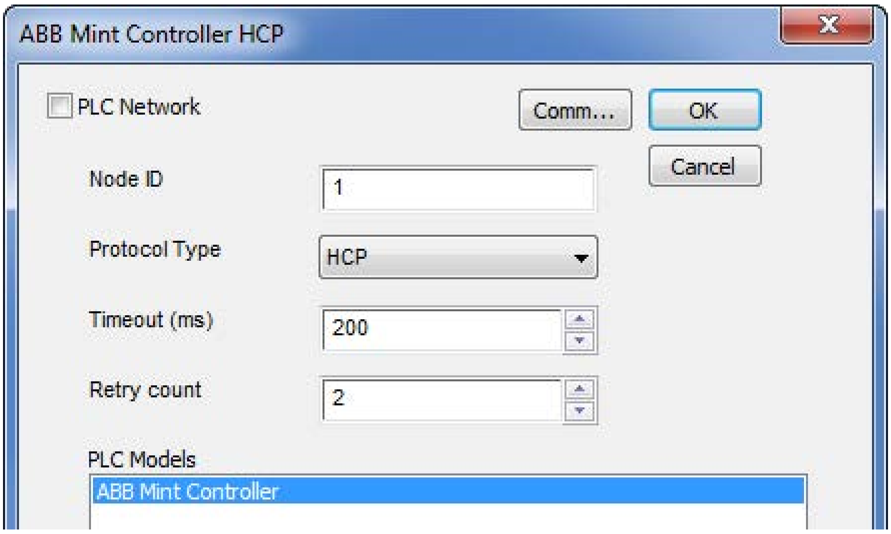
Error	Notes
Symbol file not present	Check symbol file and download again the PLC program.
“tag” not present in symbol file	Check if the tag is present in the PLC project.
Time out on Acknowledge	Controller didn’t send acknowledge.
Time out on last Acknowledge	Controller didn’t send last acknowledge.
Time out on data receiving	Controlled does not reply with data.
Connection timeout	Device not connected.

ABB Mint Controller HCP Driver

This communication protocol allows the panels to connect to the ABB motion and servo drive devices using the HCP and HCP2 communication protocols.



Settings



□ PLC Network

Node ID: 1

Protocol Type: HCP

Timeout (ms): 200

Retry count: 2

PLC Models: ABB Mint Controller

Parameter	Description
Node ID	Node ID assigned to the controller device.
Protocol Type	Protocol selection; two protocols are available: HCP and HCP2; select the one requested by the specific device
Timeout (ms)	Defines the time inserted by the protocol between two retries of the same message in case of missing response from the server device. It is expressed in milliseconds.
Retry count	This parameter defines the number of times a certain message will be sent to the controller before reporting the communication error status. A value of 1 for the parameter “No of repeats” means that the Control Panel will eventually report the communication error status if the response to the first request packet is not correct.
PLC Models	Controller model list.
PLC Network	The protocol allows the connection of multiple controllers to one operator panel. To set-up multiple connections, check **PLC network** checkbox and enter the node ID per each slave you need to access.

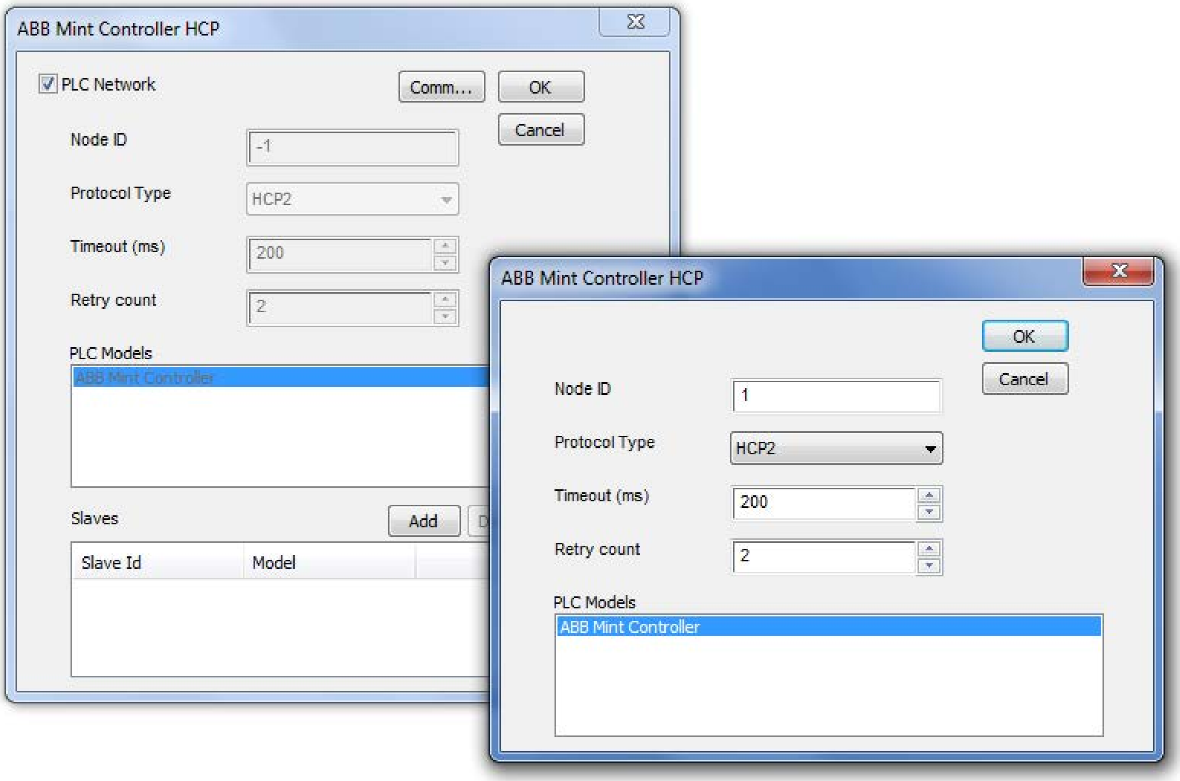


ABB Mint Controller HCP

☑ PLC Network

Node ID: -1

Protocol Type: HCP2

Timeout (ms): 200

Retry count: 2

PLC Models: ABB Mint Controller

Slaves

Slave Id	Model

ABB Mint Controller HCP

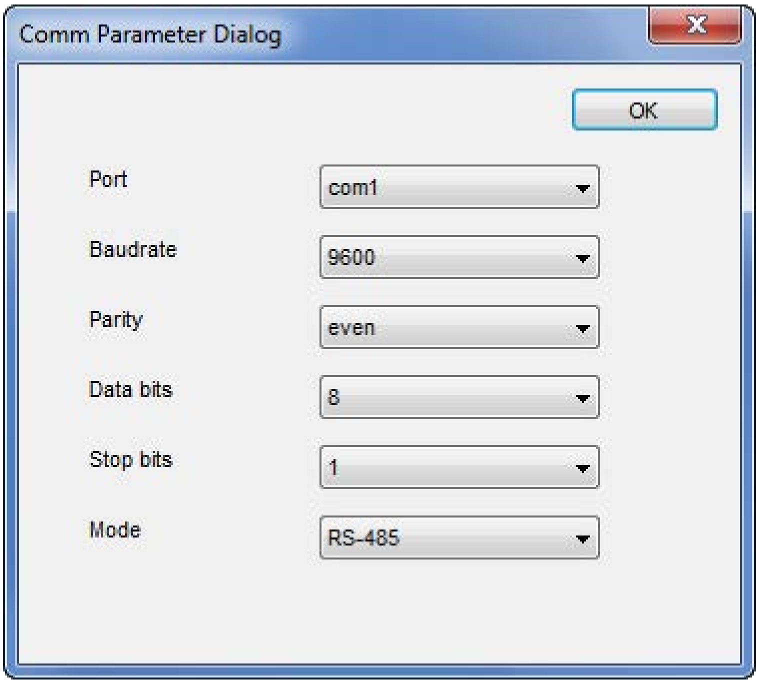
Node ID: 1

Protocol Type: HCP2

Timeout (ms): 200

Retry count: 2

PLC Models: ABB Mint Controller


Comm Parameter Dialog

Port: com1

Baudrate: 9600

Parity: even

Data bits: 8

Stop bits: 1

Mode: RS-485

Parameter	**Description**
Port	Serial port selection. COM1 is the CP600 PLC port, COM2 is the PC/printer port
Baud rate, Parity, Data bits, Stop bits	Communication parameters for the serial line.
Mode	Serial port mode. Can be selected between: RS-232, RS-485 (2 wires), RS-422 (4 wires).

Data Types

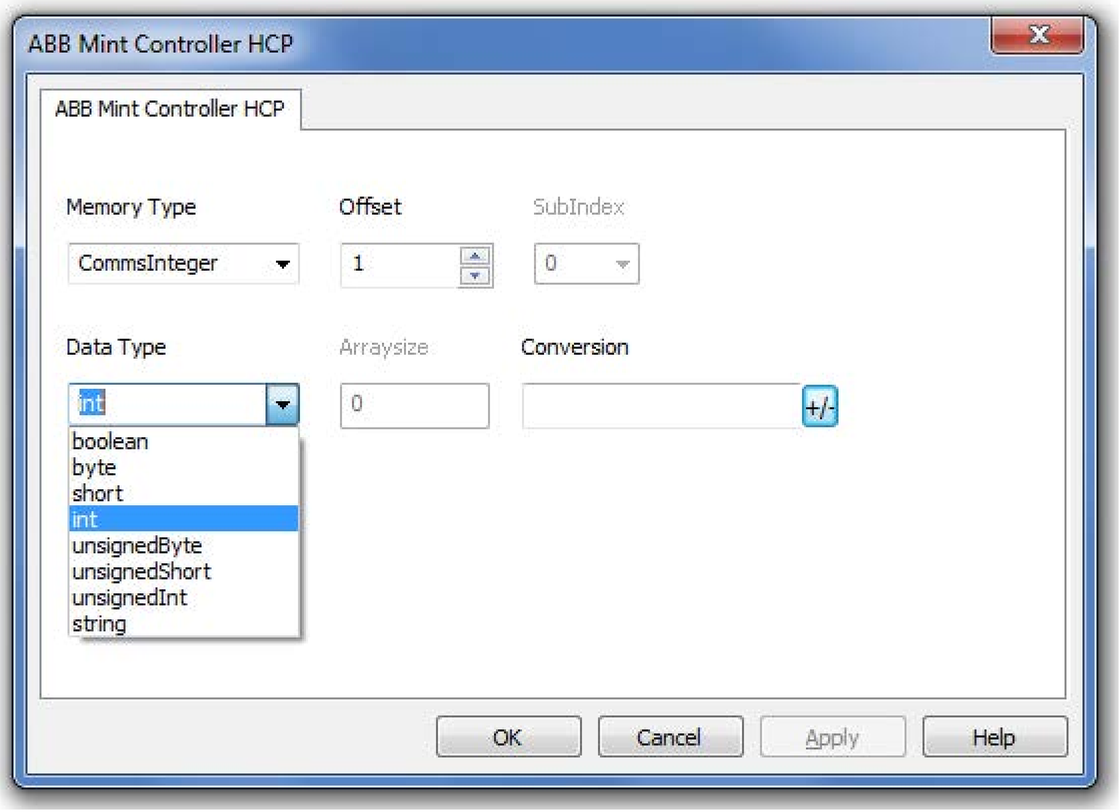
The ABB Mint Controller HCP driver provides the support for two memory types: Comms and CommsInteger.

The two memory types are actually referring to the same physical memory area in the Mint Controller. The Comms memory type should only be used with floating point values (and the Mint program on the ABB Controller should use COMMS to access this data).

The CommsInteger memory type accesses the same locations on the Mint controller but allows a variety of integer-based data types to be selected as shown in figure below.

If the Mint controller program uses the Mint COMMS keyword for a tag setup to use the CommsInteger memory type then only the bottom 23 bits will be accurate (due to floating point precision of the COMMS keyword).

If the Mint controller program uses COMMSINTEGER for a tag setup to use the CommsInteger memory type then the value is precise for the full 32 bits.



Memory Type: CommsInteger

Offset: 1

SubIndex: 0

Data Type: int

Arraysize: 0

Conversion:

Communication Status

The current communication status can be displayed using the dedicated system variables. Please refer to the User Manual for further information about available system variables and their use.

The codes supported for this communication driver are:

Error	Notes
NAK	Returned in case the controller replies with a not acknowledge.
Timeout	Returned when a request is not replied within the specified timeout period; ensure the controller is connected and properly configured to get network access.
Line Error	Returned when an error on the communication parameter setup is detected (parity, baud rate, data bits, stop bits). Ensure the communication parameter settings of the controller is compatible with panel communication setup.
Invalid response	The Control Panel did receive from the controller a response, but its format or its contents is not as expected; ensure the data programmed in the project are consistent with the controller resources.
General Error	Error cannot be identified. Should never be reported. Contact technical support.

ABB Robot Controller IRC5 driver

The ABB IRC5 robot controller communication driver has been designed for communication to the ABB Robotics robot controller family IRC5. This version supports communication towards the IRC5 robot controller I/O system (input and output signals).

Communication with the IRC5 robot controller is over a network, so the IRC5 robot controller must be available via TCP/IP.

The Communication driver cannot meet any hard real-time demands for a number of reasons:

- It executes on a non-real-time operating system.
- Communication is performed with TCP/IP over a network
- The actual controller might have more high-priority tasks to perform.

A minimal response time in the order of 10-100ms can be expected.

Please note that changes in the controller protocol or hardware, which may interfere with the functionality of this driver, may have occurred since this documentation was created. Therefore, always test and verify the functionality of the application. To accommodate developments in the controller protocol and hardware, drivers are continuously updated.

Accordingly, always ensure that the latest driver is used in the application.

Safety of personnel

A robot is heavy and extremely powerful regardless of its speed. A pause or long stop in movement can be followed by a fast hazardous movement. Even if a pattern of movement is predicted, a change in operation can be triggered by an external signal resulting in an unexpected movement. Therefore, it is important that all safety regulations are followed when entering safeguarded space.

Before beginning work with the robot, make sure you are familiar with the safety regulations as described in the IRC5 operating manuals.

ABB IRC5 Communication Driver

HMI devices can be connected to ABB robot networks as clients using this communication driver.

This implementation of the IRC5 driver provides easy handling of the connections to the ABB robot controllers providing specific support for tag import facilities.

The IRC5 driver is supported starting from version PB610 V1.90.0.778

There is a minimum requirement also for the version of operating system running in the HMI (this is normally referenced as BSP version). See in the software user manual to read the BSP version with the System Settings menu. The minimum requirements are shown in the table.

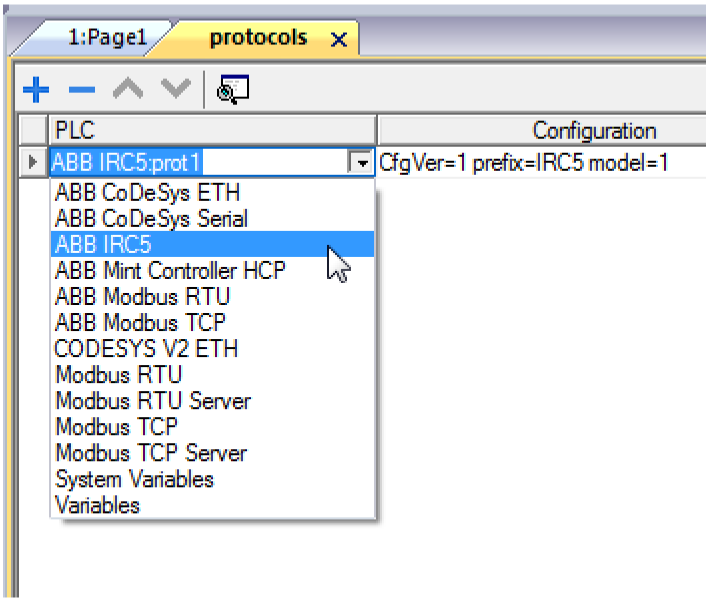
Products	Change index	Date of production	BSP version
CP650, CP660, CP675	B2 or newer	WK10 2013 (1103) or later	V2.80 or newer
CP620 … CP635	B2 or newer	WK10 2013 (1103) or later	V1.76 or newer

note

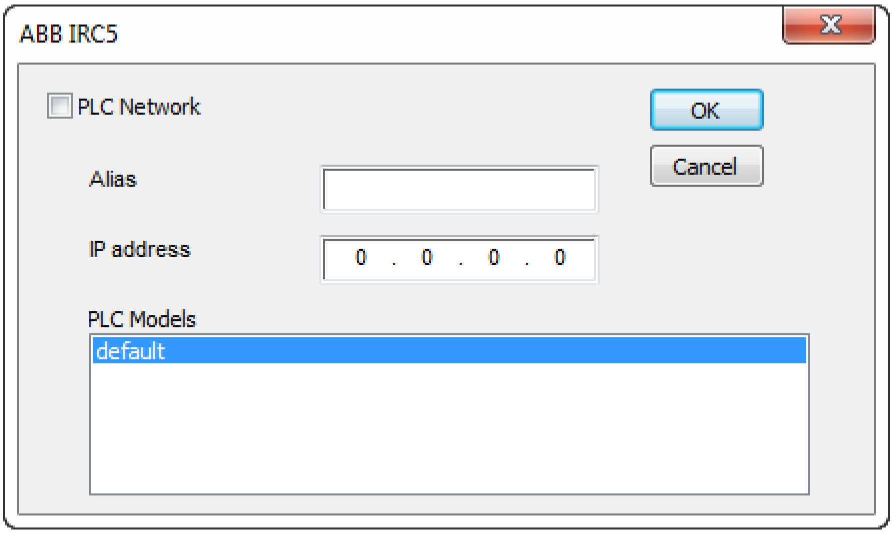
The IP submask of the control panel has to be set to 255.255.255.0. Otherwise the communication to the IRC5 controller does not work.

Protocol Editor Settings

Add (+) a driver in the Protocol editor and select the protocol called “ABB IRC5” from the list of available protocols.



The driver configuration dialog is shown in next figure.



Alias	Name to be used to identify nodes in network configurations. The name will be added as a prefix to each tag name imported for each network node
IP address	Ethernet IP address of the controller
PLC Models	Only one model type is currently available
PLC Network	The protocol allows the connection of one HMI to multiple controllers. To set-up multiple connections, check “PLC network” checkbox.

In case the HMI needs to be configured to access a network of controllers, check the "PLC Network" check box and configure the network adding the controllers as shown in figure. For each controller you add to the network you will have to specify a unique IP address.

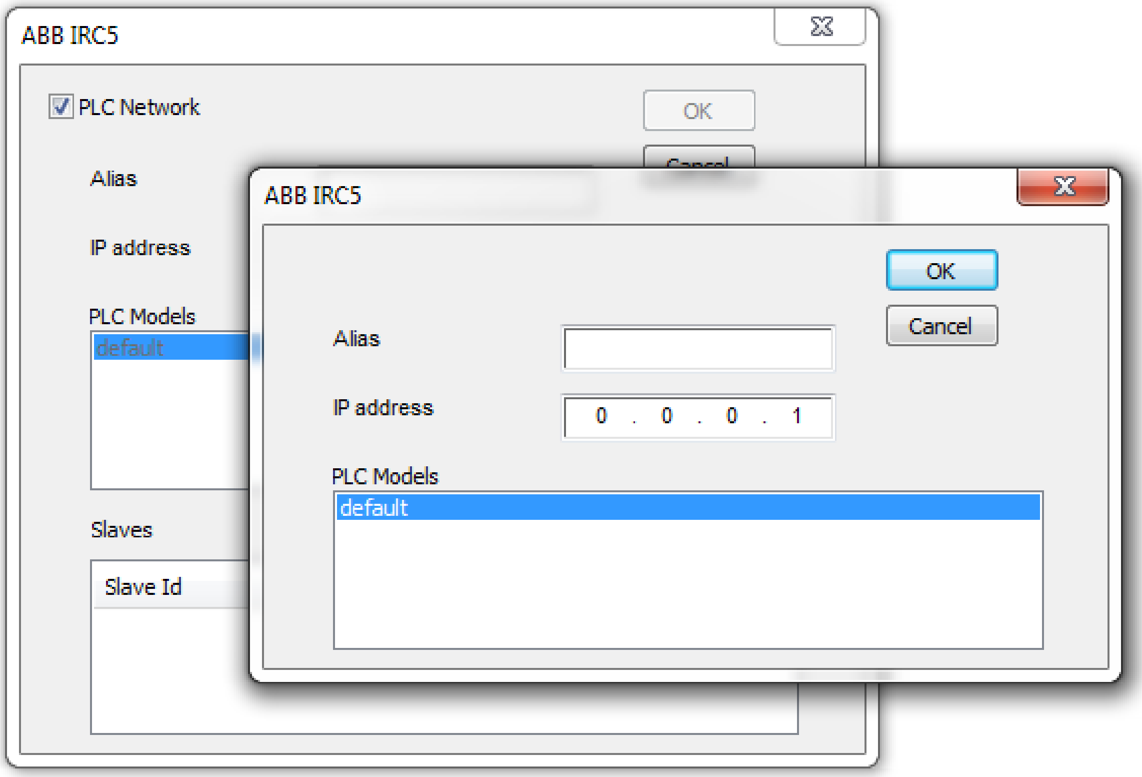


ABB IRC5

☑ PLC Network

Alias:

IP address: 0 . 0 . 0 . 1

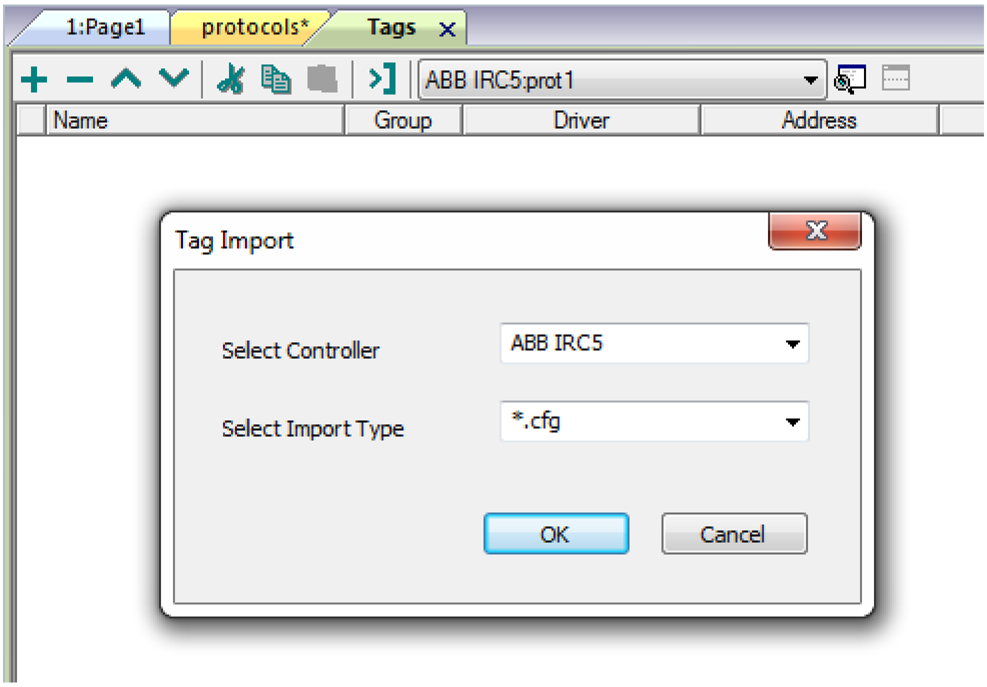
PLC Models: default

Slaves: Slave Id

Tag Import

This driver supports tag import.
The ABB Robotics controllers programming tool can generate a symbol file with extension ".CFG". The tag importer expects files in this format.

In Tag Editor select the communication driver and click on the "Import tag" button to start the importer.



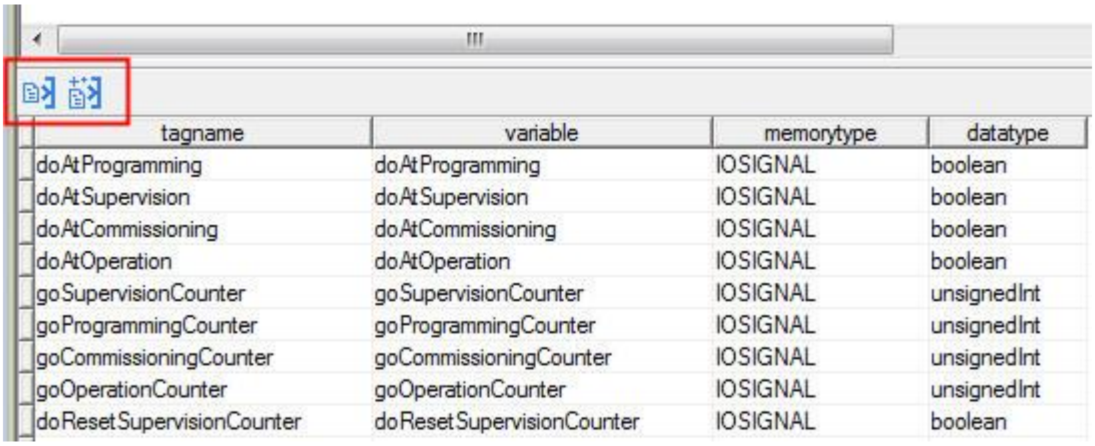
Tag Import

Select Controller: ABB IRC5

Select Import Type: *.cfg

Locate the ".CFG" file and confirm.

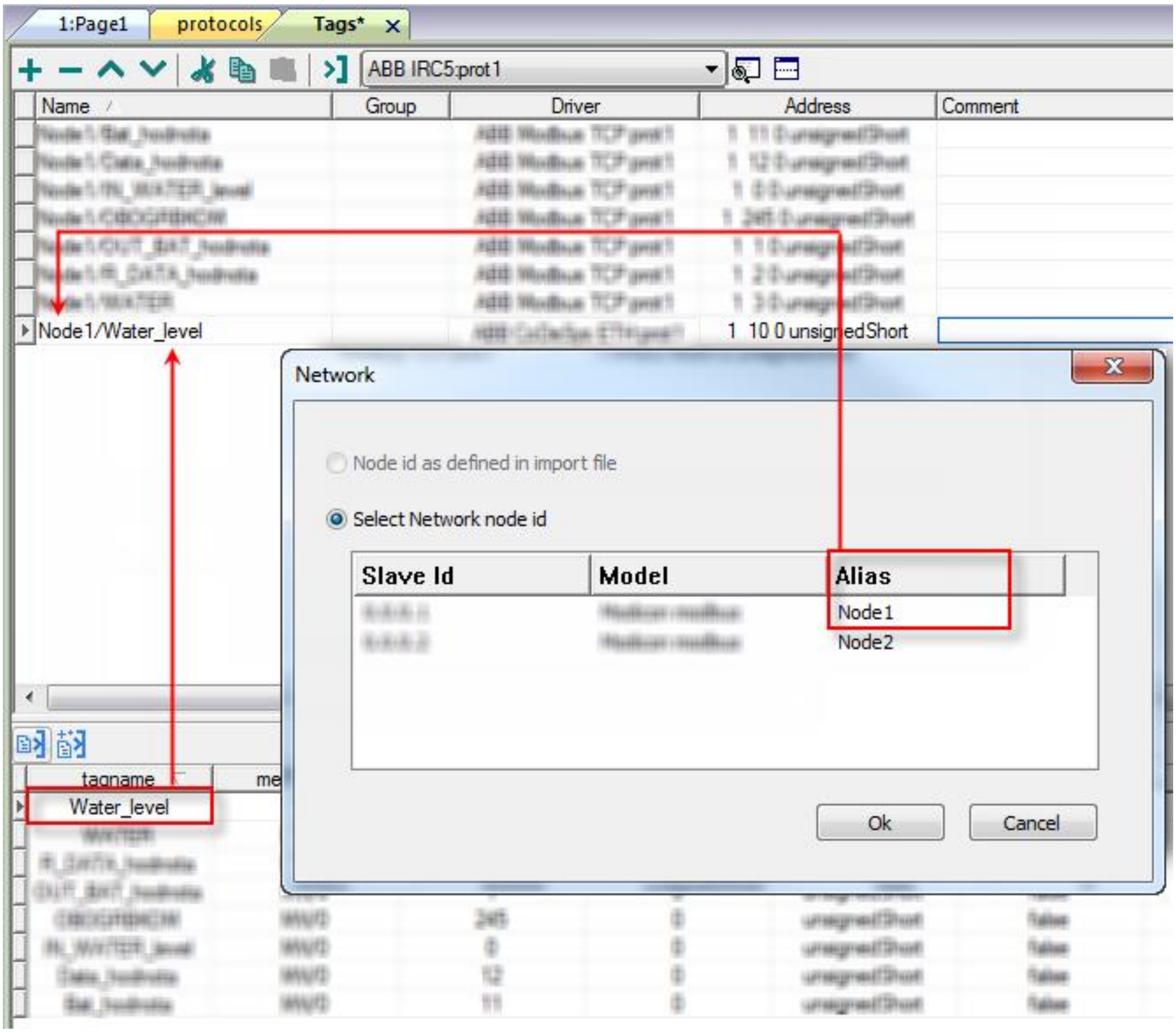
The tags present in the imported file are listed in the tag dictionary from where they can be directly added to the project using the add tags button as shown in next figure.



tagname	variable	memorytype	datatype
doAtProgramming	doAtProgramming	IOSIGNAL	boolean
doAtSupervision	doAtSupervision	IOSIGNAL	boolean
doAtCommissioning	doAtCommissioning	IOSIGNAL	boolean
doAtOperation	doAtOperation	IOSIGNAL	boolean
goSupervisionCounter	goSupervisionCounter	IOSIGNAL	unsignedInt
goProgrammingCounter	goProgrammingCounter	IOSIGNAL	unsignedInt
goCommissioningCounter	goCommissioningCounter	IOSIGNAL	unsignedInt
goOperationCounter	goOperationCounter	IOSIGNAL	unsignedInt
doResetSupervisionCounter	doResetSupervisionCounter	IOSIGNAL	boolean

Aliasing Tag Names in Network Configurations

Tag names must be unique at project level; it often happens that the same tag names are to be used for different controller nodes (for example when the HMI is connected to two devices that are running the same application). Since tags include also the identification of the node and Tag Editor does not support duplicate tag names, the import facility in Tag Editor has an aliasing feature that can automatically add a prefix to imported tags. With this feature tag names can be done unique at project level.
The feature works when importing tags for a specific protocol. Each tag name will be prefixed with the string specified by the “Alias”. As shown in the figure below, the connection to a certain controller is assigned the name “Node1”. When tags are imported for this node, all tag names will have the prefix “Node1” making each of them unique at the network/project level.



note

Aliasing tag names is only available when tags can be imported. Tags which are added manually in the Tag Editor do not need to have the Alias prefix in the tag name.
The Alias string is attached to the tag name only at the moment the tags are imported using Tag Editor. If you modify the Alias string after the tag import has been completed, there will be no effect on the names already present in the dictionary. When the Alias string is changed and tags are imported again, all tags will be imported again with the new prefix string.

Data Types

The import module supports variables of the following standard data types.

- boolean
- unsignedInt
- float

note In this first version of the IRC5 robot controller driver only IO signals can be used and only output signals of the robot controller can be modified by the CP600 panel. They can only be modified, as long as the IRC5 controller is in automatic mode or the FlexPendant is not connected and the signal access level is “ALL”. Input signals can only be monitored.

Implementation Details

An IRC5 robot controller system uses input and output signals to control the process. Signals can be of digital, analog, or group signal type. Such **IO signals** are accessible using the PanelBuilder tool.

Signal changes in the robot system are often significant, and there are many scenarios where end-users of the system need notification of signal changes.

In manual mode of the IRC5 controller, a signal value can be modified only if the Access Level of the signal is ALL and the FlexPendant is not connected. If not, the controller has to be in auto mode. To change the access level of a signal, the IRC5 robot controller tools RobotStudio or FlexPendant must be used.

A PanelBuilder application will act as a remote client as comparison with the FlexPendant which is a local client.

Remote clients do not have all the privileges of a local client, especially in manual mode. For example, in manual mode the FlexPendant can start and stop IRC5 robot controller programs by using system inputs but this is only available for a remote client in automatic mode.

An advantage of a remote client, on the other hand, is the possibility to monitor and access several robot controllers from one location.

Communication Status

The current communication status can be displayed using the dedicated system variables. Please refer to the chapter “system variables” about available types and their use.

The codes supported for this communication driver are:

Error	Notes
Can't find the node x.x.x.x "	Returned when a request is not replied within the specified timeout period; ensure the controller is connected and properly configured to get network access
I/O signal reading error	The panel did receive from the controller a response, but its format or its contents is not as expected; ensure the data programmed in the project are consistent with the controller resources
I/O signal quality not good	The panel did receive from the controller a response, but its quality is not good.
Error requesting mastership	The panel didn’t receive from the controller the correct right to write data into the controller

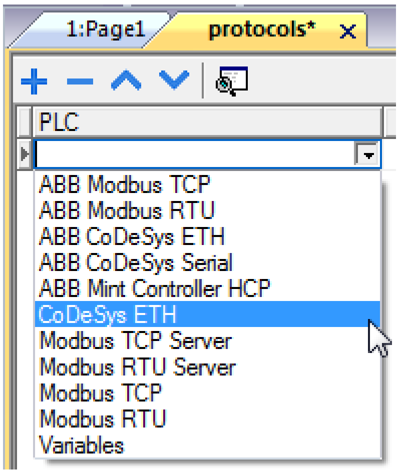
CoDeSys V2.3 Ethernet Driver

The CoDeSys communication driver for Ethernet supports communication switch controllers based on the V2.3 CoDeSys version.

> **NOTE**
>
> Note that changes in the controller protocol or hardware, which may interfere with the functionality of this driver, may have occurred since this documentation was created. Therefore, always test and verify the functionality of the application. To accommodate developments in the controller protocol and hardware, drivers are continuously updated. Accordingly, always ensure that the latest driver is used in the application.

Protocol Editor Settings

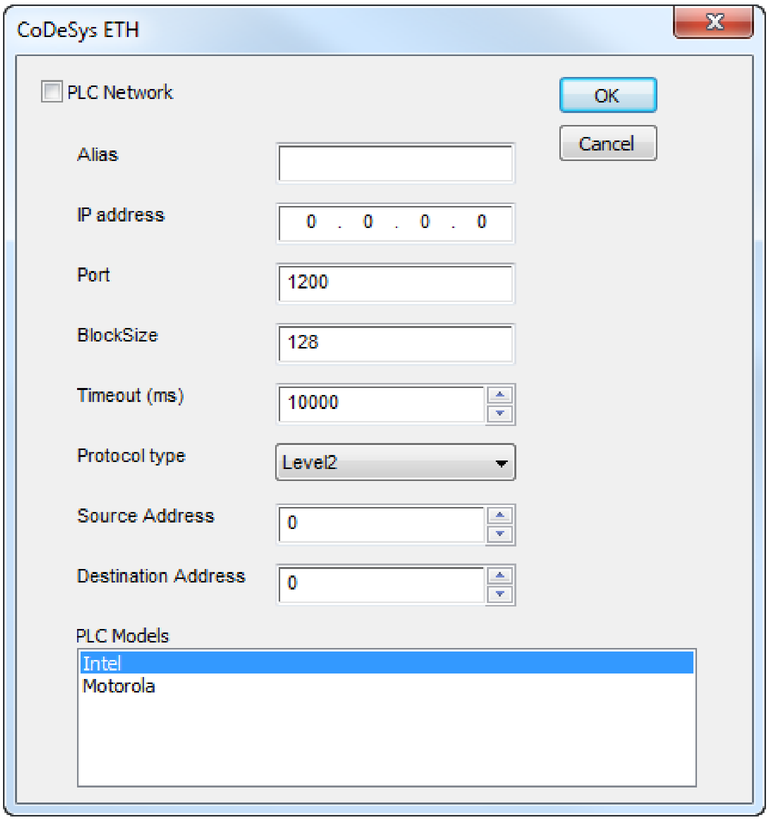
→ Add with the button **+** a driver in the protocol editor and select the protocol called **CoDeSys ETH** from the list of available protocols.



The CoDeSys ETH driver supports three different protocol types:

- Level2
- Level2 Route
- Level4

→ Select the protocol type from the dedicated combo box in the dialog.



CoDeSys ETH

□ PLC Network

Alias

IP address: 0 . 0 . 0 . 0

Port: 1200

BlockSize: 128

Timeout (ms): 10000

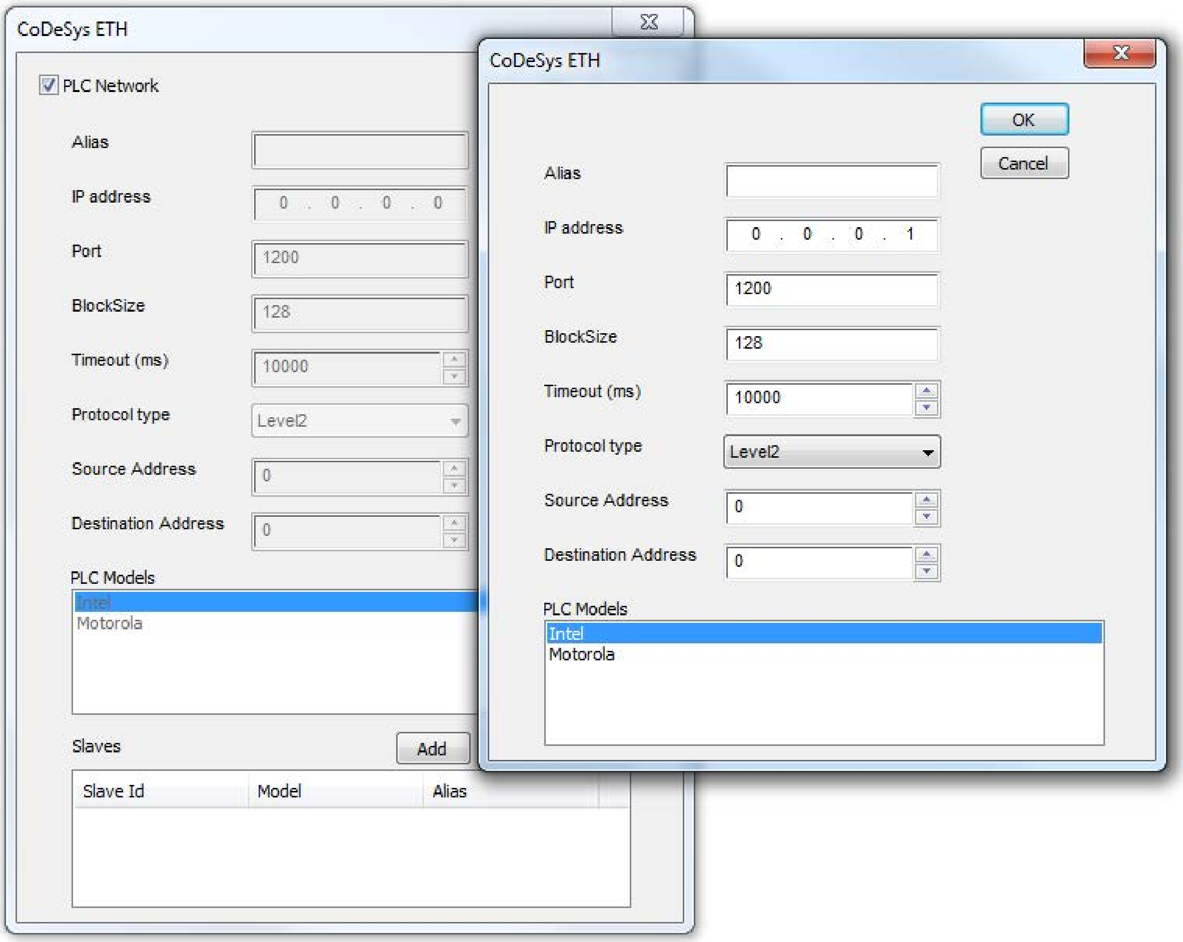
Protocol type: Level2

Source Address: 0

Destination Address: 0

PLC Models: Intel, Motorola

Parameter	Description
Alias	Name to be used to identify nodes in network configurations. The name will be added as a prefix to each tag name imported for each network node.
IP address	Ethernet IP address of the controller.
Port	This parameter allows changing the port number used for the communication. Default value for this parameter is set to 1200 and it corresponds to the default setting of CoDeSys-based controllers.
PLC Model	Defines the byte order that will be used by the communication driver when sending communication frames to the PLC.
Block Size	Enter the max. block size supported by your controller (limit is 1024 KB)
Protocol type	Shows a list with the available protocol variants. Please make sure you check what protocol variant is supported by the CoDeSys runtime you want to connect.
Timeout	The number of milliseconds between retries when communication fails.
Source Address & Destination Address	Source and Destination address are available only when TCP/IP Level 2 route is selected in Controller Setup. The destination is the node of the PLC and allows the protocol to reads variable in a sub-network. The address is used to read variables when multiple PLCs are connected in a sub-network (serial network) but only one of it have the Ethernet interface.
PLC Network	The protocol allows the connection of multiple controllers to one operator panel. To set-up multiple connections, check **Access Multiple PLC's** checkbox and enter IP address for all controllers.



CoDeSys ETH

☑ PLC Network

Alias:

IP address: 0 . 0 . 0 . 0

Port: 1200

BlockSize: 128

Timeout (ms): 10000

Protocol type: Level2

Source Address: 0

Destination Address: 0

PLC Models: Intel, Motorola

Slaves

Slave Id	Model	Alias

CoDeSys ETH

Alias:

IP address: 0 . 0 . 0 . 1

Port: 1200

BlockSize: 128

Timeout (ms): 10000

Protocol type: Level2

Source Address: 0

Destination Address: 0

PLC Models: Intel, Motorola

NOTE

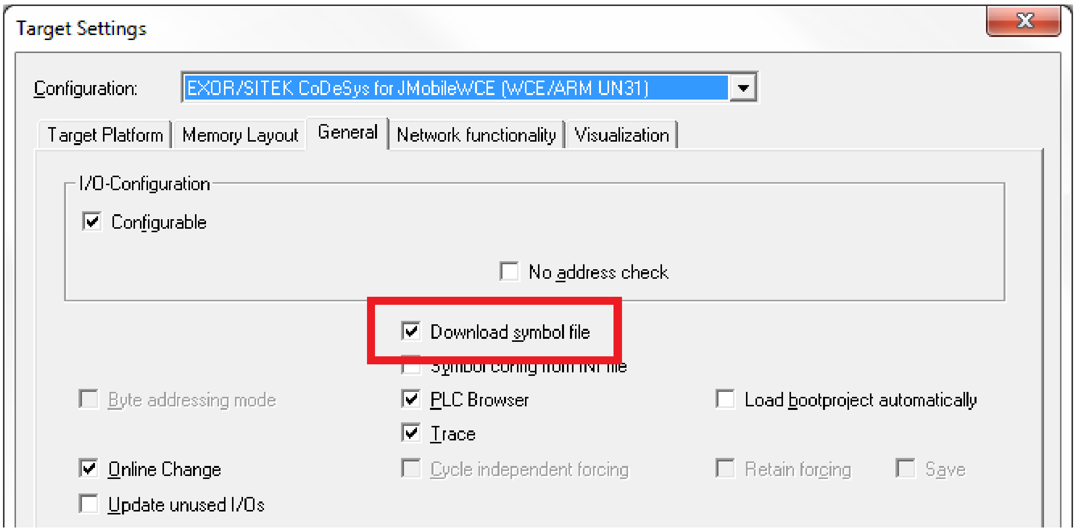
The CoDeSys Ethernet driver supports the connections to multiple controllers starting from version V1.60.

NOTE

The CoDeSys ETH driver is the best choice also when creating projects for the internal controller iPLC CoDeSys. To use the CoDeSys ETH driver for communication with iPLC it is enough to configure the IP address of the PLC as localhost (127.0.0.1). The iPLC CoDeSys supports communication with CoDeSys ETH driver with symbol based support (see next chapter) starting from V1.55 and above.

CoDeSys Software Settings

When creating the project in CoDeSys, the checkbox **Download symbol file** (under *Target/Settings/General*) must be checked.



Target Settings

Configuration: EXOR/SITEK CoDeSys for JMobileWCE (WCE/ARM UN31)

Target Platform | Memory Layout | General | Network functionality | Visualization

I/O-Configuration

☑ Configurable

□ No address check

☑ Download symbol file

□ Symbol config from INI file

□ Byte addressing mode

☑ PLC Browser

□ Load bootproject automatically

☑ Trace

☑ Online Change

□ Cycle independent forcing

□ Retain forcing

□ Save

□ Update unused I/Os

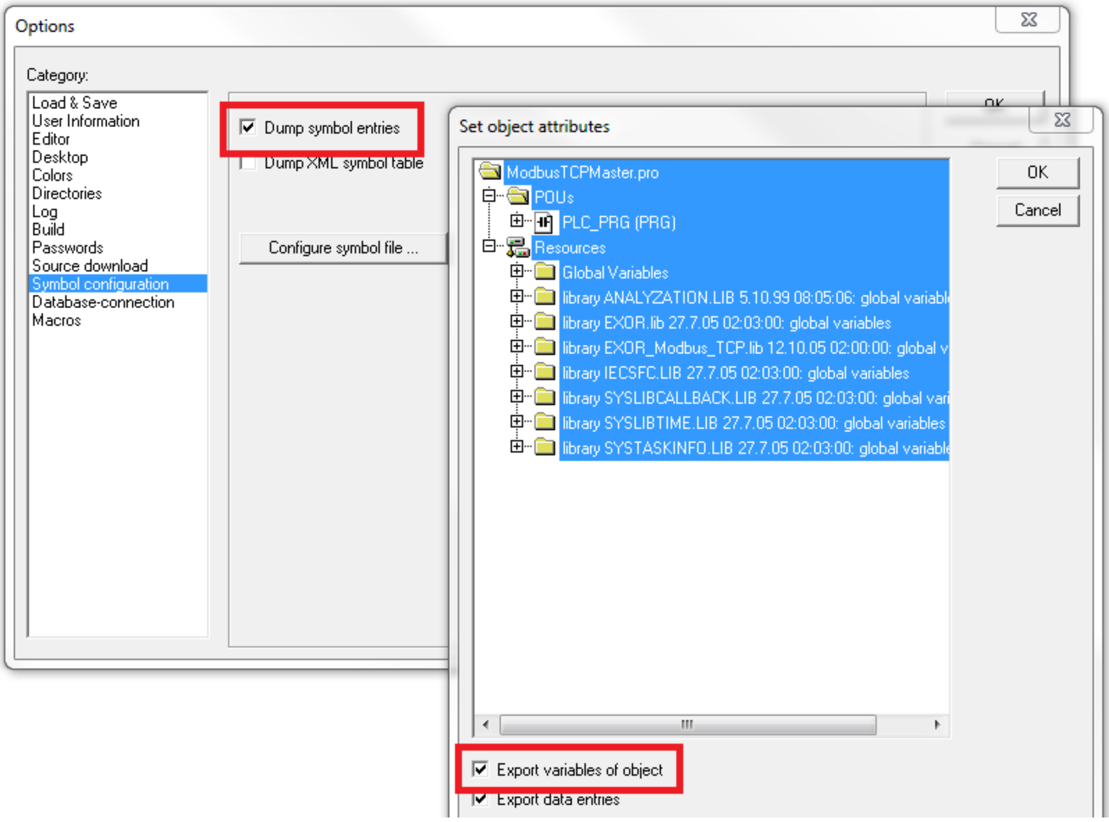
NOTE

CoDeSys Ethernet driver supports the automatic symbol file (SDB) upload from controller; any change in the tag offset due to new compilation on PLC software side does not require a symbol file re-import. Tag file has to be re-imported only in case of tag rename or addition of new tags.

Tag Import

When configuring PLC using the manufacturer's configuration software, make sure to enable *symbol file creation* (file with .SYM extension). It can be done under the CoDeSys programming software as follows:

1. Select *Project/Option/Symbol configuration* and activate the check box **Dump symbol**.



Options

Category:

Load & Save
User Information
Editor
Desktop
Colors
Directories
Log
Build
Passwords
Source download
Symbol configuration
Database-connection
Macros

☑ Dump symbol entries

□ Dump XML symbol table

Configure symbol file ...

Set object attributes

ModbusTCPMaster.pro
POUs
PLC_PRG (PRG)
Resources
Global Variables
library ANALYZATION.LIB 5.10.99 08:05:06: global variabl
library EXOR.lib 27.7.05 02:03:00: global variables
library EXOR_Modbus_TCP.lib 12.10.05 02:00:00: global v
library IECSFC.LIB 27.7.05 02:03:00: global variables
library SYSLIBCALLBACK.LIB 27.7.05 02:03:00: global vari
library SYSLIBTIME.LIB 27.7.05 02:03:00: global variables
library SYSTASKINFO.LIB 27.7.05 02:03:00: global variabl

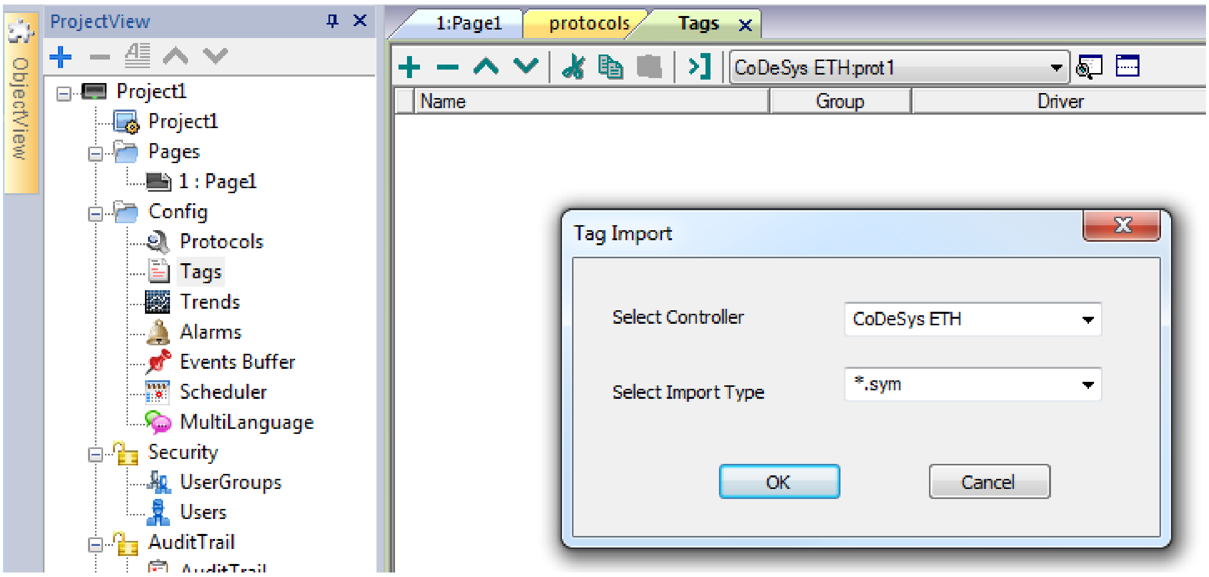
☑ Export variables of object

☑ Export data entries

2. Click on the button **Configure symbol file**.

3. Make sure the **Export variables of object** check box is marked. We recommend to un-check the check box and mark it again to be sure about the proper settings.

4. Select the driver in the Studio tag editor and click on the **Import tag** button to start the importer.

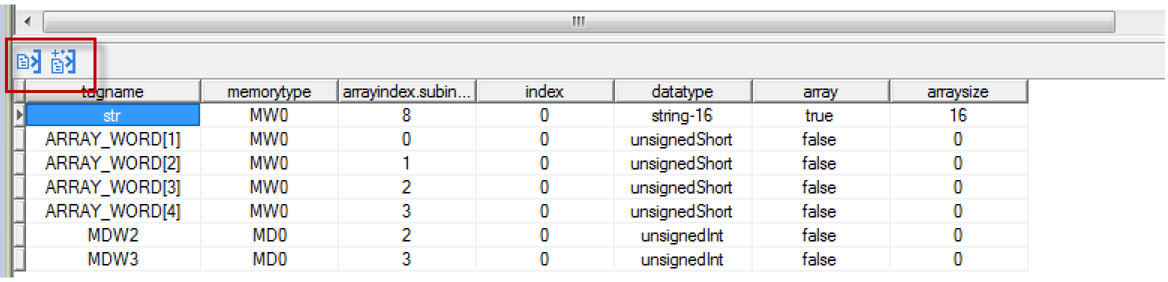


Select Controller: CoDeSys ETH

Select Import Type: *.sym

5. Locate the "sym" file and confirm with **OK** button.

The tags present in the **exported** document are listed in the tag dictionary from where they can be directly added to the project using the **add tags** button as shown in the following figure.

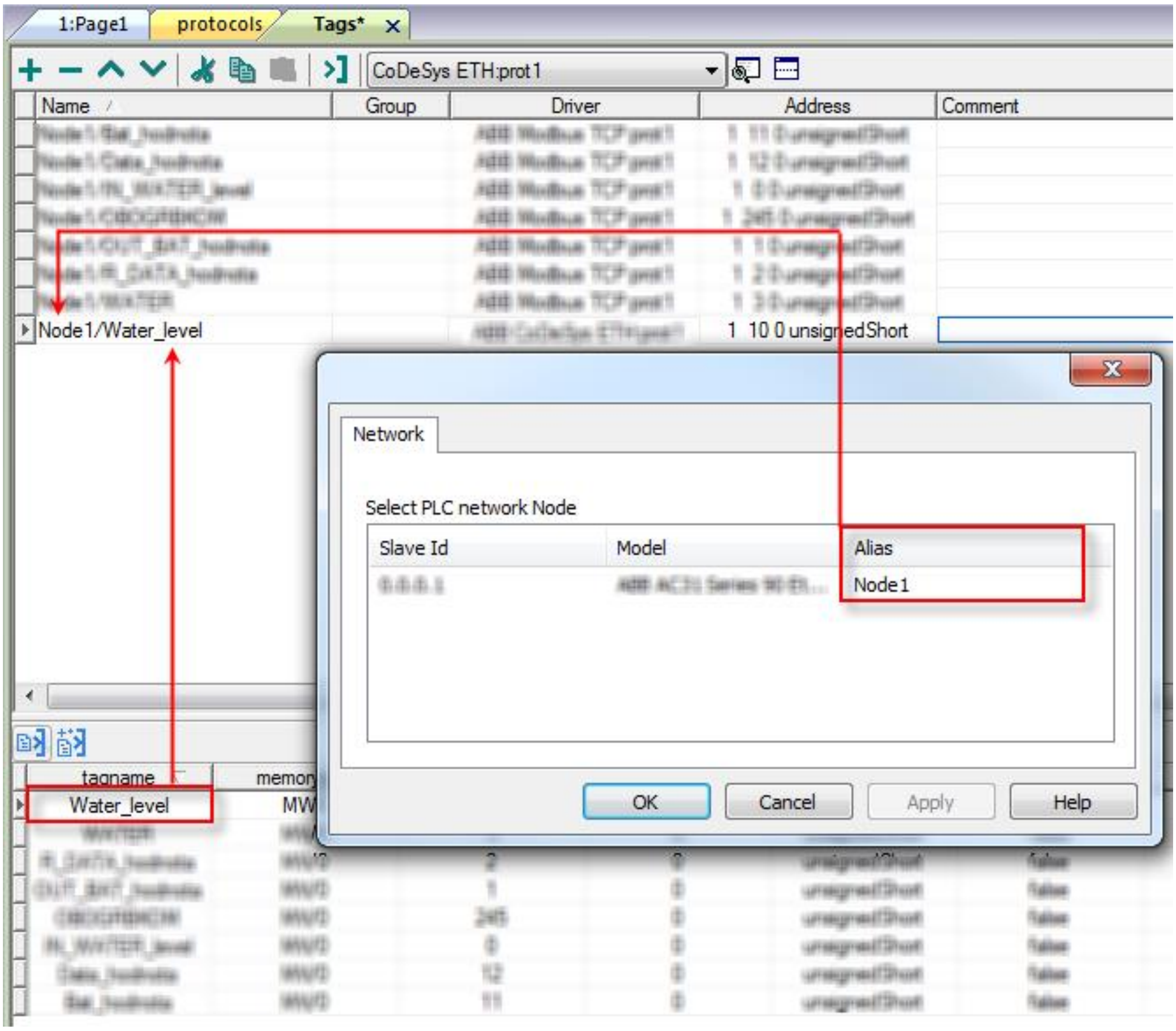


tagname	memorytype	arrayindex.subin...	index	datatype	array	arraysize
str	MW0	8	0	string-16	true	16
ARRAY_WORD[1]	MW0	0	0	unsignedShort	false	0
ARRAY_WORD[2]	MW0	1	0	unsignedShort	false	0
ARRAY_WORD[3]	MW0	2	0	unsignedShort	false	0
ARRAY_WORD[4]	MW0	3	0	unsignedShort	false	0
MDW2	MD0	2	0	unsignedInt	false	0
MDW3	MD0	3	0	unsignedInt	false	0

Aliasing Tag Names in Network Configurations

Tag names must be unique at project level; it often happens that the same tag names are to be used for different controller nodes (for example when the HMI is connected to two devices that are running the same application). Since tags include also the identification of the node and tag editor does not support duplicate tag names, the import facility in tag editor has an aliasing feature that can automatically add a prefix to imported tags. With this feature tag names can be done unique at project level.

The feature works when importing tags for a specific protocol. Each tag name will be prefixed with the string specified by the “Alias”. As shown in the figure below, the connection to a certain controller is assigned the name “Node1”. When tags are imported for this node, all tag names will have the prefix “Node1” making each of them unique at the network/project level.



CoDeSys ETH:prot1

Name	Group	Driver	Address	Comment
Node1/Water_level		[illegible]	1 10 0 unsignedShort	

Select PLC network Node

Slave Id	Model	Alias
[illegible]	[illegible]	Node1

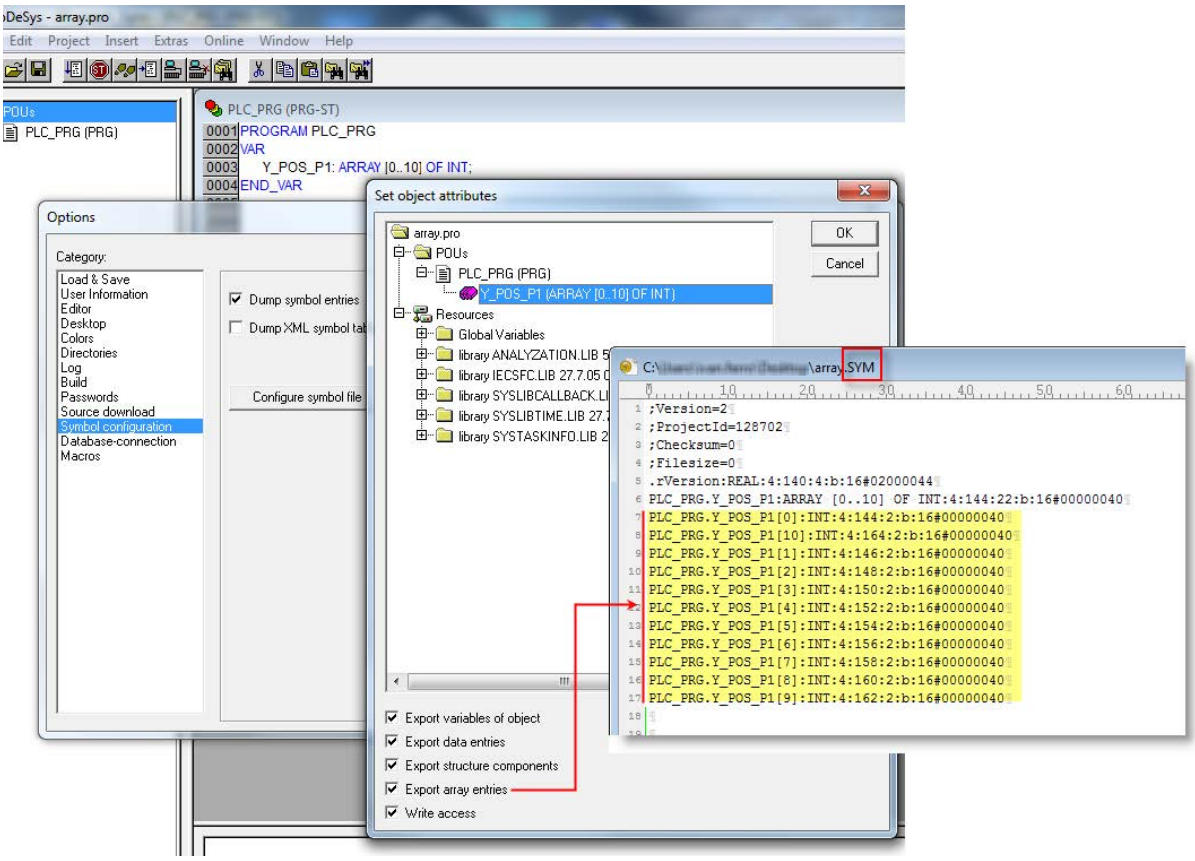
tagname	memory
Water_level	MW

NOTE

Aliasing tag names is only available when tags can be imported. Tags which are added manually in the tag editor do not need to have the Alias prefix in the tag name.
The Alias string is attached to the tag name only at the moment the tags are imported using tag editor. If you modify the Alias string after the tag import has been completed, there will be no effect on the names already present in the dictionary. When the Alias string is changed and tags are imported again, all tags will be imported again with the new prefix string.

Tag Array

Tag arrays are split into individual elements and one tag for each element is created. The figure below shows an example of one array with 10 elements.



```
0001 PROGRAM PLC_PRG
0002 VAR
0003     Y_POS_P1: ARRAY [0..10] OF INT;
0004 END_VAR
```

Category:
Load & Save
User Information
Editor
Desktop
Colors
Directories
Log
Build
Passwords
Source download
Symbol configuration
Database-connection
Macros

☑ Dump symbol entries

☐ Dump XML symbol tab

array.pro
POUs
PLC_PRG (PRG)
Y_POS_P1 (ARRAY [0..10] OF INT)
Resources
Global Variables
library ANALYZATION.LIB 5
library IECSFC.LIB 27.7.05 0
library SYSLIBCALLBACK.LI
library SYSLIBTIME.LIB 27.7
library SYSTASKINFO.LIB 2

☑ Export variables of object

☑ Export data entries

☑ Export structure components

☑ Export array entries

☑ Write access

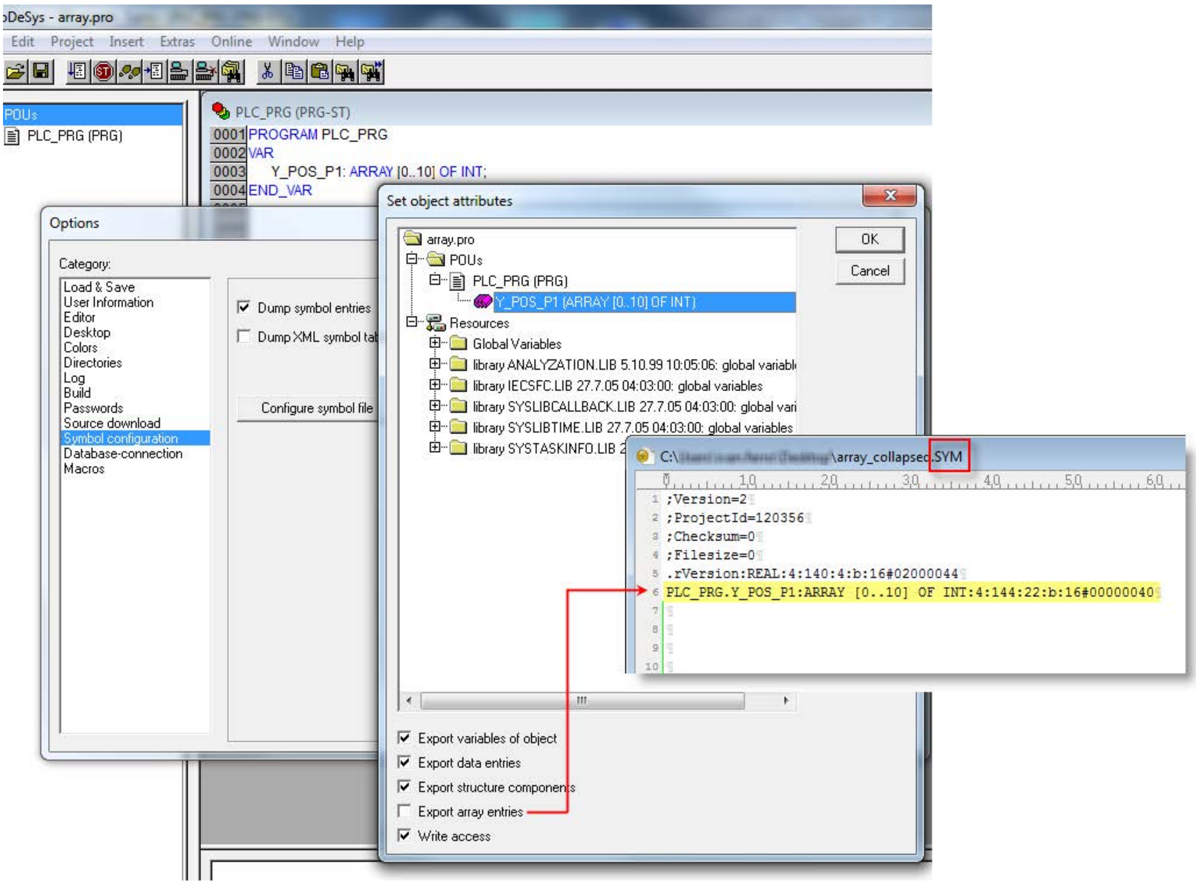
C:\[redacted]\array.SYM

```
;Version=2
;ProjectId=128702
;Checksum=0
;Filesize=0
.rVersion:REAL:4:140:4:b:16#02000044
PLC_PRG.Y_POS_P1:ARRAY [0..10] OF INT:4:144:22:b:16#00000040
PLC_PRG.Y_POS_P1[0]:INT:4:144:2:b:16#00000040
PLC_PRG.Y_POS_P1[10]:INT:4:164:2:b:16#00000040
PLC_PRG.Y_POS_P1[1]:INT:4:146:2:b:16#00000040
PLC_PRG.Y_POS_P1[2]:INT:4:148:2:b:16#00000040
PLC_PRG.Y_POS_P1[3]:INT:4:150:2:b:16#00000040
PLC_PRG.Y_POS_P1[4]:INT:4:152:2:b:16#00000040
PLC_PRG.Y_POS_P1[5]:INT:4:154:2:b:16#00000040
PLC_PRG.Y_POS_P1[6]:INT:4:156:2:b:16#00000040
PLC_PRG.Y_POS_P1[7]:INT:4:158:2:b:16#00000040
PLC_PRG.Y_POS_P1[8]:INT:4:160:2:b:16#00000040
PLC_PRG.Y_POS_P1[9]:INT:4:162:2:b:16#00000040
```

NOTE

When **Export array entries** is set, a tag for each element is created and exported into the SYM file. The entire tag list is automatically imported into tag editor.

The amount of tags can be reduced and only one tag for each one array can be created by removing the checkbox **Export array entries**, see figure below.


CoDeSys - array.pro
Edit Project Insert Extras Online Window Help
POUs
PLC_PRG (PRG)
Options
Category:
Load & Save
User Information
Editor
Desktop
Colors
Directories
Log
Build
Passwords
Source download
Symbol configuration
Database-connection
Macros
Configure symbol file
Set object attributes
OK
Cancel
array.pro
POUs
PLC_PRG (PRG)
Y_POS_P1 (ARRAY [0..10] OF INT)
Resources
Global Variables
library ANALYZATION.LIB 5.10.99 10:05:06: global variabl
library IECSFC.LIB 27.7.05 04:03:00: global variables
library SYSLIBCALLBACK.LIB 27.7.05 04:03:00: global vari
library SYSLIBTIME.LIB 27.7.05 04:03:00: global variables
library SYSTASKINFO.LIB 2

PLC_PRG (PRG-ST)

```
PROGRAM PLC_PRG
VAR
    Y_POS_P1: ARRAY [0..10] OF INT;
END_VAR
```

☑ Dump symbol entries

☐ Dump XML symbol tab

☑ Export variables of object

☑ Export data entries

☑ Export structure components

☐ Export array entries

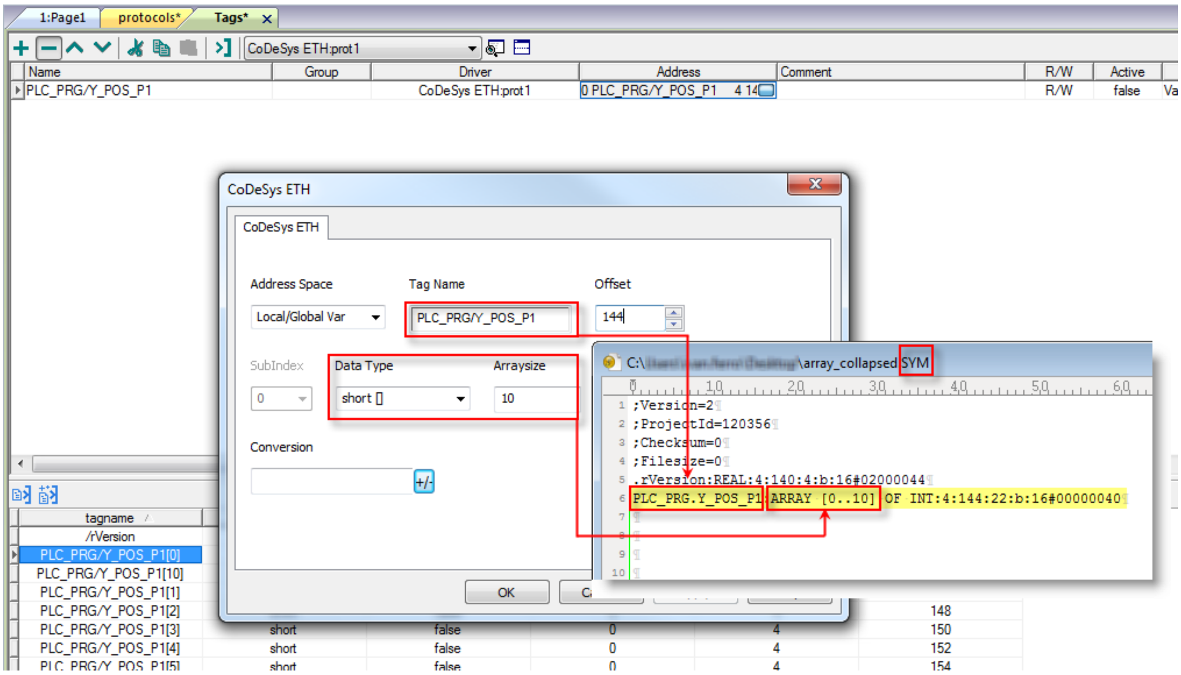
☑ Write access

C:\[redacted]\array_collapsed.SYM

```
;Version=2
;ProjectId=120356
;Checksum=0
;Filesize=0
.rVersion:REAL:4:140:4:b:16#02000044
PLC_PRG.Y_POS_P1:ARRAY [0..10] OF INT:4:144:22:b:16#00000040
```

NOTE

When **Export array entries** is not set, only one tag is created and exported into the SYM file. The array will not be automatically imported in tag editor and tags need to be manually configured in tag editor.



Name	Group	Driver	Address	Comment	R/W	Active
PLC_PRG/Y_POS_P1		CoDeSys ETH:prot1	0 PLC_PRG/Y_POS_P1 4 14		R/W	false

CoDeSys ETH

Address Space: Local/Global Var

Tag Name: PLC_PRG/Y_POS_P1

Offset: 144

SubIndex: 0

Data Type: short []

Arraysize: 10

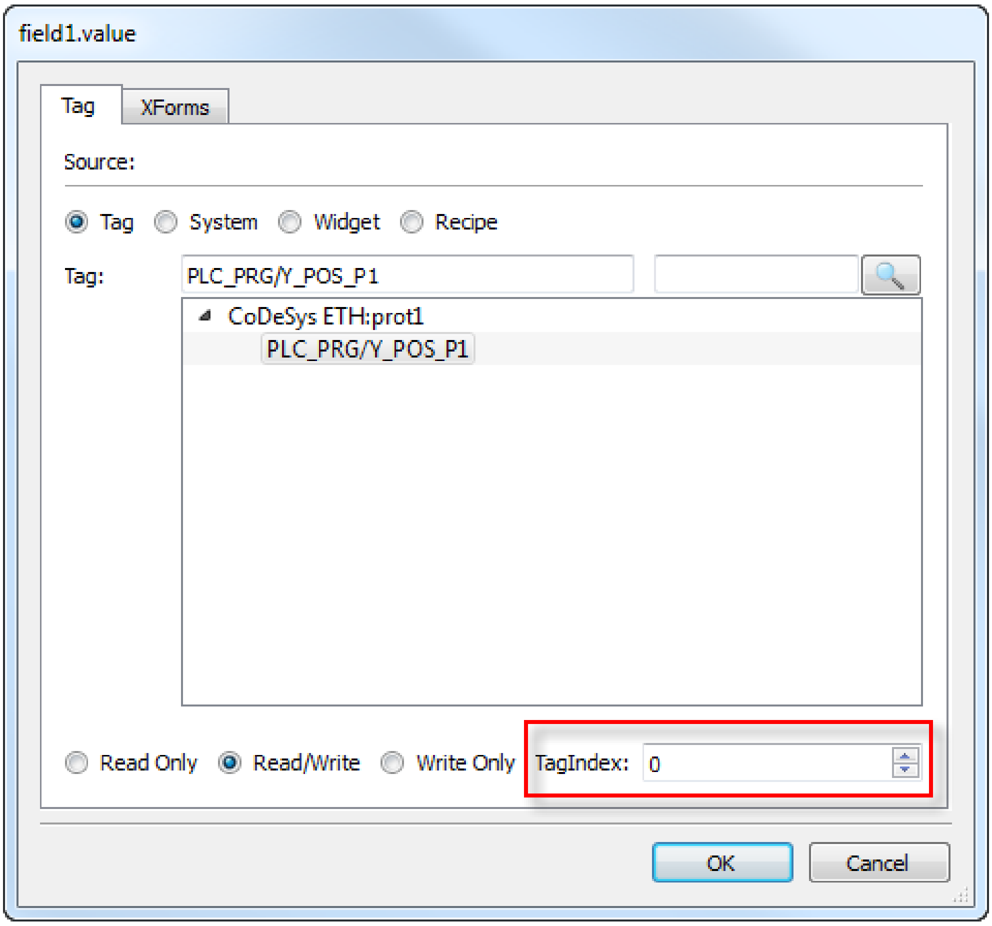
Conversion

C:\[redacted]\array_collapsed.SYM

```
;Version=2
;ProjectId=120356
;Checksum=0
;Filesize=0
.rVersion:REAL:4:140:4:b:16#02000044
PLC_PRG.Y_POS_P1:ARRAY [0..10] OF INT:4:144:22:b:16#00000040
```

tagname					
/rVersion					
PLC_PRG/Y_POS_P1[0]					
PLC_PRG/Y_POS_P1[10]					
PLC_PRG/Y_POS_P1[1]					148
PLC_PRG/Y_POS_P1[2]	short	false	0	4	150
PLC_PRG/Y_POS_P1[3]	short	false	0	4	152
PLC_PRG/Y_POS_P1[4]	short	false	0	4	154
PLC_PRG/Y_POS_P1[5]	short	false			

All tag elements can be referenced in the editor using **TagIndex** in the *Attach to Tag* dialog.



Source:

● Tag ○ System ○ Widget ○ Recipe

Tag: PLC_PRG/Y_POS_P1

CoDeSys ETH:prot1
PLC_PRG/Y_POS_P1

○ Read Only ● Read/Write ○ Write Only TagIndex: 0

Data Types

The import module supports variables of standard data types and user defined data types.

Standard Data Types

The following data types in the CoDeSys programming tool are considered standard data types by the import module:

- BOOL
- WORD
- DWORD
- INT
- UINT
- UDINT
- DINT
- STRING
- REAL
- TIME
- DATE & TIME

and a

- 1-dimensional ARRAY of the types above.

The 64-bit data types LWORD, LINT and LREAL are not supported.

String length for a STRING variable in PLC should be max. 80 characters. Declare a STRING variable either with a specified size (str: STRING(35) or default size (str: STRING)) which is 80 characters.

Special Data Types

The CoDeSys ETH driver provides one special data type called "IP Override".

The IP Override IP allows changing at runtime the IP address of the target controller you want to connect. This memory type is an array of 4 unsigned bytes, one per each byte of the IP address.

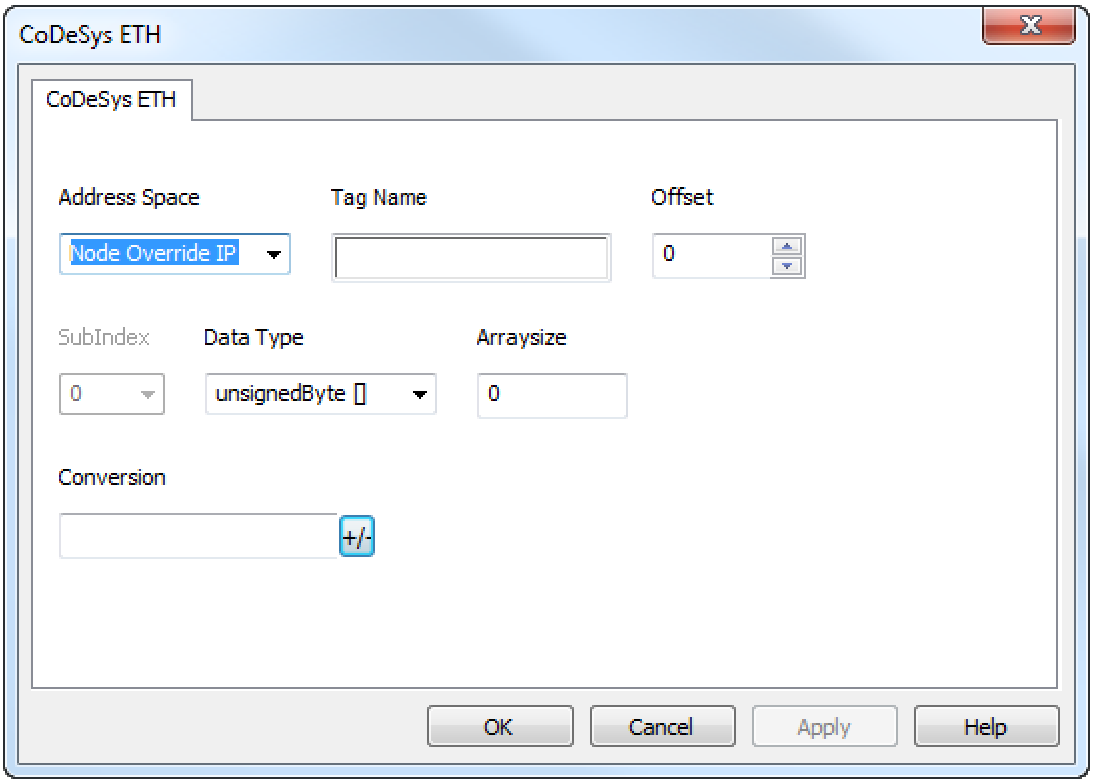
The Node Override IP is initialized with the value of the controller IP specified in the project at programming time.

If the IP Override is set to 0.0.0.0, all the communication with the node is stopped, no request frames are generated anymore.

If the IP Override has a value different from 0.0.0.0, it is interpreted as Node IP Override and the target IP address is replaced runtime with the new value.

In case the panel has been configured to access to a network of controllers, each node has its own override variable.

NOTE The IP Override values assigned at runtime are not retained through power cycles.



Address Space: Node Override IP
Tag Name:
Offset: 0
SubIndex: 0
Data Type: unsignedByte []
Arraysize: 0
Conversion:

Limitations

Max. block size is 1024 byte.

Communication Status

The current communication status can be displayed using the dedicated system variables. Please refer to the User Manual for further information about available system variables and their use.

The codes supported for this communication driver are:

Error	Notes
Symbol file not present	Check symbol file and download again the PLC program.
“tag” not present in symbol files	Check if the tag is present in the PLC project.
Time out on Acknowledge	Controller didn’t send acknowledge.
Time out on last Acknowledge	Controller didn’t send last acknowledge.
Time out on data receiving	Controlled does not reply with data.
Connection timeout	Device not connected.

Modbus TCP Server Driver

The Modbus TCP server communication driver lets you connect the panel as a server in a Modbus TCP network. It is possible for Modbus TCP clients to connect them to multiple HMI panels acting as servers. The information exchange will use standard Modbus messages over TCP/IP.

This approach will also offer an interesting way to connect the HMI panels to SCADA systems through the universally supported Modbus TCP communication protocol.

Principle of Operation

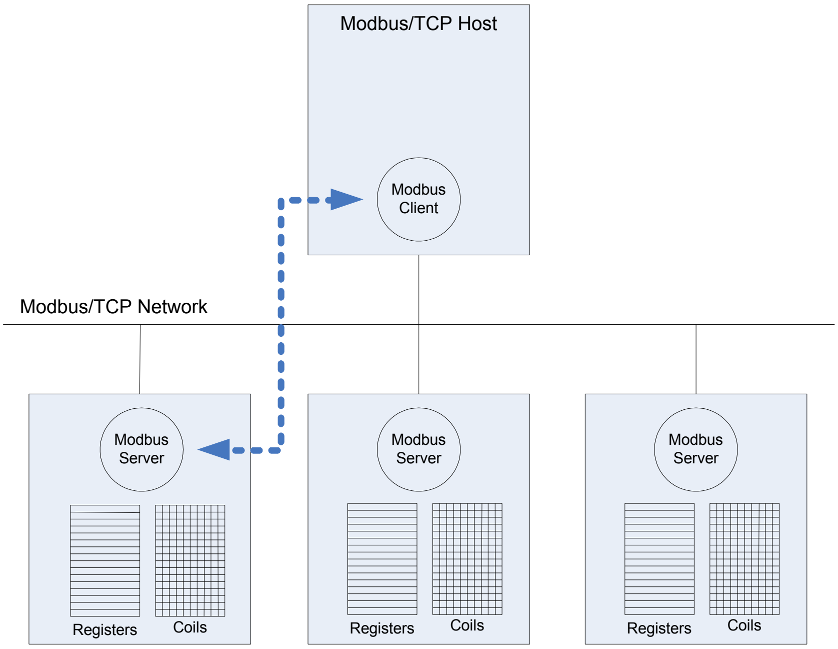
This communication driver will implement a Modbus TCP server unit in HMI device. A subset of the complete range of Modbus function codes will be supported. The available function codes will allow the transfer of data between clients on the TCP network and the server.

The diagram in the following figure shows the system architecture.

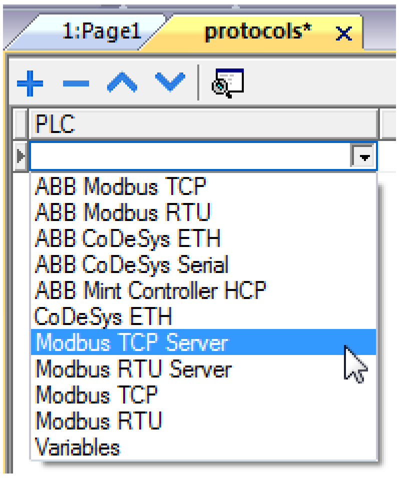
NOTE

The panel is actually simulating the communication interface of a PLC: it has two data types (coils and registers) that are respectively Boolean and 16 bit integers.

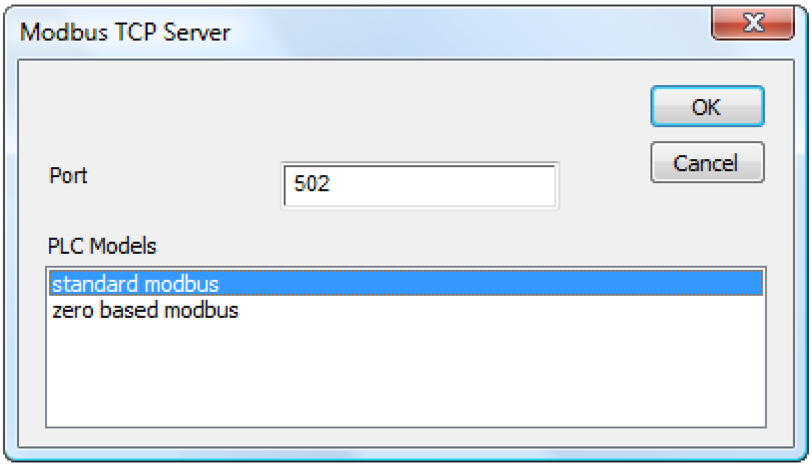
The panel will always access data in its internal memory. Data can be transferred to and from the Modbus client only on the initiative of the client itself.



System Architecture



Protocol Editor Settings



Port: 502

PLC Models: standard modbus (selected), zero based modbus

Parameter	Description
Port	By default, the Modbus TCP protocol uses port 502 for communication with the nodes in the network. Unless your network uses a different port you should leave the port setting to the default value of 502 otherwise set it to the value expected by your Modbus TCP network.
PLC Models	The *controller setup* dialog box allows you to select between two different flavours of Modbus TCP. You can select the flavour most appropriate for your needs. The *standard modbus* model type implements a holding register range of between 400001 and 401098 and an output coils range of between 1 and 9998. The *zero based modbus* model type, on the other hand, implements a holding register range of between 400000 and 401097 and an output coils range of between 0 and 9997. Please note that the address range used in the Modbus frames will always be respectively between 0 and 1097 for the holding registers and between 0 and 9997 for coils.

Communication Status

The panel is a server station in the Modbus TCP network. The current implementation of the protocol will not report any communication error code, unless the standard communication error codes related to the proper driver loading.

Implementation Details

This Modbus TCP server implementation supports only a subset of the standard Modbus function codes. Only the function codes necessary for the data exchange between the HMI and a Modbus TCP client have been implemented.

The supported function codes are listed in the table below.

Code	Function	Description
01	Read Coil Status	Reads multiple bits in the panel coil area.
02	Read Input Status	Reads multiple bits in the panel coil area.
03	Read Holding Registers	Read multiple panel registers.
04	Read Input Registers	Read multiple panel registers..
05	Force Single Coil	Force a single panel coil to either ON or OFF.
06	Preset Single Register	Preset a value in a panel register.
15	Force Multiple Coils	Force multiple panel coils to either ON or OFF
16	Preset Multiple Registers	Preset value in multiple panel registers.

The panel will return the exception code 01 (illegal function) if the function code received in the query is not supported.

The amount of memory available in the panel is as follows:

Data Type	Type	Range
Coils	Bit	0 – 9997
Registers	Word	0 – 1097

The panel protocol will return the exception code 02 (illegal data address) if the data address received in the query exceeds the predefined data ranges.

NOTE

Both PLC models available the Read Coil Status and Read Input Status function codes both access the same coil memory area in the panel memory. Also, the Read Holding Registers and Read Input Registers function codes both access the same register area in the panel memory.

NOTE

A single panel Modbus TCP server can only support a maximum of 3 concurrent connections. This means that up to 3 Clients could be connected to a panel Modbus TCP server at the same time. However, a 4th client, should it attempt to connect while the other 3 clients are connected, will be unable to do so.

Modbus RTU Server Driver

The Modbus RTU server communication driver lets you connect the panel as a slave in a Modbus RTU network. It is possible to connect multiple HMI panels to one Modbus master controller. The information exchange will use standard Modbus messages.

This approach will also offer an interesting way to connect the HMI panels to SCADA systems through the universally supported Modbus RTU communication protocol.

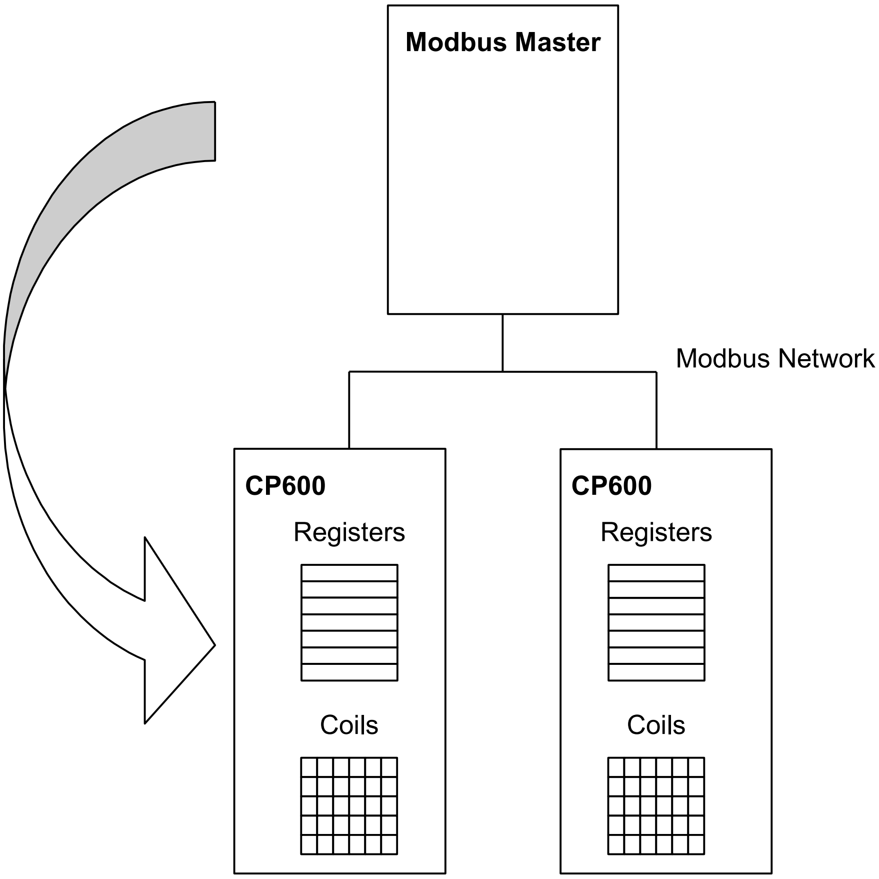
Principle of Operation

This communication driver will implement a Modbus RTU slave unit in the HMI device. A subset of the complete range of Modbus function codes will be supported. The available function codes will allow the transfer of data between the master and the slave.

The diagram in the figure shows the system architecture.

> **NOTE**
>
> The HMI is actually simulating the communication interface of a PLC: it has two data types (coils and registers) that are respectively Boolean and 16 bit integers.

The panel will always access data in its internal memory. Data can be transferred to and from the Modbus master only on initiative of the master itself.



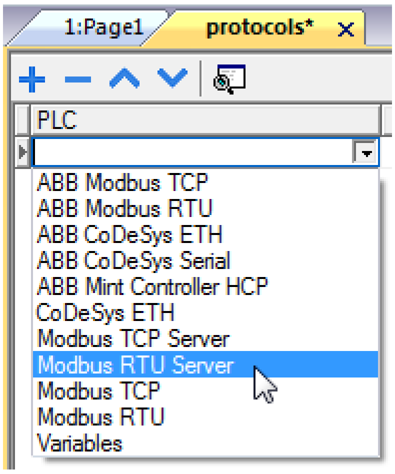
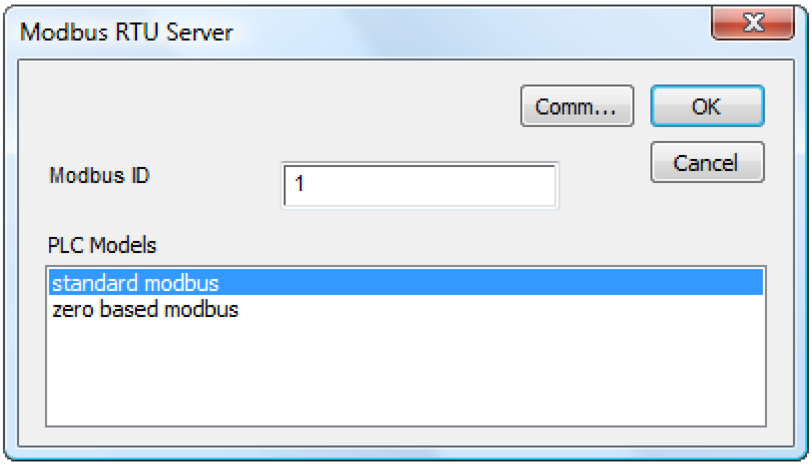


ABB Modbus TCP
ABB Modbus RTU
ABB CoDeSys ETH
ABB CoDeSys Serial
ABB Mint Controller HCP
CoDeSys ETH
Modbus TCP Server
Modbus RTU Server
Modbus TCP
Modbus RTU
Variables

Protocol Editor Settings

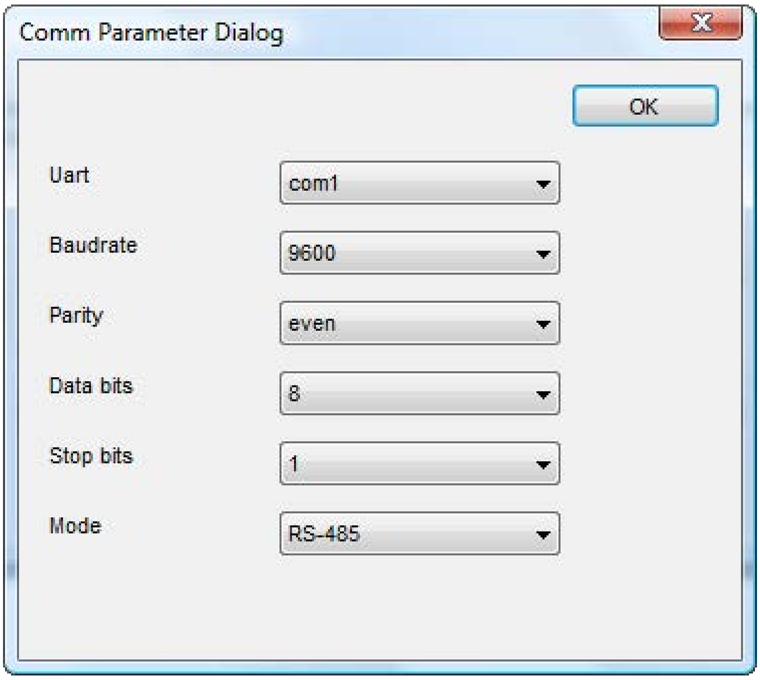


Modbus RTU Server

Modbus ID: 1

PLC Models:
- standard modbus
- zero based modbus

Parameter	Description
Modbus ID	The HMI device is a slave in the network and will exchange data with a single master controller. Every panel in the network must be assigned its own slave ID. PLC communication parameters must be set according to the values programmed in the master controller.
PLC Models	The *controller setup* dialog box allows you to select between two different flavours of Modbus addressing. You can select the flavour most appropriate for your needs. The *standard modbus* model type implements a holding register range of between 400001 and 402048 and an output coils range of between 1 and 2048. The *zero based modbus* model type, on the other hand, implements a holding register range of between 400000 and 402047 and an output coils range of between 0 and 2047. Please note that the address range used in the Modbus frames will always be respectively between 0 and 1097 for the holding registers and between 0 and 9997 for coils.



Comm Parameter Dialog

Uart: com1

Baudrate: 9600

Parity: even

Data bits: 8

Stop bits: 1

Mode: RS-485

Parameter	Description
Uart	Serial port selection. COM1 is the HMI PLC port, COM2 is the PC/printer port.
Baud rate, Parity, Data bits, Stop bits	Communication parameters for the serial line.
Mode	Serial port mode. Can be selected between: RS-232, RS-485 (2 wires), RS-422 (4 wires).

Communication Status

The actual communication status can be displayed using the dedicated system variables. Please refer to the manual for further information about available system variables and their use.

This communication protocol acts as server and does not return any specific protocol error message.

Implementation Details

This Modbus RTU slave implementation supports only a subset of the standard Modbus function codes. Only the function codes necessary for the data exchange between the panel and the Modbus master have been implemented.

The supported function codes are listed in the table below.

Code	Function	Description
01	Read Coil Status	Read multiple bits in the panel coil area.
03	Read Holding Registers	Read multiple panel registers.
05	Force Single Coil	Force a single panel coil to either ON or OFF.
06	Preset Single Register	Preset a value in a panel register.
08	Loopback Diagnostic Test	Only sub function 00 (return query data) is supported.
15	Force Multiple Coils	Force multiple panel coils to either ON or OFF.
16	Preset Multiple Registers	Preset value in multiple panel registers.
17	Report Slave ID	Returns some diagnostic information of the controller present at the slave address.

The HMI protocol will return the exception code 01 (illegal function) if the function code received in the query is not supported.

The HMI protocol will return the exception code 03 (illegal data value) if a sub function other than 00 is specified for function 08.

The amount of memory available in the HMI device is as follows:

Data Type	Type	Range
Coils	Bit	0 – 2047
Registers	Word	0 – 2047

The HMI protocol will return the exception code 02 (illegal data address) if the data address received in the query exceeds the predefined data ranges.

Modbus TCP Driver

Various Modbus TCP-capable devices can be connected to the operator panel. To set-up your Modbus TCP device, please refer to the documentation you have received with the device.

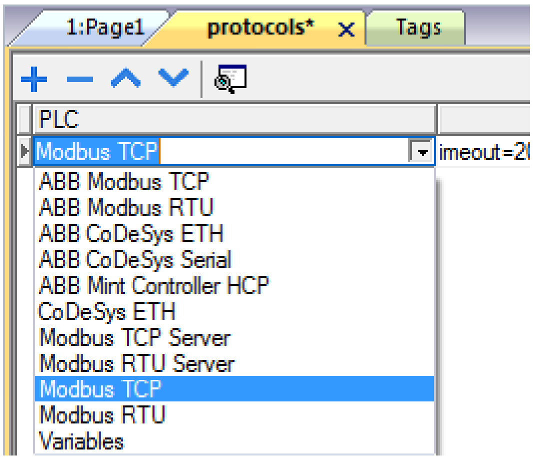
The HMI protocol identifies the Modbus TCP devices using their IP addresses. You should take note of these addresses as you assign them because you will need them later in the set-up phase of the user interface application.

Different physical media, gateways, routers and hubs can be used in the communication network. Also, other devices can independently make simultaneous use of the network. However, it is important to ensure that the traffic generated by these devices does not degrade the communication speed (round-trip time) to an unacceptable level. Too slow communication between the panel and the Modbus TCP device may result in low display update rate.

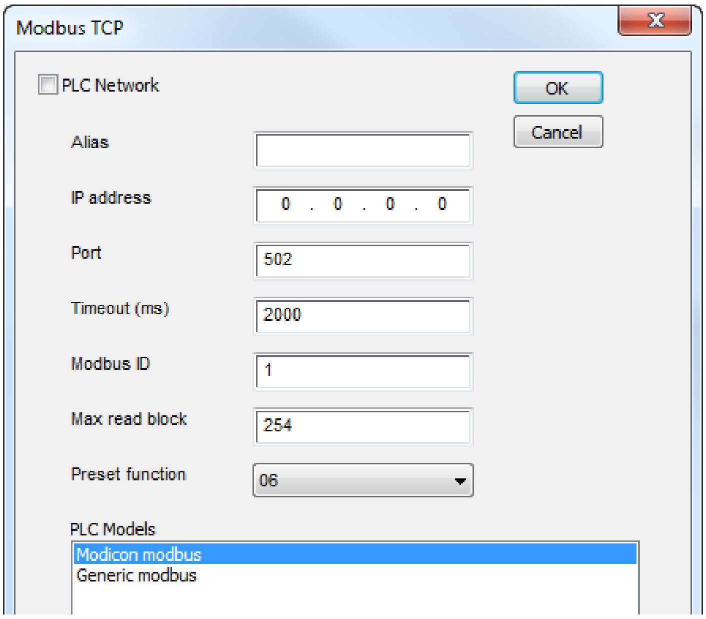
- The implementation of the protocol operates as a Modbus TCP client only.
- The HMI Modbus TCP protocol uses the standard port number 502 as the destination port.
- The HMI Modbus TCP protocol supports the standard commonly referred as “Ethernet II”.

Protocol Editor Settings

→ Add with the button **+** a driver in the protocol editor and select the protocol called **Modbus TCP** from the list of available protocols.



The driver configuration dialog is shown in the following figure.


Modbus TCP

□ PLC Network

Alias:

IP address: 0 . 0 . 0 . 0

Port: 502

Timeout (ms): 2000

Modbus ID: 1

Max read block: 254

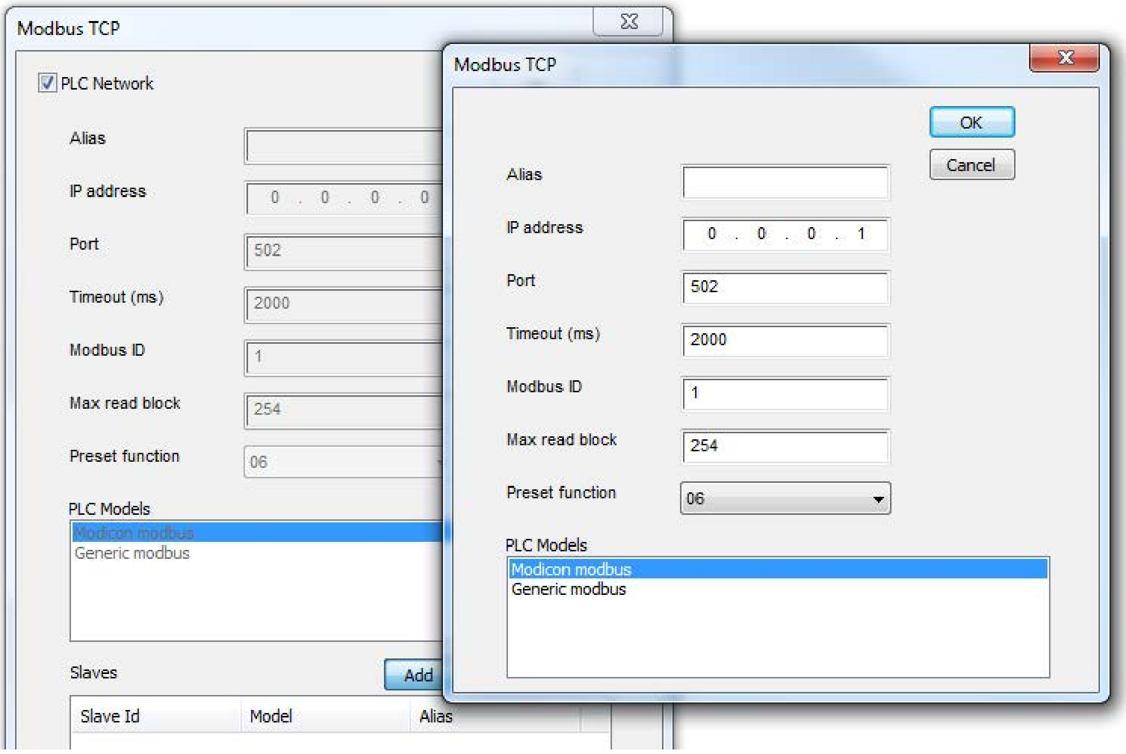
Preset function: 06

PLC Models

Modicon modbus

Generic modbus

Parameter	Description
Alias	Name to be used to identify nodes in network configurations. The name will be added as a prefix to each tag name imported for each network node.
IP address	Ethernet IP address of the controller.
Port	Allows to change the default port number used by the Modbus TCP driver; it could be useful whenever the communication passes through routers or Internet gateways where the default port number is already in use.
Timeout (ms)	Defines the time inserted by the protocol between two retries of the same message in case of missing response from the server device. It is expressed in milliseconds.
Modbus ID	The Modbus ID is rarely used and in most cases can be left zero. The value of the Modbus ID is simply copied into the unit identifier field of the Modbus TCP communication frame. Usually it is used when communicating over Ethernet-to-Serial bridges and is then representing the slave ID.
Max read block	Maximum length in bytes of a data block request.
Preset Function	Specifies what Modbus function will be used for write operations to the holding registers data type. The user can select between the function 06 (preset single register) and function 16 (preset multiple registers). If the Modbus function 06 is selected, the protocol will always use function 06 to write to the controller, even to write multiple consecutive registers. If the Modbus function 16 is selected, the protocol will always use function 16 to write to the controller, even for a single register write request and the Byte Count parameter of the query is set to two. Using function 16 may result in higher communication performance.
PLC Models	Two PLC models are available. The *Modicon modbus* model has an addressing space that starts from offsets 1 for all the memory types. The *Generic modbus* model has an addressing space that starts from offset 0 for all the memory types.
PLC Network	The protocol allows the connection of multiple controllers to one operator panel. To set-up multiple connections, check **PLC network** checkbox and enter IP Address for all controllers.



Modbus TCP

☑ PLC Network

Alias

IP address 0 . 0 . 0 . 0

Port 502

Timeout (ms) 2000

Modbus ID 1

Max read block 254

Preset function 06

PLC Models

Modicon modbus

Generic modbus

Slaves

Slave Id	Model	Alias

Modbus TCP

Alias

IP address 0 . 0 . 0 . 1

Port 502

Timeout (ms) 2000

Modbus ID 1

Max read block 254

Preset function 06

PLC Models

Modicon modbus

Generic modbus

Tag Import

The Modbus TCP driver supports the generic import of tags when provided in CSV format according to the following format:

- NodeID, TagName, MemoryType, Address, DataFormat,...,[Comment]

The fields in brackets are optional as well as the fields between Data Format and Comment.

The meaning of each field is described below.

Field	Description
NodeID	Identifies the node which the tag belongs to.
TagName	A string that describes the tag itself.
MemoryType	Should be one of the following: • a. OUTP • b. INP • c. IREG • d. HREG
Address	Offset compatible with Modbus notation.
DataFormat	Data type in internal ICOM notation: • a. boolean • b. byte • c. short • d. int • e. unsignedByte • f. unsignedShort • g. unsignedInt • h. float • i. double • j. string • k. binary • l. Time • m. boolean [] • n. byte [] • o. short [] • p. unsignedShort [] • q. unsignedInt [] • r. float [] • s. double [] • t. Time []
Comment	A string for additional description; optional.

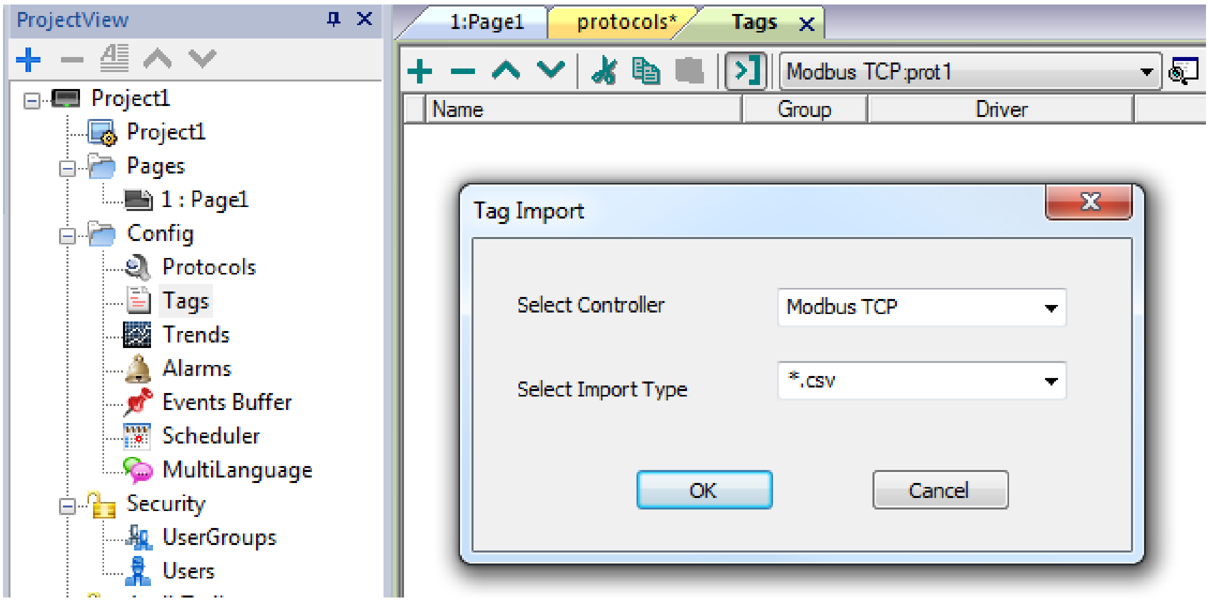
Example of CSV line:

- 2,Holding Register 1, HREG, 400001, unsignedShort,

NOTE

Here we have a line without comment. Notice the comma at the end of line. In fact, though comment is missing, the comma is mandatory.

1. Select the driver in the tag editor and click on the **Import tag** button to start the importer.

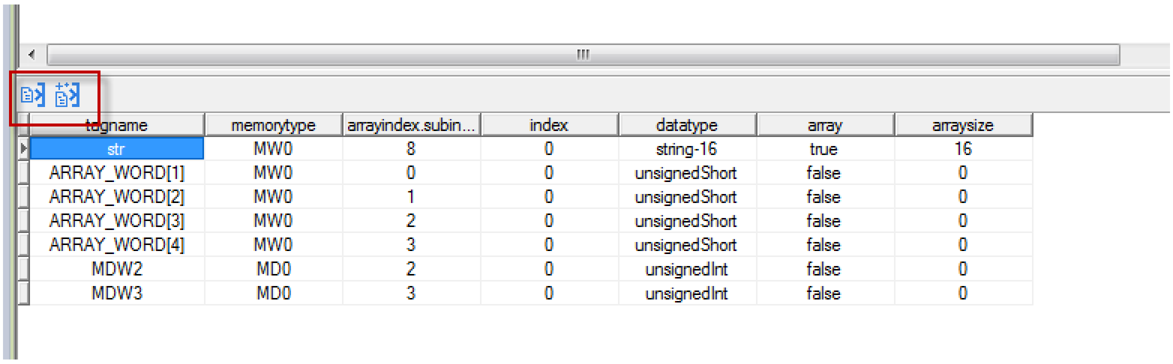


Select Controller: Modbus TCP

Select Import Type: *.csv

2. Locate the ".csv" file and confirm with **OK** button.

The tags present in the exported document are listed in the tag dictionary from where they can be directly added to the project using the **add tags** button as shown in the following figure.

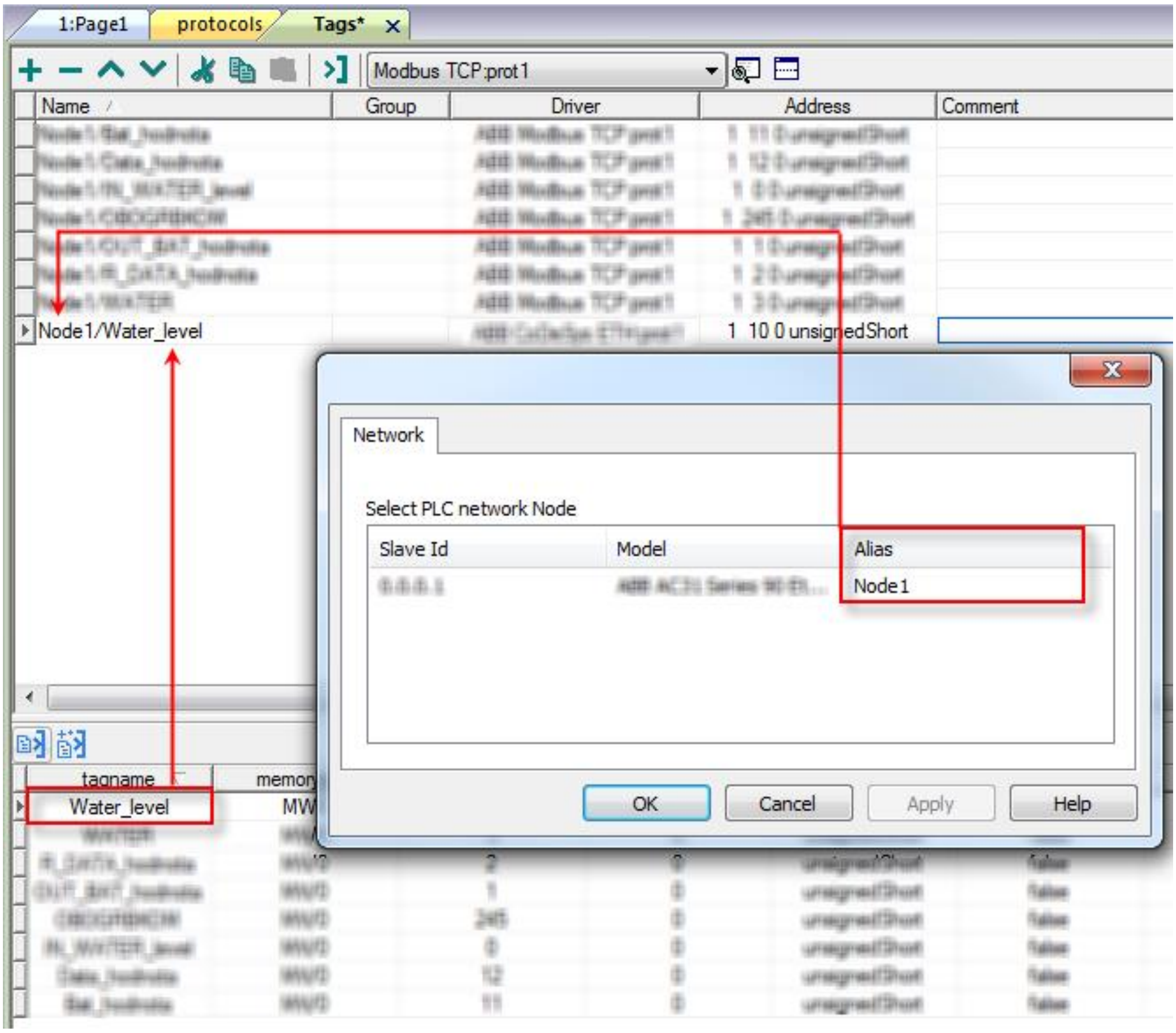


tagname	memorytype	arrayindex.subin...	index	datatype	array	arraysize
str	MW0	8	0	string-16	true	16
ARRAY_WORD[1]	MW0	0	0	unsignedShort	false	0
ARRAY_WORD[2]	MW0	1	0	unsignedShort	false	0
ARRAY_WORD[3]	MW0	2	0	unsignedShort	false	0
ARRAY_WORD[4]	MW0	3	0	unsignedShort	false	0
MDW2	MD0	2	0	unsignedInt	false	0
MDW3	MD0	3	0	unsignedInt	false	0

Aliasing Tag Names in Network Configurations

Tag names must be unique at project level; it often happens that the same tag names are to be used for different controller nodes (for example when the HMI is connected to two devices that are running the same application). Since tags include also the identification of the node and tag editor does not support duplicate tag names, the import facility in tag editor has an aliasing feature that can automatically add a prefix to imported tags. With this feature tag names can be done unique at project level.

The feature works when importing tags for a specific protocol. Each tag name will be prefixed with the string specified by the "Alias". As shown in the figure below, the connection to a certain controller is assigned the name "Node1". When tags are imported for this node, all tag names will have the prefix "Node1" making each of them unique at the network/project level.



Name	Group	Driver	Address	Comment
Node1/Water_level			1 10 0 unsignedShort	

Slave Id	Model	Alias
		Node1

tagname	memory
Water_level	MW

NOTE

Aliasing tag names is only available when tags can be imported. Tags which are added manually in the tag editor do not need to have the Alias prefix in the tag name.
The Alias string is attached to the tag name only at the moment the tags are imported using tag editor. If you modify the Alias string after the tag import has been completed, there will be no effect on the names already present in the dictionary. When the Alias string is changed and tags are imported again, all tags will be imported again with the new prefix string.

Special Data Types

The Modbus TCP driver provides one special data type called "Node Override IP".

The Node Override IP permits to change at runtime the IP address of the target controller. This memory type is an array of 4 unsigned bytes, one per each byte of the IP address.

The Node Override IP is initialized with the value of the controller IP specified in the project at programming time.

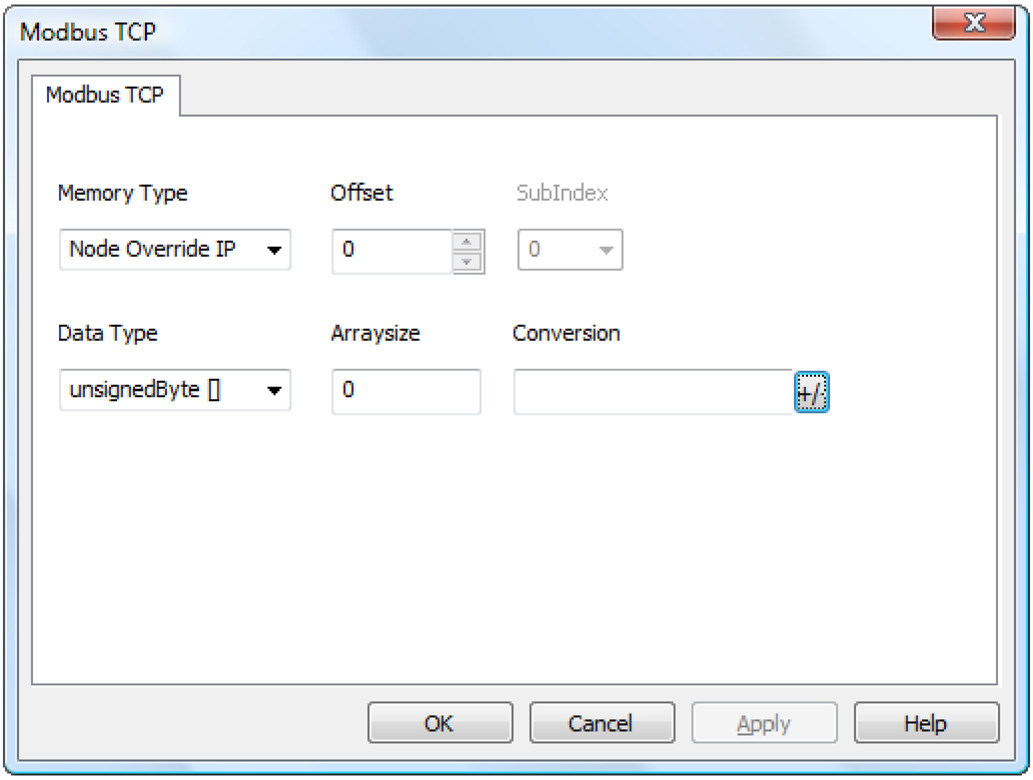
If the Node Override is set to 0.0.0.0, all the communication with the slave is stopped, no request frames are generated anymore.

If the Node Override has a value different from 0.0.0.0, it is interpreted as Node IP Override and the target IP address is replaced runtime with the new value.

In case the panel has been configured to access to a network of controllers, each node has its own override variable.

NOTE

The Node Override values assigned at runtime are not retained through power cycles.



Modbus TCP

Memory Type: Node Override IP

Offset: 0

SubIndex: 0

Data Type: unsignedByte []

Arraysize: 0

Conversion:

Communication Status

The current communication status can be displayed using the dedicated system variables. Please refer to the user manual for further information about available system variables and their use.

The codes supported for this communication driver are:

Error	Notes
NAK	Returned in case the controller replies with a not acknowledge.
Timeout	Returned when a request is not replied within the specified timeout period; ensure the controller is connected and properly configured to get network access.
Invalid response	The panel did receive from the controller a response, but its format or its contents is not as expected; ensure the data programmed in the project are consistent with the controller resources.
General Error	Error cannot be identified; should never be reported; contact technical support.

Implementation Details

This Modbus TCP implementation supports only a subset of the standard Modbus TCP Function Codes.

The supported Function Codes are listed in the table below.

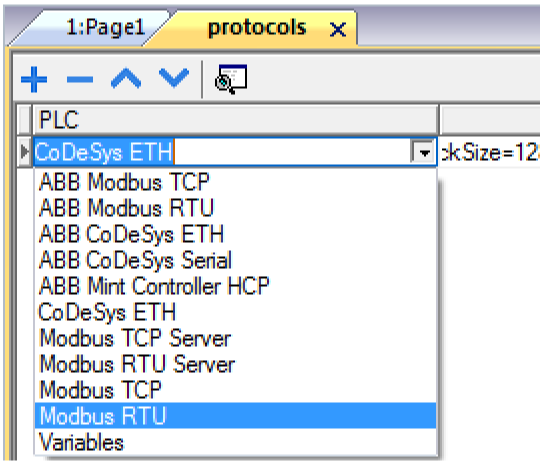
Code	Function	Description
01	Read Coil Status	Read multiple bits in the panel coil area.
02	Read Input Status	Read the ON/OFF status of the discrete inputs (1x reference) in the slave.
03	Read Holding Registers	Read multiple registers.
04	Read Input Registers	Read the binary contents of input registers (3x reference) in the slave.
05	Force Single Coil	Force a single soil to either ON or OFF.
06	Preset Single Register	Preset a value in a register.
16	Preset Multiple Registers	Preset value in multiple registers.

Modbus RTU Driver

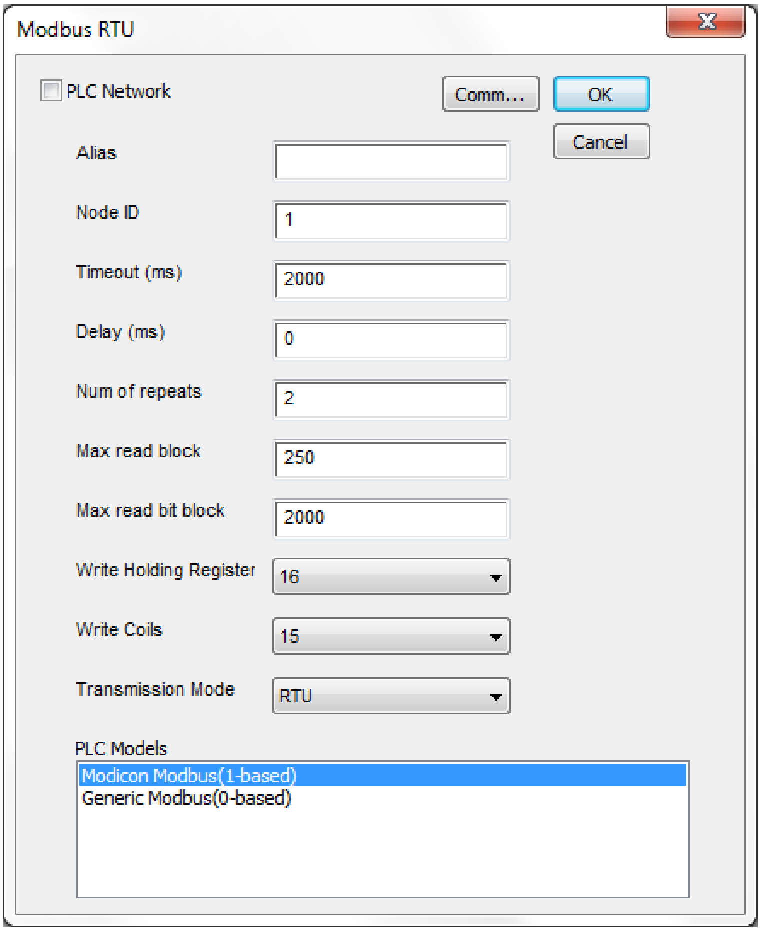
The operator panels can be connected to a Modbus network as the network master using this generic driver.

Protocol Editor Settings

→ Add with the button **+** a driver in the protocol editor and select the protocol called **Modbus RTU** from the list of available protocols.



The driver configuration dialog is shown in the following figure.



Modbus RTU

□ PLC Network

Alias

Node ID 1

Timeout (ms) 2000

Delay (ms) 0

Num of repeats 2

Max read block 250

Max read bit block 2000

Write Holding Register 16

Write Coils 15

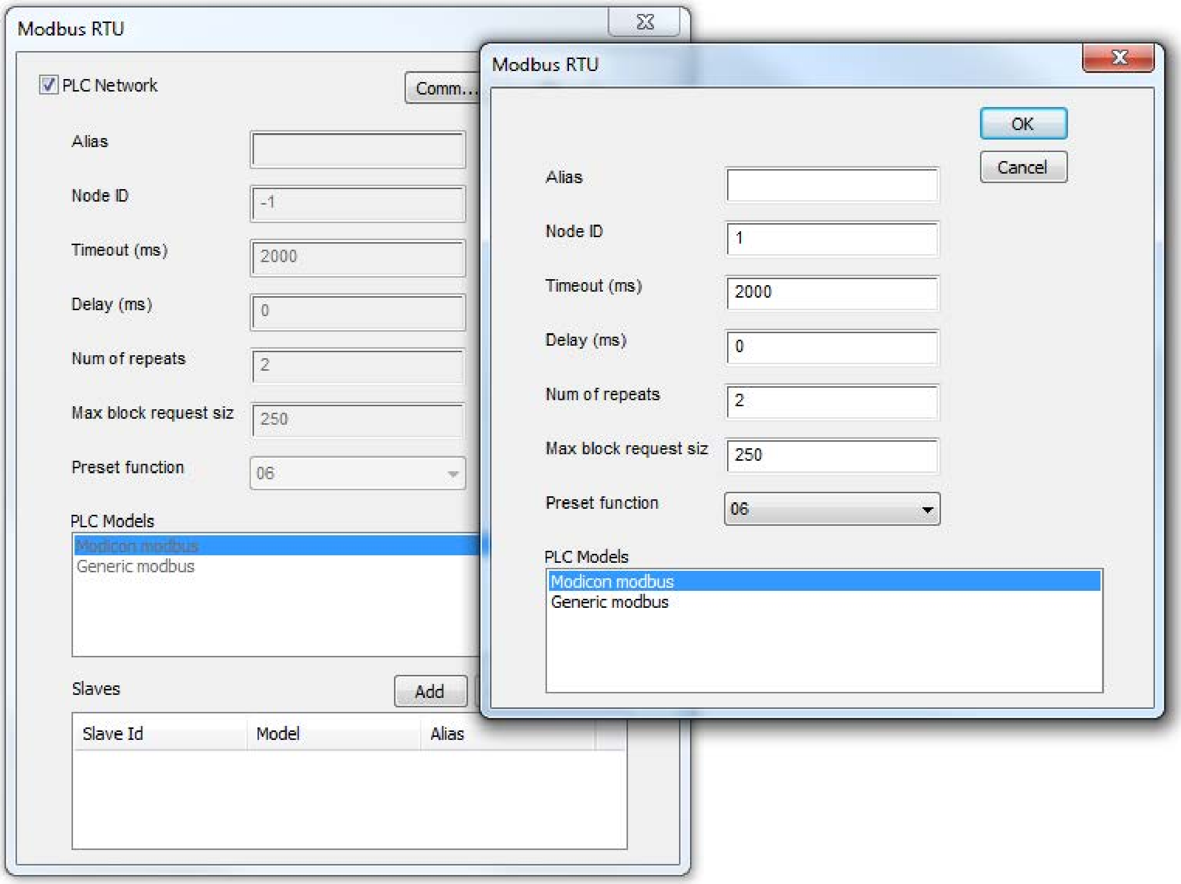
Transmission Mode RTU

PLC Models

Modicon Modbus(1-based)

Generic Modbus(0-based)

Parameter	Description
Alias	Name to be used to identify nodes in network configurations. The name will be added as a prefix to each tag name imported for each network node.
Node ID	Modbus node of the slave device.
Timeout (ms)	Defines the time inserted by the protocol between two retries of the same message in case of missing response from the server device. It is expressed in milliseconds.
Delay (ms)	This parameter defines a fixed time delay in the communication between the end of the last received frame and the starting of a new request; when set to 0, the new request will be issued as soon as the internal system is able to reschedule it.
Num of repeats	This parameter defines the number of times a certain message will be sent to the controller before reporting the communication error status. A value of 1 for the parameter “No of repeats” means that the panel will eventually report the communication error status if the response to the first request packet is not correct.
Max read block	Maximum length in bytes of a data block request. It applies only to read access of Holding Registers
Max read bit block	Maximum length in bits of a block request. It applies only to read access of Input Bits and Output Coils.
Write Holding Register	Specifies what Modbus Function will be used for write operations to the Holding Registers data type. The user can select between the function 06 (preset single register) and function 16 (preset multiple registers). If Modbus function 06 is selected, the protocol will always use function 06 to write to the controller, even to write multiple consecutive registers. If Modbus function 16 is selected, the protocol will always use function 16 to write to the controller, even for a single register write request and the “Byte Count” parameter of the query is set to two. Using function 16 may result in higher communication performance.
Write Coils	Specifies what Modbus Function will be used for write operations to the Output Coils data type. The user can select between the function 05 (force single coil) and function 15 (write multiple coils). If Modbus function 05 is selected, the protocol will always use function 05 to write to the controller, even to write multiple consecutive coils. If Modbus function 15 is selected, the protocol will always use function 15 to write to the controller, even for a single coil write request. Using function 15 may result in higher communication performance.
Transmission Mode	RTU: use RTU mode ASCII: use ASCII mode Note: when PLC network is active, all nodes will be configured with the same Transmission Mode.
PLC Models	Two PLC models are available. The *Modicon modbus* model has an addressing space that starts from offsets 1 for all the memory types. The *Generic Modbus* model has an addressing space that starts from offset 0 for all the memory types.
PLC Network	The protocol allows the connection of multiple controllers to one operator panel. To set-up multiple connections, check **PLC network** checkbox and create your network using the command “Add” per each slave device you need to include in the network.



Modbus RTU

☑ PLC Network

Alias

Node ID -1

Timeout (ms) 2000

Delay (ms) 0

Num of repeats 2

Max block request siz 250

Preset function 06

PLC Models

- Modicon modbus
- Generic modbus

Slaves

Slave Id	Model	Alias

Modbus RTU

Alias

Node ID 1

Timeout (ms) 2000

Delay (ms) 0

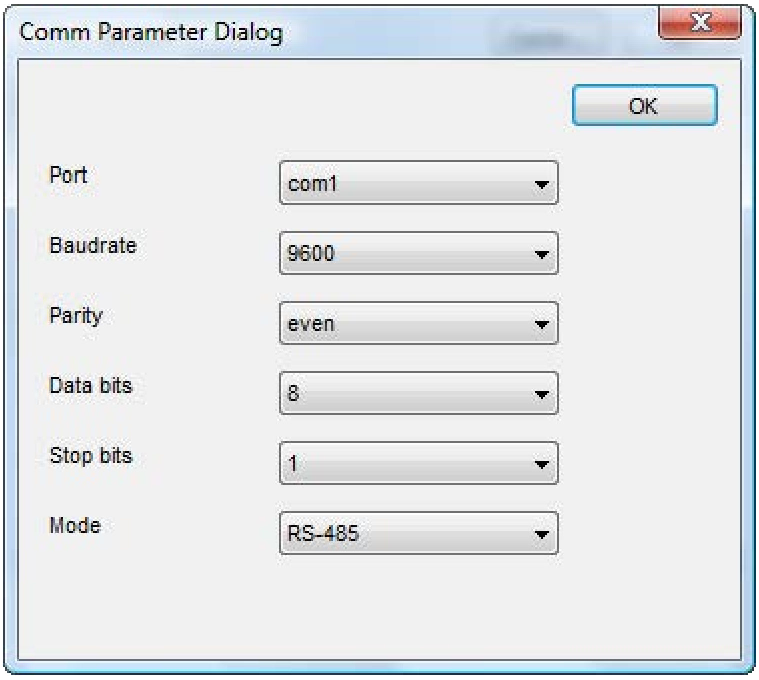
Num of repeats 2

Max block request siz 250

Preset function 06

PLC Models

- Modicon modbus
- Generic modbus



Comm Parameter Dialog

Port com1

Baudrate 9600

Parity even

Data bits 8

Stop bits 1

Mode RS-485

Parameter	Description
Port	Serial port selection. COM1 is the panel PLC port, COM2 is the PC/printer port.
Baud rate, Parity, Data bits, Stop bits	Communication parameters for the serial line.
Mode	Serial port mode. Can be selected between: RS-232, RS-485 (2 wires), RS-422 (4 wires).

Tag Import

The Modbus RTU driver supports the generic import of tags when provided in CSV format according to the following format:

- NodeID, TagName, MemoryType, Address, DataFormat,...,[Comment]

The fields in brackets are optional as well as the fields between Data Format and Comment.

The meaning of each field is described below.

Field	**Description**
NodeID	Identifies the node which the tag belongs to.
TagName	A string that describes the tag itself.
MemoryType	Should be one of the following: • a. OUTP • b. INP • c. IREG • d. HREG
Address	Offset compatible with Modbus notation.
DataFormat	Data type in internal ICOM notation: • boolean • byte • short • int • unsignedByte • unsignedShort • unsignedInt • float • double • string • binary • Time • boolean [] • byte [] • short [] • unsignedShort [] • unsignedInt [] • float [] • double [] • Time []
Comment	A string for additional description; optional.

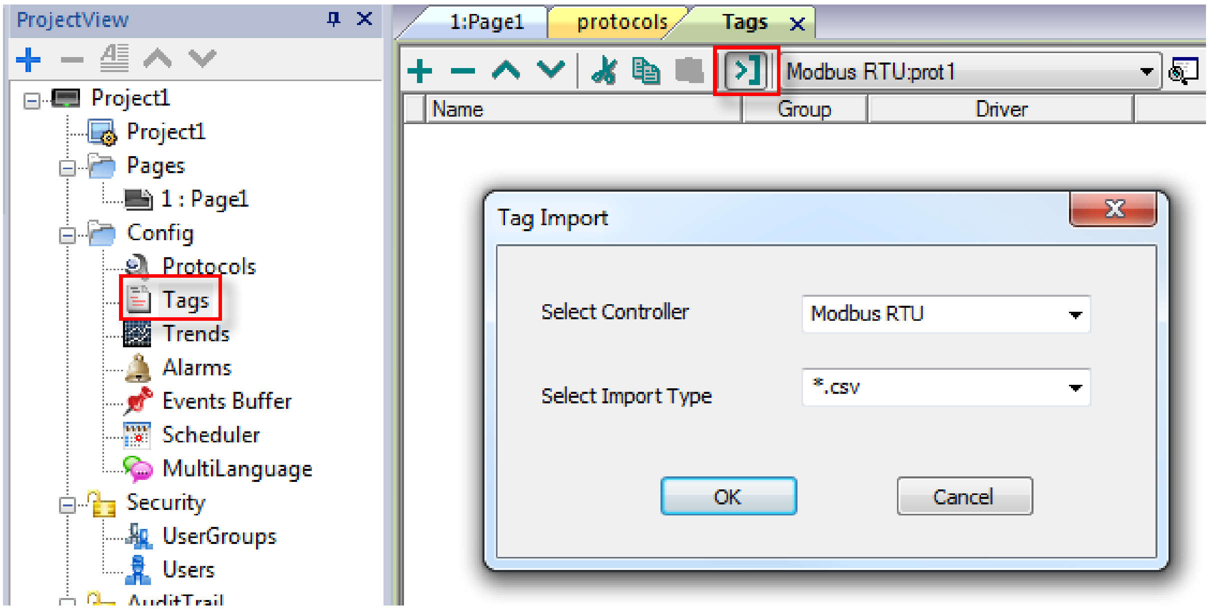
Example of CSV line:

- 2,Holding Register 1, HREG, 400001, unsignedShort,

NOTE

Here we have a line without comment. Notice the comma at the end of line. In fact, though comment is absent, the comma is mandatory.

1. Select the driver in the tag editor and click on the **Import tag** button to start the importer.



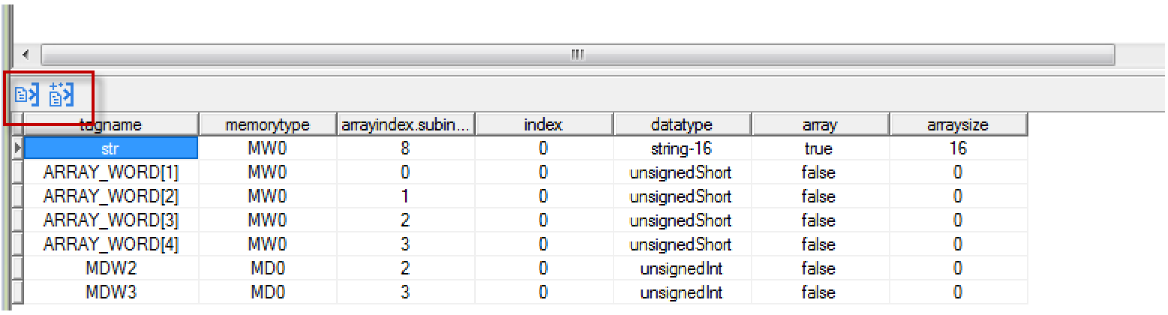
Tag Import

Select Controller: Modbus RTU

Select Import Type: *.csv

2. Locate the ".csv" file and confirm with **OK.**

The tags present in the exported document are listed in the tag dictionary from where they can be directly added to the project using the **add tags** button as shown in the following figure.

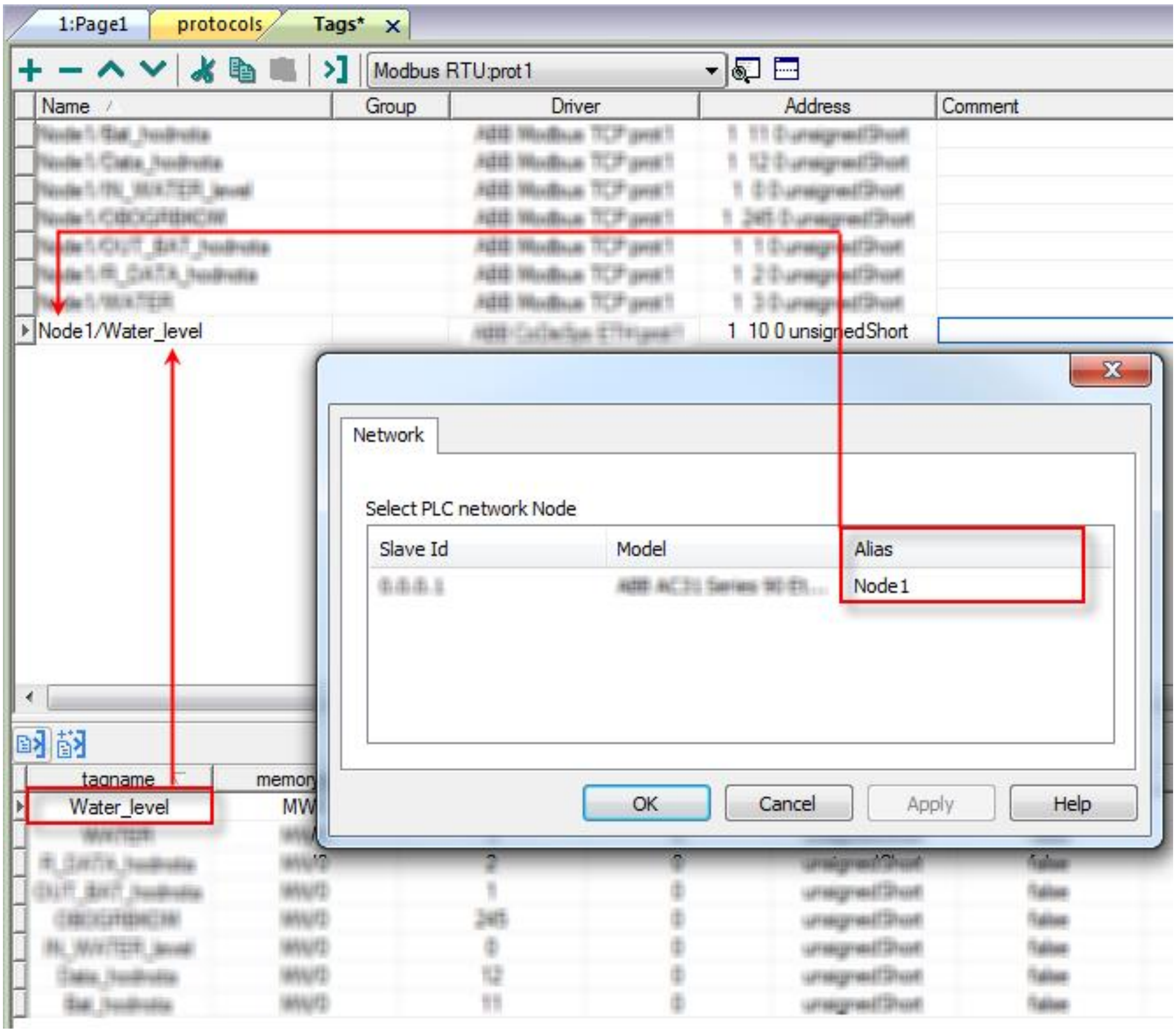


tagname	memorytype	arrayindex.subin...	index	datatype	array	arraysize
str	MW0	8	0	string-16	true	16
ARRAY_WORD[1]	MW0	0	0	unsignedShort	false	0
ARRAY_WORD[2]	MW0	1	0	unsignedShort	false	0
ARRAY_WORD[3]	MW0	2	0	unsignedShort	false	0
ARRAY_WORD[4]	MW0	3	0	unsignedShort	false	0
MDW2	MD0	2	0	unsignedInt	false	0
MDW3	MD0	3	0	unsignedInt	false	0

Aliasing Tag Names in Network Configurations

Tag names must be unique at project level; it often happens that the same tag names are to be used for different controller nodes (for example when the HMI is connected to two devices that are running the same application). Since tags include also the identification of the node and tag editor does not support duplicate tag names, the import facility in tag editor has an aliasing feature that can automatically add a prefix to imported tags. With this feature tag names can be done unique at project level.

The feature works when importing tags for a specific protocol. Each tag name will be prefixed with the string specified by the "Alias". As shown in the figure below, the connection to a certain controller is assigned the name "Node1". When tags are imported for this node, all tag names will have the prefix "Node1" making each of them unique at the network/project level.



Modbus RTU:prot1

Name	Group	Driver	Address	Comment
Node1/Water_level			1 10 0 unsignedShort	

Network

Select PLC network Node

Slave Id	Model	Alias
		Node1

tagname	memory
Water_level	MW

NOTE	Aliasing tag names is only available when tags can be imported. Tags which are added manually in the tag editor do not need to have the Alias prefix in the tag name. The Alias string is attached to the tag name only at the moment the tags are imported using tag editor. If you modify the Alias string after the tag import has been completed, there will be no effect on the names already present in the dictionary. When the Alias string is changed and tags are imported again, all tags will be imported again with the new prefix string.

Special Data Types

The Modbus RTU driver provides one special data type called "Node Override ID".

The Node Override ID allows changing at runtime the address of the target controller. This memory type is an unsigned byte.

The Node Override ID is initialized with the value of the controller ID specified in the project at programming time.

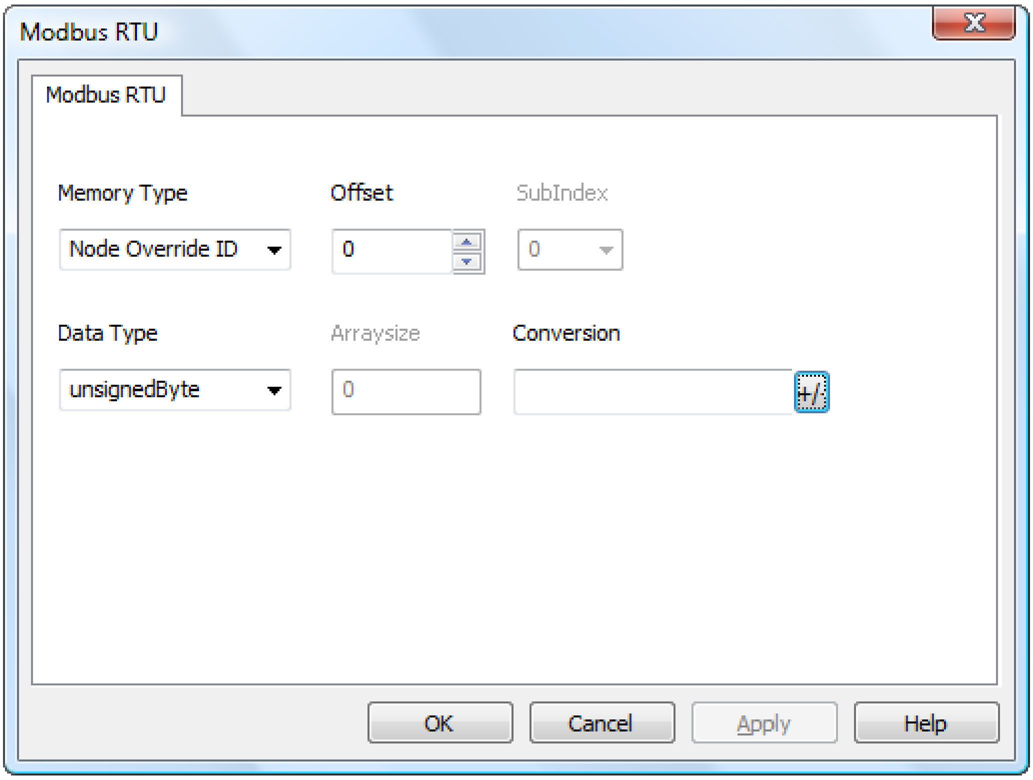
If the Node Override is set to 0, all the communication with the slave is stopped, no request frames are generated anymore.

If the Node Override has a value different from 0, it is interpreted as Node ID Override and the target ID address is replaced runtime with the new value.

In case the panel has been configured to access to a network of controllers, each node has its own override variable.

NOTE

The Node Override values assigned at runtime are not retained through power cycles.



Memory Type: Node Override ID

Offset: 0

SubIndex: 0

Data Type: unsignedByte

Arraysize: 0

Conversion:

Communication Status

The current communication status can be displayed using the dedicated system variables. Please refer to the User Manual for further information about available system variables and their use.

The codes supported for this communication driver are:

Error	**Notes**
NAK	Returned in case the controller replies with a not acknowledge.
Timeout	Returned when a request is not replied within the specified timeout period; ensure the controller is connected and properly configured to get network access.
Line Error	Returned when an error on the communication parameter setup is detected (parity, baud rate, data bits, stop bits); ensure the communication parameter settings of the controller is compatible with panel communication setup.
Invalid response	The panel did receive a response from the controller, but its format or its content is not as expected; ensure the data programmed in the project are consistent with the controller resources.
General Error	Error cannot be identified; should never be reported; contact technical support.

Implementation Details

This Modbus RTU implementation supports only a subset of the standard Modbus function codes.

The supported function codes are listed in the table below.

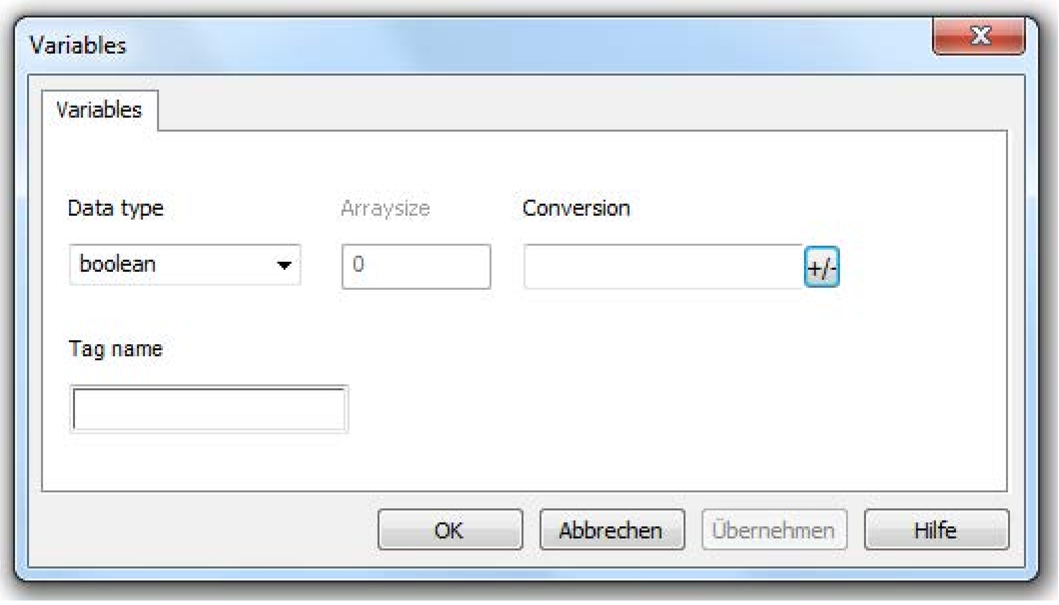
Code	Function	Description
01	Read Coil Status	Read multiple bits.
02	Read Input Status	Read the ON/OFF status of the discrete inputs (1x reference) in the slave.
03	Read Holding Registers	Read multiple registers.
04	Read Input Registers	Read the binary contents of input registers (3x reference) in the slave.
05	Force Single Coil	Force a single coil to either ON or OFF.
06	Preset Single Register	Preset a value in a register.
16	Preset Multiple Registers	Preset value in multiple registers.

- Communication speed with controllers is supported up to 57600 baud.
- Floating point data format is compliant to the IEEE standard.

Variables

Selecting *Variables* in the protocols menu gives the user the possibility, to define variables independant of any PLC and any defined protocol.

This option is helpful for creating applications e.g. for demonstration purposes. By means of this popup window the variables become specified.



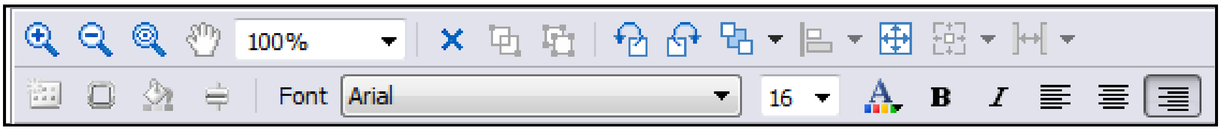
Programming Concepts

The programming guidelines for PB610 Panel Builder 600 are based on a few basic concepts, which are recurrent in many parts of the system.

Attach To

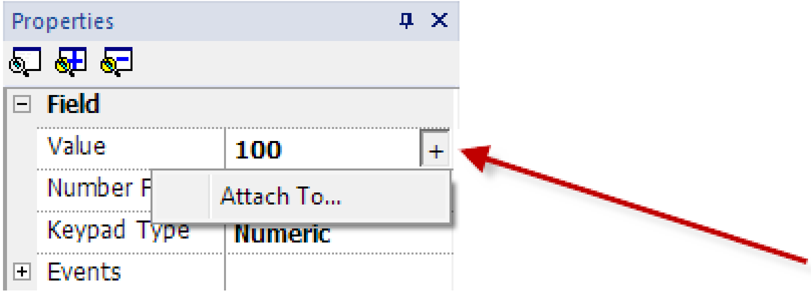
In PB610 Panel Builder 600 the basic programming techniques are used in configuring the properties for an object in page. Objects' properties can be changed at programming time or configured to be dynamic.

→ To change a property use the page toolbar or the property pane which shows the properties available for the selected object.

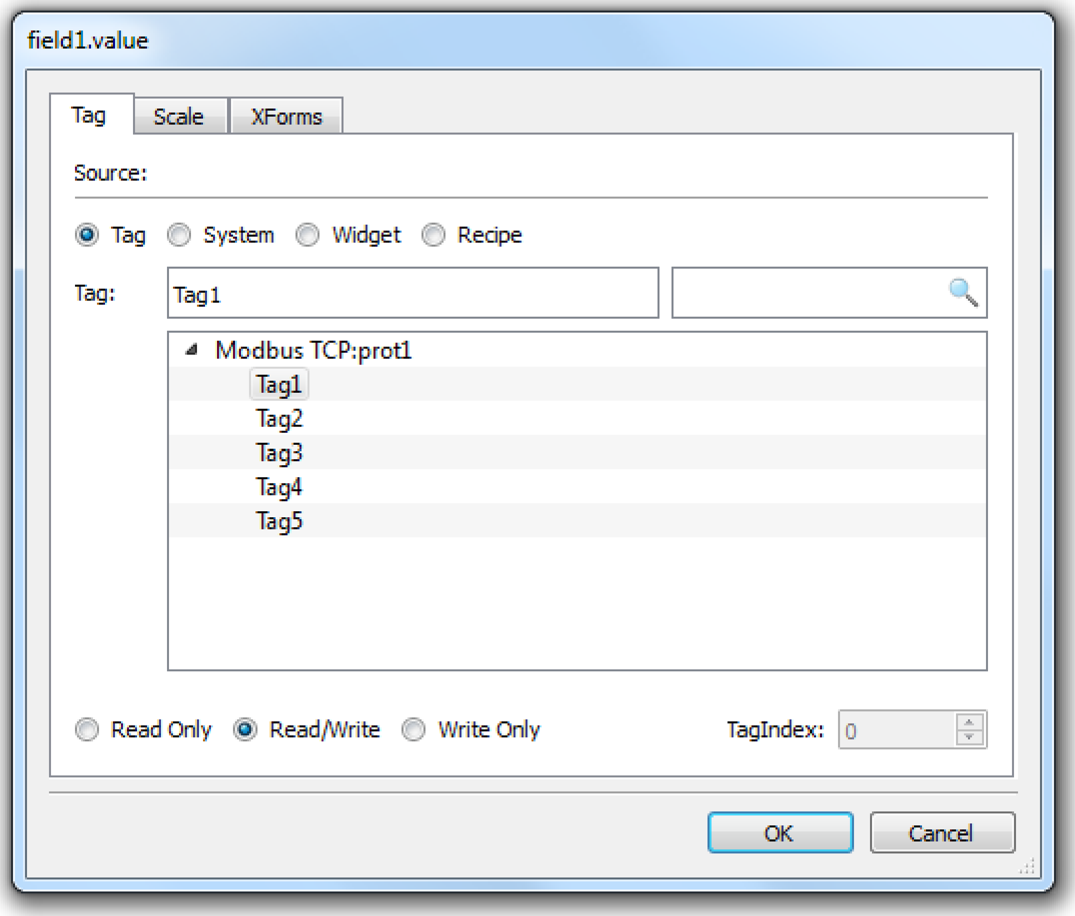


The page toolbar permits a quick change at programming time of the most commonly used object's properties. When you need a complete view of all the properties of a certain object you need to use the property pane. The property pane allows you to change a property at programming time and to attach the property to a dynamic element.

From the property pane, when you click on the right side of a property cell, you get the possibility to "attach" the property to a second element. This operation is done using the "Attach To" dialog.



Tab "Tag"



field1.value

Source: ● Tag ○ System ○ Widget ○ Recipe

Tag: Tag1

○ Read Only ● Read/Write ○ Write Only

TagIndex: 0

The tab “Tag” allows to attach the property to an element. The *Source* can be selected using the radio buttons. The elements to which the property can be attached are:

- Tags
- System variables (see chapter “System Variables” for an explanation of the meanings of all system variables)
- Properties from another widget
- Elements of a recipe

The radio buttons at the bottom allow to set the access type. The *TagIndex* selection is used in case of arrays to determine the array element.

While adding tags, the protocols used in the project are shown in the tag dialog and when expanding each protocol the corresponding tags can be seen. The tags will be arranged in alphabetical order inside each protocol.

There is an option to search the tag to be attached by its name as shown in the figure above. This makes it easy to find tags.

The search can be done in two ways:

1. Start typing the tag name in the left box:
   - ➤ This will “jump” into the list to the first tag starting with the entered characters.
2. Type in the search box any part of a tag name:
   - ➤ This will automatically apply a filter to the view so that only the tags which contain the entered characters are displayed.

NOTE

Tag sorting, tag grouping and tag filtering are features available starting from version 1.80.

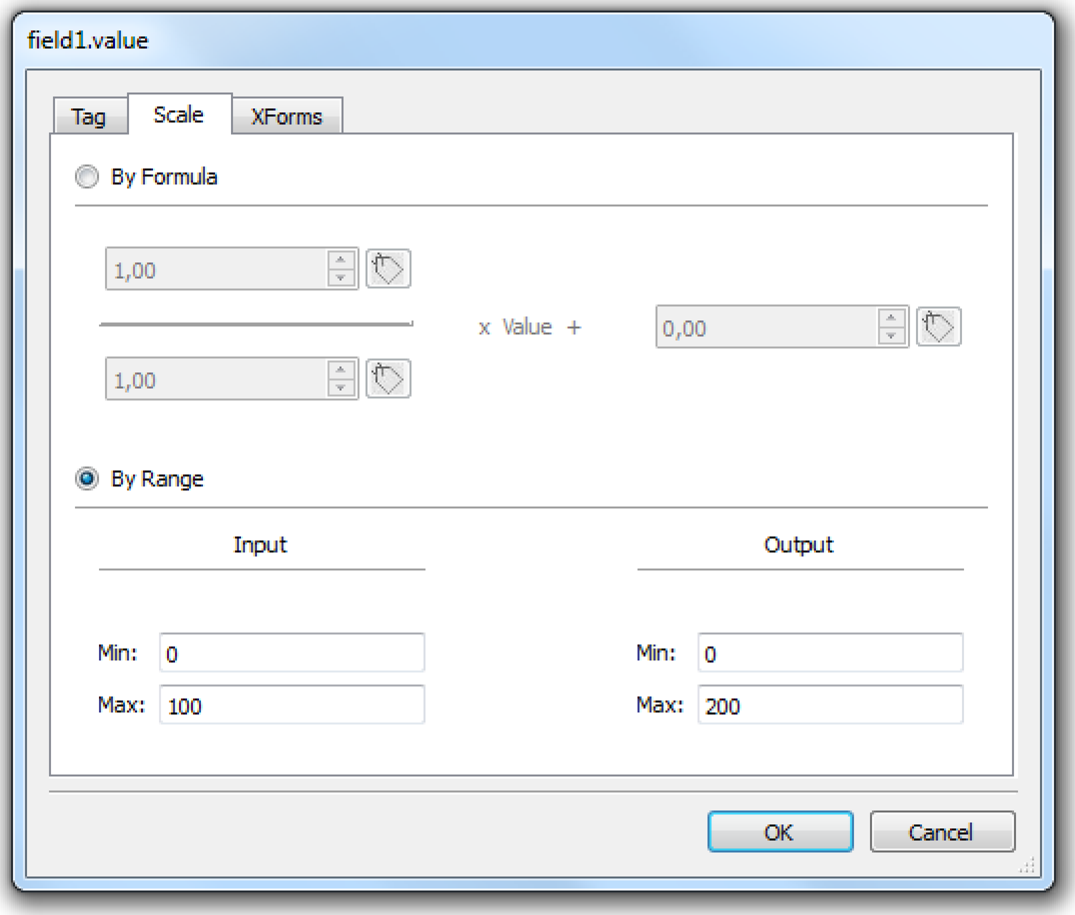
Tabs “Scale”, “XForms”

“Scale/XForms” allow to apply transformations to the numeric value of the source element before they are applied to the property of widget. Transformations can be simple linear relationships or generic transformations. Linear scaling can be configured when selecting the **Scale** tab and they can be specified in terms of a formula or by range. In case the range mode is selected, you just need to specify the input and output range while the system will automatically calculate the factors for the formula.

Special transformations are available when you click on the **Transform** radio button. Currently supported transformations are:

Color conversion: Allows to define a map between numeric values of the tag and colors to be assigned to the property. This feature is used to change the color of a button, for example, based on the value of a tag. If the tag is an integer, you can have many different colors based on the tag value.

Bit and byte index: Allows extracting a single bit or byte content from a word depending on the specified bit or byte number.



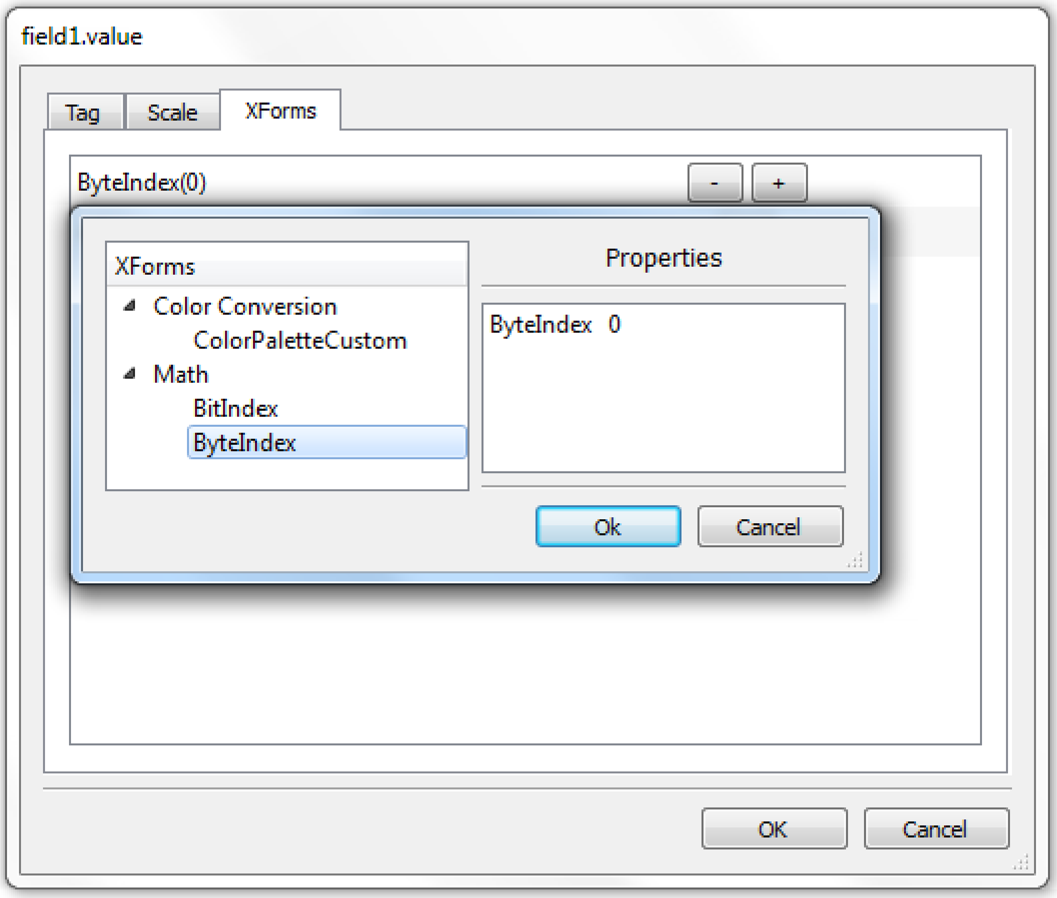
○ By Formula

1,00 / 1,00 x Value + 0,00

● By Range

	Input		Output
Min:	0	Min:	0
Max:	100	Max:	200

Example of transformation:
scaling (100/10*value + 5), byteIndex(0), bitIndex(1)   that is equivalent to:
bitIndex (byteIndex (100/10*value+5,0) , 1)



ByteIndex 0

Events

In a Panel Builder 600 application, Events are the way to trigger Actions at the application level.

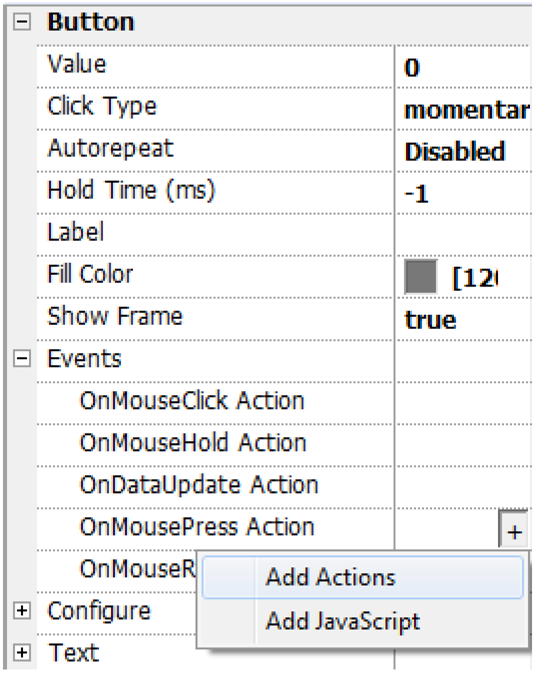
Main types of Events:

- Events related to buttons/ touch (Click, Press, Release, Release)
- Events related to external input devices like keyboards & mouse (Click, Press, Hold, Release, Wheel)
- Events related to data changes (OnDataUpdate)
- Events related to switch pages (OnActivate, OnDeactivate)
- Events related to alarms
- Events related to scheduler

Whenever the system generates an Event, you can attach one of the following actions to the event:

- an Action/Macro (or sequence of) selected from a list of predefined actions
- a JavaScript function

The figure below shows an example of an Action activated by pressing a button.



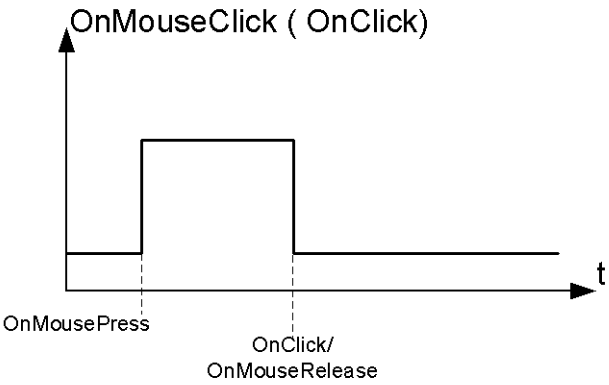
⊟ Button	
Value	0
Click Type	momentar
Autorepeat	Disabled
Hold Time (ms)	-1
Label	
Fill Color	■ [12
Show Frame	true
⊟ Events	
OnMouseClick Action	
OnMouseHold Action	
OnDataUpdate Action	
OnMousePress Action	+
OnMouseR	
⊞ Configure	
⊞ Text	

Example of an action activated by pressing a button.

By associating actions to events, the programmer configures user interactions with the program.

OnClick/OnMouseClick

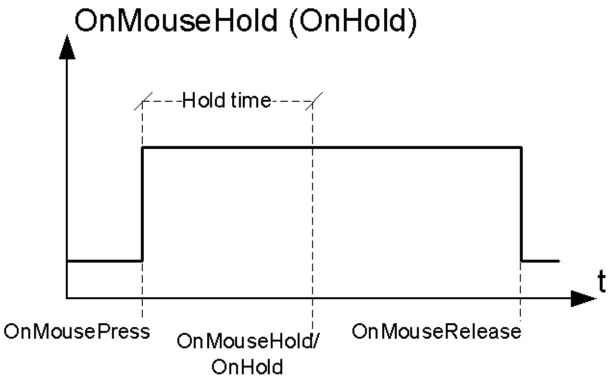
This Event occurs when the button/key is pressed and released quickly.



OnHold/OnMouseHold

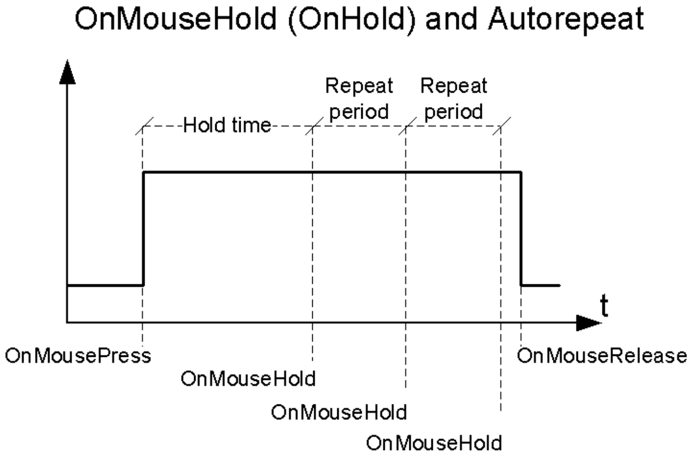
This Event occurs when the button/key is pressed and held pressed for a certain Hold time. Actions programmed for this Event will be executed only after the Hold time has expired.

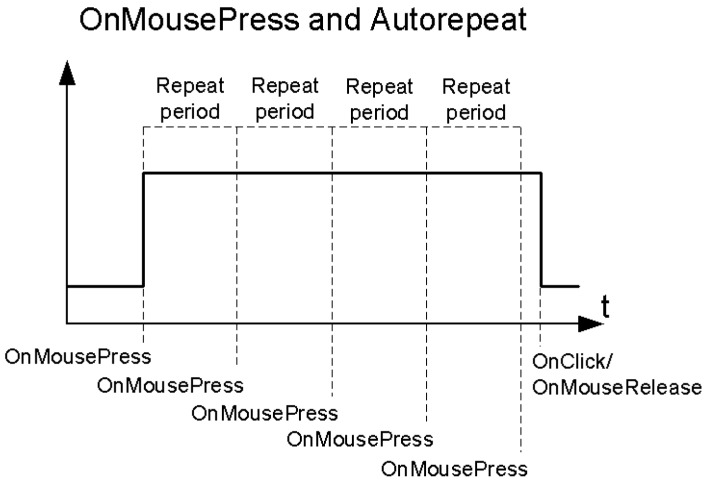
Hold time is configured in project properties but can be redefined for each button/key. When a value **-1** is specified as Hold time for a certain button, the project default value will be used.



Autorepeat

It is possible to enable autorepeat for Press event or for Hold event of a button/key.
Autorepeat Time is specified in Project properties but can be redefined for each button/key.





OnWheel

This Event occurs when a wheel (example: a USB mouse wheel or a handheld wheel) value change. A wheel usually is used to increment/decrement a value in data entry or attached to a tag.

OnActivate

This event triggers when a page is loaded and before widgets being initialized with the values read from Tag Manager.

OnDataUpdate

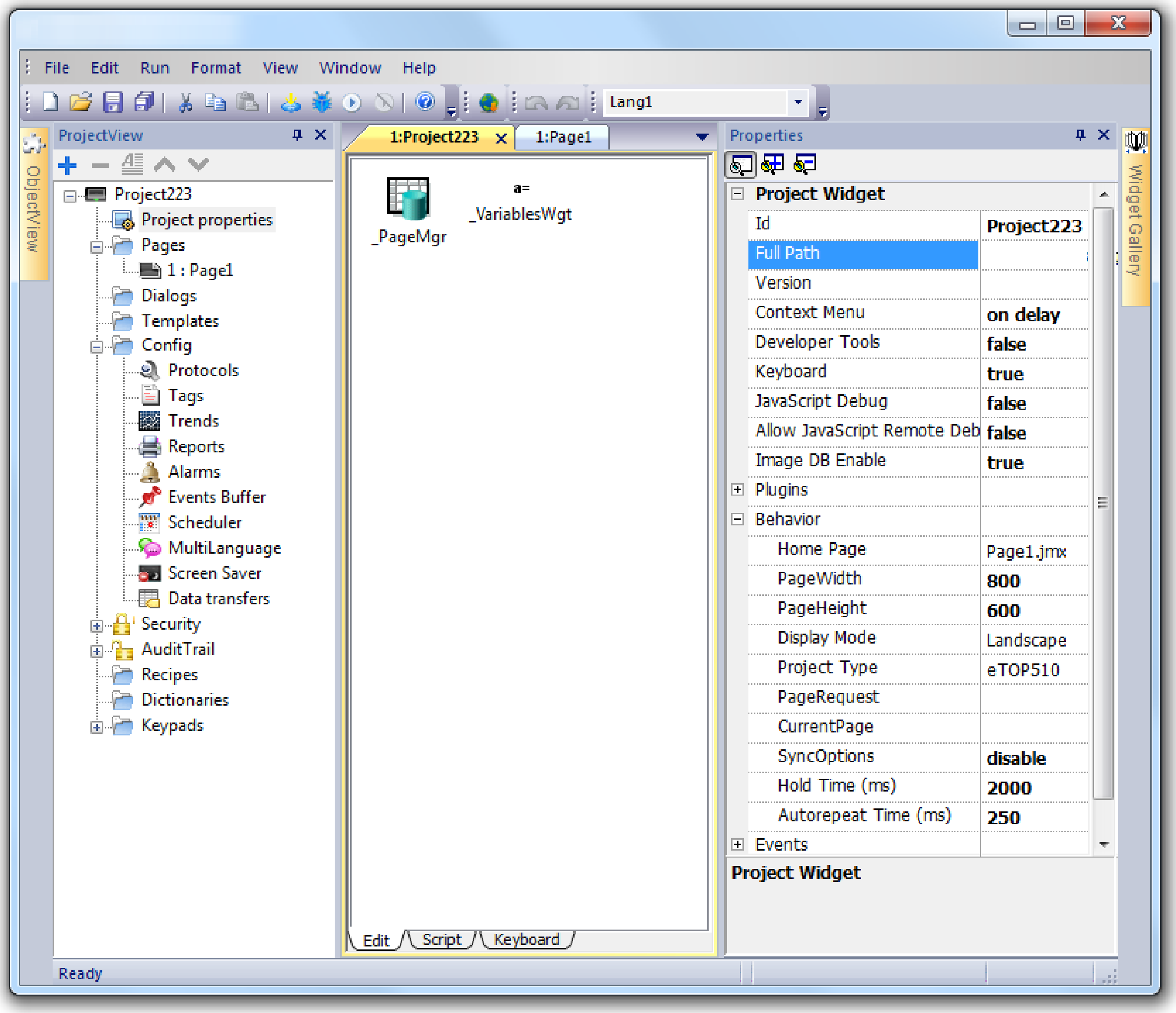
This event triggers when a data field attached to a widget changes. Upon page change, data is updated asynchronously at a time that depends on the time needed to read data from protocol. As a consequence, the OnDataUpdate event can be triggered or not, depending on whether data becomes available from protocol respectively after or before widgets being initialized for first time. In particular page change notifications are more likely to happen with slow protocols and Windows Remote Client.

Moreover, note that the value we read during OnActivate can be the same we get from a subsequent OnDataUpdate event, since OnDataUpdate notifications are sent asynchronously.

Project Properties / Project Widget

Project properties contain settings for the project. Project properties are available from **Project View**.

The **Properties** window on right side of the Panel Builder 600 contains the list of project level user-configurable data.



Project Widget	
Id	Project223
Full Path	
Version	
Context Menu	on delay
Developer Tools	false
Keyboard	true
JavaScript Debug	false
Allow JavaScript Remote Deb	false
Image DB Enable	true
Plugins	
Behavior	
Home Page	Page1.jmx
PageWidth	800
PageHeight	600
Display Mode	Landscape
Project Type	eTOP510
PageRequest	
CurrentPage	
SyncOptions	disable
Hold Time (ms)	2000
Autorepeat Time (ms)	250
Events	

Project Widget

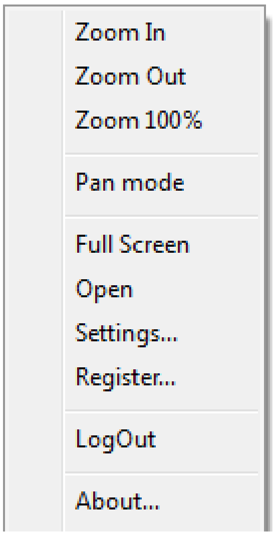
Version

The version field is available for users to report the project version.

Context Menu

The default method for users to access runtime settings is press and hold for a few seconds on an empty area of the runtime screen. Using this property you can choose howcontext menu should appear:

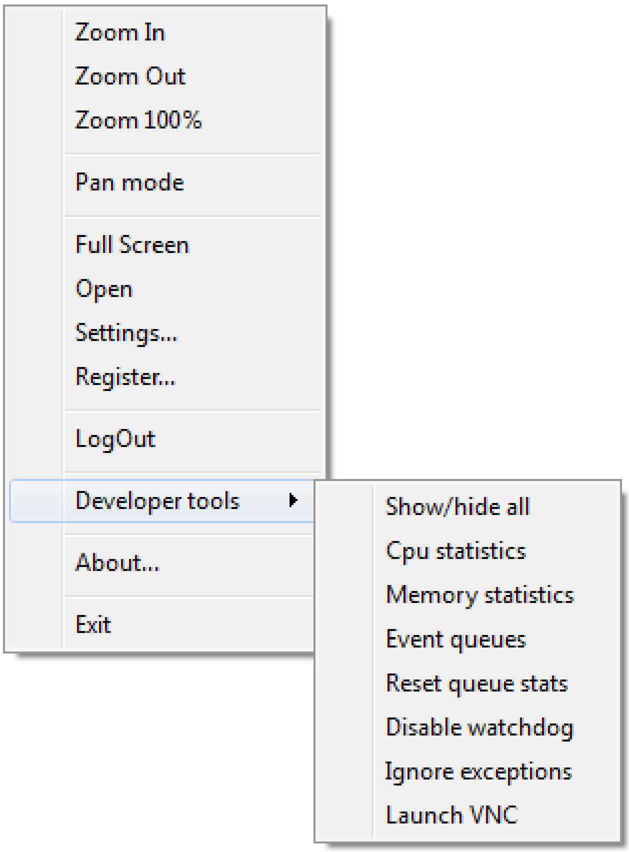
Context menu	Explanation
on delay (default)	press and hold
on macro command	via macro/action controlled by HMI application



Developer Tools

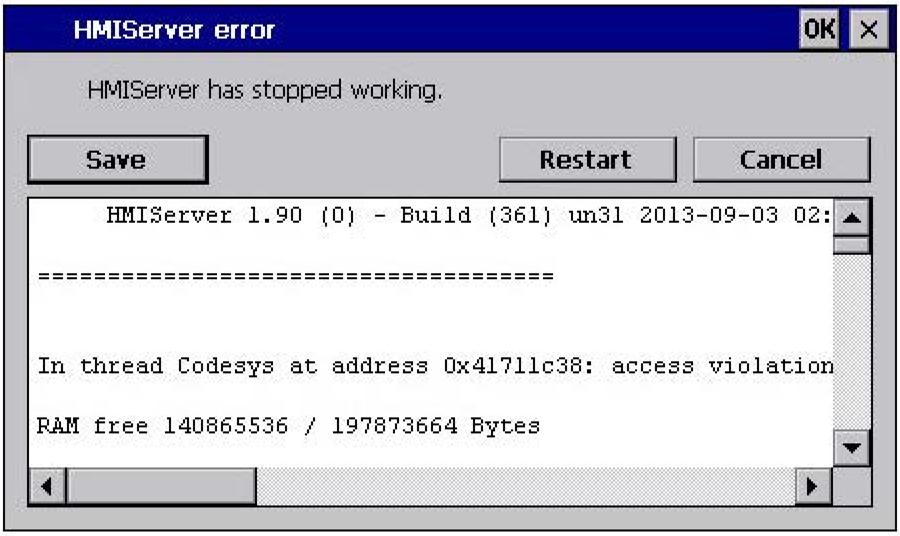
Developer tools are a collection of utilities useful for debugging problems at runtime. Developer tools are available in context menu at runtime. Not all items are available for all products and platforms.

To enable developer tools, set to true **Project properties ->Developer tools.**



One of the most important items in the developer tools is related to the **watchdog**. This item allows the developer to disable it to avoid a system restart in case of a runtime crash, to have time to save the **crash report** or check system status information (example: memory available, CPU load, events queue size etc.).

Crash report dialog appears automatically in case of a system freeze or crash and allows users to save a log file of crash. The crash report may contain information important for technical support.



HMIServer has stopped working.

```
    HMIServer 1.90 (0) - Build (361) un31 2013-09-03 02:

====================================

In thread Codesys at address 0x41711c38: access violation

RAM free 140865536 / 197873664 Bytes
```

Keyboard

Enable the use of keyboard Macros at runtime when using external keyboards.

JavaScript Debug

Enable the JavaScript debugger at runtime for current project.

NOTE For UN20 target (WCE MIPS hmi panels like CP650 … CP675), local debugger has been disabled. However, remote debugger is available to debug JS from a PC connected to HMI panel via Ethernet.

NOTE Remote debugger not supported in Windows Client and ActiveX.

Allow JS Remote Debugger

Enable the JavaScript remote debugger for current project.

NOTE For UN20 target (WCE MIPS hmi panels like CP650 … CP675), local debugger has been disabled. However, remote debugger is available to debug JS from a PC connected to HMI panel via Ethernet.

NOTE Remote debugger not supported in Windows Client and ActiveX.

Image DB enable

Enabled by default, this property activates an engine used by the runtime to optimize project performance. Available in the Project Properties, should be disabled just by tech support for debugging in case of a problem. Disabling it can create performance problems at runtime.

FreeType Font Rendering

The “FreeType Font Rendering” property is used to switch between old font engine used by Panel Builder 600 & Runtime up to v1.80 (native OS-based font engine) and the font rendering based on FreeType.

All projects created with Panel Builder 600 v1.90 or newer use the FreeType font engine as default while all projects created with older versions of Panel Builder 600 continue to use old font engine after the conversion to avoid potential backward compatibility issues in font rendering.

Moving to the FreeType Font Rendering is recommended to all users; to enable it set true in “FreeType Font Rendering” in Project Properties, save and verify that all texts are shown correctly in all HMI project pages.

Example of rendering issues that could appear when switching between old and new font engine are:

- text require few more/less pixels for rendering and this could change text layout
- size to fit could result in change in size of widgets.
- better rendering using antialiasing (feature not available in v1.80). Antialiasing can be disabled in v1.90 for texts (it is a property of text widgets).

Software Plug-in Modules

The software plug-in concept allows users to choose if certain software modules must be downloaded to runtime together with the project. Examples of software plug-ins are:

- PDF Reader
- VNC Server
- ActiveX

Not all software plug-in modules are compatible with all targets. New software plug-in modules will be added in the future to extend optional features of the product.

Once enabled, a software plug-in is considered as part of runtime. You can use Panel Builder 600 to install it in the target using one of the following procedures:

- Installing runtime
- Updating runtime
- Update Package

System is not able to detect automatically if a software plug-in is required by the HMI application, so it is up to the user selecting the software plug-in manually from project properties when required.

Software Plug-in support has been designed for embedded HMI panels where storage is limited and reducing software footprint is critical. This option is not supported in Win32 Runtime.

Behavior -> Home Page

Define homepage of project. The homepage is the first page loaded at runtime (after log-in page if security is enabled in project).

When **Security** is enabled, it is possible to specify a different homepage for each groups of users, in this case this property is ignored. Refer to User **Management** for more details.

Behavior -> Page Width / Page Height

Define default size in pixel of an HMI page. Default is target type dependent (depend on HMI panel screen resolution).

Behavior -> Display Mode

Define HMI panel orientation, Landscape or Portrait.

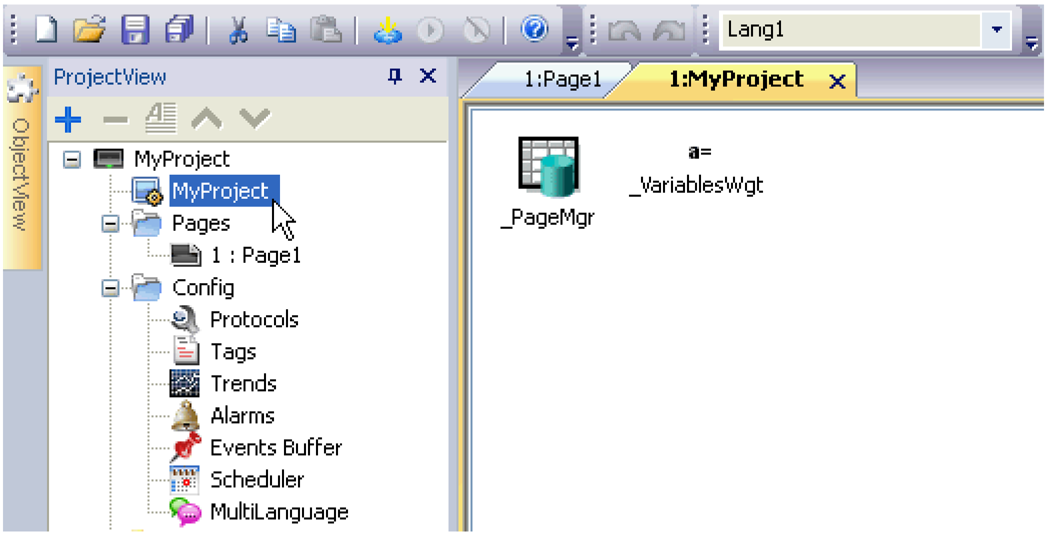
Behavior -> Project Type

Define target type / HMI panel model. Based on model, many features and properties of project are automatically adjusted to fit it in the right way.

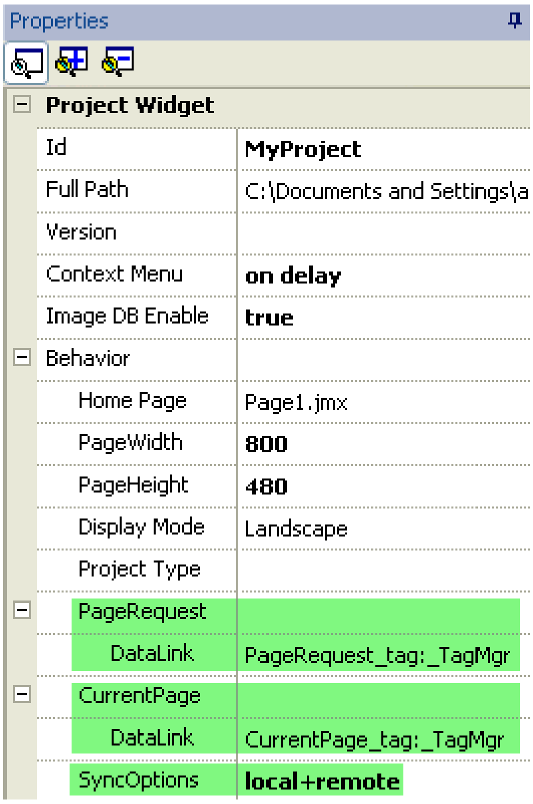
Behavior -> PageRequest, CurrentPage and SyncOptions

The HMI projects contain properties that let you know which page is currently displayed on the HMI and to force the HMI to switch to a specific page. These properties can be used to synchronize pages showed on the HMI and Windows Client or to control an HMI with a PLC.

Double click on project name present into ProjectView pane to open the project properties page:



Expand the properties view of the Properties pane, by clicking on "Show Advanced Properties" button:



Project Widget	
Id	MyProject
Full Path	C:\Documents and Settings\a
Version	
Context Menu	on delay
Image DB Enable	true
Behavior	
Home Page	Page1.jmx
PageWidth	800
PageHeight	480
Display Mode	Landscape
Project Type	
PageRequest	
DataLink	PageRequest_tag:_TagMgr
CurrentPage	
DataLink	CurrentPage_tag:_TagMgr
SyncOptions	local+remote

Following properties, highlighted in green in the picture above, can be configured:

PageRequest

This property determines the page to be shown on the HMI and on Windows Client. Attached Tag must contain an integer value, clearly within the range of the available project pages. The attached Tag must be available at least as a Read resource.

CurrentPage

This property represents the page number actually displayed on the HMI or on Windows Client or on both. Attached Tag must be available at least as a Write resource and must have data type that allows containing an integer value. SyncOptions value can be set as one of following options:

- Local: if you want that CurrentPage represents the number of page actually displayed on HMI,
- Remote: if you want that CurrentPage represents the number of page actually displayed on Windows Client.

SyncOptions

This property determines the synchronization of the project pages with the value contained in the CurrentPage property.

- Disable: CurrentPage value is ignored.
- Local: CurrentPage value corresponds to the page displayed on HMI.
- Remote: CurrentPage value corresponds to the page displayed on Windows Client.
- Local + Remote: CurrentPage is changed according to page displayed on HMI and on Windows Client, if different pages are displayed; CurrentPage refers to the last page loaded.

Example 1

Force page change from PLC to HMI and Windows Client

- PageRequest: attached to Tag "A"
- CurrentPage: empty
- SyncOptions: Disabled

Changing value of "A", HMI and Windows Client will show page requested

Example 2

Force page change from PLC to HMI and Windows Client. Read current page loaded on HMI

- PageRequest: attached to a Tag "A"
- CurrentPage: attached to a Tag "B" as Read/Write
- SyncOptions: Local

Changing value of "A", HMI and Windows Client will show page requested. On "B" will be written page currently shown by HMI.

Example 3

Force page change from PLC to HMI and Windows Client. Read current page loaded on Windows Client

- PageRequest: attached to a Tag "A"
- CurrentPage: attached to a Tag "B" as Read/Write
- SyncOptions: Remote

Changing value of "A", HMI and Windows Client will show page requested. On "B" will be written page currently shown by Windows Client.

Example 4

Force page change from PLC to HMI and Windows Client. Windows Clientpage Synchronization with HMI (not vice versa)

- PageRequest: attached to a Tag "A" as Read/Write
- CurrentPage: attached to the same Tag "A" as per PageRequest
- SyncOptions: Local

Changing value of "A", HMI and Windows Client will show page requested. Changing page on HMI same page will be forced on Windows Client.

Example n.5

Force page change from PLC to HMI and HMI Client HMI page Synchronization with Windows Client (not vice versa)

- PageRequest: attached to a Tag "A" as Read/Write
- CurrentPage: attached to the same Tag "A" as per PageRequest
- SyncOptions: Remote

Changing value of "A", HMI and Windows Client will show page requested. Changing page on Windows Client same page will be forced on HMI.

Example n.6

Synchronize displayed page between HMI and on Windows Client

- PageRequest: attached to a Tag "A" as Read/Write
- CurrentPage: attached to the same Tag "A" as per PageRequest
- SyncOptions: Local+Remote

Changing page on HMI, same page will be shown on Windows Client and vice versa.

Behavior -> Hold Time and Autorepeat Time

Define default values for hold time and autorepeat time for buttons and external keyboards. However, for each button/key, they can be redefined in related widget instance.

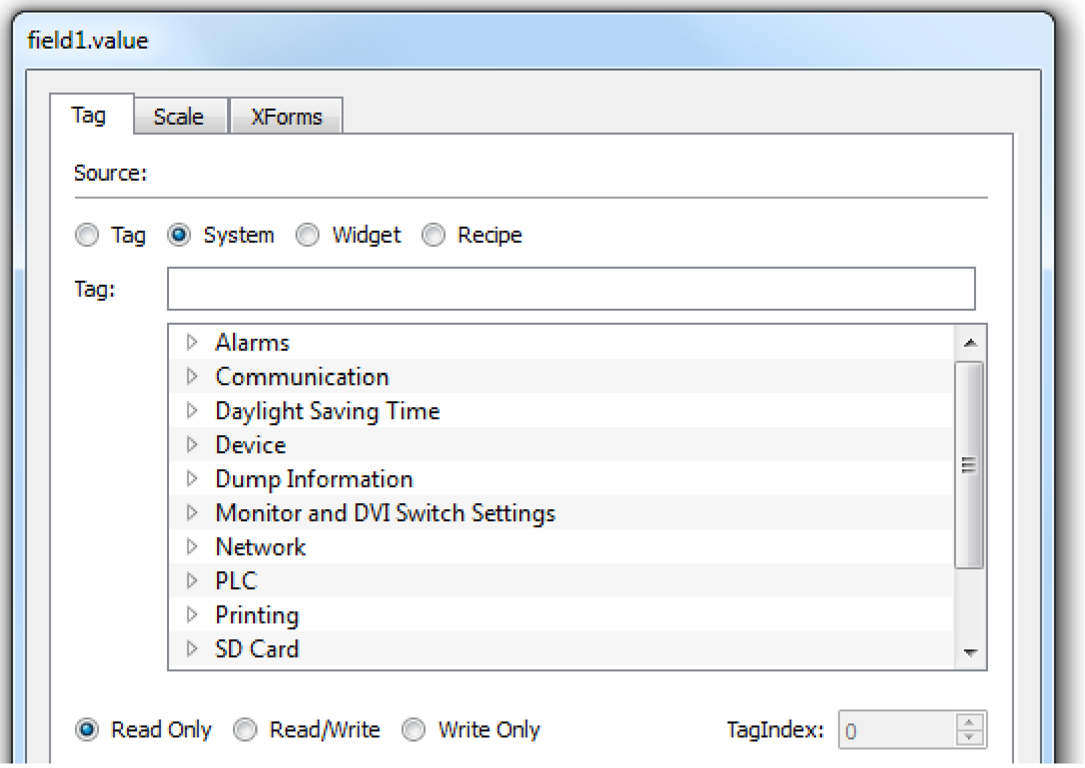
Events -> OnWheel

A wheel is used by special products like handhelds or attaching an USB mouse with a wheel input device. Usually a wheel is used to increase/descries of a fixed value a Tag without the need to use an external keyboard device.

From **Project Widget** is possible to attach to a change of wheel status an action like BiStepTag to increase/decrease a tag value.

System Variables

System variables are special system tags containing generic information about the runtime. System variables are available in the *Attach to* dialog from the *Source* selection.



field1.value

Tag | Scale | XForms

Source:

○ Tag ● System ○ Widget ○ Recipe

Tag:

- Alarms
- Communication
- Daylight Saving Time
- Device
- Dump Information
- Monitor and DVI Switch Settings
- Network
- PLC
- Printing
- SD Card

● Read Only ○ Read/Write ○ Write Only

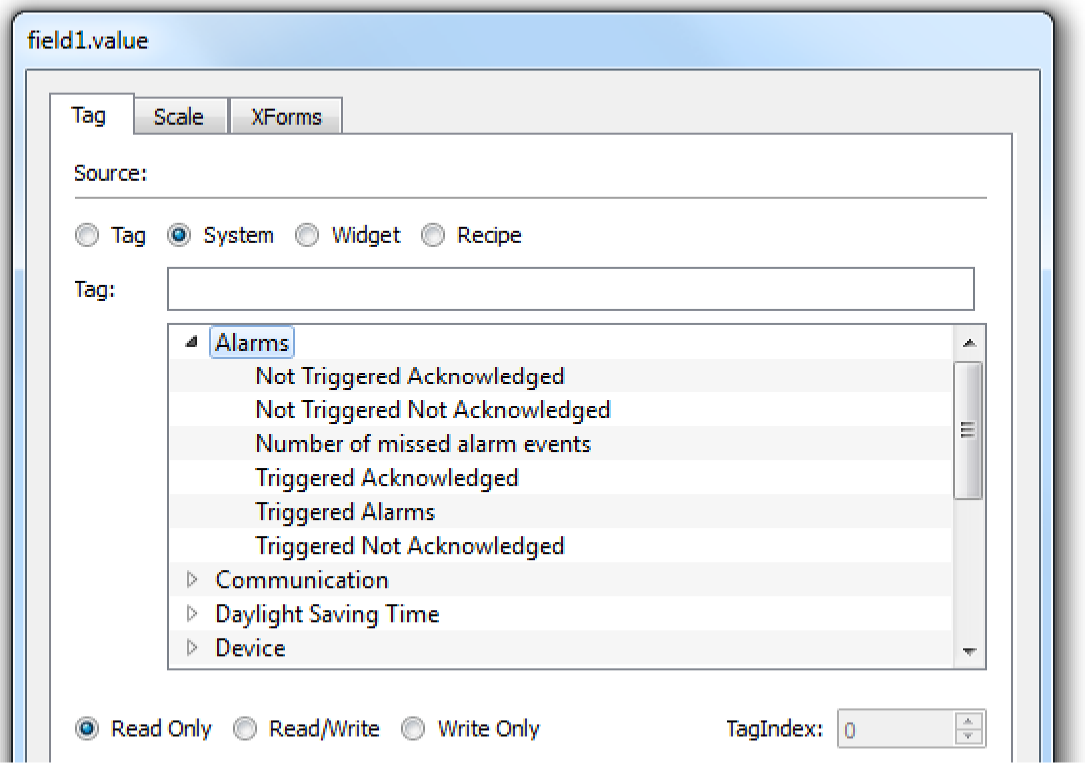
TagIndex: 0

System variables are divided in the following categories.

Starting from v1.90, system variables are available also as a standard protocol in protocol editor. This protocol can be used for data transfer betweens system variables and tags from devices or when is necessary to specify a custom refresh rate for a system variable.

Alarms

Variables return information on the actual number of alarms according to the status.



field1.value

Tag | Scale | XForms

Source:

○ Tag ● System ○ Widget ○ Recipe

Tag:

- Alarms
  - Not Triggered Acknowledged
  - Not Triggered Not Acknowledged
  - Number of missed alarm events
  - Triggered Acknowledged
  - Triggered Alarms
  - Triggered Not Acknowledged
- Communication
- Daylight Saving Time
- Device

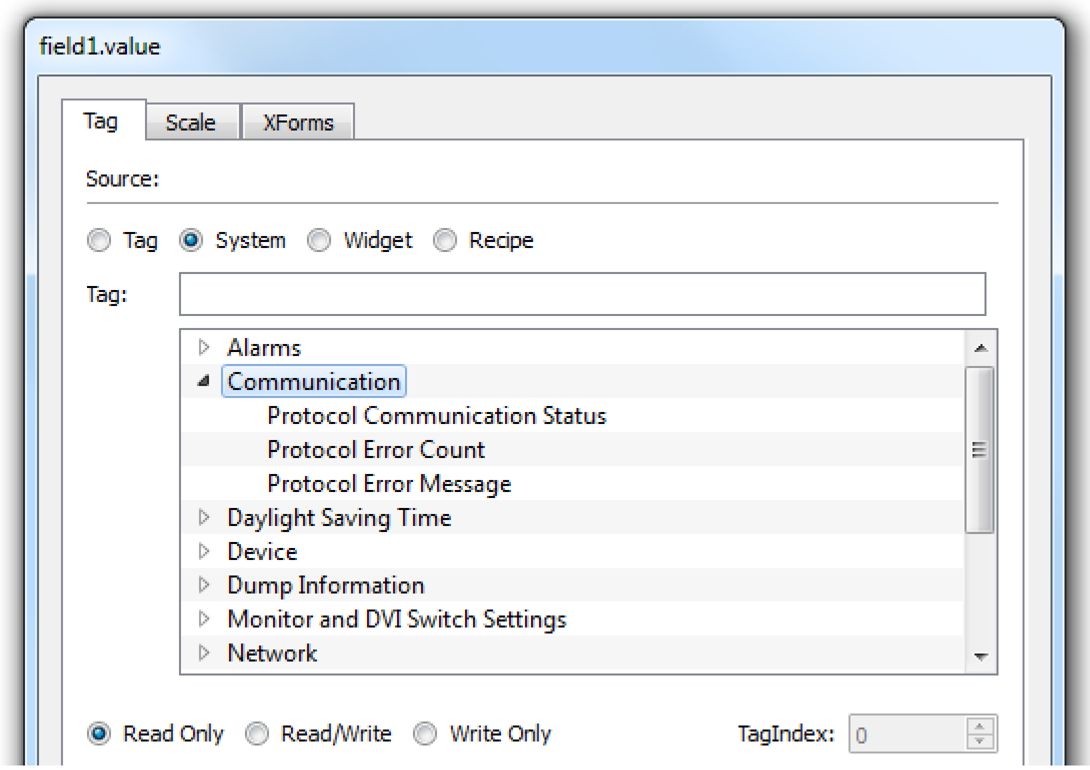
● Read Only ○ Read/Write ○ Write Only TagIndex: 0

System variable	Explanation
Not Triggered Acknowledged	Total number of alarms “Not Triggered Acknowledged”
Not Triggered Not Acknowledged	Total number of alarms “Not Triggered Not Acknowledged”
Triggered Acknowledged	Total number of alarms “Triggered Acknowledged”
Triggered Not Acknowledged	Total number of alarms “Triggered Not Acknowledged”
Triggered Alarms	Total number of alarms “Triggered”
Number of missed alarm events	Total number of missed alarm events

All these system variables are int type (32 bit), read only.

Communication

Variables return information on the status of the communication between the HMI device and the controllers configured in the protocol editor.



Source: ○ Tag ● System ○ Widget ○ Recipe

Tag:

● Read Only ○ Read/Write ○ Write Only TagIndex: 0

Protocol Communication Status

The variable is read only int (32 bit), can have three values:

- 0: No protocol running; it may occur if the protocol driver has not been properly downloaded to the target system.
- 1: Protocol has been properly loaded and started; no communication errors
- 2: At least one communication protocol is reporting an error

Protocol Error Message

This variable returns an ASCII string containing a description of the current communication error. The communication protocol acronym is reported between square brackets to recognize the source of the error in case of multiple protocols configurations. The variable is a read only string. If no errors are present, the string will be blank.

Protocol Error Count

This variable returns the number of communication errors that occurred since the last time it was reset.

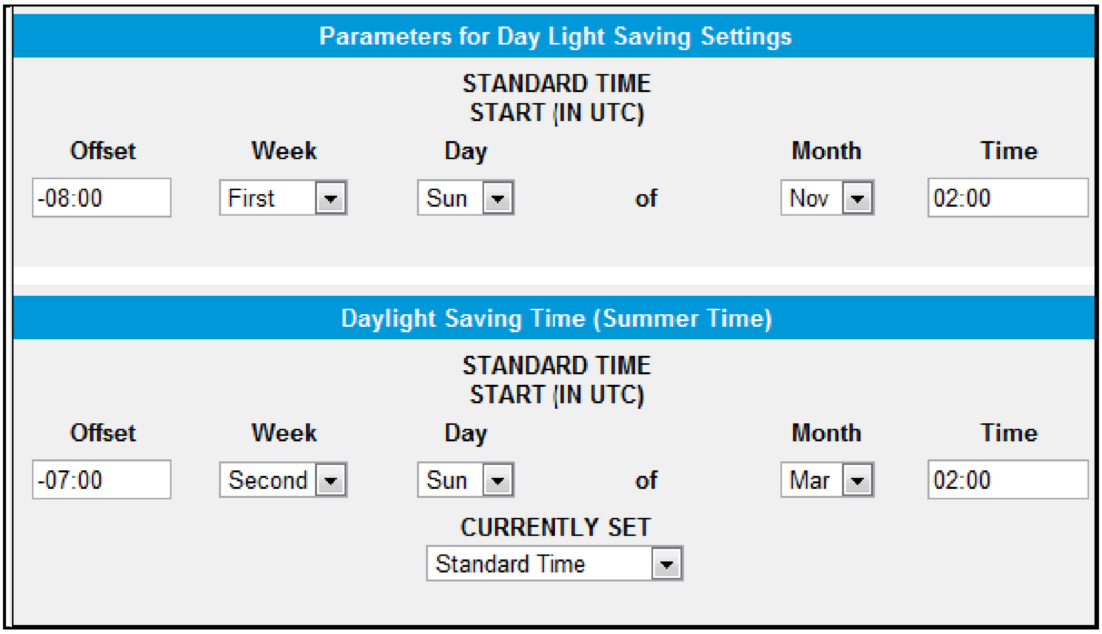
The variable is a read only integer. The reset of this variable is only possible using the dedicated Action “Reset Protocol Error Count“.

Daylight Saving Time

Variables return information on the system clock and allow adjusting it from the application. They contain information on the “local” time. All the variables are read/write; this means that you cannot change them to update the system RTC.

All the variables are bytes (8 bit) except for the DLS and standard offset that are shorts (16 bit).

Standard time is the “solar time” and other is daylight saving time.



Parameters for Day Light Saving Settings

STANDARD TIME START (IN UTC)

Offset	Week	Day		Month	Time
-08:00	First	Sun	of	Nov	02:00

Daylight Saving Time (Summer Time)

STANDARD TIME START (IN UTC)

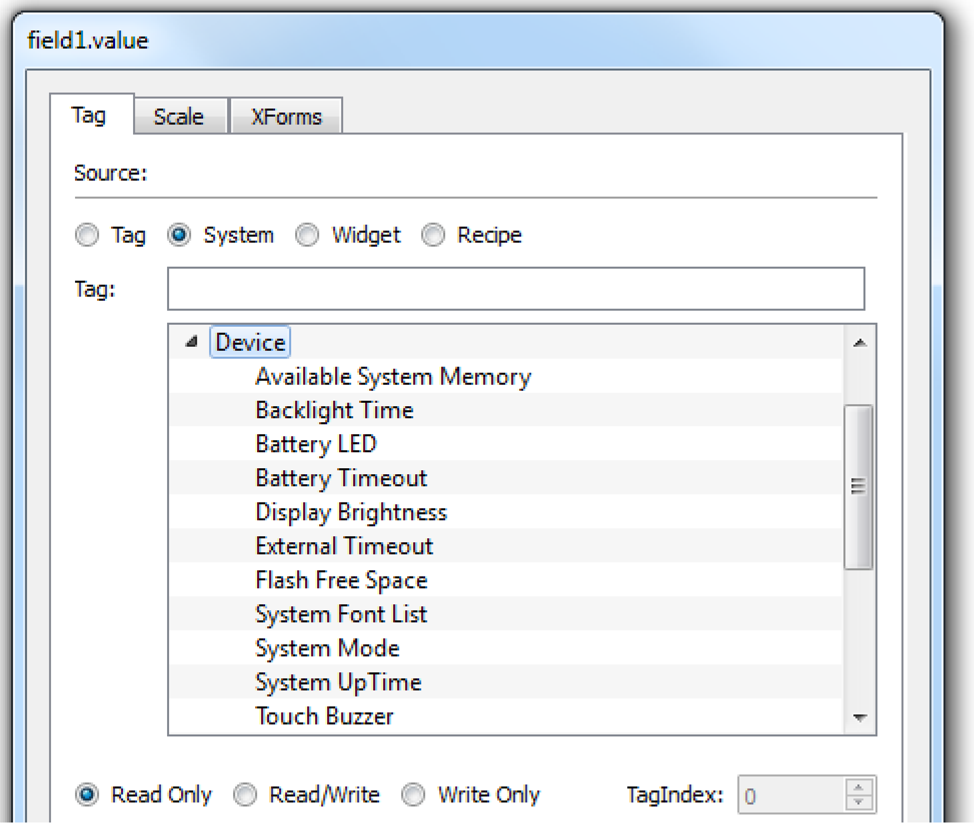
Offset	Week	Day		Month	Time
-07:00	Second	Sun	of	Mar	02:00

CURRENTLY SET: Standard Time

Variable	Explanation
Standard offset	represents the offset in minutes when standard time is set, with respect to GMT. (with respect to the picture it is -8*60 = - 480 minutes)
Standard week	the week in which the Standard time starts (w.r.t. the picture it is First = 1)
Standard Month	the month in which the standard time starts (range of the variable is [0 -11] so w.r.t. the picture it is November = 10)
Standard Day	day of week in which the standard time starts (w.r.t. the picture it is Sunday = 0)
Standard hour	hour in which the standard time starts (w.r.t. the picture in Time field it is 02 = 2)
Standard minute	minute in which the standard time starts (w.r.t. the picture in Time field it is 00 = 0)
Dst offset	represents the offset in minutes when DLS time is set, with respect to GMT. (w.r.t. the picture it is -7*60 = - 420 minutes)
Dst week	week in which the DLS time starts (w.r.t. the picture it is Second = 2).
Dst Month	month in which the DLS time starts (range of the variable is [0 -11] so w.r.t. the picture it is March = 2)
Dst Day	day of week in which the DLS time starts (w.r.t. the picture it is Sunday = 0)
Dst hour	hour in which the DLS time starts (w.r.t. the picture in Time field it is 02 = 2)
Dst minute	minute in which the DLS time starts (w.r.t. the picture in Time field it is 00 = 0)

Device

Variables can be used to adjust specific device settings and obtain operational information.



field1.value

Source:

○ Tag ● System ○ Widget ○ Recipe

Tag:

- Device
  - Available System Memory
  - Backlight Time
  - Battery LED
  - Battery Timeout
  - Display Brightness
  - External Timeout
  - Flash Free Space
  - System Font List
  - System Mode
  - System UpTime
  - Touch Buzzer

● Read Only ○ Read/Write ○ Write Only

TagIndex: 0

Available System Memory

Returns the free available RAM memory in bytes; it is a 64 bit data; it is a read only variable.

Backlight Time:

Returns the activation time in hours of the display backlight lamp since production of the unit; it is a read only variable.

Battery LED

Enables/disables the use of the front LED indicator to report the low battery status. It can have values 0 (disabled) or 1 (enabled).

Battery Timeout

Reserved.

Display Brightness

This variable is an integer of R/W type. Its range goes from 0 to 255. It can be used to check brightness level and adjust it from the application. Typical use is connected to a slider widget.

When set to a low level (0..3), backlight assume a low but visible value for around 8 seconds (to let user change it otherwise nothing is visible in display) and after that display appear as switch-off. However, also with value 0 backlight is still on and counter of backlight life time increase.

External Timeout

This variable allows setting the not operational time after which the display backlight is automatically turned off. The backlight is then automatically turned on when user presses on the touchscreen.
The variable is an int of R/W type.
-1 = switch off backlight and disable touch (switch display off). Backlight counter is stopped.

0 = switch backlight on (so switch display on)

1..n = set a timeout for switch off backlight, so work like a screensaver timer

Flash Free Space

Returns the free space left in the device internal flash.

System Font List

List of system fonts. The variable is a read only string.

System Mode

Returns a value informing the operation status of the runtime. Possible values are:

- 1: Booting
- 2: Configuration mode
- 3: Operating mode
- 4: Restart
- 5: Shutdown

System UpTime:

Returns the total time in hours in which the system has been powered since production of the unit. It is a read only variable.

Touch Buzzer

Allows you to enable/disable the touch audible feedback. It can have values 0 (disabled) or 1 (enabled).

NOTE

Starting from BSP 1.66.6 ARM / 2.73.1 MIPS, buzzer control has been extended as below.

Buzzer Control

- 0: Buzzer off
- 1: Buzzer on
- 2: Buzzer blink (on and off times programmed by System Variables Buzzer On and Off)

Buzzer Off Time

duration in milliseconds of off time when blink has been selected:

minimum value: 100
maximum value: 5000
default value: 1000

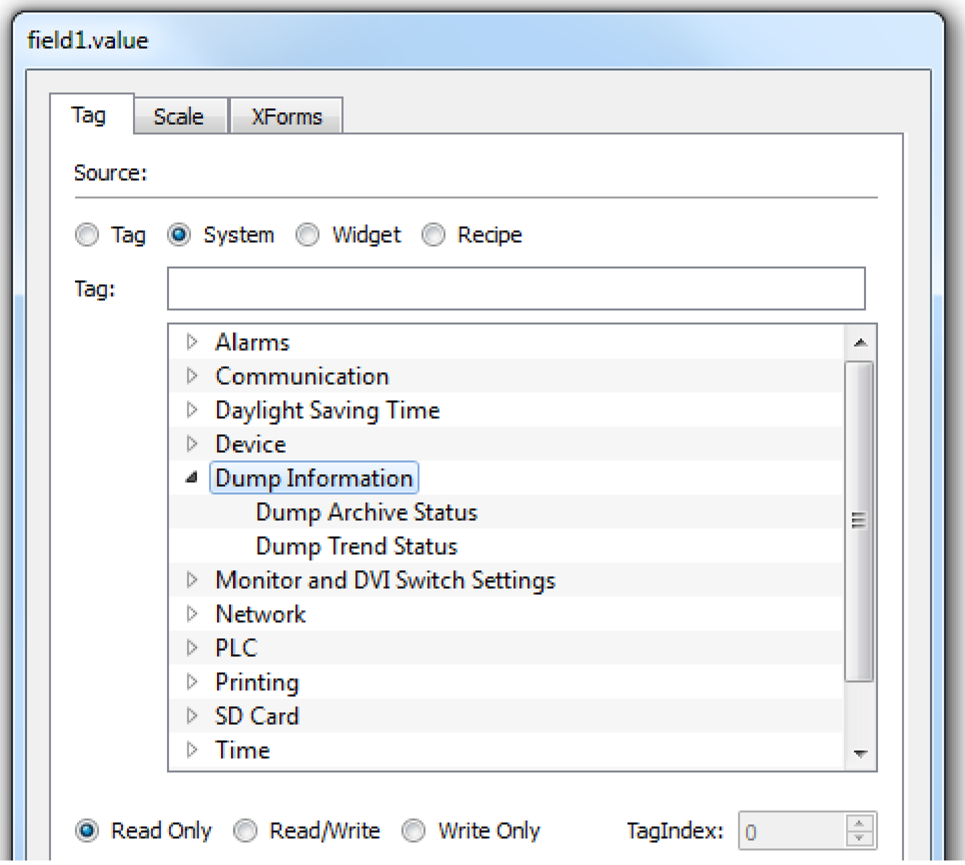
Buzzer On Time

duration in milliseconds of off time when blink has been selected:

minimum value: 100
maximum value: 5000
default value: 1000

Dump Information

Variables return information about the status of the copy process to external drives (USB) for trend and archive buffers.



Source:

○ Tag ● System ○ Widget ○ Recipe

Tag:

● Read Only ○ Read/Write ○ Write Only TagIndex: 0

Dump Trend Status

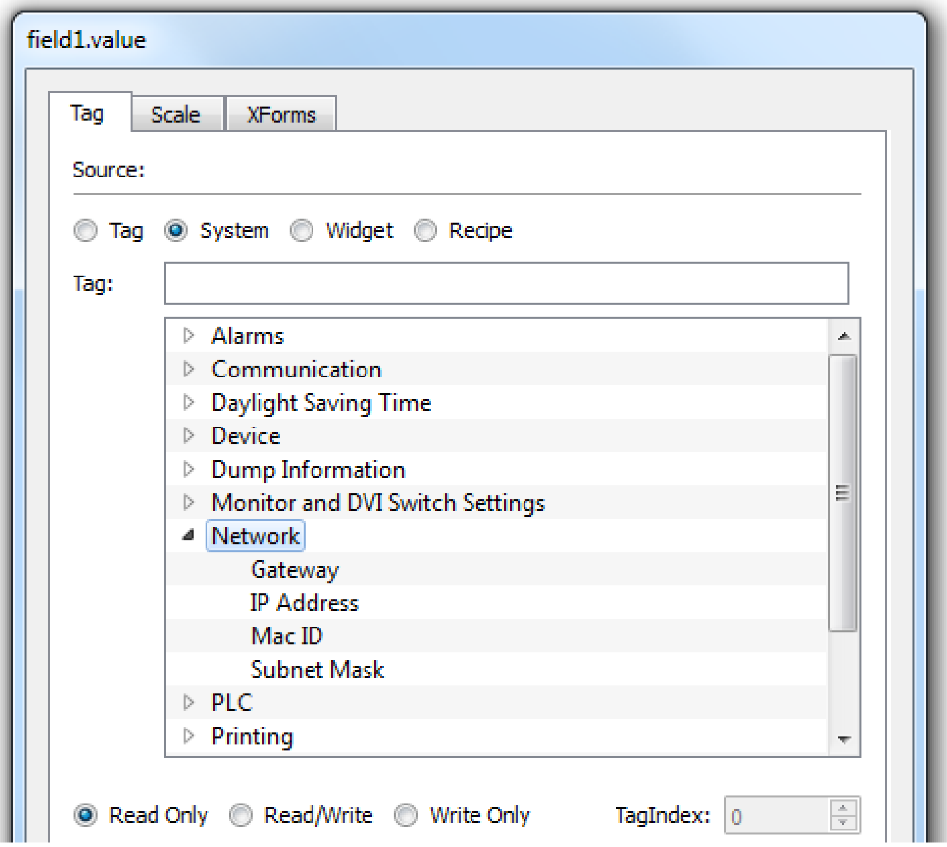
This variable returns value 1 during the copy process of the trend buffers. If the copy duration time is less than one second, the system variable does not change its value.

Dump Archive Status

This variable returns value 1 during the copy process of the archive buffers. If the copy duration time is less than one second, the system variable does not change its value.

Network

Variables allow to show and set network device parameters. Except for the MAC ID, they are all of R/W type.



field1.value

Tag | Scale | XForms

Source:

○ Tag ● System ○ Widget ○ Recipe

Tag:

- Alarms
- Communication
- Daylight Saving Time
- Device
- Dump Information
- Monitor and DVI Switch Settings
- Network
  - Gateway
  - IP Address
  - Mac ID
  - Subnet Mask
- PLC
- Printing

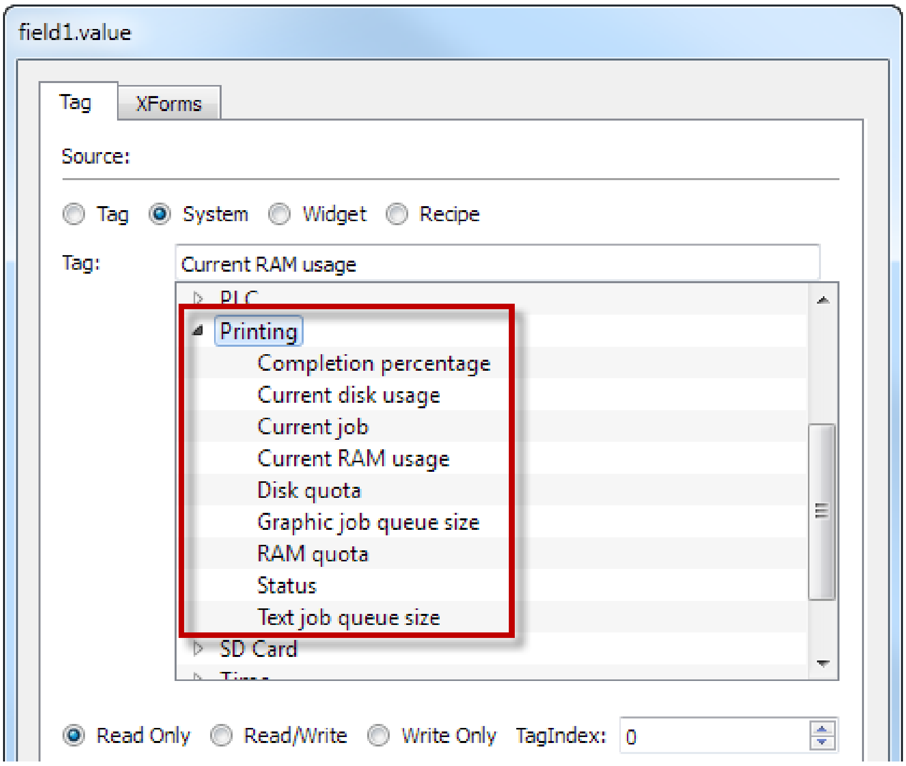
● Read Only ○ Read/Write ○ Write Only

TagIndex: 0

Variable	Explanation
Gateway	Gateway address of the main Ethernet interface of device
IP Address	IP address of the main Ethernet interface of device
Mac ID	MAC ID of the main Ethernet interface of device
Subnet Mask	Subnet Mask of the main Ethernet interface of device

Printing

Variables return information about the Printing system. All the variables are read only.



field1.value

Tag | XForms

Source:

○ Tag ● System ○ Widget ○ Recipe

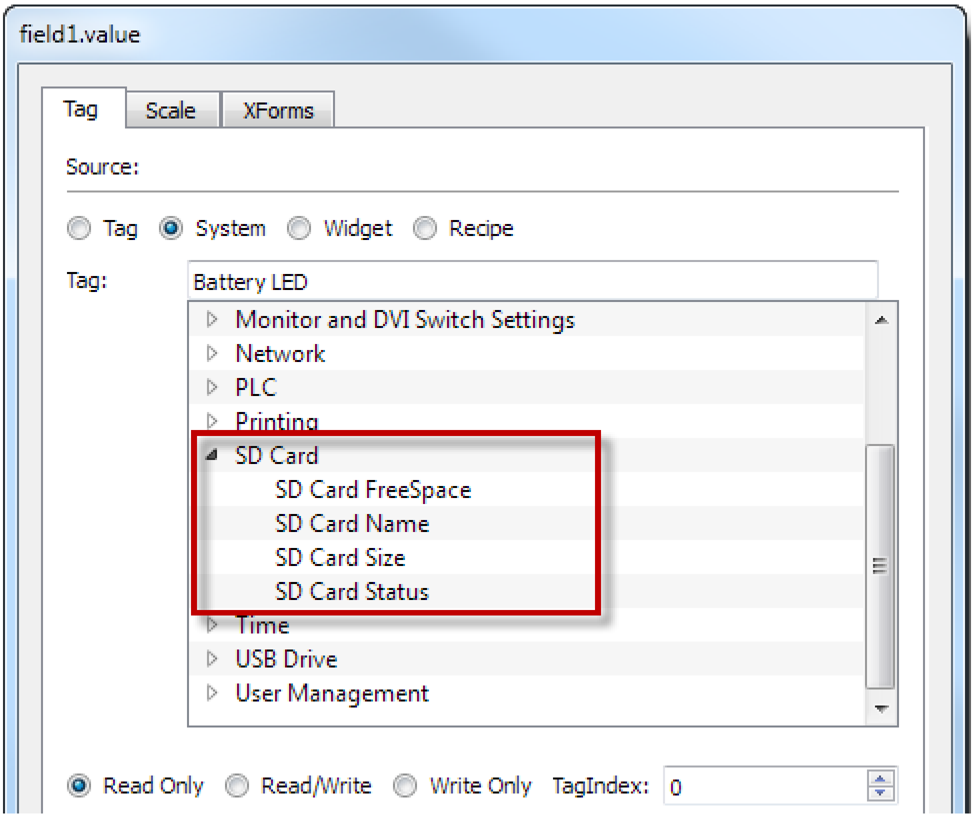
Tag: Current RAM usage

● Read Only ○ Read/Write ○ Write Only TagIndex: 0

Variable	Explanation
Completion percentage	The percentage of competition of the current print job. It ranges from 0 to 100.
Current disk usage	The size (in bytes) of folder where PDF reports are stored (it is reportspool if option *Spool media type* is *Flash*).
Current job	The name of the report the job is processing.
Current RAM usage	The size (in bytes) of the RAM used to process the current job.
Disk quota	The maximum size (in bytes) of the folder where PDF reports are stored.
Graphic job queue size	The number of the available graphic jobs in the printing queue.
RAM quota	The maximum size (in bytes) of the RAM used to generate reports.
Status	A string representing the status of the printing system. The possible values are **idle**, **error**, **paused** and **printing**.
Text job queue size	The number of the available text jobs in the printing queue.

SD Card

Variables return information on the external SD Card plugged into the panel. They are 64 bit variables, except the drive name which is a string. All the variables are read only.



field1.value

Tag | Scale | XForms

Source:

○ Tag ● System ○ Widget ○ Recipe

Tag: Battery LED

- Monitor and DVI Switch Settings
- Network
- PLC
- Printing
- SD Card
  - SD Card FreeSpace
  - SD Card Name
  - SD Card Size
  - SD Card Status
- Time
- USB Drive
- User Management

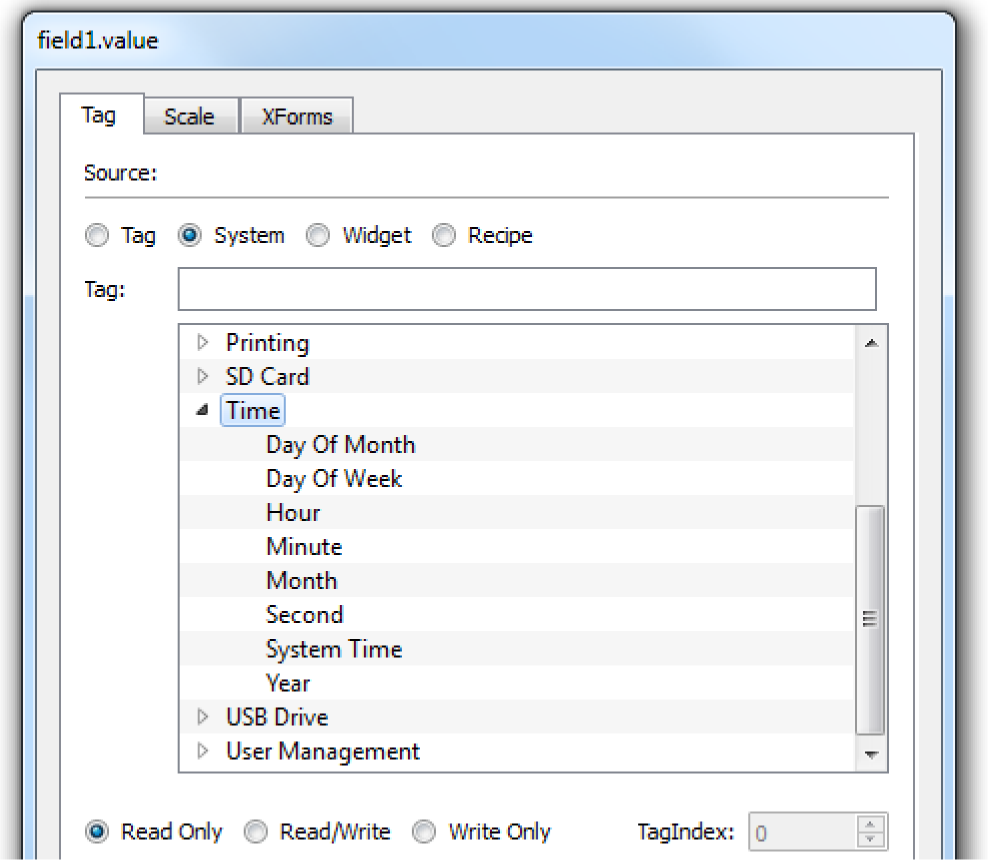
● Read Only ○ Read/Write ○ Write Only TagIndex: 0

Variable	Explanation
SD Card FreeSpace	Size in bytes of the available space
SD Card Name	Name of the SD card
SD Card Size	Size in bytes of the card plugged in the slot
SD Card Status	Status of the SD card

Time

Variables return information on the system time expressed in UTC format.

They are all Int (32 bits) of read/write type, except for the system time which is a 64 bit variable, still of read/write type. This is actually the UTC time which is also available as date/time from the other variables.



field1.value

Source: ○ Tag ● System ○ Widget ○ Recipe

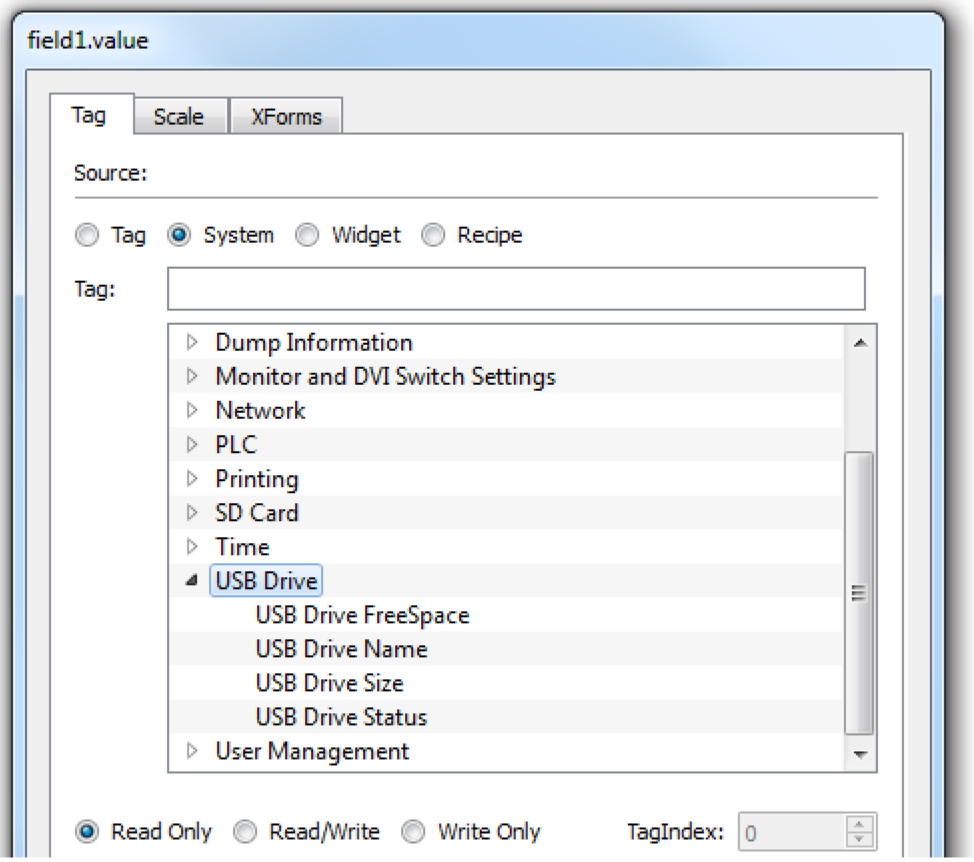
Tag:

● Read Only ○ Read/Write ○ Write Only TagIndex: 0

Variable	Explanation
Day Of Month	Day of the month (1..31)
Day of Week	Day of the week (0=Sunday, .. , 6=Saturday)
Hour	Hour (0..23)
Minute	Minute (0..59)
Month	Current month (1..12)
Second	Second (0..59)
System Time	System time
Year	Current Year

USB Drive

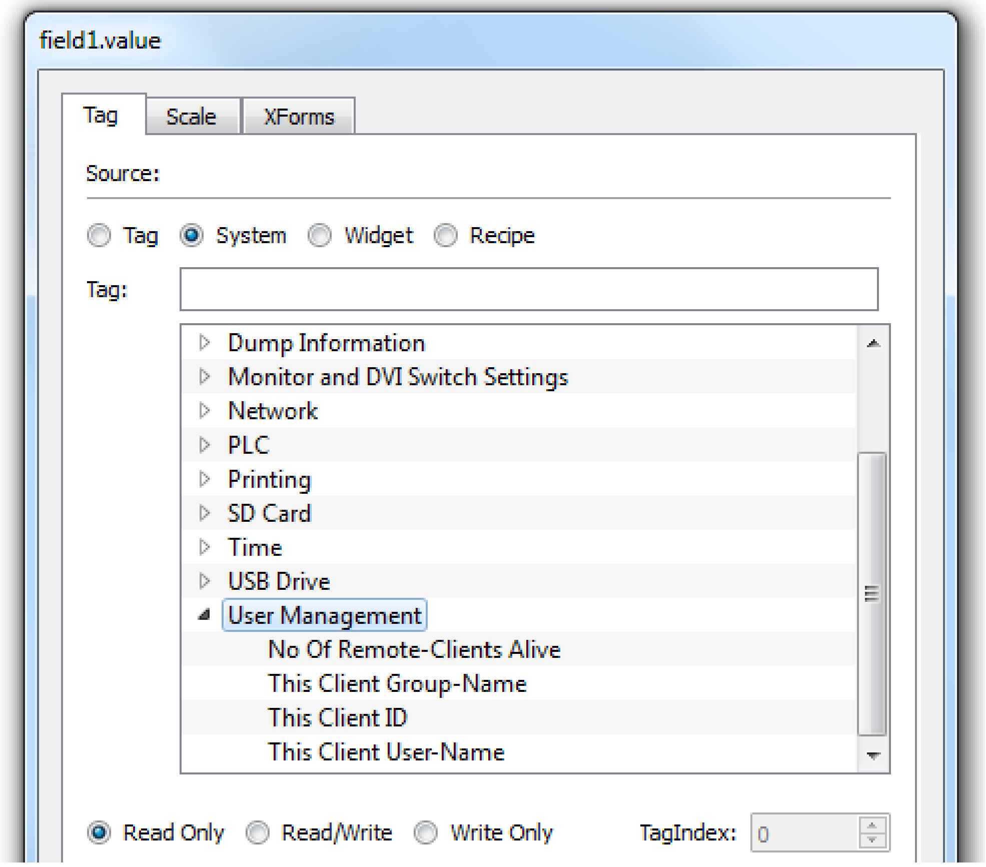
Variables return information on the external USB drive connected to the panel, They are 64 bit variables, except the drive name which is a string. All the variables are read only.



Variable	Explanation
USB Drive free space	Size in bytes of the available space
USB Drive Name	Name of the USB device
USB Drive Size	Size in bytes of the device plugged in the USB port
USB Drive Status	Status of the USB device

User Management

Variables return information on users and groups.



field1.value

Tag | Scale | XForms

Source:

○ Tag ● System ○ Widget ○ Recipe

Tag:

- Dump Information
- Monitor and DVI Switch Settings
- Network
- PLC
- Printing
- SD Card
- Time
- USB Drive
- User Management
  - No Of Remote-Clients Alive
  - This Client Group-Name
  - This Client ID
  - This Client User-Name

● Read Only ○ Read/Write ○ Write Only TagIndex: 0

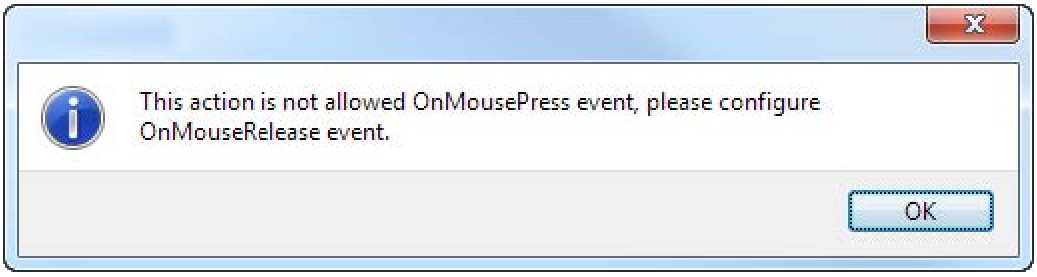
Variable	Explanation
No of Remote-Clients Alive	Number of Windows Client connected to the server. This is a read only short (16 bit).
This Client Group-Name	Name of the group to which the current logged user belongs to. This is a read only string.
This Client ID	The variable is valid with reference to the Windows Client scope. Local and remote clients connected to the same "server" (same runtime) get a unique ID returned by this variable. This is a read only short (16 bit).
This Client User-Name	Name of the user logged to the Client where the system variable is displayed This is a read only string.

Actions

Actions are used to interact with the system; they can be executed when events are triggered.

When considering events generated by buttons (pressed or released) not all the actions are available for both states.

In case the selected action is not supported for the actual state, the software will report a warning message as shown in the following figure.



This action is not allowed OnMousePress event, please configure OnMouseRelease event.

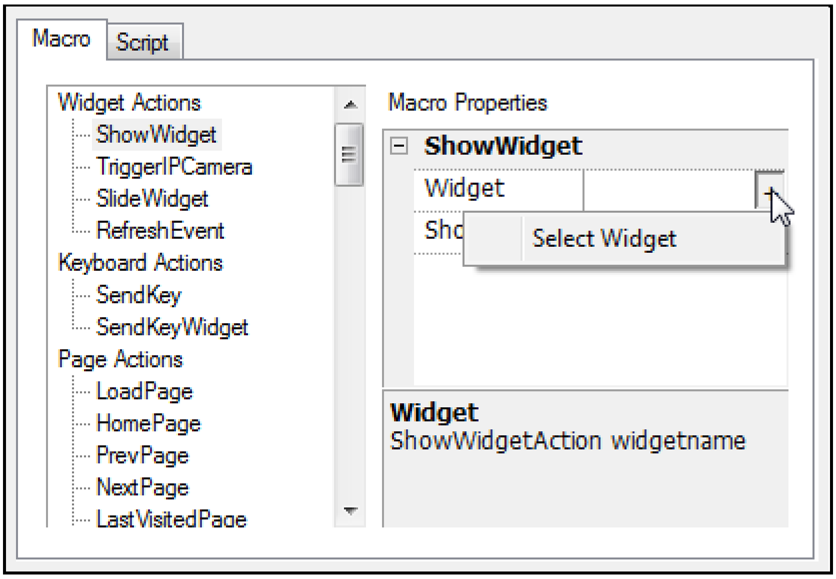
Widget Actions

The following chapter will include the description of a set of actions dedicated to handling widget visibility and control.

Show Widget

This macro allows you to show or hide the page widgets.

1. In the macro properties, select the widget you want to show or hide.
2. Set the *Show* properties as follows: **False** to hide and **True** to show widget.

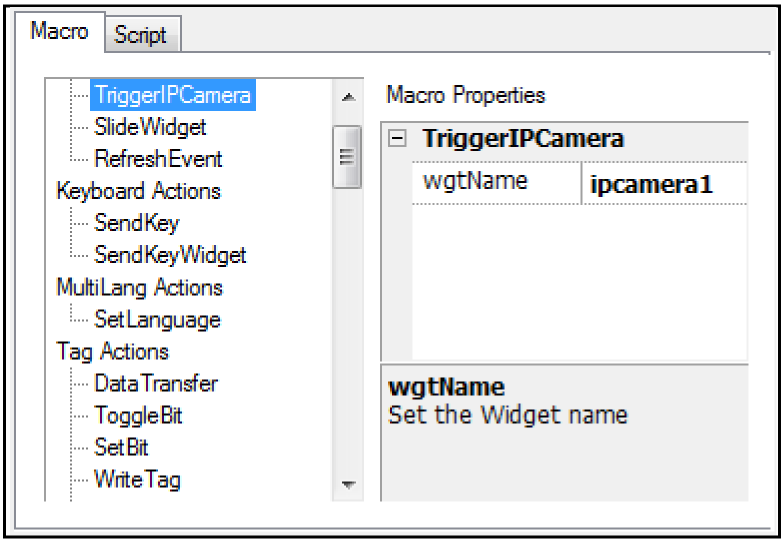


Widget
ShowWidgetAction widgetname

Trigger IP Camera

This macro allows you to start the image capture from an IP camera.

→ Select the IP camera from the macro properties to trigger the capture from the IP camera.

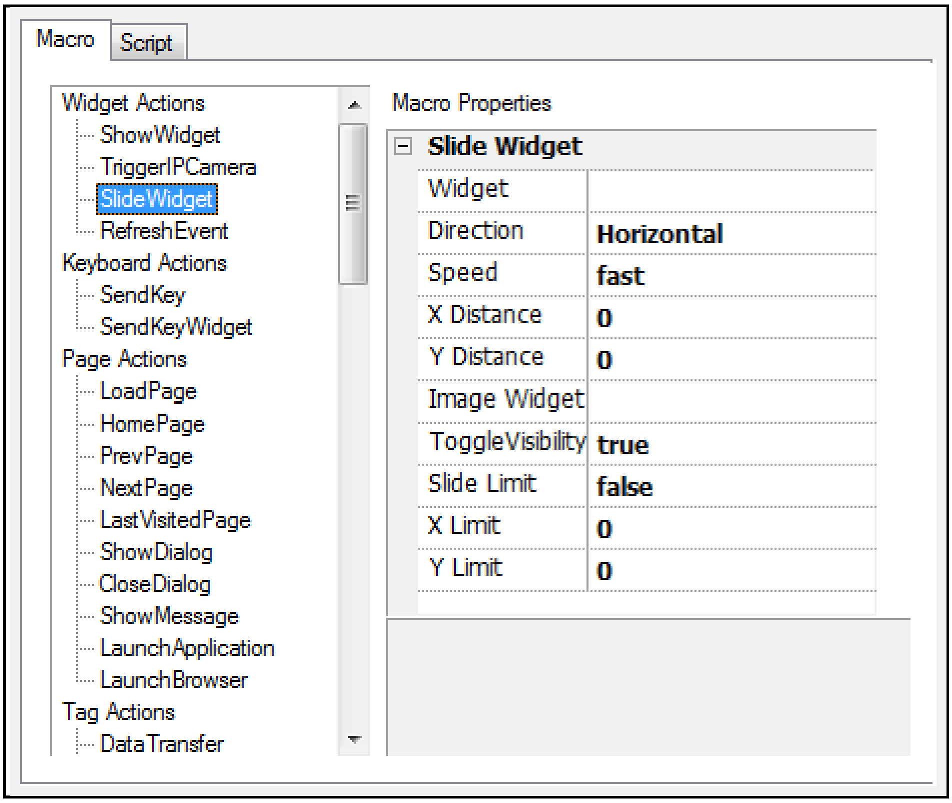


wgtName: ipcamera1

wgtName
Set the Widget name

Slide Widget

This macro allows you to show the sliding effect of a widget, or of a widget group, in HMI Runtime.



Slide Widget

Widget	
Direction	Horizontal
Speed	fast
X Distance	0
Y Distance	0
Image Widget	
ToggleVisibility	true
Slide Limit	false
X Limit	0
Y Limit	0

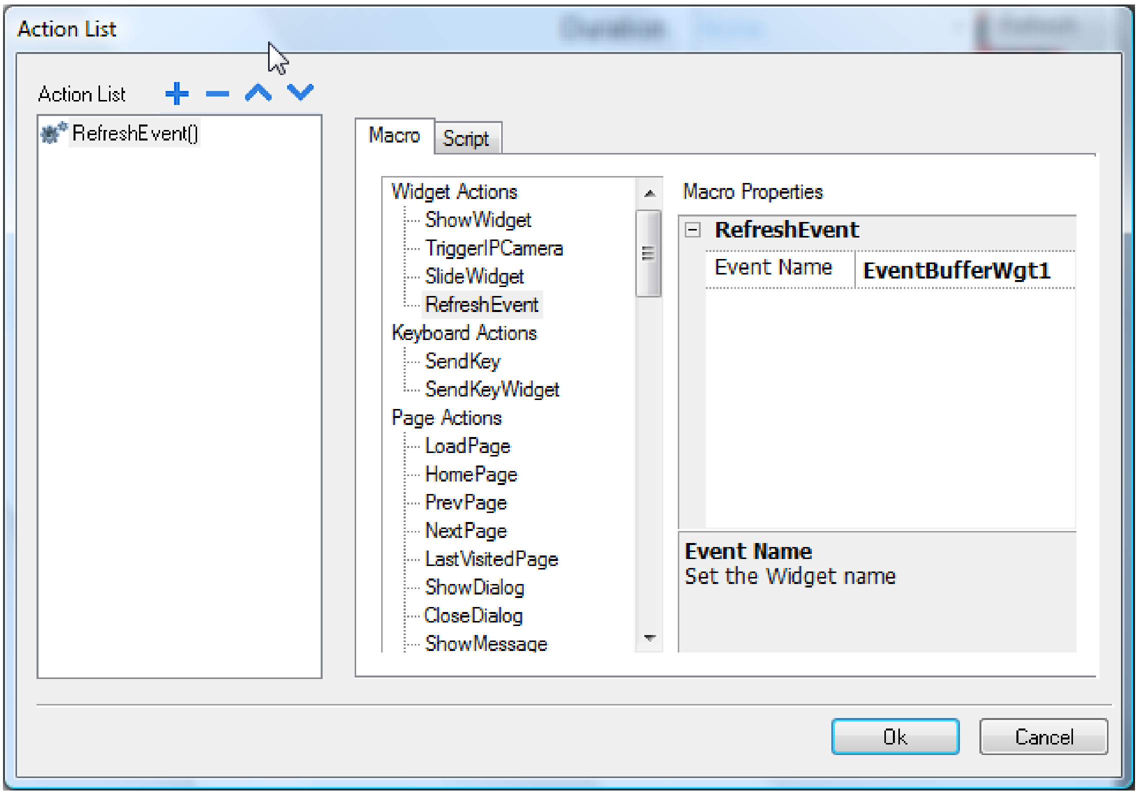
Parameter	Value
Widget	The widget to slide
Direction	Sliding direction
Speed	The transition speed of the sliding widget
X Distance	The travel distance of the X coordinate of pixel
Y Distance	The travel distance of the Y coordinate of pixel
Slide Limit	Enable/Disable limiting the movement with respect to the coordinates (X and Y) of the widget.
X-Limit	When specified, allows to stop automatically the slide action when the widget reaches the specified position.
Y-Limit	When specified allows stopping automatically the slide action when the widget reaches the specified position.
Toggle-Visibility	Toggle the visibility of the widget at the end of each slide action.
Image Widget	Allows to show an image during the movement; the specified image will be shown during the slide operation between start and end point of the movement.

NOTE

The widget, or grouped widget can actually be outside of the page in the project and slide in and out of view.

Refresh Event

The RefreshEvent macro allows you to refresh the selected Event widget. The Event Widget is a component of the Alarm History Widget. (see paragraph Configure Alarms History Widget.).

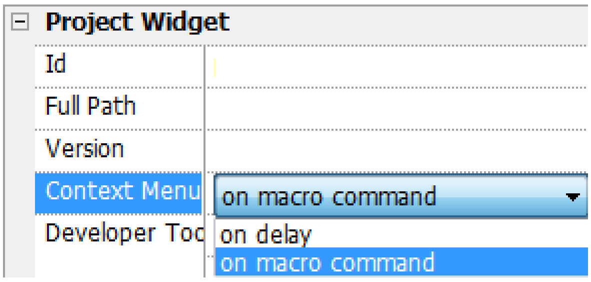


RefreshEvent	
Event Name	EventBufferWgt1

Event Name
Set the Widget name

ContextMenu

Context menu is used to configure runtime parameters like Zoom level, to update runtime & project using an update package, to show log window, to access the rotating menu for basic HMI configurations like IP Address or device local time etc. By default the context menu appears when the user press/click and hold for few seconds in the runtime area (in an area free of widgets like buttons). However, in the project area, it is possible to disable the OnHold event and configure the HMI to open the context menu just when the macro ContextMenu is called by the user. Usually this macro is attached to a button and protected to be used just by system administrators



Project Widget	
Id	
Full Path	
Version	
Context Menu	on macro command
Developer Toc	

ReplaceMedia

Replace media macro is used to replace Images and Media files. When called, a dialog appears to select source folder.

Keyboard Actions

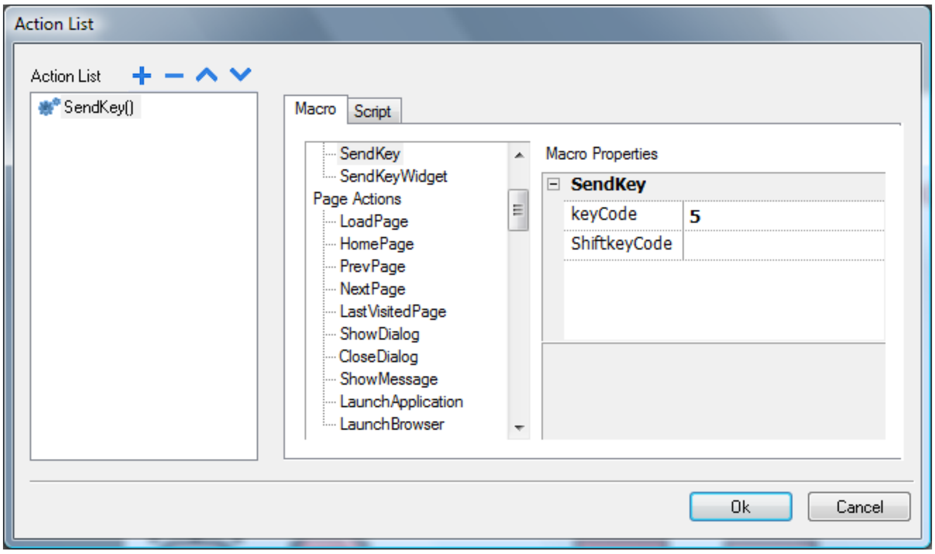
The Keyboard macro actions include “SendKey” and “SendKeyWidget”

Send Key

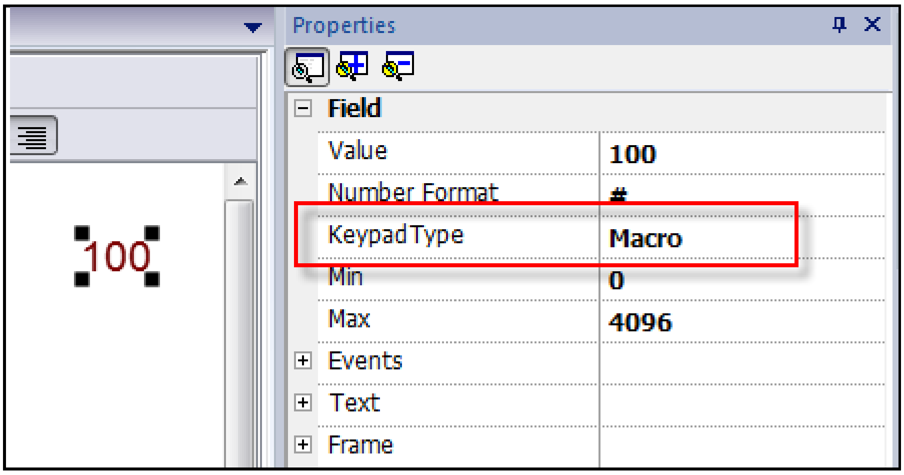
This macro is used to enter the predefined character to the read/write widget. Define the predefined key code and shift key code to the macro actions property.

1. In runtime, click the R/W numeric widget.
2. Execute the macro to send the predefined keys to the numeric widget.
   - ➤ The action works on the field currently being edited.

> **NOTE**
>
> To use this macro, you must define the keypad type as “Macro” in the numeric widget properties.



SendKey	
keyCode	5
ShiftkeyCode	



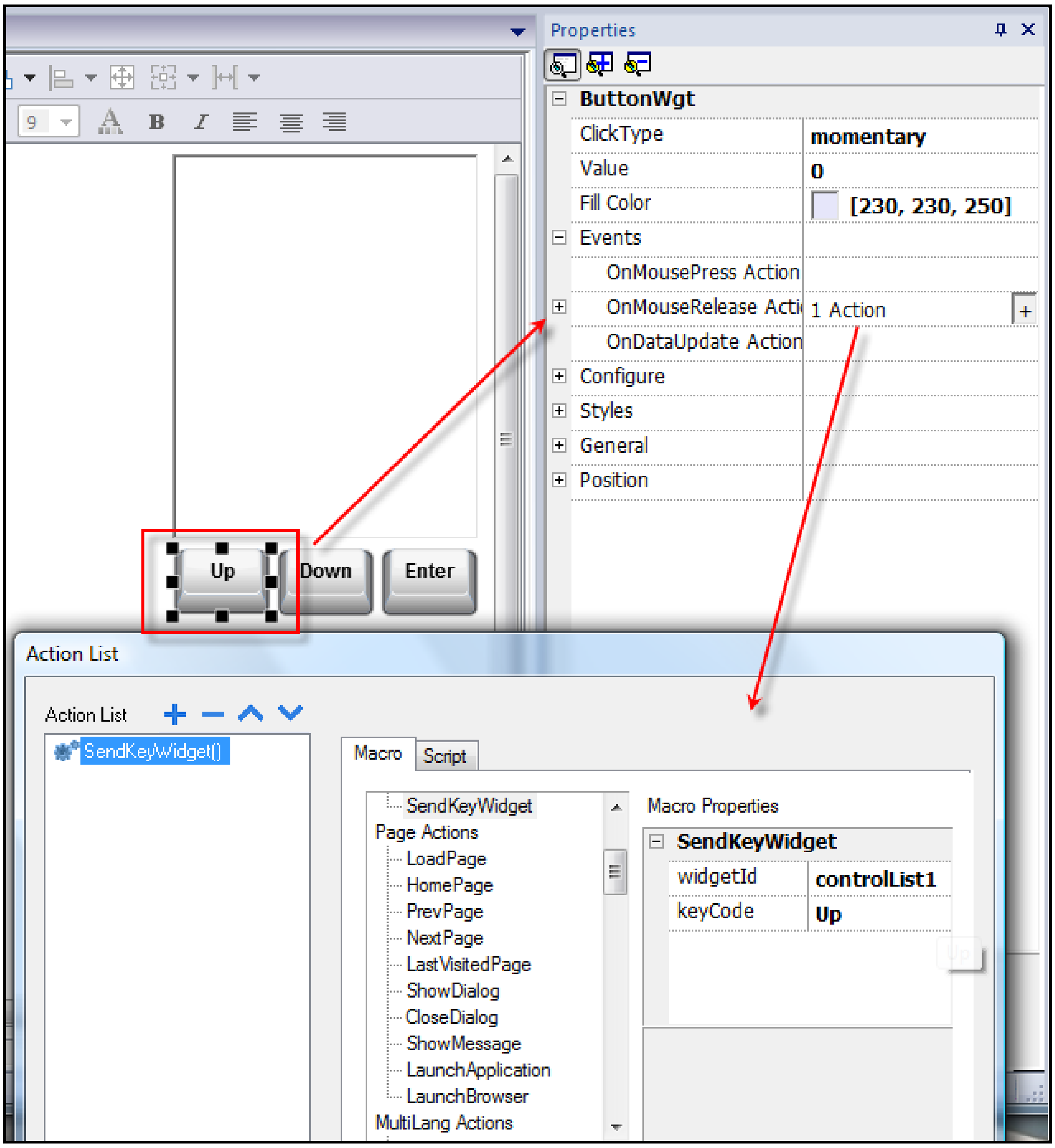
Field	
Value	100
Number Format	#
KeypadType	Macro
Min	0
Max	4096
Events	
Text	
Frame	

Send Key Widget

This macro is used to enter the predefined character for a specific widget. To use the macro, define the widget ID and the key code in the macro properties.

Example

Control list widget (available in the advanced category of the widgets gallery) is a good example of how this macro command can be used. Here **Up** and **Down** buttons have been implemented using the *SendKeyWidget* macro.



ButtonWgt

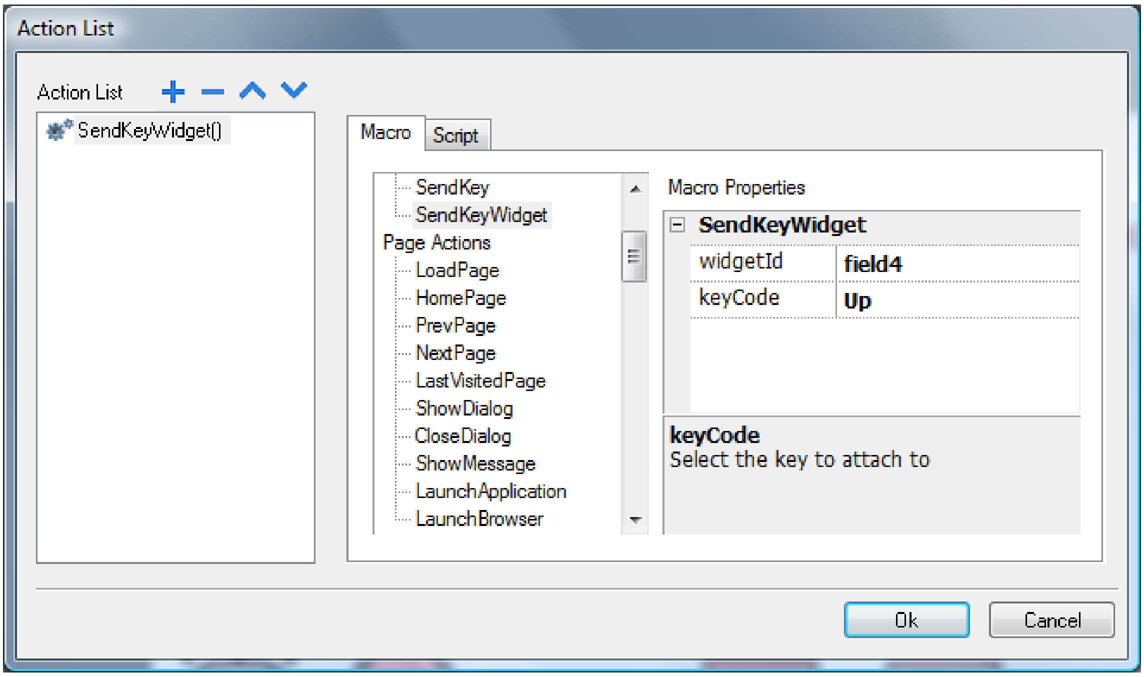
ClickType	momentary
Value	0
Fill Color	[230, 230, 250]
Events	
OnMousePress Action	
OnMouseRelease Acti	1 Action
OnDataUpdate Action	
Configure	
Styles	
General	
Position	

Action List: SendKeyWidget()

Macro Properties

SendKeyWidget

widgetId	controlList1
keyCode	Up



Action List: SendKeyWidget()

Macro Properties

SendKeyWidget

widgetId	field4
keyCode	Up

keyCode
Select the key to attach to

NOTE: To use the SendKey macro, you must define the keypad type as “Macro” in the numeric widget properties.

ShowKeyPad

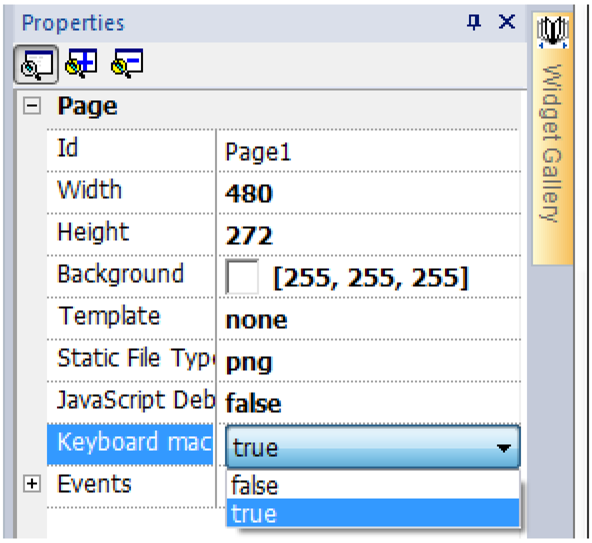
ShowKeyPad is used to show the default operating system touch keypad. Some operating systems might not support it

Keyboard Macros

Keyboard macros enable and disable the use of keyboard macros at runtime when using external keyboards.

You can also enable/disable macro execution related to keyboard PB610 side at the project level and at the level of the single page.

A dedicated property is available in the project property sheet and in the page property sheet.



Page	
Id	Page1
Width	480
Height	272
Background	[255, 255, 255]
Template	none
Static File Typ	png
JavaScript Deb	false
Keyboard mac	true
Events	

Page Actions

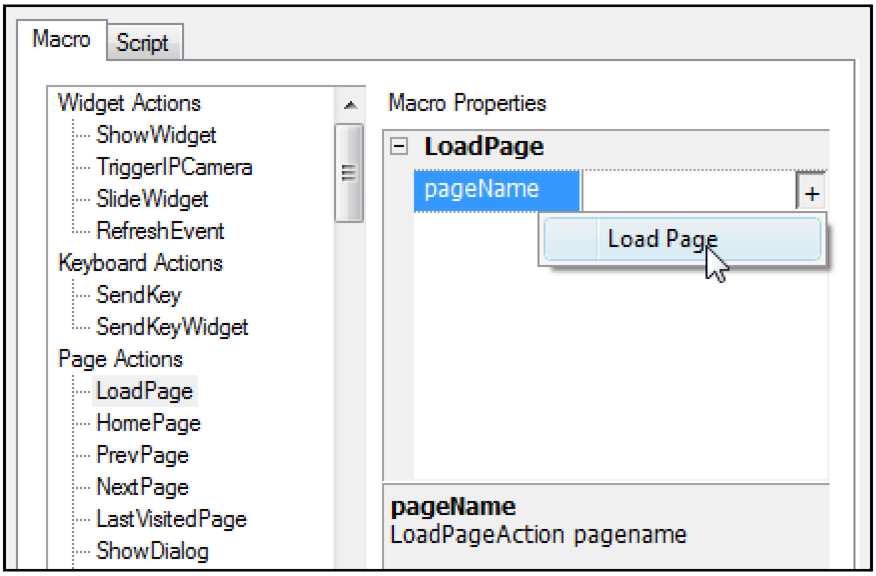
This macro is used for page navigation and load-specific pages.

NOTE

Page actions are programmable ONLY in the released state.
The page actions macro is available for “Alarms”, “Schedulers” and “Mouse Release Events”.

Load Page

This macro allows you to load the selected page when the macro is executed.

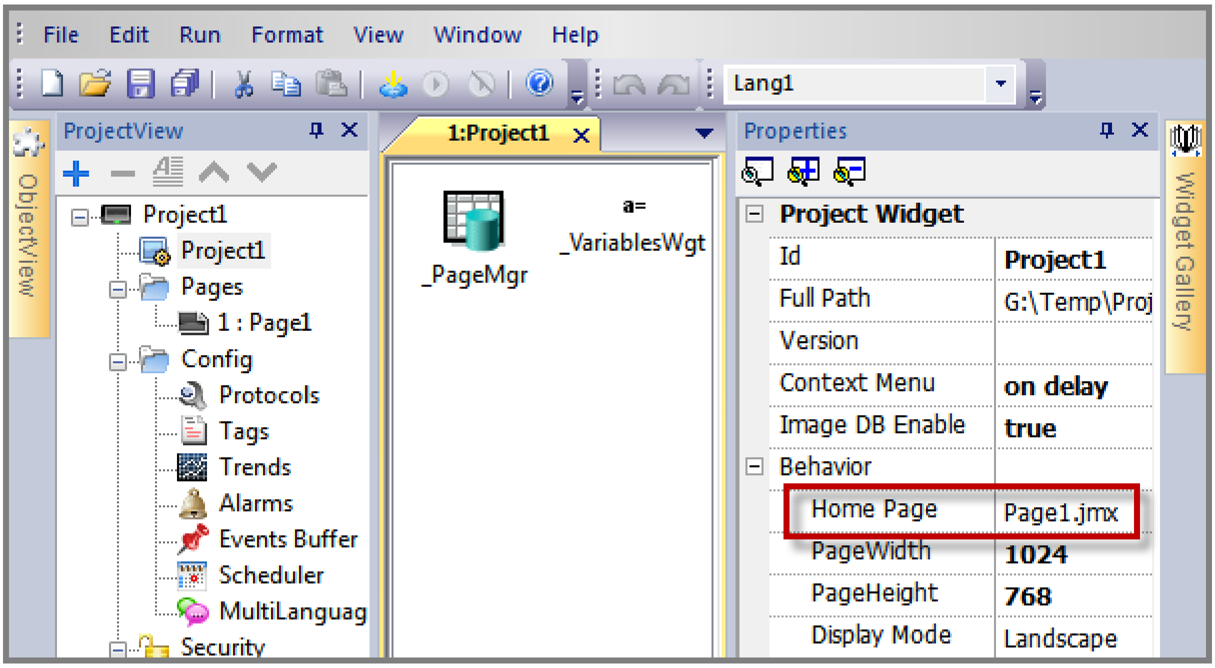


pageName
LoadPageAction pagename

Home Page

This property allows you to specify the home page. By default, the home page is the first page. You can change the home page in the project configuration properties.

1. To change the home page, double click on the project name item in Project View.
2. Once in properties, choose the home page.



Project Widget	
Id	Project1
Full Path	G:\Temp\Proj
Version	
Context Menu	on delay
Image DB Enable	true
Behavior	
Home Page	Page1.jmx
PageWidth	1024
PageHeight	768
Display Mode	Landscape

PrevPage

This macro allows you to navigate the HMI Runtime to the previous page.

NextPage

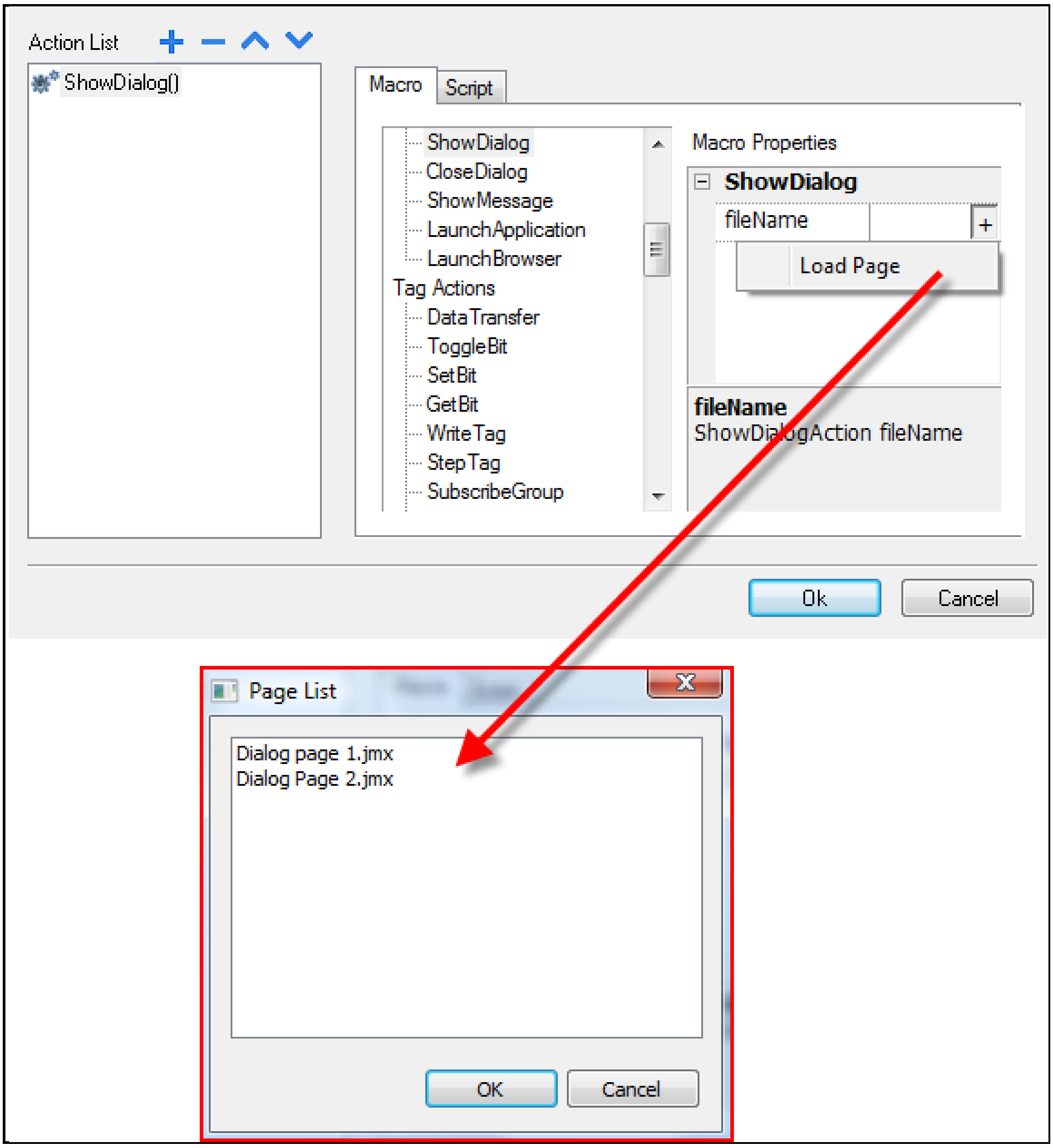
This macro allows you to navigate the HMI Runtime to the next page.

LastVisitedPage

This macro allows you to load the page previously displayed on HMI Runtime.

ShowDialogPage

This macro allows you to display dialog pages defined in the project. After the execution of this macro, the HMI Runtime displays the specified dialog page.



fileName
ShowDialogAction fileName

Dialog page 1.jmx
Dialog Page 2.jmx

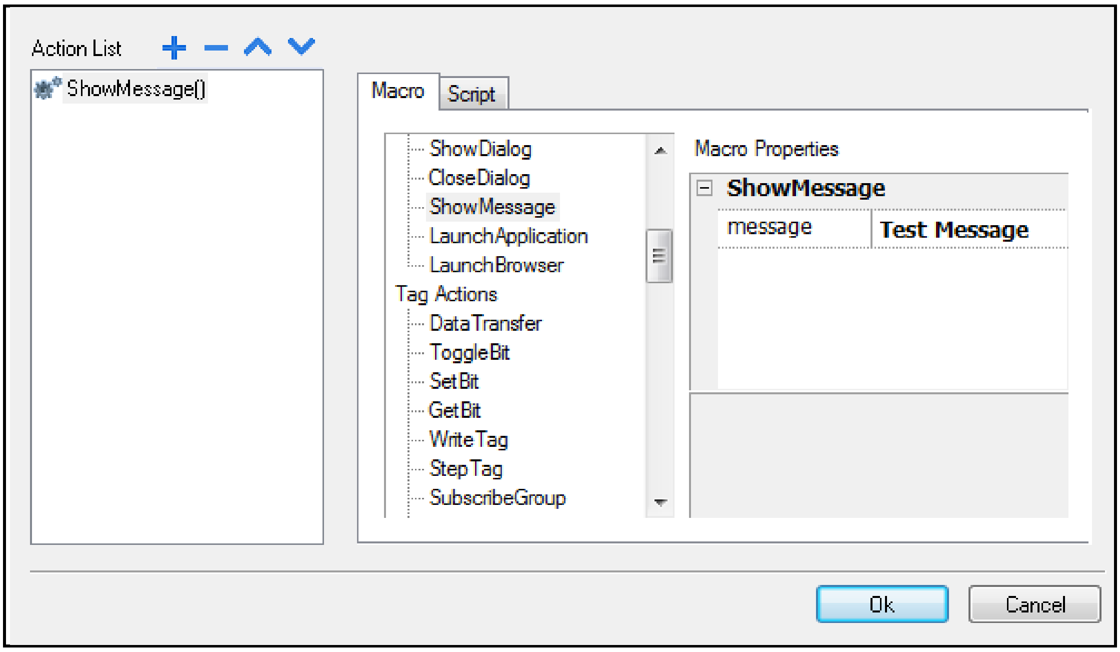
CloseDialog

This macro is applicable only on dialog pages. The *Close Dialog* page allows you to close the dialog page currently displayed.

ShowMessage

This macro allows you to display warning message pop-ups when the macro is executed.

→ Type the message that you wish to have displayed while executing the macro.

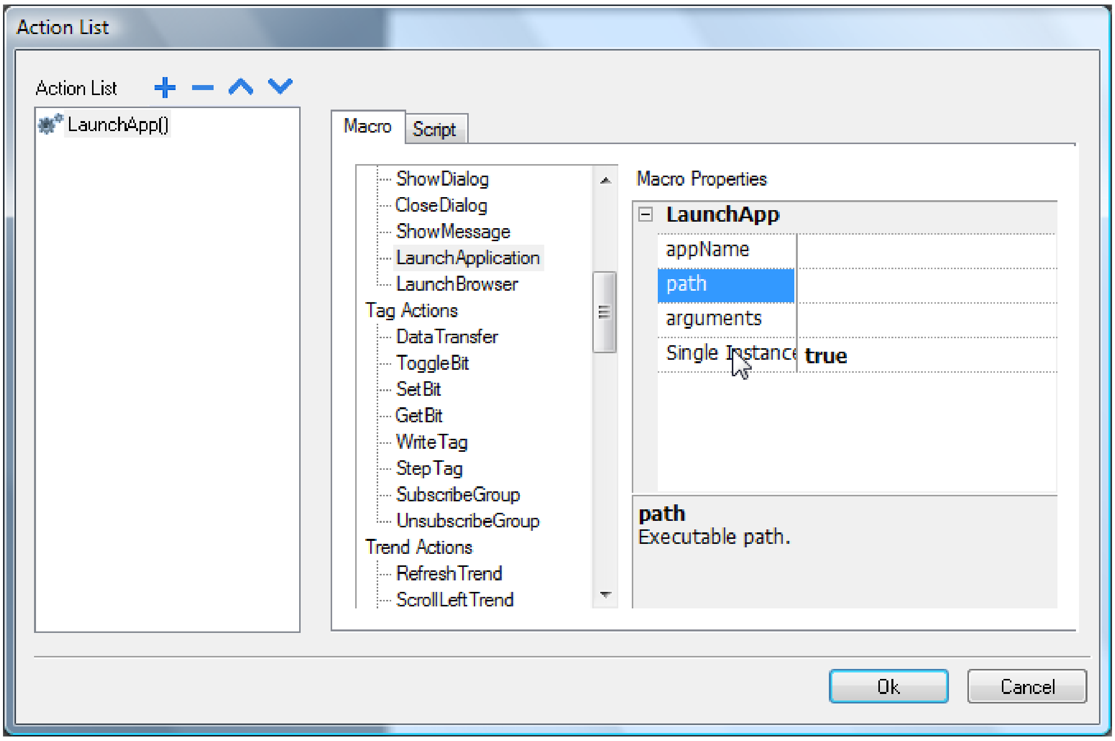


ShowMessage

message: Test Message

LaunchApplication

This macro allows the user to launch an external third party application when the macro is executed.



LaunchApp

appName	
path	
arguments	
Single Instance	true

path
Executable path.

The following inputs must be provided in order to execute the specific application.

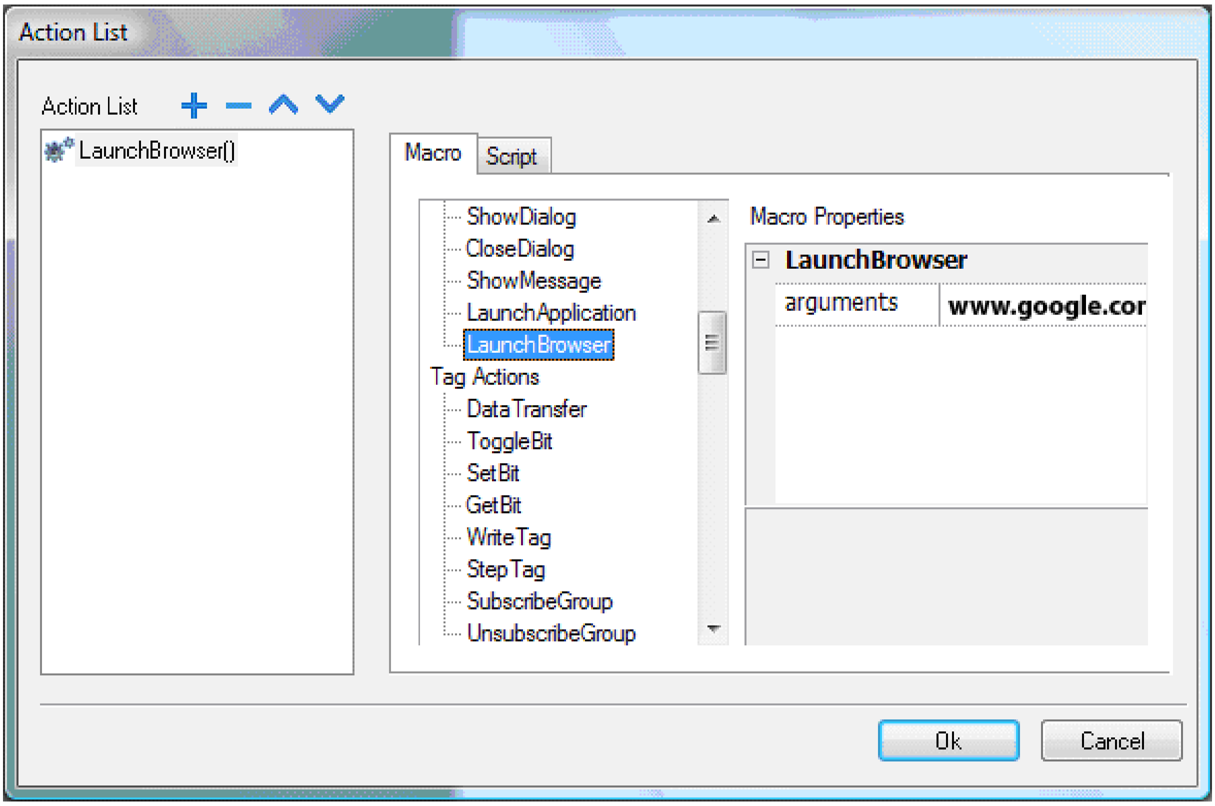
- **App Name:** The executable file name with extension. Example: If you want to run notepad application, the argument should be “notepad.exe”.
- **Path:** The application path; when the target platform is Windows CE, the path is \flash\qthmi. This is the folder that you see and have access to, when connecting to the panel via FTP
- **Arguments:** Some applications may need arguments to be passed. For example, to open a pdf file, specify the file name so that, while launching the application, the file name set in the argument is loaded on the application. For example, \flash\qthmi\Manual.pdf will open the document “Manual.pdf”.

- **Single Instance:** This argument allows the application to start in single instances or multiple instances. When single instance is selected the system first verifies whether the application is already running. If it is running, then the application gets the focus (the operating system puts it in foreground to user attention); if it is not running, then the application is launched.

Launch Browser

This macro will launch the default web browser. You can define the URL address of the webpage in the arguments.

> **NOTE** This macro only works in platforms that have a default web browser as application. Not all platforms are equipped with a default web browser. For example, this macro is supported in OS based on Windows CE Pro with Internet Explorer enabled.



LaunchBrowser
arguments: www.google.cor

LaunchVNC

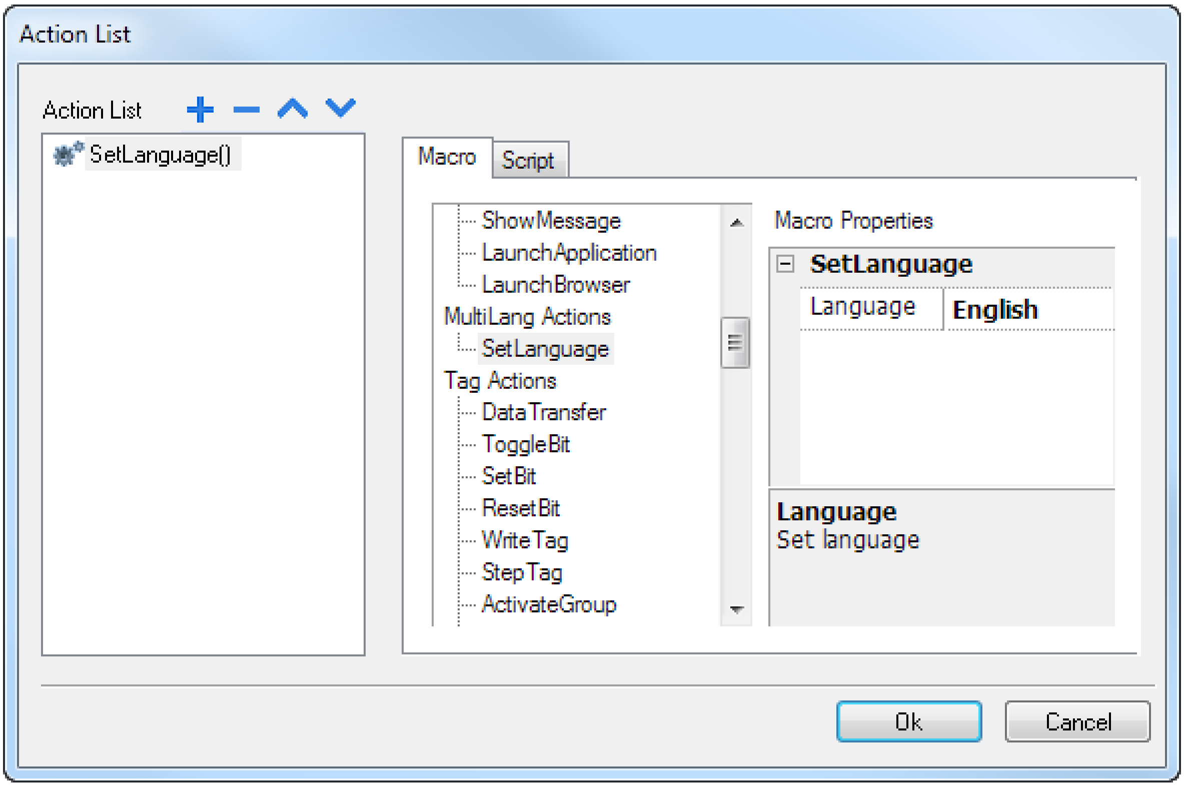
Use this macro to execute VNC server configuration form. This macro is working only on embedded devices WinCE based.

MultiLang Actions

The Multi-Language (MultiLang) actions are used to select and modify the languages used in the application.

SetLanguage

The SetLanguage macro allows you to set the current display language. In Macro Properties, enter in the Language. At runtime, while executing the macro, the selected language will be applied to all applicable Widgets.



SetLanguage

Language	English

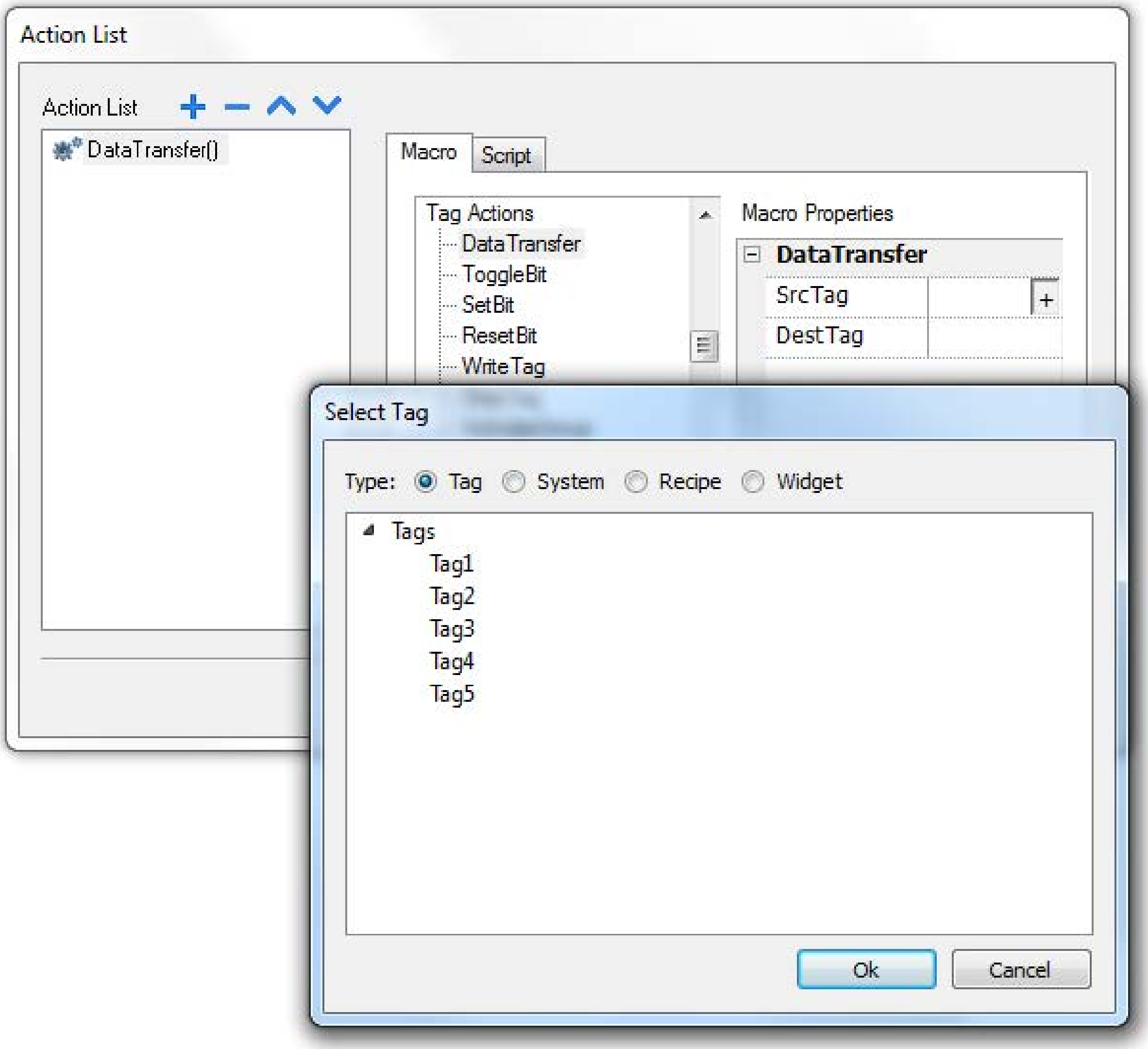
Language
Set language

Tag Actions

These macros are used to interact with the application's tags.

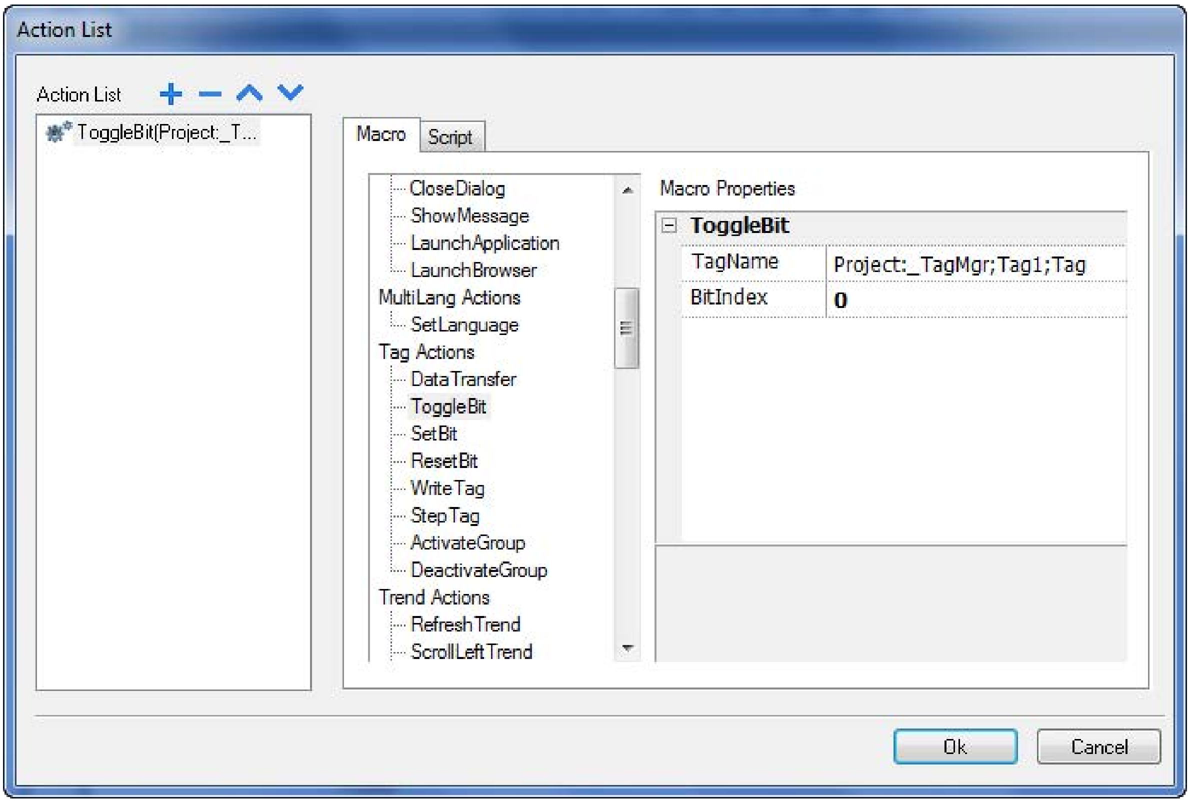
Data Transfer

This macro allows you to exchange data between two controllers, between registers within a controller, or from system variables to controllers (and vice versa). *SrcTag* refers to the source tag and *DestTag* refers to destination tag. The various tag types include a controller tag, a system tag, a recipe tag and widget property.



Toggle Bit

This macro allows you to “toggle” (meaning set or reset) a bit of a word. The bit index allows you to select the bit to be inverted: this implies a read-modify-write operation; the read value is inverted and then written back to the controller tag.



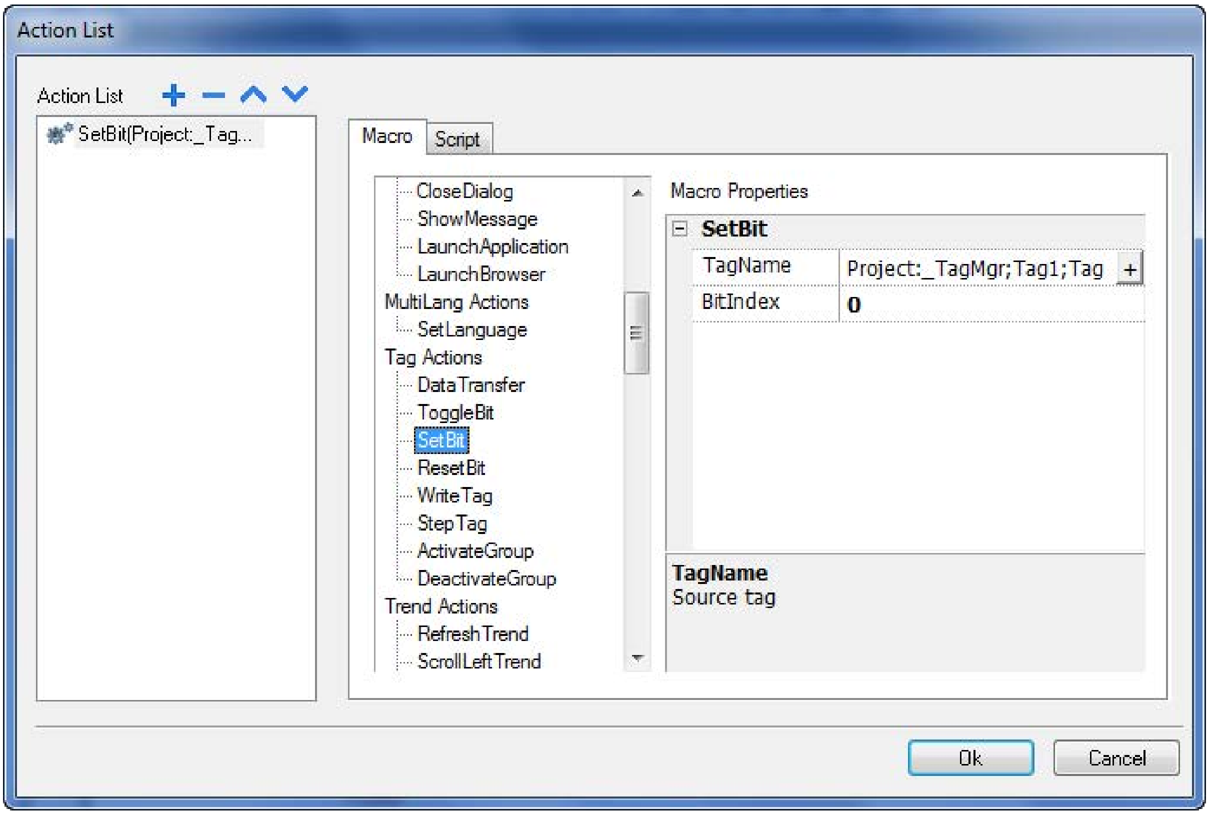
ToggleBit

TagName	Project:_TagMgr;Tag1;Tag
BitIndex	0

SetBit

This macro allows you to set the selected bit. When the macro is executed, the selected bit value is set to "1".

The BitIndex property allows you to select the bit position inside the tag.



SetBit

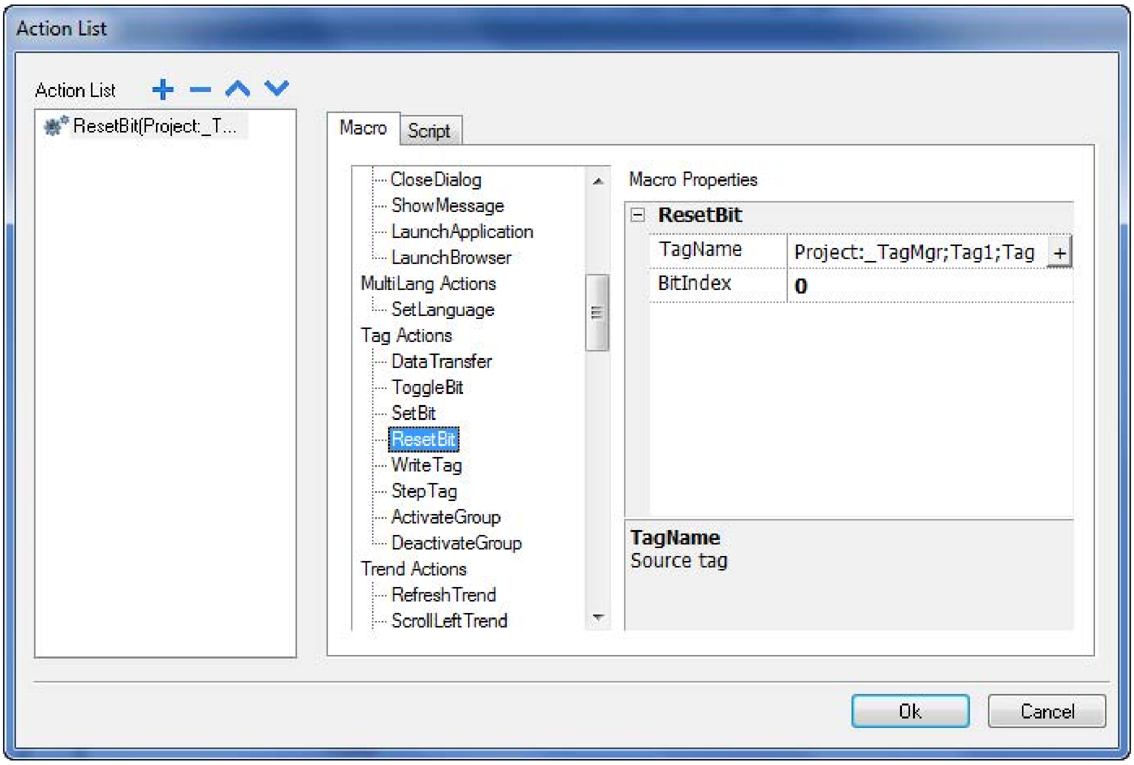
TagName	Project:_TagMgr;Tag1;Tag
BitIndex	0

TagName
Source tag

ResetBit

This macro allows you to reset the selected bit. When the macro is executed the selected bit value is set to “0”.

The BitIndex property allows you to select the bit position inside the tag.



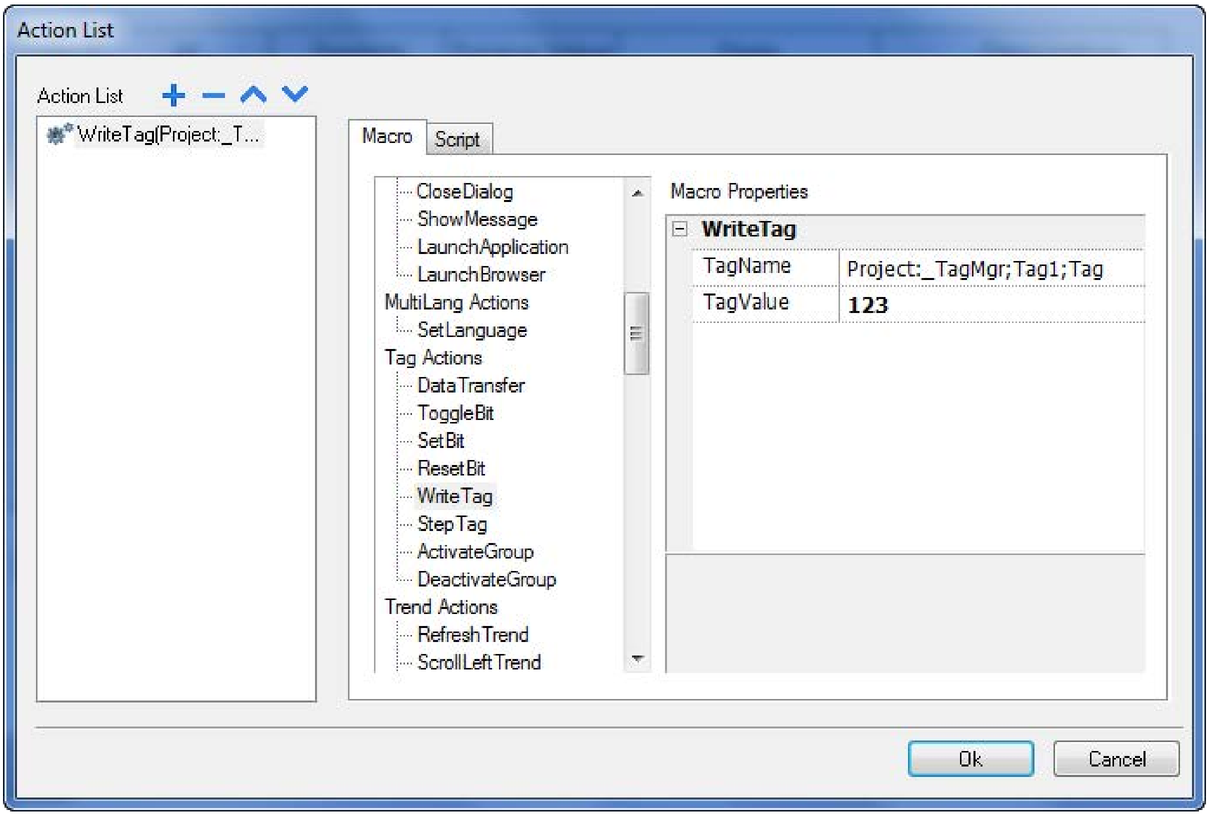
ResetBit

TagName	Project:_TagMgr;Tag1;Tag
BitIndex	0

TagName
Source tag

WriteTag

This macro allows you to write constant values for the controller memory. In the action list, specify the tag name and the constant value to be written.



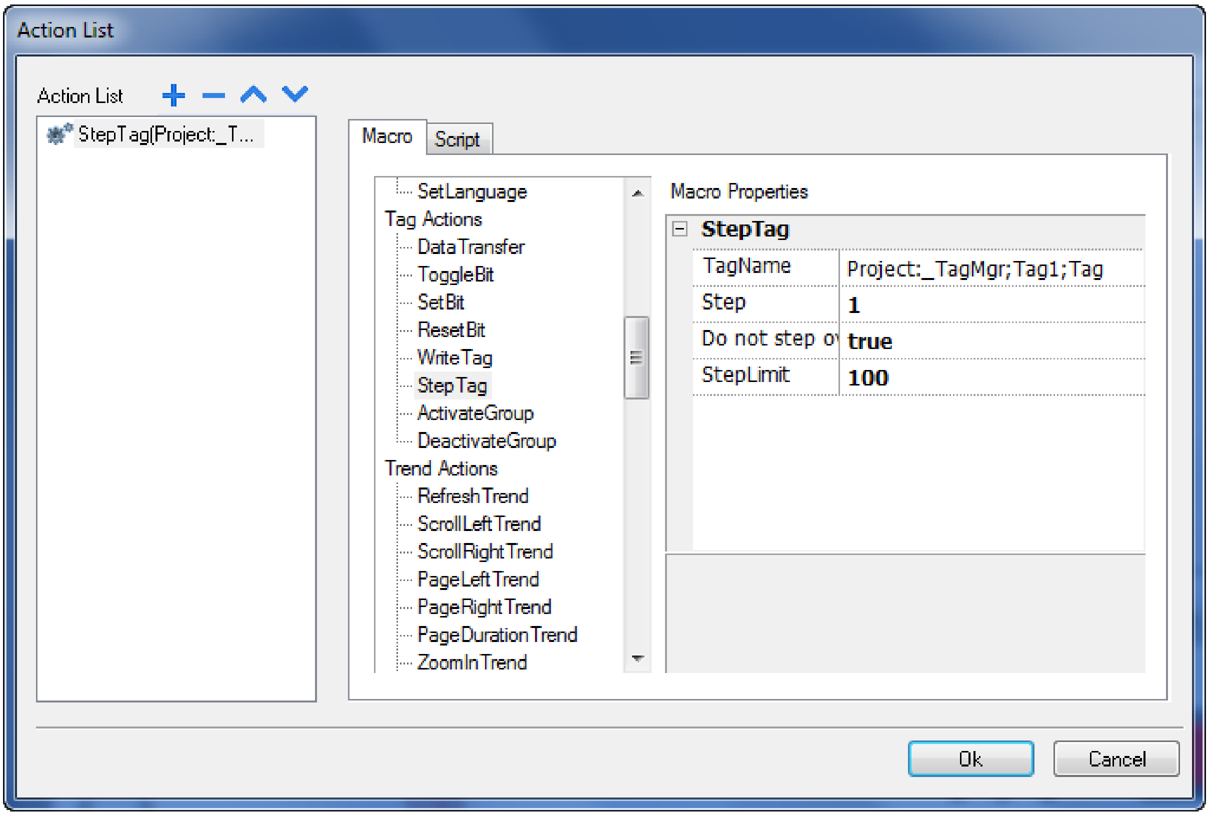
WriteTag

TagName	Project:_TagMgr;Tag1;Tag
TagValue	123

StepTag

This macro allows you to increment or decrement, in steps, the content of a tag value.

- **Tag Name:** Tag name that you want to step.
- **Step:** Step value.
- **Do not step over limit:** Step limit enables.
- **Step Limit:** If the step over limit is true, the macro will work until the tag value reaches the specified level.



StepTag

TagName	Project:_TagMgr;Tag1;Tag
Step	1
Do not step o\	true
StepLimit	100

ActivateGroup

This macro activates tags update for a group of Tags.

Usually tags are updated when used in the current page (or always when defined in Tag Editor as Active =True). Using this macro it is possible to force the system to keep a group of tags always active (updated) independant if they are used on the current page or not.

DeactivateGroup

Deactivate a group of tags. Using this macro system stops reading a group of tags that had been previously activated

Trend Actions

These macros are used for both live data trends and the historical trends widget.

RefreshTrend

This macro is used to refresh the historical trend window. You have to specify the trend widget in the macro properties. This macro can be used in any of available graph widgets like Historical trends, Scatter Diagram and Consumption Meter.

ScrollLeftTrend

This macro is used to scroll the trend window to the left side, by a one-tenth (1/10) page duration. For the live data trend, this macro works when the trend is paused.

> **NOTE**
>
> With the Real-Time trend it is recommended to pause the trend using the macro PauseTrend, otherwise the window is continuously shifted to the current value.

ScrollRightTrend

This macro is used to scroll the trend window to the right side, by a one-tenth (1/10) page duration. For the live data trend, this macro works when the trend is paused.

> **NOTE**
>
> With the Real-Time trend it is recommended to pause the trend using the macro PauseTrend, otherwise the window is continuously shifted to the current value.

PageLeftTrend

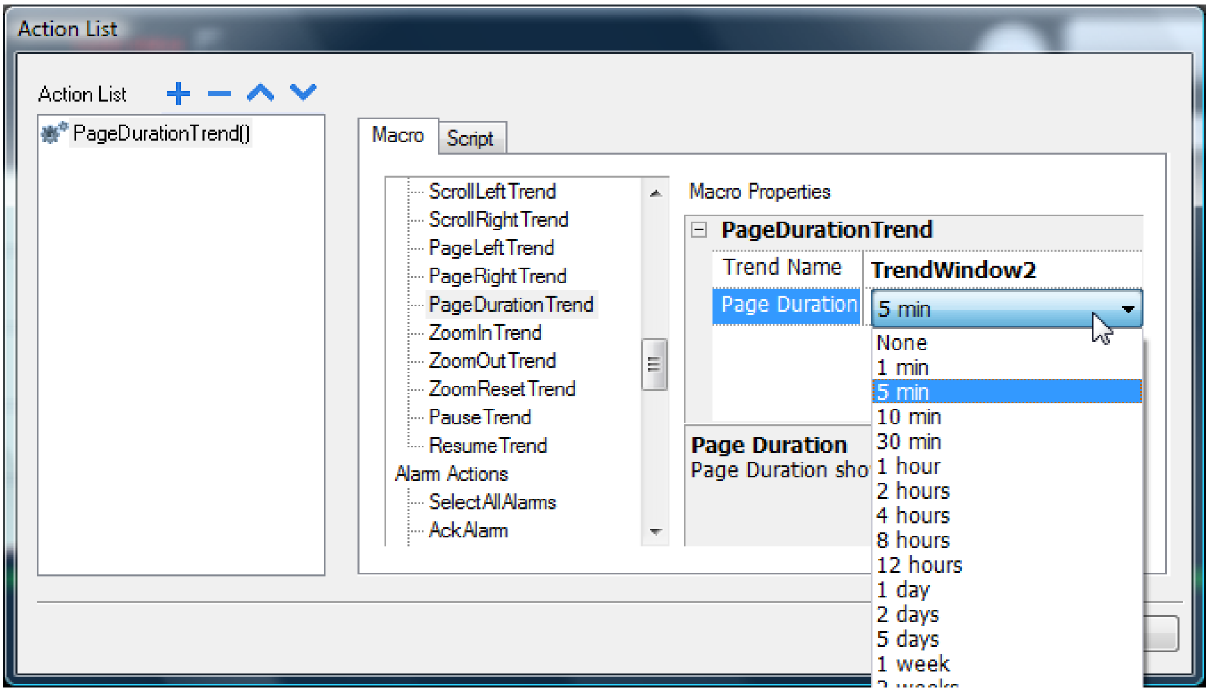
This macro allows you to scroll the trend window by a one-page duration. Example: If the page duration is 10 minutes, with this macro you can scroll the trend left for 10 minutes.

PageRightTrend

This macro allows you to scroll the trend window by a one-page duration. Example: If the page duration is 10 minutes, with this macro you can scroll the trend right for 10 minutes.

PageDurationTrend

This macro is used to set the page duration of the trend window. In macro properties, you must define the trend window and duration.



Action List

Action List: PageDurationTrend()

Macro Properties

PageDurationTrend

Trend Name: TrendWindow2

Page Duration: 5 min

Page Duration

Page Duration sho

NOTE

You can also use a combo box widget to select the page duration in runtime.

ZoomInTrend

This macro allows you to reduce the page duration.

ZoomOutTrend

This macro allows you to make the page duration longer.

ZoomResetTrend

This macro allows you to reset the zoom level back to the original zoom level.

PauseTrend

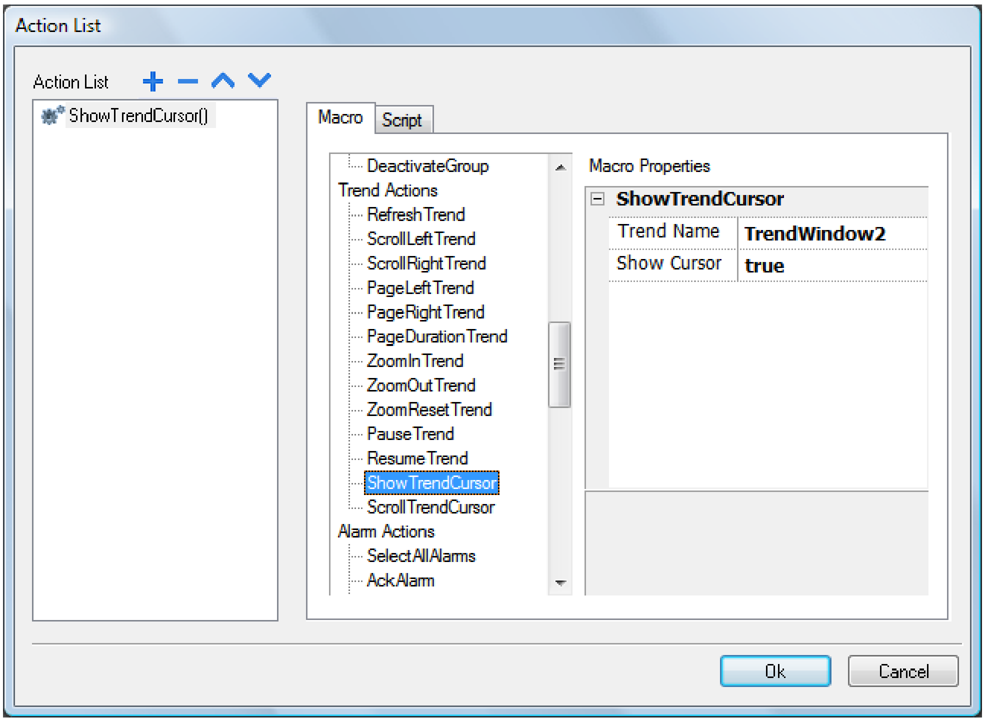
This macro allows you to stop plotting the trend curves in the trend window. The trend logging operation is not stopped from the panel when this macro command is used.

ResumeTrend

This macro allows you to resume a trend plotting you previously paused. After executing this macro, the trend window will start to plot the data to the trend once again.

ShowTrendCursor

This macro allows the user to know the value of the curve at a given point on the X-axis. Use this macro to activate the trend cursor. In runtime, upon executing the macro, a vertical line (cursor) will display in the trend widget. When the graphic cursor is enabled, the scrolling of the trend is stopped. You can implement this macro to move the graphic cursor over the curves, or to move the entire trend window.



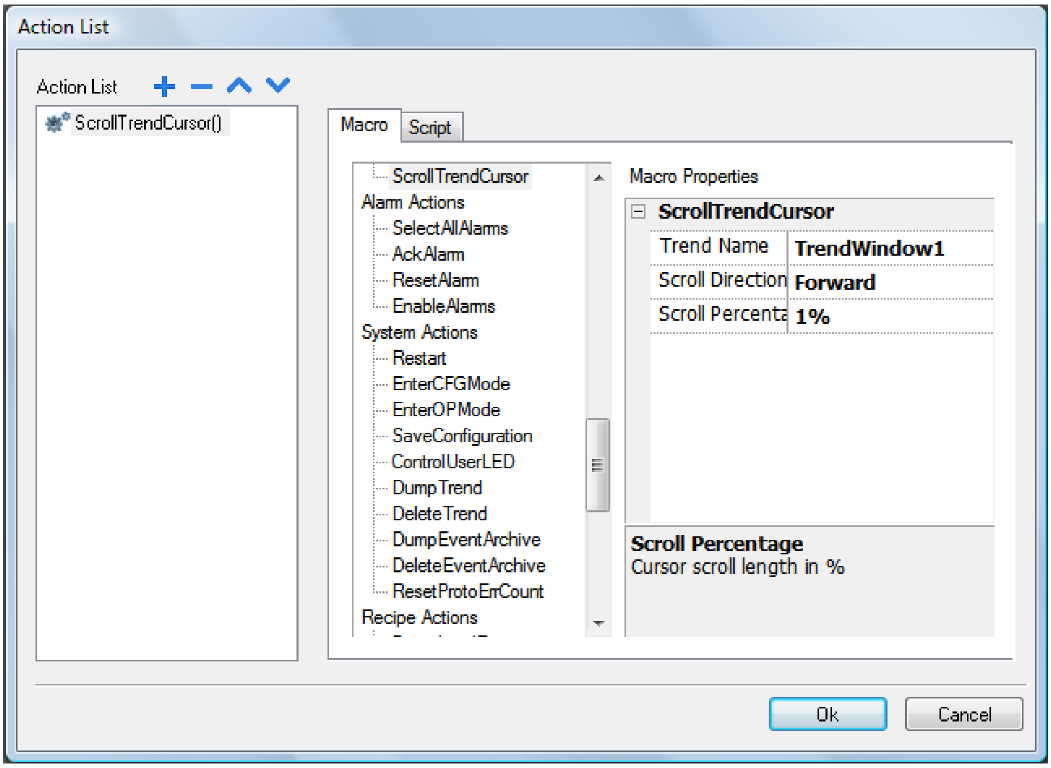
Macro Properties

ShowTrendCursor

Trend Name	TrendWindow2
Show Cursor	true

ScrollTrendCursor

This macro allows the user to scroll through the trend cursor in forward or reverse, thereby moving the cursor to the point at which you want the value of the trend. The Y-cursor value will display the trend value at the point of the cursor. The scrolling percentage can be set at 1 % or 10 %. The percentage is calculated based on the trend window duration.



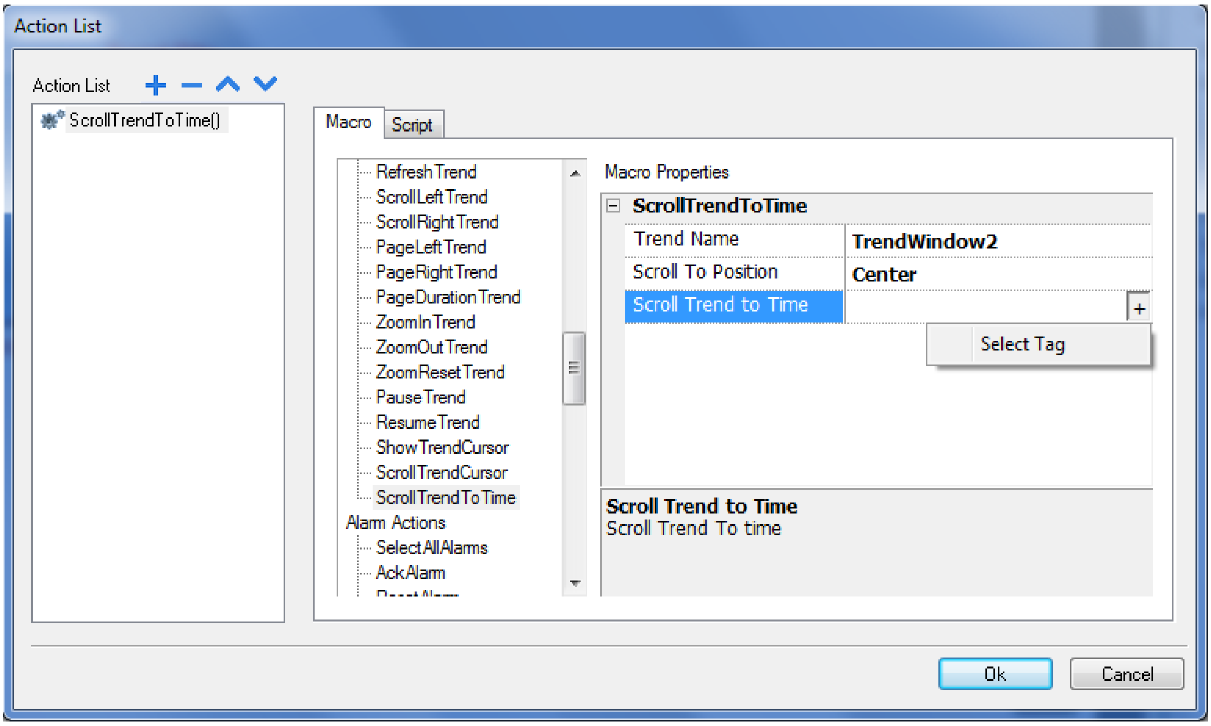
ScrollTrendCursor

Trend Name	TrendWindow1
Scroll Direction	Forward
Scroll Percenta	1%

Scroll Percentage
Cursor scroll length in %

ScrollTrendToTime

This macro is used to scroll the Trend Window to a particular point in time. When you execute this macro the Trend Window will move to the time specified in the Macro Properties.



ScrollTrendToTime

Trend Name	TrendWindow2
Scroll To Position	Center
Scroll Trend to Time	

Scroll Trend to Time
Scroll Trend To time

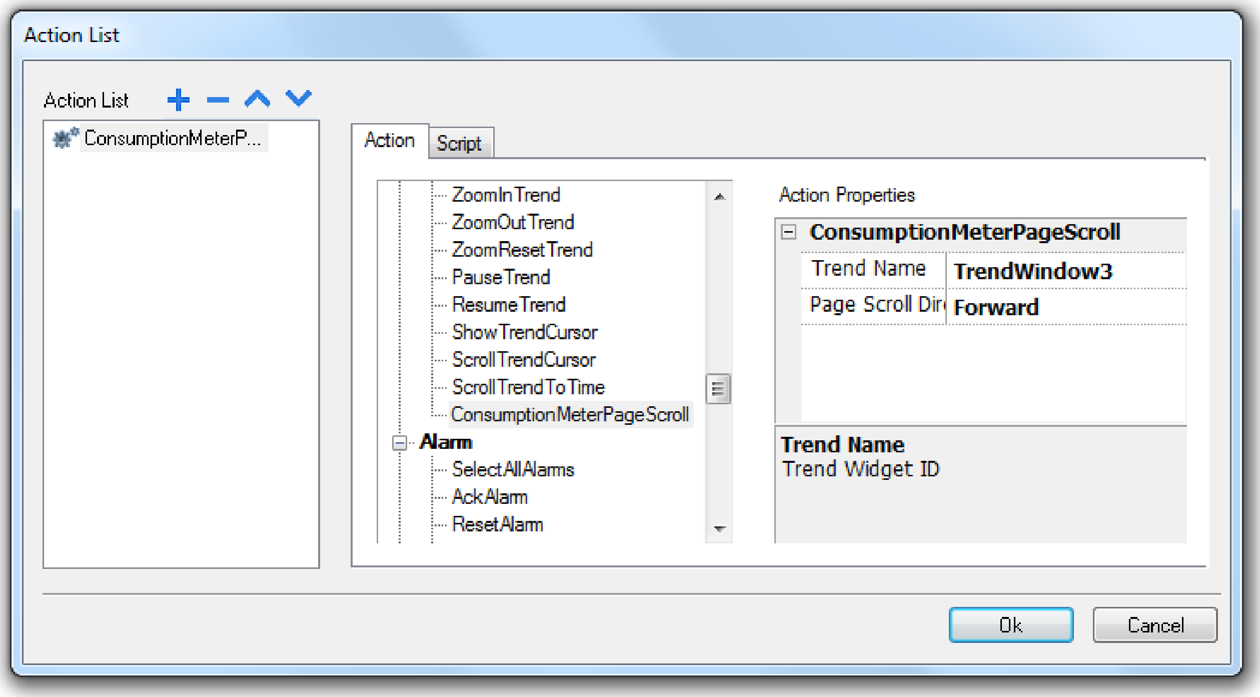
This Action may be very useful when you need to scroll at a specific position in a trend window based on the time at which a certain event occurs. This can be achieved by configuring an action for that alarm (event) that executes a Data Transfer of the system time into a Tag; when selecting that tag as “ScrollTrendtoTime” parameter (see above figure) the trend windows will be centered at the time in which the event has been triggered.

Consumption MeterPageScroll

This macro is used to scroll page back / forward in consumption meter widget.

Available parameters are:

- Trend Name: Trend widget ID (ex. TrendWindow3)
- Page Scroll Direction: Forward / Reverse



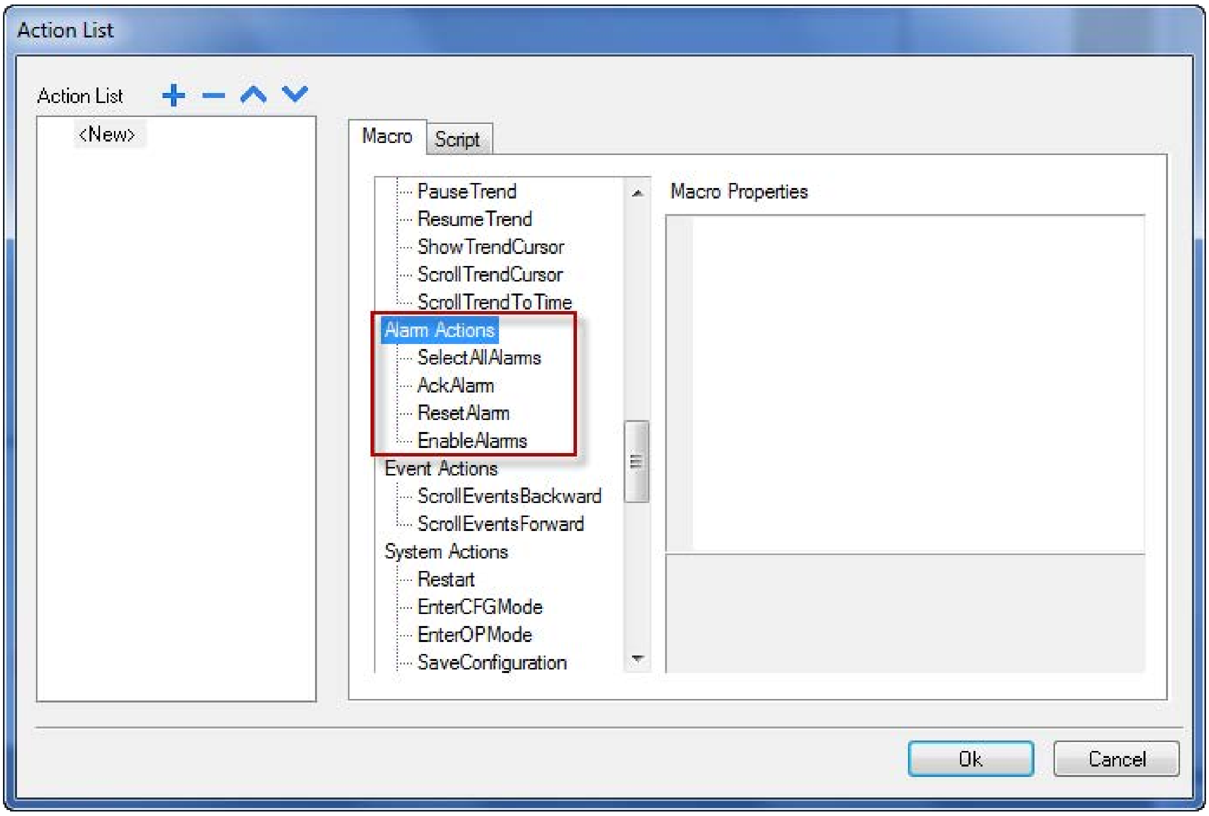
ConsumptionMeterPageScroll

Trend Name	TrendWindow3
Page Scroll Dir	Forward

Trend Name
Trend Widget ID

Alarm Actions

These macros are used to acknowledge or reset the alarms. The actions listed here can be used to build a custom widget for the alarms display. You can observe an example of how these are used in the default alarm widget, available in the widget gallery.



SelectAllAlarms

This macro allows you to select all the alarms in the alarm widget.

AckAlarm

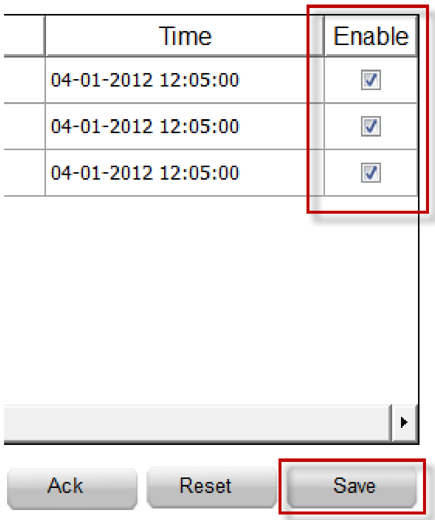
This macro allows you to acknowledge the selected alarms.

ResetAlarm

This macro allows you to reset the selected acknowledged alarms.

EnableAlarms

The Enable Alarm Action is used within the **Save** button of the alarm widget; it is required to properly save at runtime the changes done in the “Enable” check boxes from the “Enable” column in the alarm widget.



Time	Enable
04-01-2012 12:05:00	☑
04-01-2012 12:05:00	☑
04-01-2012 12:05:00	☑

Event Actions

ScrollEventsBackward

This macro used by the alarm history widget to scroll events/alarms backward in table view (event buffer widget).

ScrollEventsForward

This macro used by the alarm history widget to scroll events/alarms forward in table view (event buffer widget).

System Actions

These macros allow you to use the system properties in runtime.

Restart

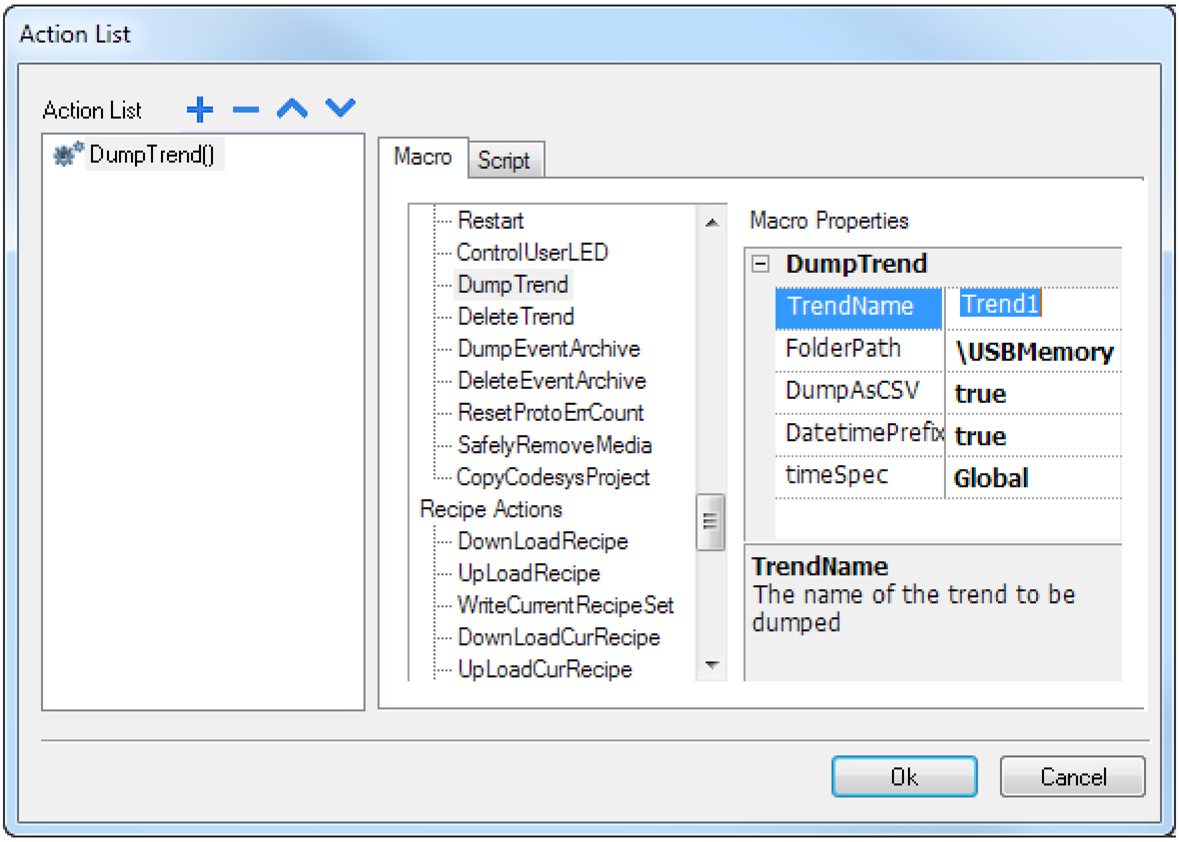
This macro allows you to restart Runtime. After executing the macro, the Runtime goes to configuration mode and restarts.

Dump Trend

This macro is used to store the historical trend data to external drives, such as a USB drive or an SD card. In the macro properties, you must configure the historical trend name you want to store and the destination folder path. If you use a USB drive plugged into the USB port, the path will be "\USBMemory", or if you use a SD Card, the path will be \Storage Card, followed by the specified folder in the memory.

NOTE	The execution of the dump action will automatically force a flush to disk of the data temporarily maintained in the RAM memory. See the chapter "Trend Editor" for further information about the policy used to save sampled data to disk.

NOTE	The external drives plugged on the USB port of the panel must be FAT or FAT32 formatted. NTFS format is not supported.



DumpTrend	
TrendName	Trend1
FolderPath	\USBMemory
DumpAsCSV	true
DatetimePrefix	true
timeSpec	Global

TrendName
The name of the trend to be dumped

DumpAsCSV: If this option is set true, then the buffer will be directly dumped in the specified location as a **.CSV* file in the format specified below. If it is set as False, then the dumped trend file comes in a binary format; the result of the dump operation is actually a couple of files, one with extension *.dat* and one with extension *.inf*. An external utility is then required to convert it to a CSV format. The two above mentioned files are both required by the utility to operate the conversion.

DateTimePrefixFileName: When this option is enabled the dumped file will have the date and time as prefix to the name of the file. Example: If you are making a dump at 10.10AM on 1-1-2012, then the file name will look like D2012_01_01_T10_10_Trend1.csv.
[DYear_Month_Day_THour_Minute_Filename].

This helps to know the time at which the dump was executed and also to identify which is the latest one.

TimeSpec

This option defines the time format used when dumping the trend to file.

- Local: the time values exported are the time of the HMI device.
- Global: the time values exported are in the Coordinated Universal Time (UTC) format.

Example:

Local	2012-10-11T05:13:43.724-07:00
Global	2012-10-11T12:13:43.724Z

NOTE

The utilities required to convert the dumped files to CSV are available in the Panel Builder 600 folder called “Utils” under the directory where the software is installed.

The utility needed to convert trend buffers is called *TrendBufferReader.exe*. The *TrendBufferReader.exe* utility can be invoked using a batch file with the following syntax:

```
TrendBufferReader -r Trend1 Trend1.csv 1
```

where Trend1 is the name of the trend buffer without extension as dumped (original file name is *trend1.dat*) and *Trend1.csv* is the name desired for the output file.

The resulting CSV file has 5 columns with the following meaning:

”Data Type“, “Value”, “Timestamp (UTC)”, “Sampling Time (ms)”, “Quality”

Column Name	**Description**
DataType	A code that informs about the data type of the sampled tag according to the following codes: 0: Empty 1: Boolean 2: Byte 3: Short 4: Int 5: Unsigned Byte 6: Unsigned Short 7: Unsigned Int 8: Float 9: Double
Value	Value of the sample.
Timestamp(UTC)	Timestamp in UTC format.
SamplingTime(ms)	Sampling interval time in milliseconds.
Quality	Informs about the tag value quality. The information is coded according the OPC DA standard; the information is stored in a byte data (8 bits) currently defined in the form of 3 bit fields: “Quality”, “Sub status” and “Limit status”. The 8 quality bits are arranged as follows: **QQSSSSLL**

For a complete and detailed description of all the single fields, please refer to the OPC DA official documentation. Here below we report the most commonly used quality values returned by the HMI acquisition engine:

QUALITY CODE	QUALITY	DESCRIPTION
0	BAD	The value is bad but no specific reason is known
4	BAD	There is some server specific problem with the configuration. For example the tag its question has been deleted from the configuration file (tags.xml).
8	BAD	This quality may reflect that no value is available at this time, for reasons like the value may have not been provided by the data source.
12	BAD	A device failure has been detected.
16	BAD	Timeout occurred before device responded.
24	BAD	Communications have failed.
28	BAD	There are no data found to provide upper or lower bound value (trend interface specific flag).
32	BAD	No data have been collected (i.e. archiving not active. Trend interface specific flag). When the HMI return online after a reboot or from a condition where sampling stopped, a sample with quality value 32 is added to indicate this temporary offline status.
64	UNCERTAIN	There is no specific reason why the value is uncertain.
65	UNCERTAIN	There is no specific reason why the value is uncertain. (The value has ‘pegged’ at some lower limit)
66	UNCERTAIN	There is no specific reason why the value is uncertain. (The value has “pegged” at some high limit.)
67	UNCERTAIN	There is no specific reason why the value is uncertain. (The value is a constant and cannot move.)
84	UNCERTAIN	The returned value is outside the limits defined for it. Note that in this case the “Limits” field indicates which limit has been exceeded but the value can move farther out of this range.
85	UNCERTAIN	The returned value is outside the limits defined for it. Note that in this case the “Limits” field indicates which limit has been exceeded but the value can move farther out of this range. (The value has “pegged” at some lower limit.)
86	UNCERTAIN	The returned value is outside the limits defined for it. Note that in this case the “Limits” field indicates which limit has been exceeded but the value can move farther out of this range. (The value has “pegged” at some high limit.)
87	UNCERTAIN	The returned value is outside the limits defined for it. Note that in this case the “Limits” field indicates which limit has been exceeded but the value can move farther out of this range. (The value is a constant and cannot move.)
192	GOOD	

DeleteTrend

This macro allows you to delete saved trend data from the file.

→ In macro properties, define the trend name from which you want to delete the trend logs.

DumpEventArchive

The DumpEventArchive macro is used to export the Historical Alarm log and Audit Trail data to external drives, such as a USB memory or SD card. If you use a USB drive the path will be \USBMemory or if you use an SD Card the path will be \Storage Card, followed by the specified folder in the memory.

NOTE The external drives plugged on the USB port of the panel must be FAT or FAT32 formatted. NTFS format is not supported.

→ In the macro properties, configure the event buffer name that you want to dump and the destination folder path. The DumpConfigFile property must be set to true when you plan to convert the dumped files to CSV.

DumpAsCSV: If this option is set true, then files will be directly dumped in the location in **.CSV* format.

If it is set as False, then the events archive is dumped in a binary format; an external utility is then required to convert it to a CSV format.

DateTimePrefixFileName: When this option is enabled then dumped file will have the date and time as prefix to the name of the file. For example: If you are making a dump at 10.10AM on 1-1-2012, then the file name will look like D2012_01_01_T10_10_alarmBuffer1.csv.
[DYear_Month_Day_THour_Minute_Filename]

This helps to know the time at which the dump was made and also to identify which one is the latest.

NOTE This option is only supported when exporting to CSV directly.

timeSpec

This option defines the time format used when dumping the event archive to file.

- Local: the time values exported are the time of the HMI device.
- Global: the time values exported are in the Coordinated Universal Time (UTC) format.

Example:

Local	2012-10-11T05:13:43.724-07:00
Global	2012-10-11T12:13:43.724Z

When exporting alarm buffers in binary format assuming the DumpConfigFile option is set to true (recommended settings) the result of the dump action execution are 2 folders. One is called “data“ and contains the data files. The 2. is called “config” and does contain the configuration files needed by the utility to reconstruct the entire information for proper conversion to CSV.

Once the two folders are copied from the root of the USB memory stick to the computer disk, the folder structure looks as follows:

```
.\config\
      alarms.xml
      eventconfig.xml
.\data\
      AlarmBuffer1.dat
      AlarmBuffer1.inf
.\
AlarmBufferReader.exe
```

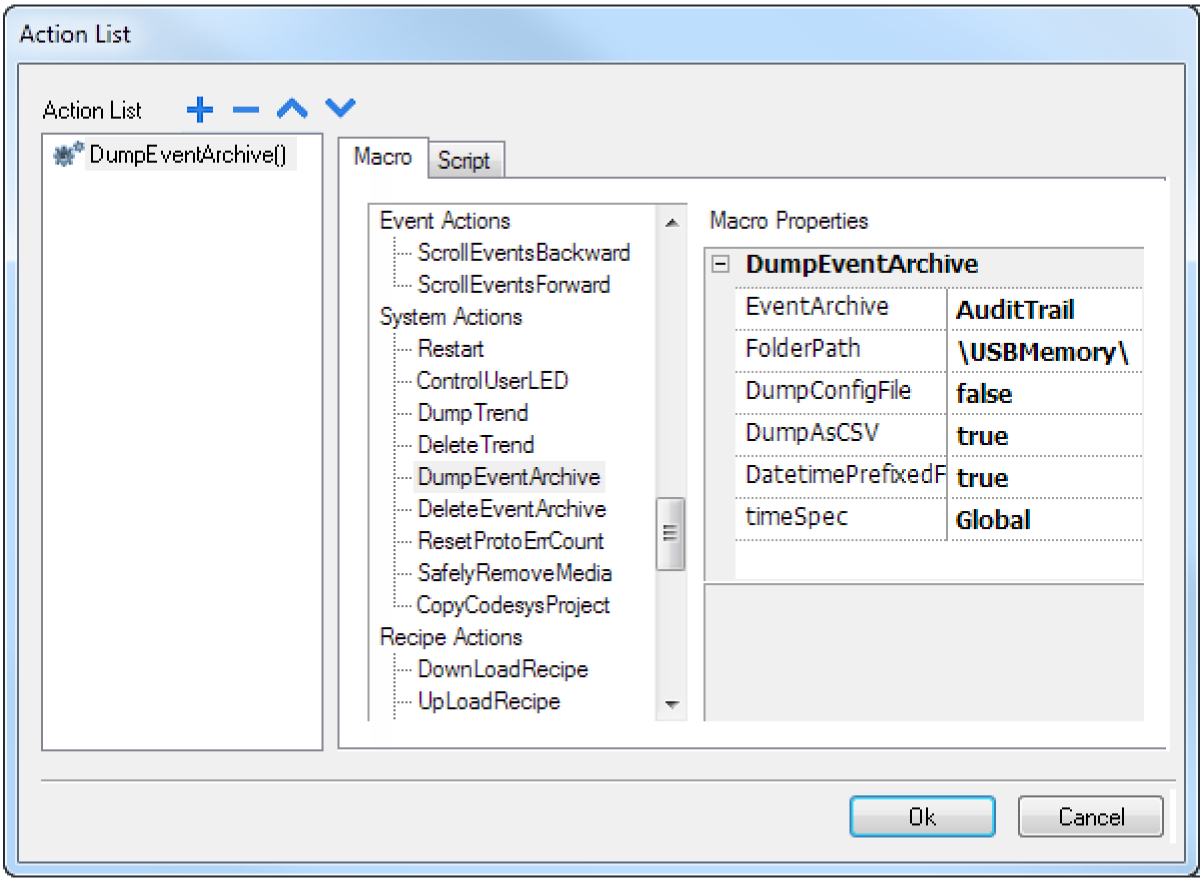
NOTE

The utility is distributed in Panel Builder 600 Suite under the “..\ABB\ Panel Builder 600 Suite\Utils”.

The AlarmBufferreader can be called from command line with the following syntax:

```
AlarmBufferReader AlarmBuffer1 FILE ./AlarmBuffer1.csv
```

AlarmBuffer1 is the name of the dumped *.dat* file without extension and *AlarmBuffer1.csv* is the desired output file name.



Macro Properties

DumpEventArchive	
EventArchive	AuditTrail
FolderPath	\USBMemory\
DumpConfigFile	false
DumpAsCSV	true
DatetimePrefixedF	true
timeSpec	Global

The utility called *AuditTrailBufferReader.exe* has to be used for audit trail buffers.

NOTE

The macro action has to be configured with the option **DumpConfigFile** set to true.

The result of the dump is a directory structure similar to the one generated for events.

The utility can be called from the command line according to the following syntax:

AuditTrailBufferReader AuditTrail FILE ./AuditTrail.csv

AuditTrail is the name of the dumped buffer without extension and *AuditTrail1.csv* is the desired output file name.

DeleteEventArchive

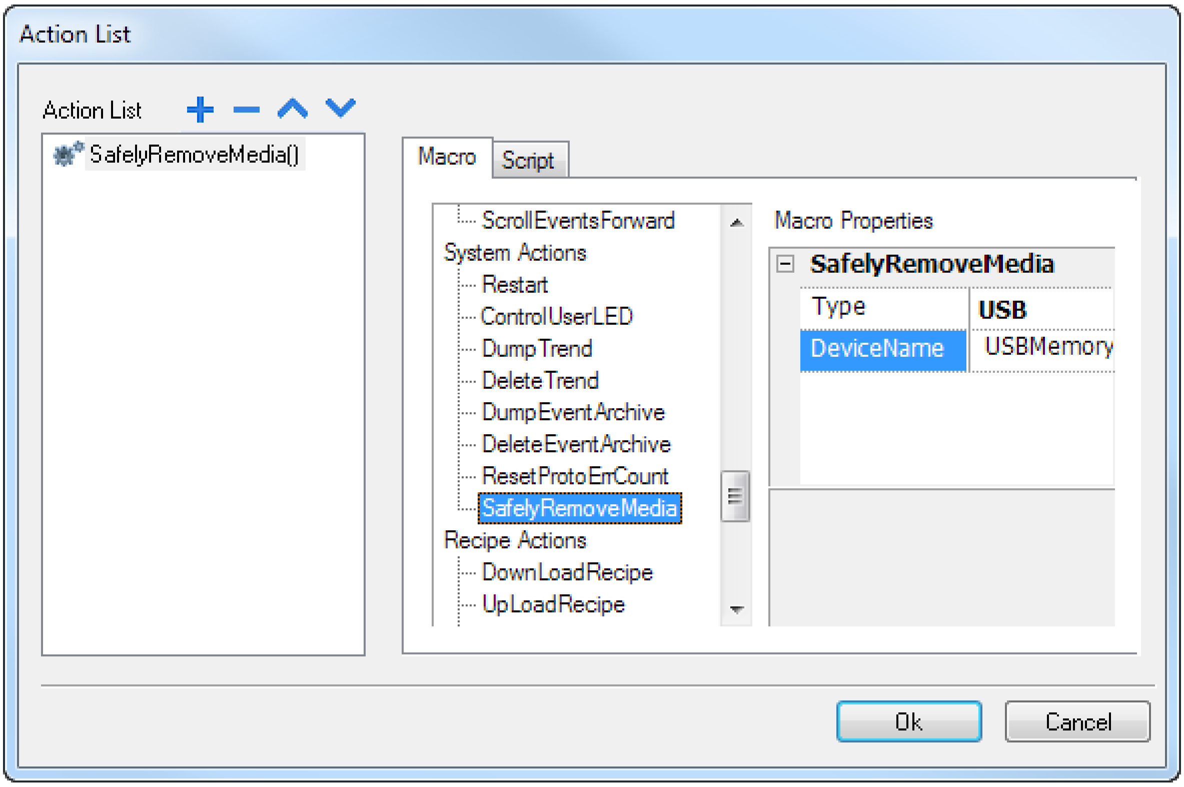
This macro allows you to delete saved event buffers log data from the file. In macro properties, define the event buffer name that you want to delete from the event logs.

ResetProtoErrCount

This macro is used to reset the protocol error count on the system variables. See the chapter “System Variables“ for further information about system variables.

SafelyRemoveMedia

If you unplug an SD Card or a USB drive from the HMI while it's transferring or saving information, you might risk losing some information. This macro provides a way to help you safely remove such devices.



SafelyRemoveMedia

Type	USB
DeviceName	USBMemory

Recipe Actions

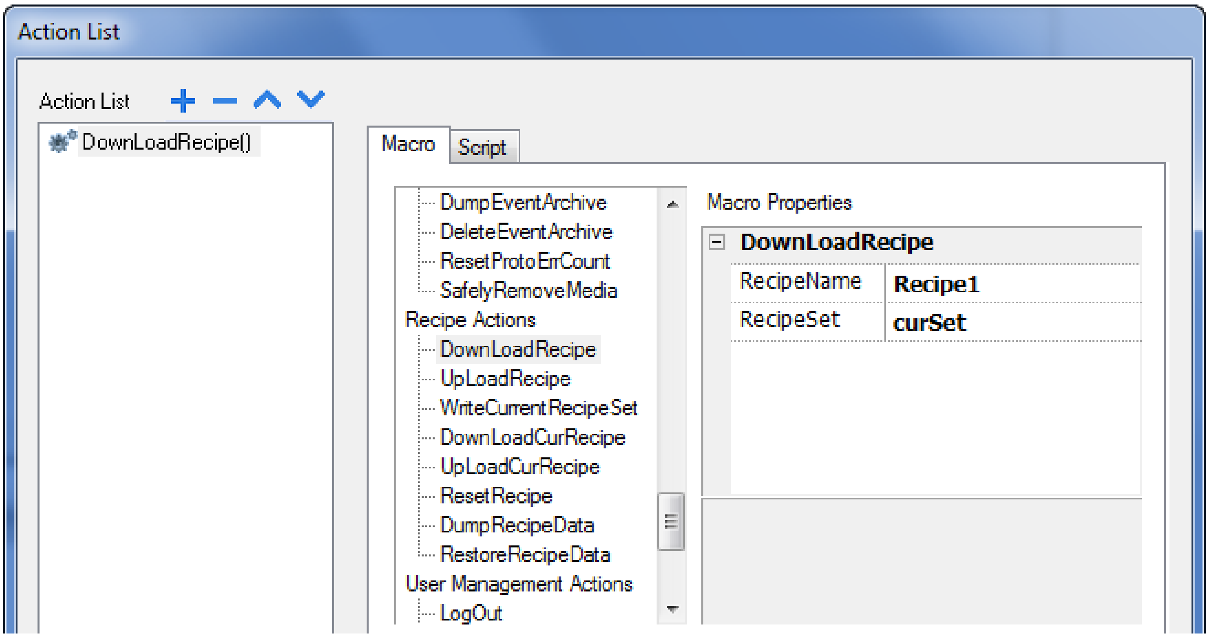
These macros are used in recipes.

DownLoadRecipe

This macro allows you to transfer the set of recipe data to controller tags.

→ In macro properties, select the recipe in the recipe name field and select the recipe set you want to download.

→ To download a selected recipe set, select **curSet** in the recipe set.



⊟ **DownLoadRecipe**

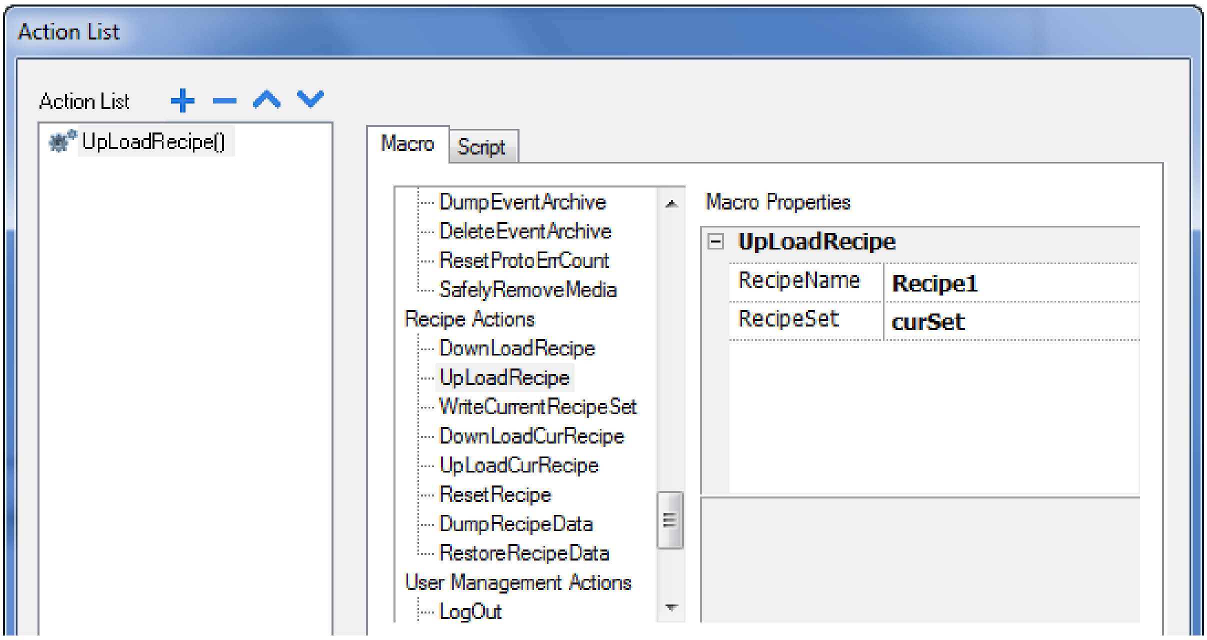
RecipeName	**Recipe1**
RecipeSet	**curSet**

Upload Recipe

This macro allows you to transfer the set of controller data to the recipe data.

→ In macro properties, select the recipe in the recipe name and select the recipe set you want to upload.

→ To upload a selected recipe set, select **curSet** in the recipe set.



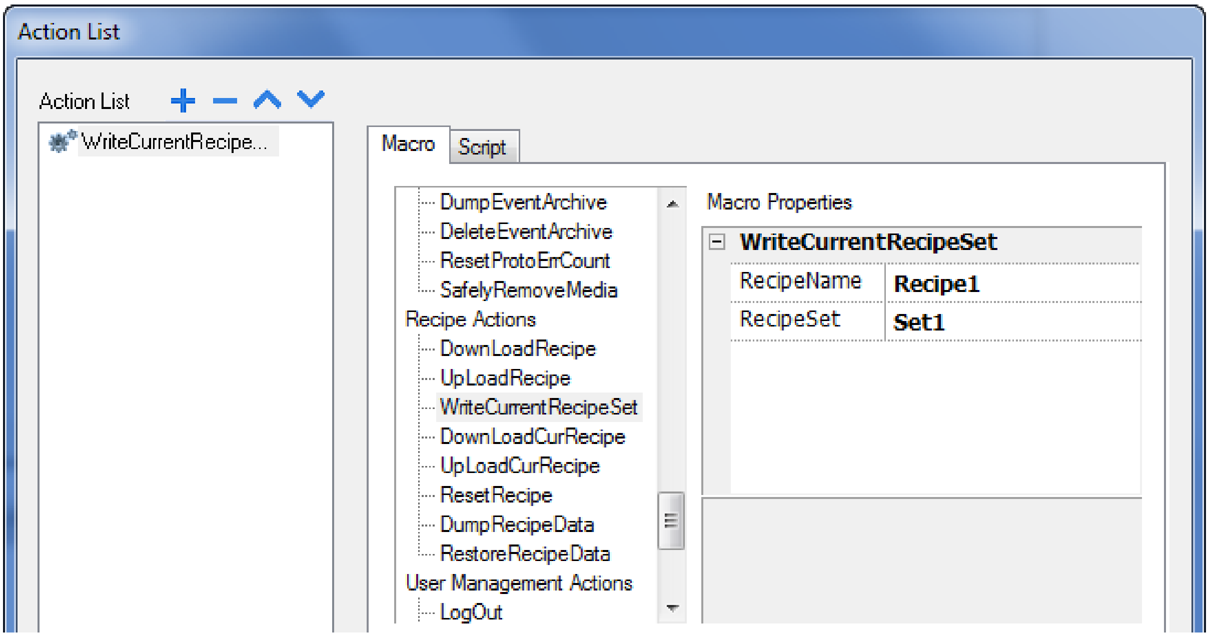
⊟ **UpLoadRecipe**

RecipeName	**Recipe1**
RecipeSet	**curSet**

WriteCurrentRecipeSet

This macro allows you to set the selected recipe as current recipe set.

→ In macro properties, select the recipe and recipe set you want to set as the current recipe in PB610 Panel Builder 600 Runtime.

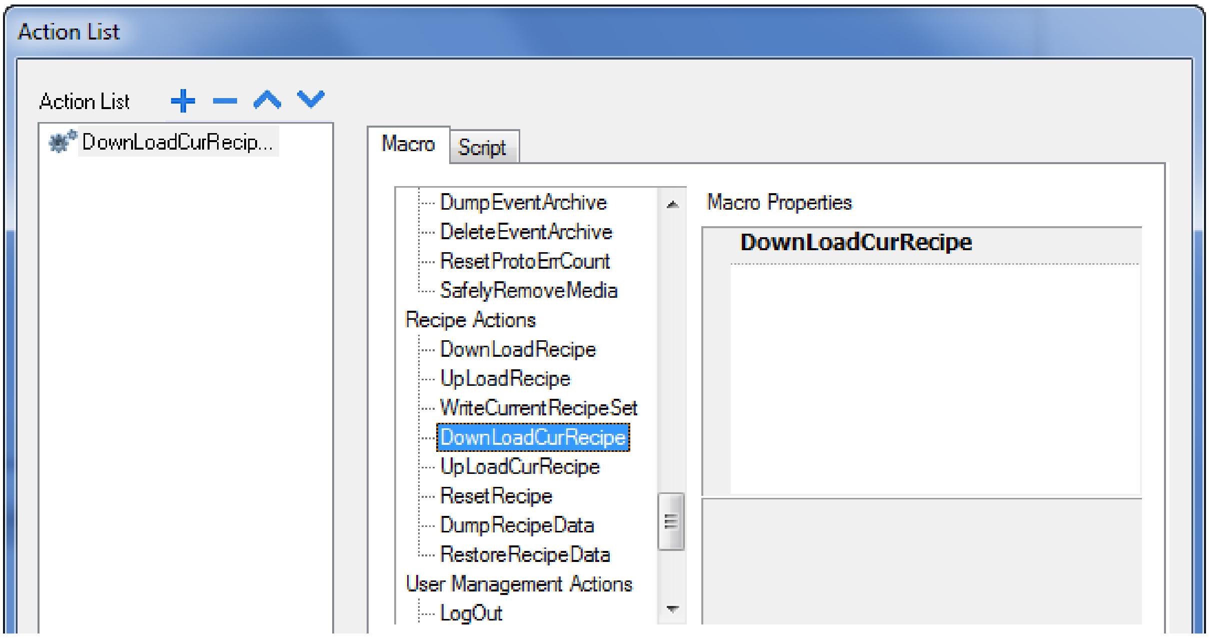


WriteCurrentRecipeSet

RecipeName	Recipe1
RecipeSet	Set1

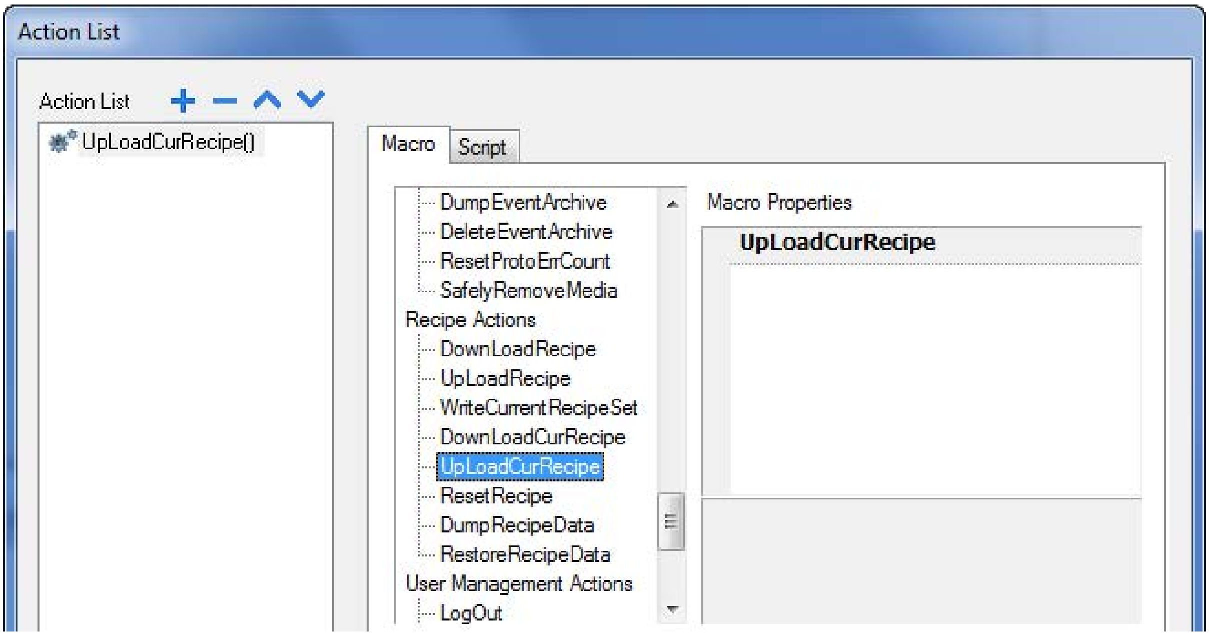
DownLoadCurrentRecipe

This macro allows you to transfer the set of recipe data to the controller. No parameter is required to set this in the macro parameters. This will download the currently selected recipes and recipe set to the controller.



UploadCurrentRecipe

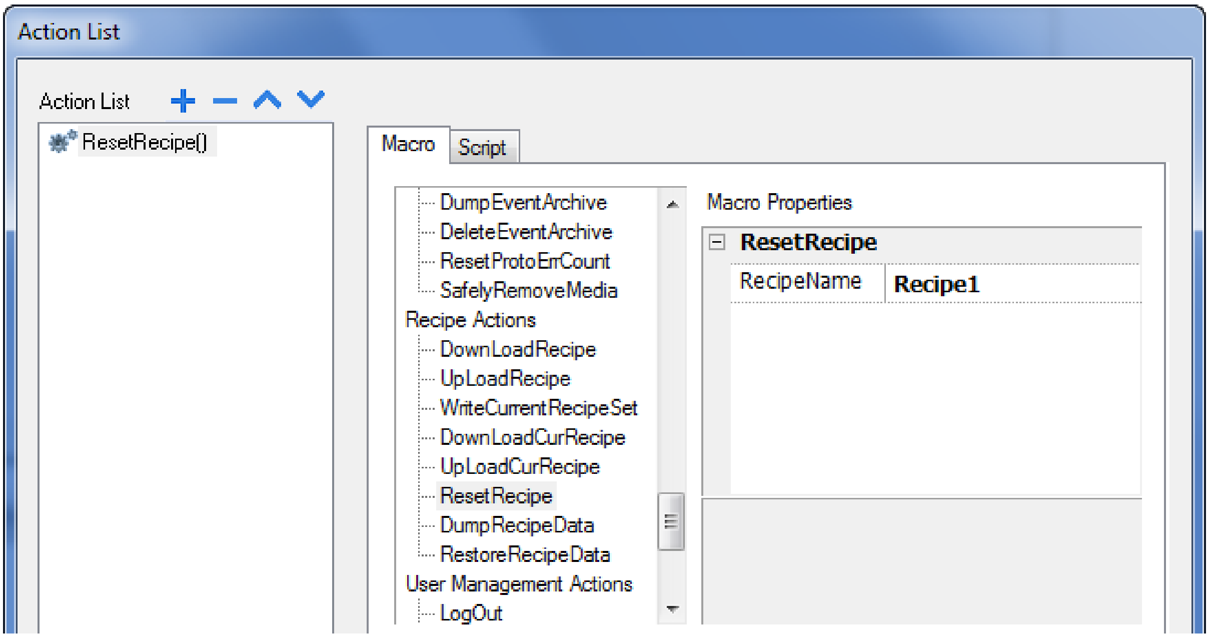
This macro allows you to transfer the set of controller tags data to recipes. No parameter is required to set this in the macro parameters. This will upload the currently selected recipes from the controller.



ResetRecipe

This macro allows you to restore the factory settings for the recipe data. The uploaded recipes will be replaced with the original recipe data.

→ In the macro property, select the recipe you want to reset to factory settings.



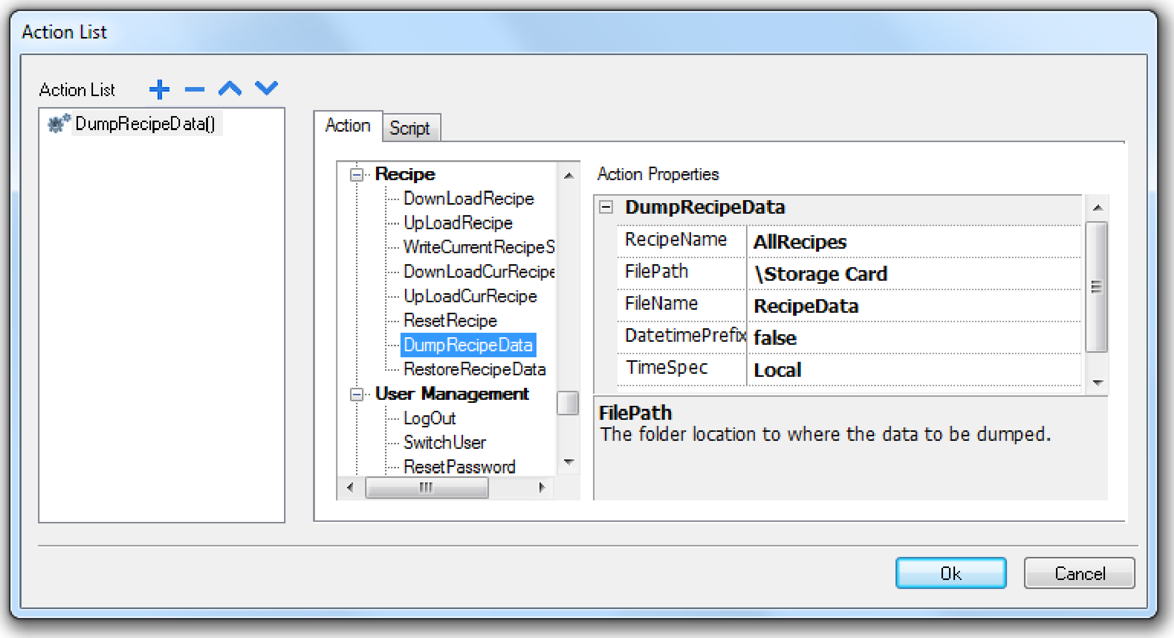
RecipeName Recipe1

DumpRecipeData

This macro is used to dump recipes to internal or external storages.

→ In macro properties, define the file location where to save the dumped file.

➤ Recipe data is saved in CSV format.



DumpRecipeData

RecipeName	AllRecipes
FilePath	\Storage Card
FileName	RecipeData
DatetimePrefix	false
TimeSpec	Local

FilePath
The folder location to where the data to be dumped.

NOTE	The external drives plugged on the USB port of the panel must be FAT or FAT32 formatted. NTFS format is not supported.

DateTimePrefixFileName: When this option is enabled then the dumped file will have the date and time as prefix of the filename.
For example: If you are making a dump at 10.10AM on 1-1-2012, then the file name will look like: D2012_01_01_T10_10_recipe1.csv.
[DYear_Month_Day_THour_Minute_Filename]

This helps to know the time at which the dump was executed and also to identify which one is the latest.

TimeSpec define time format, **Local** for HMI time and **Global** for UTC time.

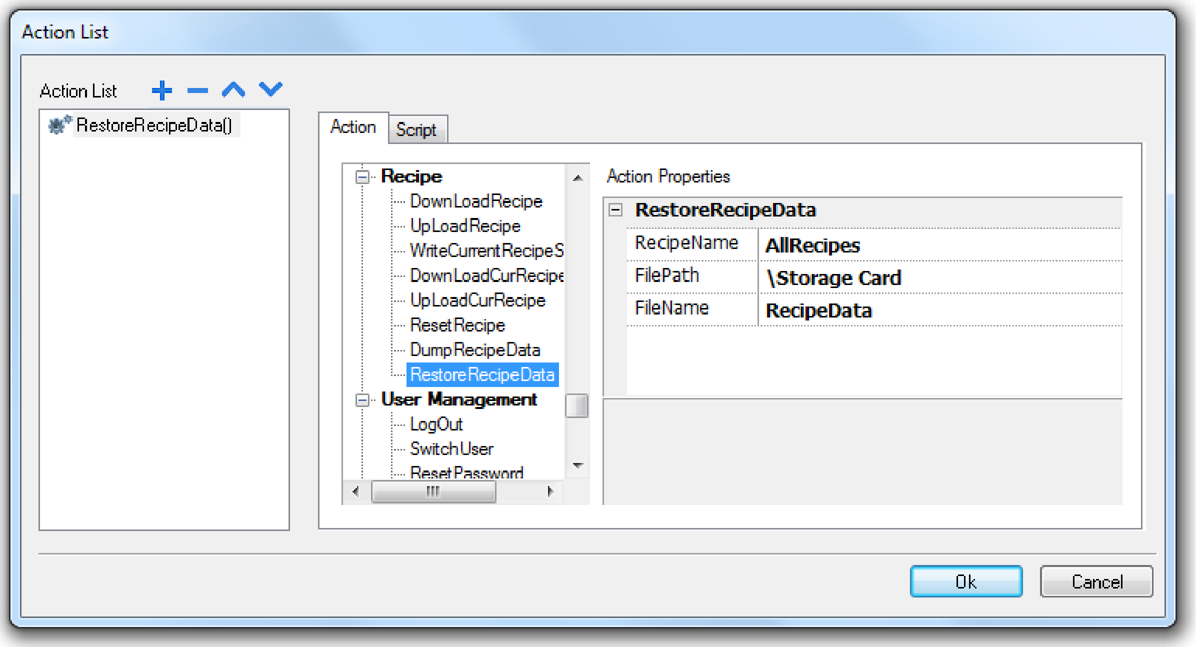
RestoreRecipeData

This macro allows you to restore the recipe data previously.

→ In macro properties, provide the file full path of the recipe filesRecipes to restore can be in any external storage like USB, SD or network paths.

NOTE	The external drives plugged on the USB port of the panel must be FAT or FAT32 formatted. NTFS format is not supported.

Example: The figure shows the case of an external USB memory stick.



RestoreRecipeData

RecipeName	AllRecipes
FilePath	\Storage Card
FileName	RecipeData

User Management Actions

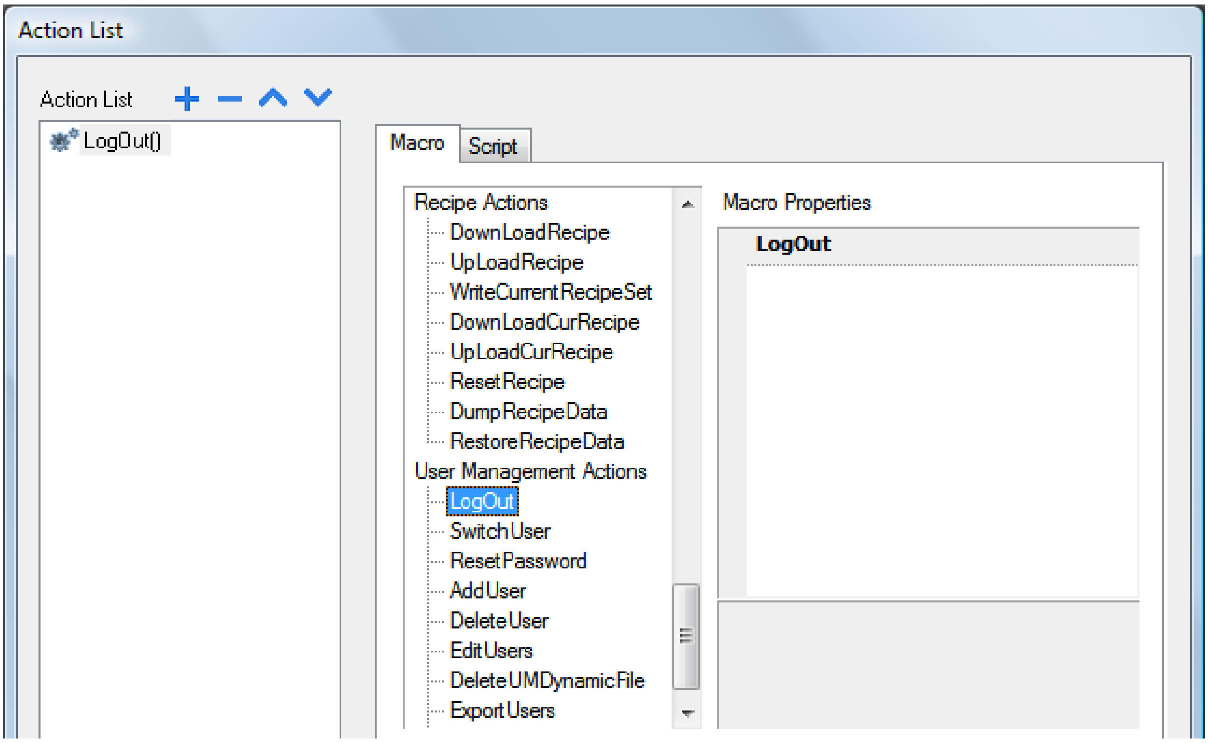
These macros can be utilized for user management and security settings in PB610 Panel Builder 600 Runtime.

LogOut

This macro allows you to log off the current user in Runtime.

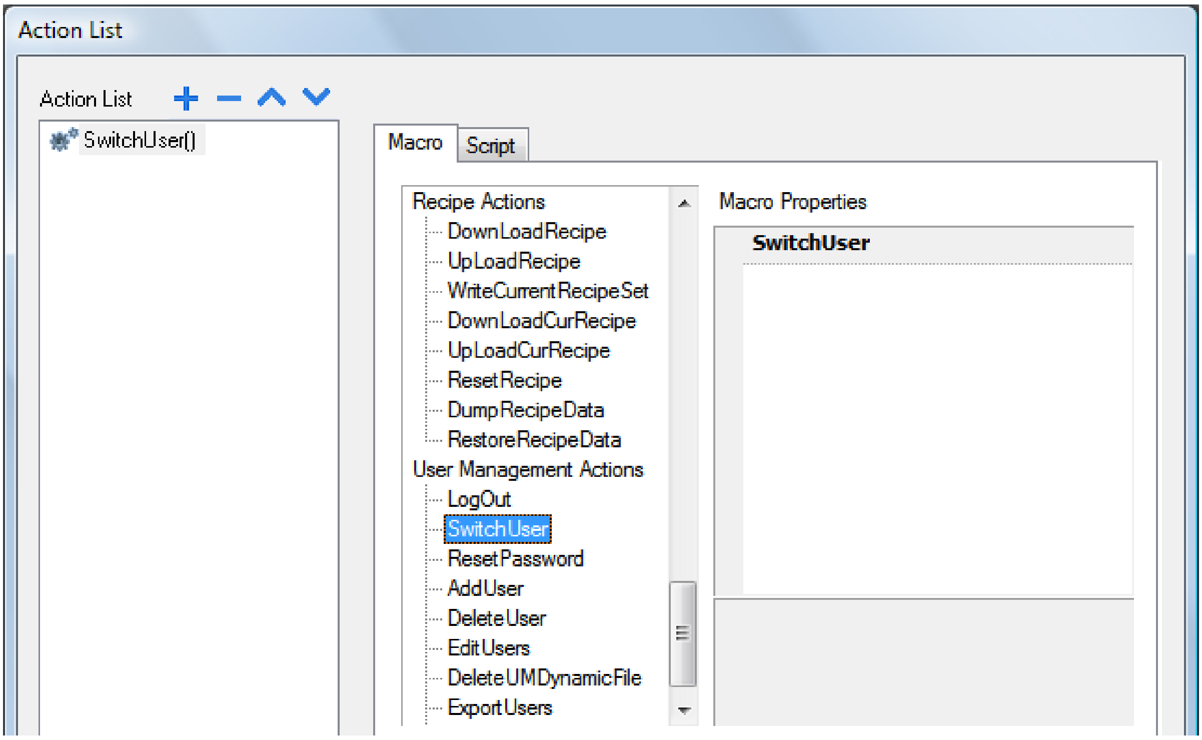
After executing the Logout macro, the HMI behavior depends on the fact that a Default user is configured in the project or not.

If there is a Default user, the Logout automatically logs in the Default user. If there is not a Default user or you log out from the Default user, then the log in screen is shown.

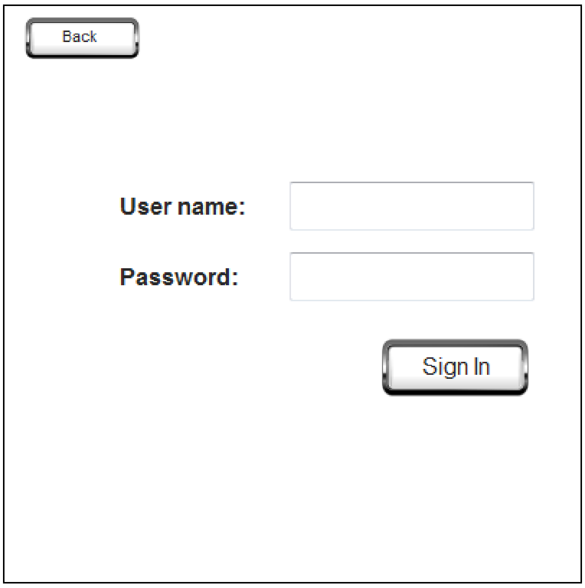


SwitchUser

This macro allows you to switch between two users without logging out the logged-in user. The server continues running with the previously logged-in user, until the next user logs in. This means, after executing the *Switch User* macro, the Runtime will display the user login template. Internally, the server runs with the previously logged-in user. This action is useful for ensuring that there is always one user logged onto the system.

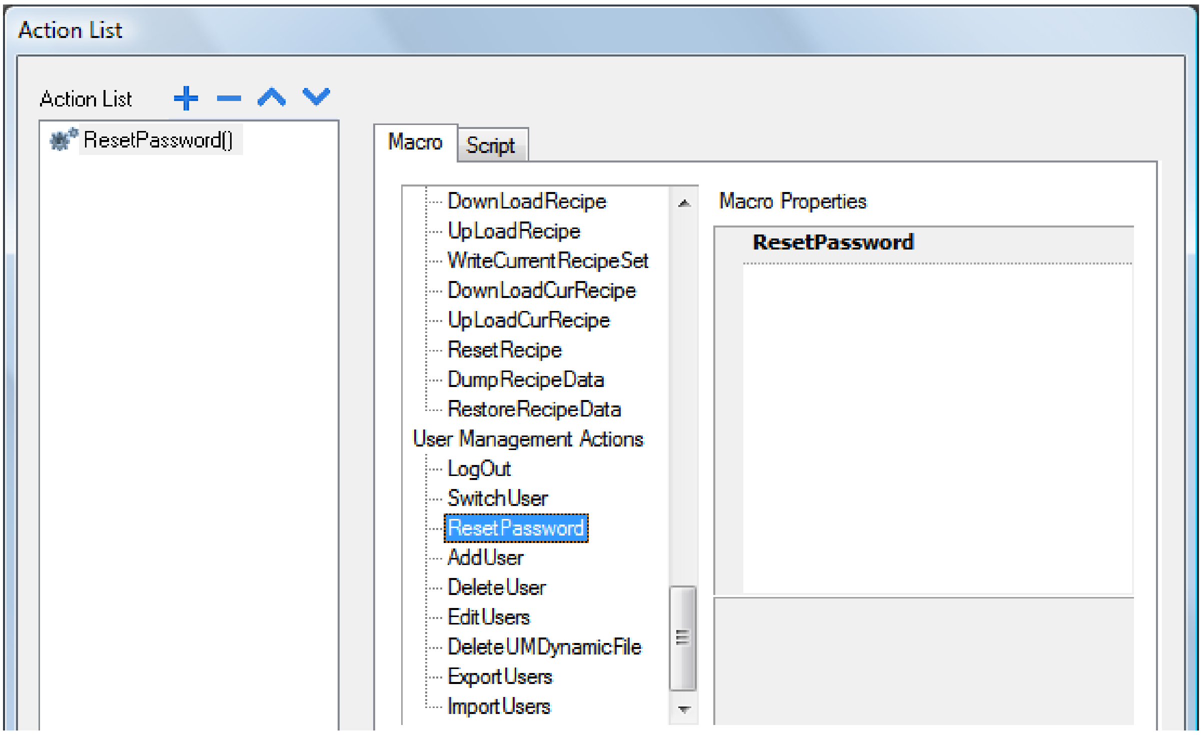


→ Click on the **Back** button to go back to the previously logged-in user.



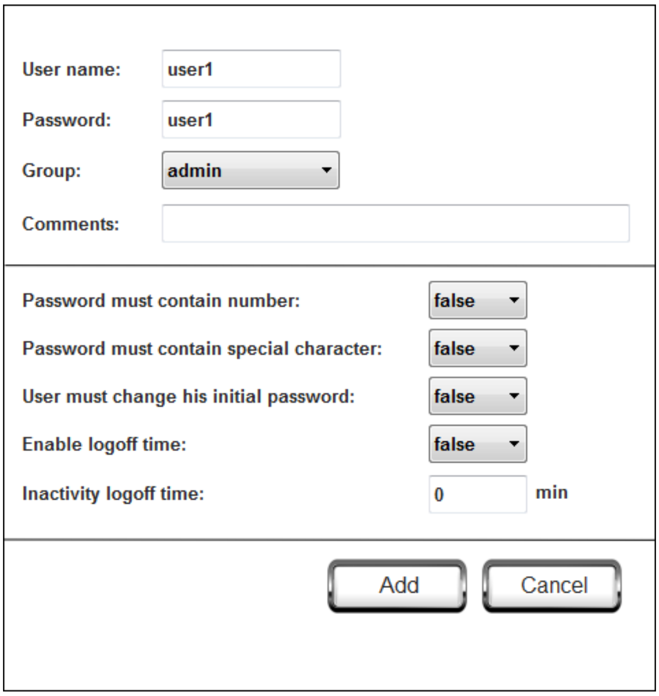
ResetPassword

This macro allows the current user to restore his or her original password; this macro will restore factory settings for the current user's password. No parameter is required to set this macro.



AddUser

This macro is used to add users at Runtime. When this macro is executed, a template page pops up, where parameters for the user can be set. The parameters include username, password, group, comments and the following flags: “password must contain numbers”, “password must contain special character”, “user must change his initial password”, “enable logoff time”, “inactivity logoff time”.



User name: user1

Password: user1

Group: admin

Comments:

Password must contain number: false

Password must contain special character: false

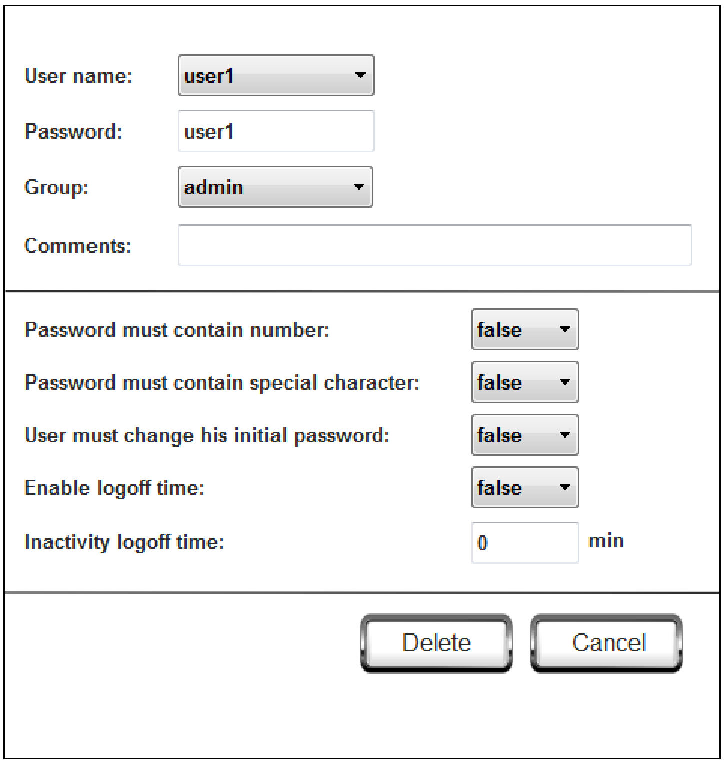
User must change his initial password: false

Enable logoff time: false

Inactivity logoff time: 0 min

DeleteUser

This macro is used to delete users in PB610 Panel Builder 600 Runtime. Upon executing this macro, a template page will pop up where you can select the user you wish to delete. No parameters are required to set this macro. After executing the macro, the Delete User form will be displayed, as shown in the figure below.



User name: user1

Password: user1

Group: admin

Comments:

Password must contain number: false

Password must contain special character: false

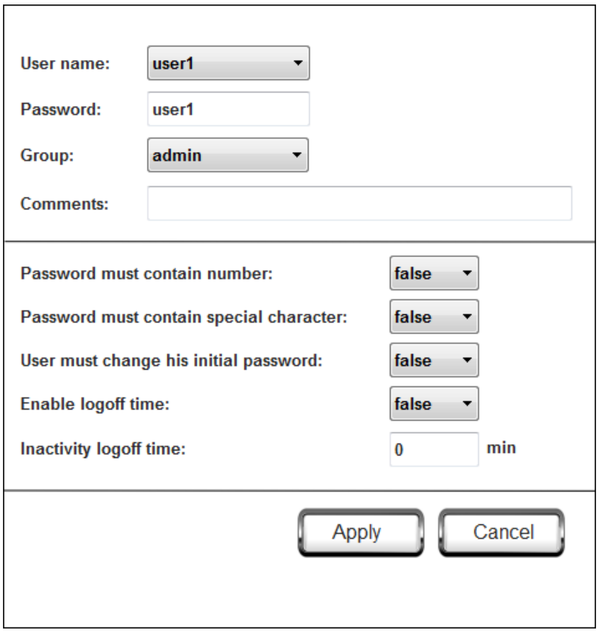
User must change his initial password: false

Enable logoff time: false

Inactivity logoff time: 0 min

EditUsers

This macro is used to edit users in PB610 Panel Builder 600 Runtime. When executing this macro, a template page pops up. Here you can select a user and modify this user’s parameters. The parameters include username, password, group, comments and the following flags: “password must contain numbers”, “password must contain special character”, “user must change his initial password”, “enable logoff time”, “inactivity logoff time”. After executing the macro, a User Edit form will pop up, as shown in the figure below.



User name: user1

Password: user1

Group: admin

Comments:

Password must contain number: false

Password must contain special character: false

User must change his initial password: false

Enable logoff time: false

Inactivity logoff time: 0 min

DeleteUserManagementDynamicFile

The DeleteUMDynamicFile macro allows you to delete the dynamic user management file. This means that the users created, edited, or deleted in Runtime will be erased, and the server will restore the settings from the project, originally downloaded from PB610 Panel Builder 600. No macro properties are required.

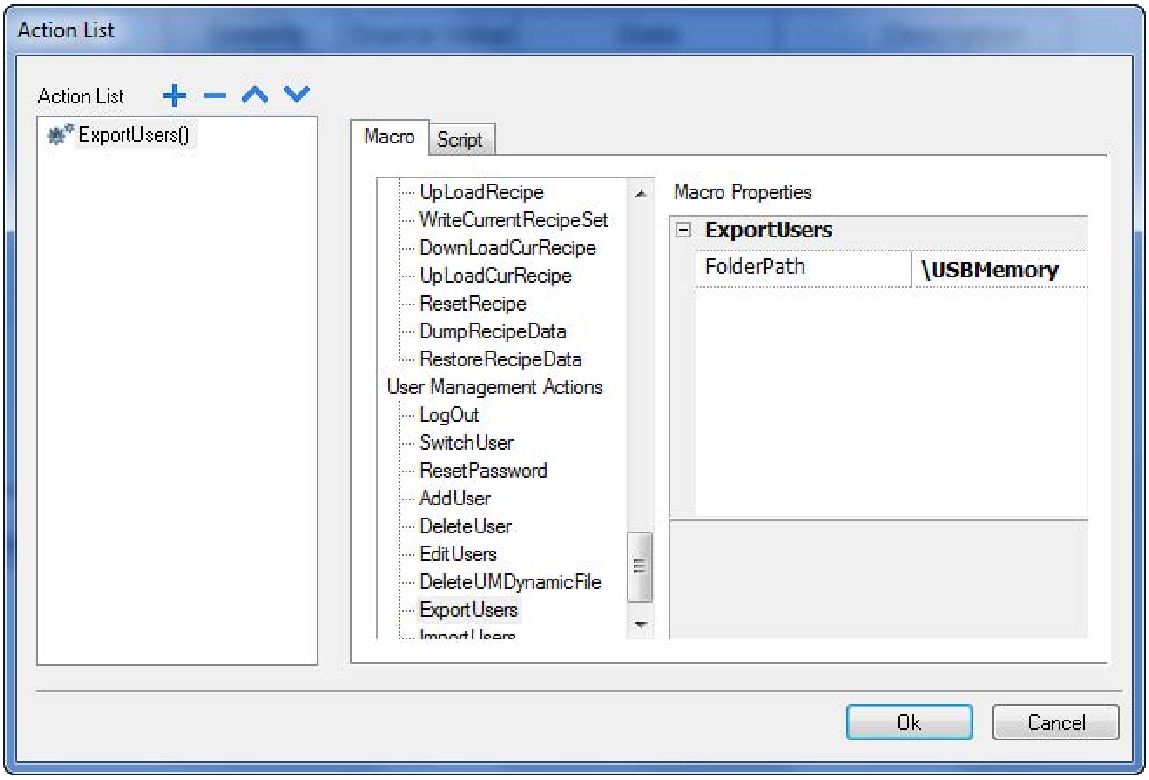
ExportUsers

This macro allows you to export user details to an xml file *usermgnt_user.xml*. User details will be in an encrypted form. In this macro property, the destination folder path must be set to the location where the *usermgnt_user.xml* file is saved. If using a USB memory stick plugged in to the USB port, the path will be *\USBMemory*, followed by the specified folder in the memory (or left empty for root folder).

NOTE

The external drives plugged on the USB port of the panel must be FAT or FAT32 formatted. NTFS format is not supported.

Since the file is encrypted, there is no way to edit the user configuration from this exported file. This action is most useful for making a backup to be used for a later restore (see next chapter).



ExportUsers

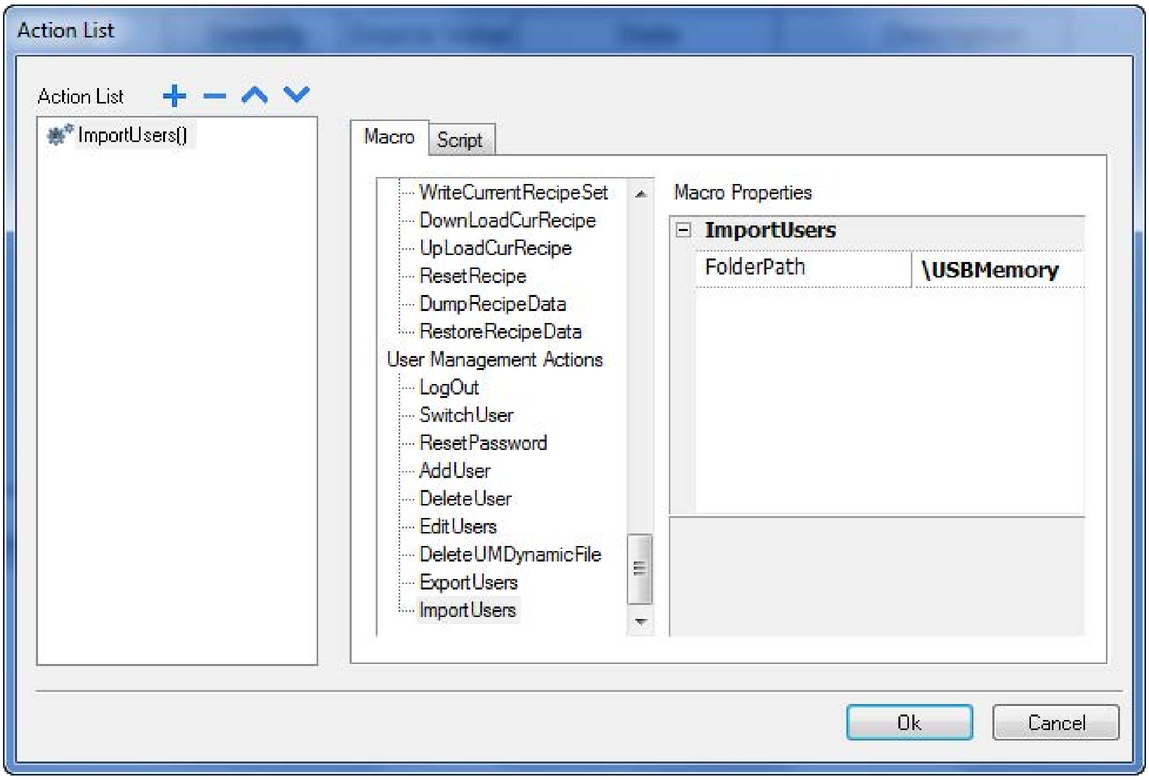
FolderPath \USBMemory

ImportUsers

This macro allows you to import user details from an xml file *usermgnt_user.xml*. The folder path where the *usermgnt_user.xml* file is located must be specified within the macro properties. If using a USB memory stick plugged into the USB port, the path will be *\USBMemory*, followed by the specified folder in the memory (or left empty for root folder).

NOTE

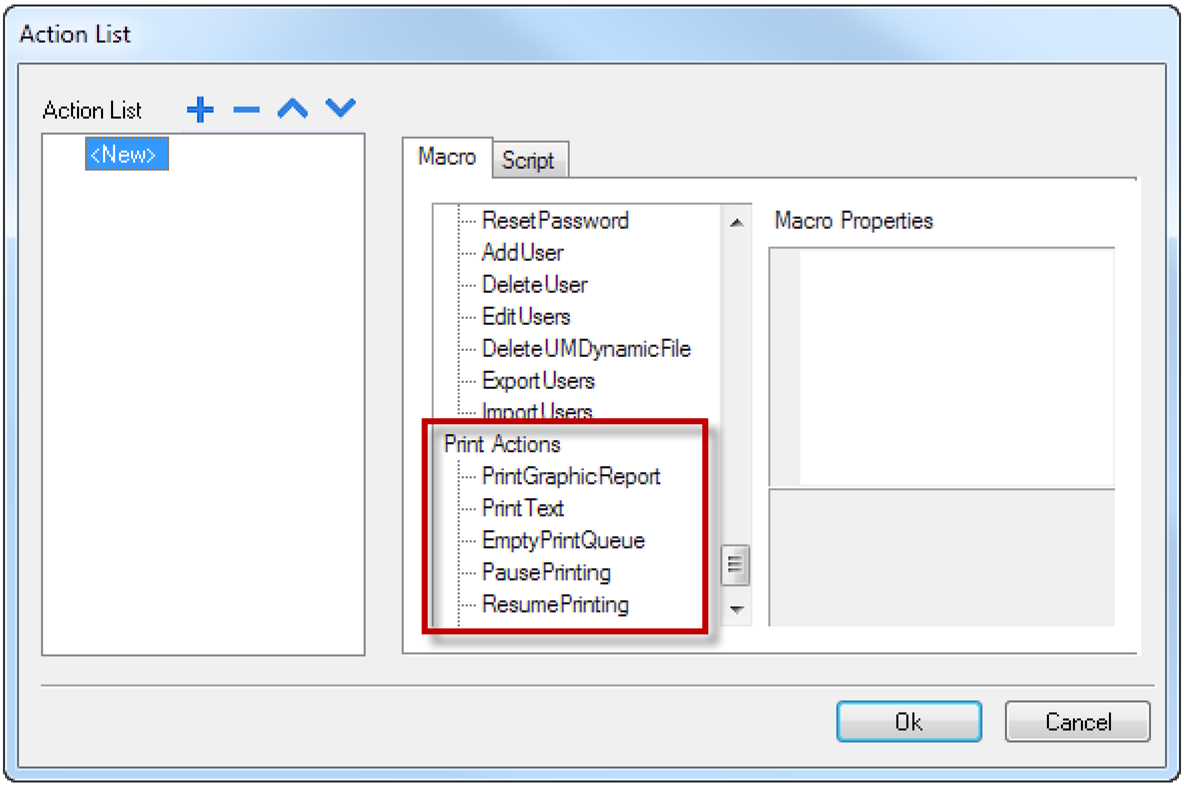
The external drives plugged on the USB port of the panel must be FAT or FAT32 formatted. NTFS format is not supported.



ImportUsers

FolderPath	\USBMemory

Print Actions



PrintGraphicReport

The PrintGraphicReport macro allows you to print a graphic report. You have to specify the report name in the combo box reportName. The option silent (default value is true), if set to false, allows you to open a dialog at runtime which asks the user to adjust printer properties.

PrintText

The PrintText macro allows you to print the string written in the field text. The option silent (default value is true) allows, if set to false, you to open a dialog at runtime which asks the user to adjust printer properties.

NOTE

PrintText requires a printer supporting line printing.
PrintText works in line printing mode using a standard protocol common to all printers that support it. No custom drivers required for line printing.

NOTE

In line printing, text is printed immediately line by line or after a timeout custom for each printer model (could take also minutes for some models not design for line printing).

EmptyPrintQueue

The EmptyPrintQueue macro allows you to empty the current printing queue. If the macro is executed in the middle of the execution of a job, then the queue will be cleared at the end of the job.

PausePrinting

The PausePrinting macro allows you to put on hold the current printing queue. If the macro is executed in the middle of the execution of a job, then the queue is paused at the end of the job.

ResumePrinting

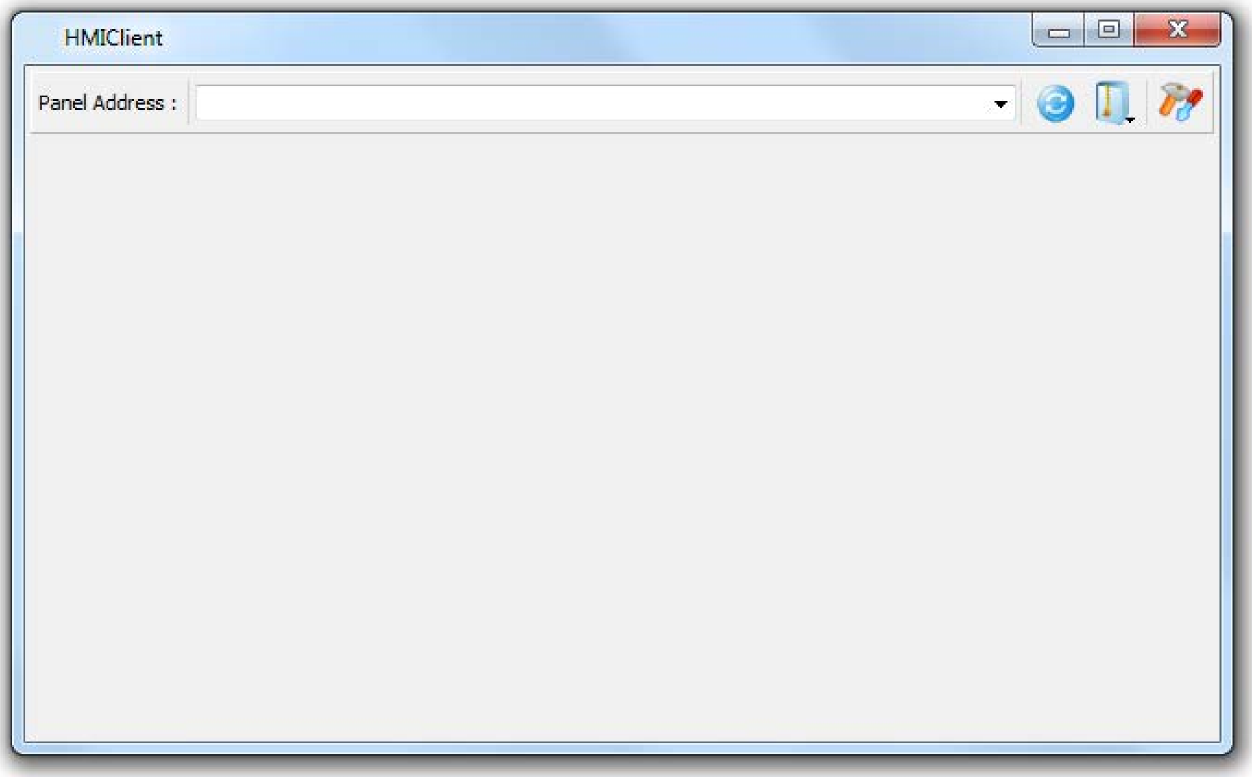
The ResumePrinting macro allows you to start the queue if previously it was put on hold.

AbortPrinting

The AbortPrinting macro allows you stop the execution of the current job and remove it from the queue. If the queue has another job, then, after aborting, the next one starts immediately.

Using Windows Client

The Windows Client provides remote access to the Runtime, and is included in the Panel Builder 600 installation. The Windows Client consists of a simple standalone application; although it uses the same graphic rendering system as the server, it relies on a specified Runtime as Server for live data.



Windows Client for Windows is available in the Runtime folder of the Panel Builder 600 root folder. Execute the Windows Client application from the Runtime folder or from the start up menu (Panel Builder 600 - Windows Client). The client will open in a browser-like style window. Type the server IP address (the panel's IP address) in the address bar (for example: http://192.168.1.12). The Client will connect to the server and the same graphical application running on the Server panel will be loaded in the client window.

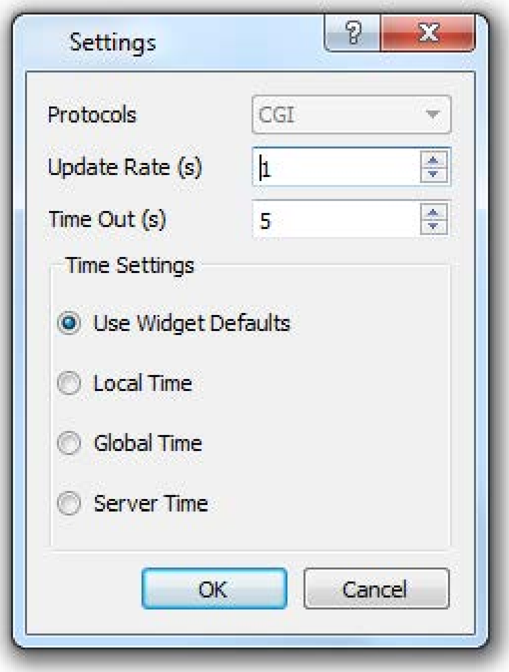
Windows Client acts as a remote client and communicates to the server, sharing the local visualization with those Tag values that are maintained or updated by the communication protocol.

The HMI projects contain properties that let you know which page is currently displayed on the HMI and to force the HMI to switch to a specific page. These properties can be used to synchronize pages showed on the HMI and Windows Client or to control an HMI with a PLC. Please refer to SyncOptions for more details.

Time Zone Options for Windows Client

Starting from version V1.60 the Windows Client provides an additional option to handle the visualization of the timestamp information of a project.

From the "Settings" dialog you now have access to a set of new options shown in the next figure:



Protocols: CGI

Update Rate (s): 1

Time Out (s): 5

Time Settings

● Use Widget Defaults

○ Local Time

○ Global Time

○ Server Time

Update Rate (default 1 second)

Polling frequency used by Windows Client to synchronize data from server.

Time Out (default 5 seconds)

Max time wait from Windows Client before considering a request lost and repeating it.

The **Time Settings** information is used by the client to adapt the widget timestamp information according to the desired behavior.

Option for Time Settings	Explanation
Use Widget Defaults	For each widget use time information according to the widget settings provided at the time of programming.
Local Time	Translates all timestamps used in the project into the PC local time where the client is installed.
Global Time	Translates all timestamps used in the project into UTC format.
Server Time	Translates all timestamps used in the project into the same used by HMI panel/server in order to show the same time.

NOTE

This feature requires you to set the HMI RTC with the correct time zone and DST (Daylight Savings Time) options.

Using the Integrated FTP Server

The HMI runtime system features an integrated FTP server that can be used to get access to the internal flash disk data.

> **NOTE**
>
> Folders present on the Flash disk external to the runtime directory are not accessible via FTP; external USB drive and SD Storage Card are not accessible via FTP.

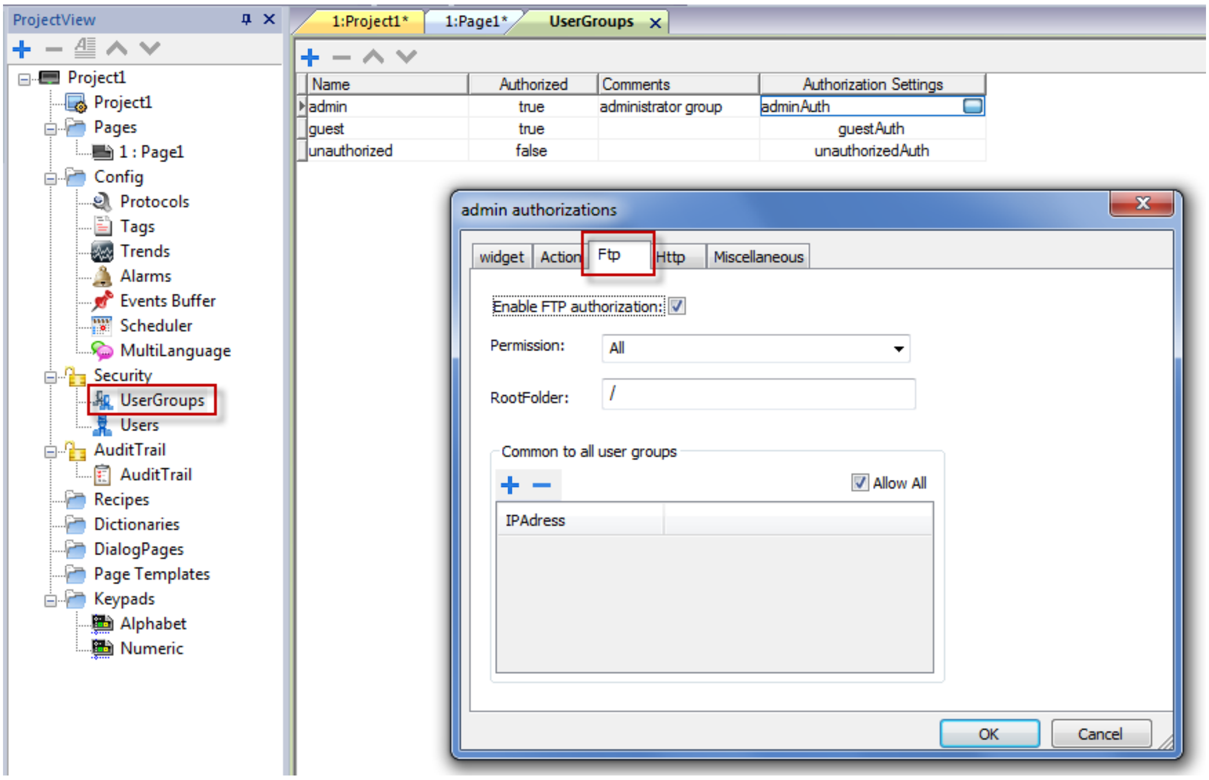
You can use any standard FTP client program to connect to the panel FTP server. The FTP server responds to the standard port 21 when using the IP address assigned to the panel as host.

> **NOTE**
>
> The server supports only ONE connection at a time; if you are using an FTP client which is configured to multiply the connections to the server in order to speed up the transfer operation, you will need to disable this feature in the client program or set the maximum number of connections per session to 1.

The FTP server is configured by default to accept incoming connection from the following accounts:

- User name: admin
- Password: admin

FTP permissions and account information can be changed from the “UserGroups” under the “Security” item of the project folder as shown in the following figure.



Name	Authorized	Comments	Authorization Settings
admin	true	administrator group	adminAuth
guest	true		guestAuth
unauthorized	false		unauthorizedAuth

Enable FTP authorization: ☑

Permission: All

RootFolder: /

Common to all user groups

☑ Allow All

IPAdress

Additional information can be found later in this document in the chapter titled “FTP Authorizations”.

Using ActiveX Client for Internet Explorer

In the standard distribution of Panel Builder 600, a Windows Client and an ActiveX Client is provided. ActiveX components are NOT installed by default to the Target devices, in order to save space in the flash memory.

Installing ActiveX

The ActiveX component is distributed with the Panel Builder 600 installation package. The related files are located in the Runtime folder of the Panel Builder 600 installation directory. The files, “HMIAX.cab" and “HMIClientAX.html", should be copied into the workspace folder of the Target device, where the Runtime is installed. The file copy can be done using the panel FTP server.

Starting from v1.90 of Panel Builder 600 has been introduced software **plug-in** support (ref. to chapter on software Plug-ins chapter for more details) to simplify ActiveX installation. Just enable ActiveX plug-in from project properties and install/update runtime to add ActiveX files to the runtime and transfer it into the target without the need of manual copy of it via FTP.

NOTE

This ActiveX requires Microsoft Visual C++ 2008 Redistributable Package (x86) installed on your system. You may need to download the Microsoft Visual C++ 2008 Redistributable Package (x86) from the Microsoft web site.

NOTE

The ActiveX plug-in require about 10MB of space. Enable it only if required by the HMI application to keep the smallest footprint for the application.

HTTP Access to ActiveX files

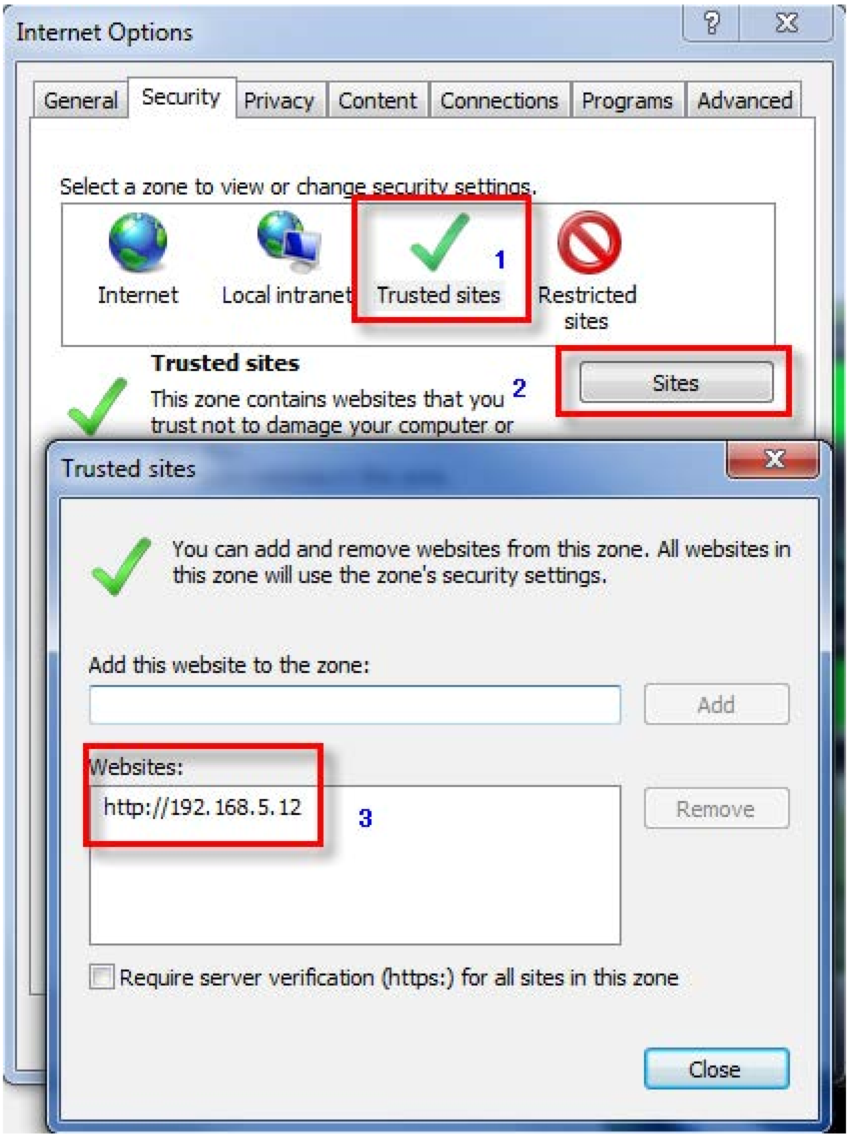
When security is enabled, ActiveX files “HMIAX.cab” and “HMIClientAX.html” have to be accessible from the http server embedded into the runtime. Refer to **HTTP Authorizations** chapter for more details.

Internet Explorer Settings

Internet Explorer settings must be changed, adding the panel's IP to the list of the trusted sites.

In Tools – Internet Option Security tab choose, "Trusted sites". Then click on the "Sites" button.

Type in the IP address of the Target device, the location where the ActiveX component has been installed and it will be loaded to the browser.



You can add and remove websites from this zone. All websites in this zone will use the zone's security settings.

Add this website to the zone:

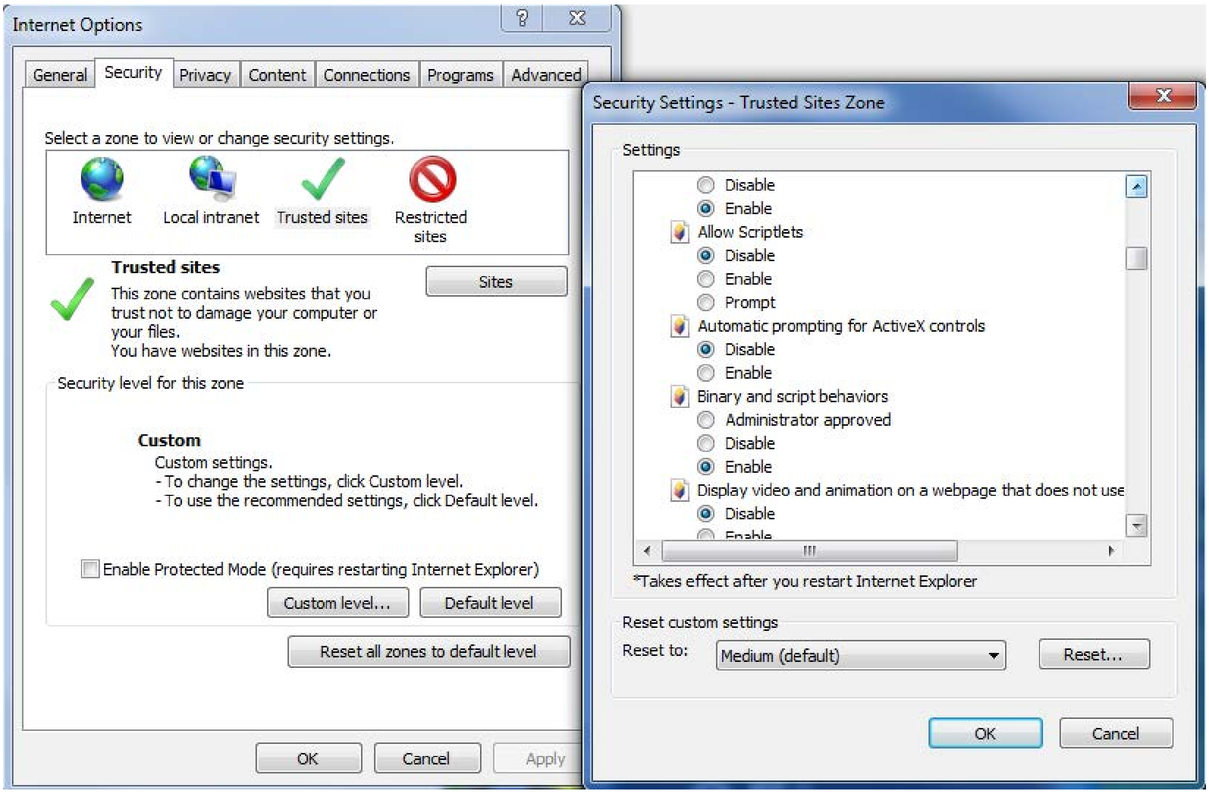
Websites:

http://192.168.5.12

□ Require server verification (https:) for all sites in this zone

Security Setting for Trusted Site Zone

Set your Internet Explorer Browser as seen in the following images:



Select a zone to view or change security settings.

Trusted sites
This zone contains websites that you trust not to damage your computer or your files.
You have websites in this zone.

Security level for this zone

Custom
Custom settings.
- To change the settings, click Custom level.
- To use the recommended settings, click Default level.

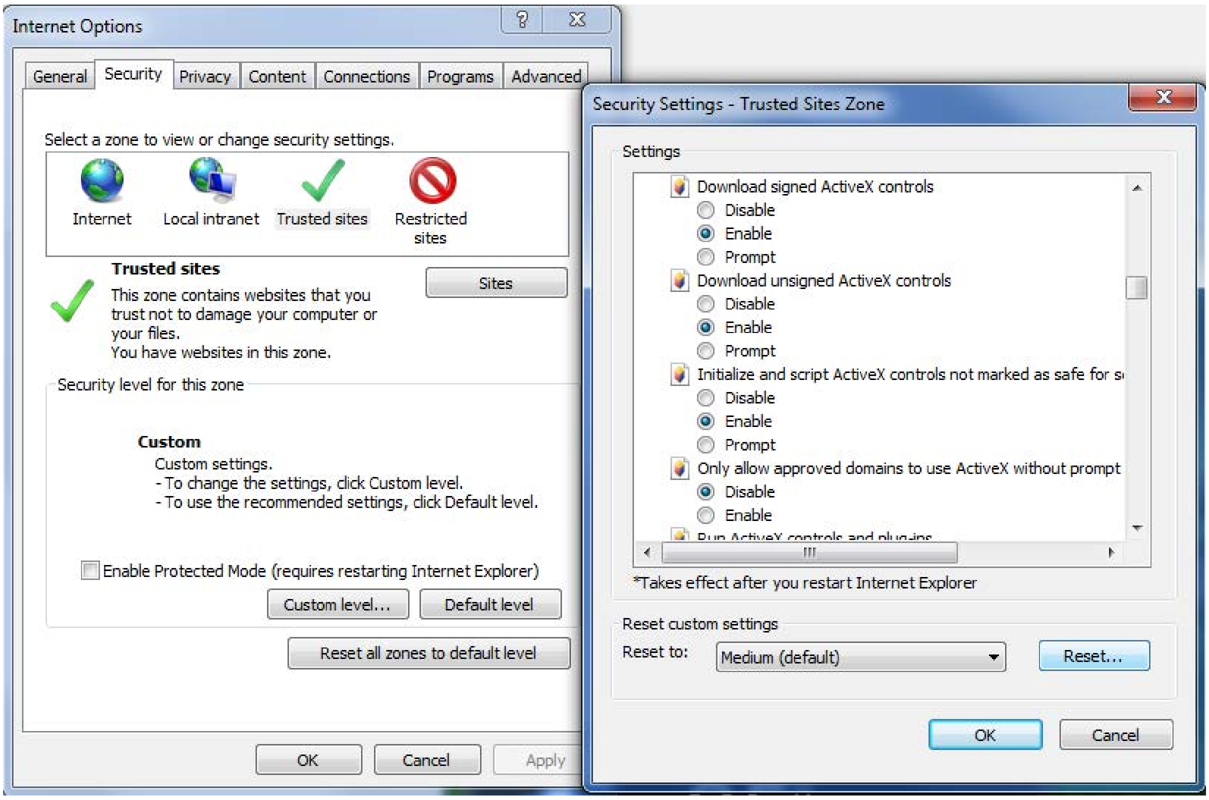
□ Enable Protected Mode (requires restarting Internet Explorer)

Settings

○ Disable
● Enable
Allow Scriptlets
● Disable
○ Enable
○ Prompt
Automatic prompting for ActiveX controls
● Disable
○ Enable
Binary and script behaviors
○ Administrator approved
○ Disable
● Enable
Display video and animation on a webpage that does not use
● Disable
○ Enable

*Takes effect after you restart Internet Explorer

Reset custom settings
Reset to: Medium (default)



Select a zone to view or change security settings.

Trusted sites
This zone contains websites that you trust not to damage your computer or your files.
You have websites in this zone.

Security level for this zone

Custom
Custom settings.
- To change the settings, click Custom level.
- To use the recommended settings, click Default level.

□ Enable Protected Mode (requires restarting Internet Explorer)

Settings

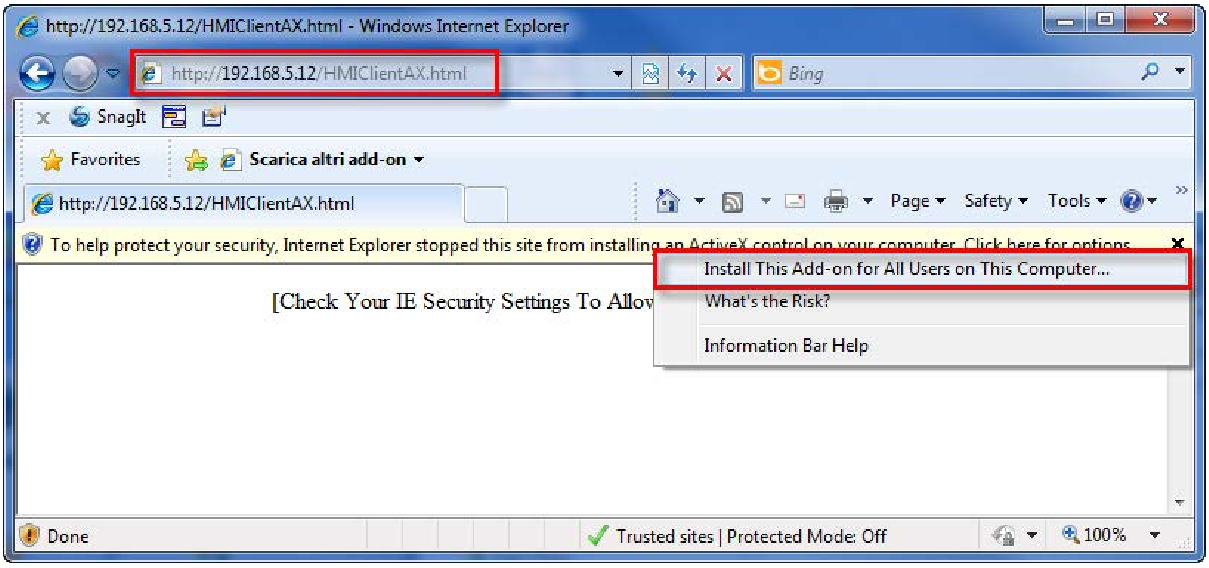
Download signed ActiveX controls
○ Disable
● Enable
○ Prompt
Download unsigned ActiveX controls
○ Disable
● Enable
○ Prompt
Initialize and script ActiveX controls not marked as safe for s
○ Disable
● Enable
○ Prompt
Only allow approved domains to use ActiveX without prompt
● Disable
○ Enable
Run ActiveX controls and plug-ins

*Takes effect after you restart Internet Explorer

Reset custom settings
Reset to: Medium (default)

Install Active X in Internet Explorer

In Internet Explorer, allow the installation of the ActiveX component when the question pops up in your browser.



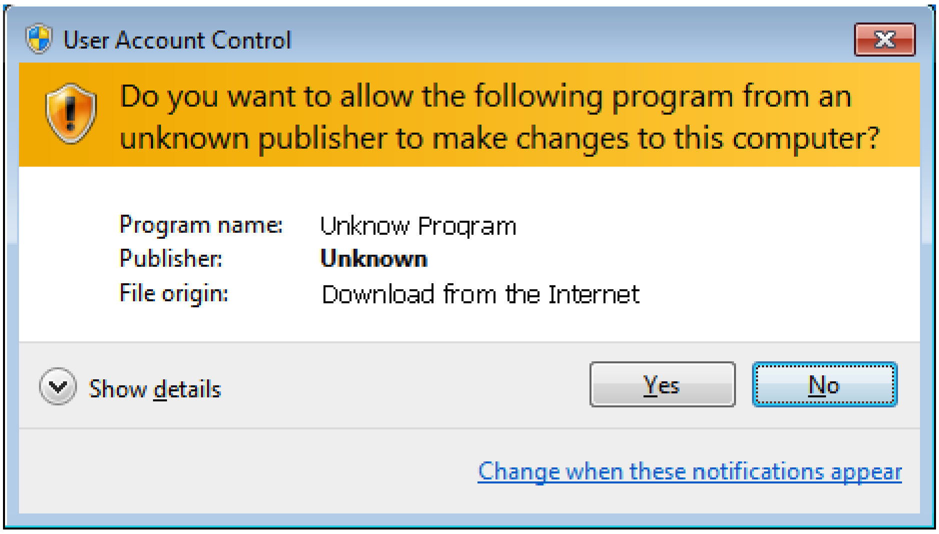
http://192.168.5.12/HMIClientAX.html - Windows Internet Explorer

To help protect your security, Internet Explorer stopped this site from installing an ActiveX control on your computer. Click here for options...

[Check Your IE Security Settings To Allow

http://192.168.5.12/HMIClientAX.html
Bing
SnagIt
Favorites
Scarica altri add-on
http://192.168.5.12/HMIClientAX.html
Page
Safety
Tools
Install This Add-on for All Users on This Computer...
What's the Risk?
Information Bar Help
Done
Trusted sites | Protected Mode: Off
100%

In case you are using a Vista or Windows 7 operating system, you need to click on Yes on User Account Control, as shown in the following picture.



User Account Control

Do you want to allow the following program from an unknown publisher to make changes to this computer?

Program name: Unknow Program
Publisher: **Unknown**
File origin: Download from the Internet

Uninstalling Active X

To remove the ActiveX component from your system, you must delete it from the computer. By default, the component is installed in the following folder:

C:\Program Files\ABB\HMIClientAX

ActiveX information

The ActiveX is able to show projects at a maximum pixel resolution of 1200 x 800.

Using VNC for Remote Access

VNC is a software for remote control. With VNC, you can see the HMI application remotely and control it with your local mouse and keyboard, just like you would if you were in front of it.

VNC is useful for administration and technical support. To be used it requires that a server is started on the HMI device; a viewer is used for connecting from a remote location.

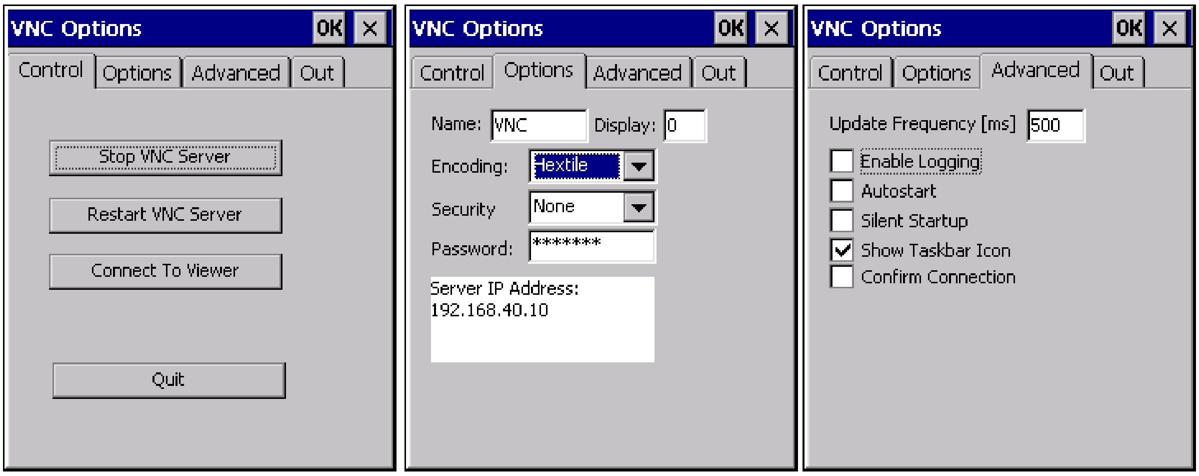
VNC Server

Starting from v1.90, the VNC Server has been added as plugin (ref. to Plugins chapter for more details) to allow developers of hmi applications to choose if enable & download it as part of the runtime. Just enable it from project properties -> plugins and install/update runtime to download it into the target.

VNC server is located in folder **\Flash\qthmi\VNC** and can be activated using macro **launchVNC.**

LaunchVNC macro is used to open the VNC configuration dialog. From the configuration dialog you can:

- Start / Stop / Restart VNC Server in Control Tab
- Enable security and set password in Options tab that will be used later for access using a VNC viewer.
- Enable Autostart in Advanced tab to activate VNC server automatically every time the HMI panel start.
    - **Silent Startup** (usefull only when Autostart is enabled) prevents the VNC dialog from appearing at panel start-up and keeps it open in the background.



Name: VNC
Display: 0
Encoding: Hextile
Security None
Password: *******
Server IP Address:
192.168.40.10

Update Frequency [ms] 500
□ Enable Logging
□ Autostart
□ Silent Startup
☑ Show Taskbar Icon
□ Confirm Connection

OK button on top/right of VNC server configuration dialog is used to confirm and save changes.

Advanced configurations are provided for expert users when VNC server is used in conjunction with a VNC repeater to bypass firewall problems or to optimize VNC performances based on network configuration.

NOTE

The VNC server uses port 5900/TCP.

NOTE

The Password of VNC server is null as default. Password can be changed via VNC viewer or with an external usb keyboard attach to the hmi panel.

NOTE

For developers, a macro **LaunchVNC** is available in **Developer tools**.

NOTE

Show Taskbar icon flag is used by tech support when debugging problems out of KIOSK mode. This flag is not usefull for standard hmi users.

NOTE

The VNC Server has been design for embedded HMI panels WCE based. Win32 platform not supported for VNC Server.

NOTE

The VNC Server allows only one single client. It´s not allowed, to connect two or more at the same time.

NOTE

Drag and drop is not supported yet by VNC server.

VNC Viewer

A VNC viewer is not provided as part of Panel Builder 600. However, many types of VNC viewers are freely available. One example of compatible VNC viewer is TightVNC.

Alarms

The alarm feature is designed to provide alerts through pop-up messages, typically to issue a warning, or to indicate any abnormal conditions or any malfunctions in the system under control. Whenever a bit goes high, or the value of a tag crosses the limit of deviation defined in the alarm configuration, the respective alarm message(s) will be displayed in a special dialog. Or, alternatively, you can program certain macro actions to be executed when the alarms are triggered.

In PB610 Panel Builder 600 there is no default action associated to a triggered alarm. The visualization of a specific page containing the alarm widget is optional, and the specific action executed when the trigger condition is verified can be any one of the actions found on the action list.

The alarm configuration of an alarm determines whether or not the alarm has to be acknowledged. It can also be used to determine how the alarm appears when displayed on the HMI device, like background and foreground color. Alarm configuration also determines whether and when the corresponding alarm is logged in the event list.

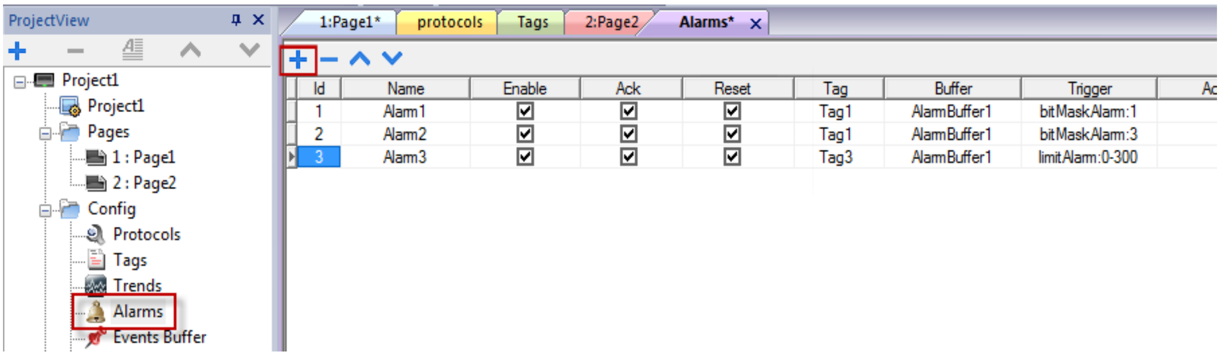
For alarms displaying critical or hazardous operating and process statuses, a stipulation can be made requiring the plant operator to acknowledge the alarm.

The alarms are configured in the alarm manager and, are a component of all screens of a project. More than one alarm can be displayed simultaneously in the alarm widget, depending on its configured size. An event can trigger the closing and reopening of the alarm window.

In PB610 Panel Builder 600, working with alarms is similar to working with events. In general, there is no absolute need to have a pop-up dialog when an alarm is triggered. Any “background” action (from the list of available actions) can be associated with this event.

Alarm Configuration Editor

1. To open the editor double click on alarms in the Project Workspace.
2. Add the alarms by clicking the **+** button.



Id	Name	Enable	Ack	Reset	Tag	Buffer	Trigger	Ac
1	Alarm1	☑	☑	☑	Tag1	AlarmBuffer1	bitMaskAlarm:1	
2	Alarm2	☑	☑	☑	Tag1	AlarmBuffer1	bitMaskAlarm:3	
3	Alarm3	☑	☑	☑	Tag3	AlarmBuffer1	limitAlarm:0-300	

Name

Specifies the name of the specific alarm.

Enable

A user can enable or disable the triggering of particular alarms. Alarms can be enabled or disabled in operation mode as well (for more information, see chapter “Enable / Disable Alarms in Runtime”).

Acknowledgment

For an alarm that needs to be acknowledged by the operator (when the alarm is triggered), select the check box to enable the acknowledgment. If checked, an operator is required to acknowledge this alarm any time it is triggered, before it will be cleared from the active alarm widget.

Reset

The alarms' editor includes from V1.50 an additional column called "Reset". This check box, specific to each alarm, works in conjunction with the acknowledge check box. After an alarm requiring acknowledgment has been acknowledged, it will be cleared from the alarm list. If the “Reset” check box is checked, the alarm will continue to be listed in the alarm list, as “Not Triggered Ack”, until the **Reset** button present in the alarm widget is pressed.

Buffer

Specifies the buffer file to which the alarm history will be saved.

Trigger

This selection determines the triggering condition for an alarm. Three alarm types are available:

- **Limit Alarm**
  A limit alarm is triggered when the monitored tag value goes OUTSIDE of its given boundaries (low limit and high limit). If the tag value is equal to its low or high limit, the alarm is not triggered.
- **Bitmask Alarm**
  To get a valid trigger, the bitwise AND operator compares each bit of the bitmask with the tag value corresponding to that alarm. If both bits are on, the alarm is set to true. If the bitmask alarm is selected, you can specify one or more bit positions inside the tag. If one of the bits is set, the alarm is triggered. The bit position must be given in decimal format. If more bits are specified, each position must be separated by a “,”.Bitmask is a position, so it starts from zero (0).
- **Deviation Alarm**
  For the deviation alarm, a predefined “set point”, as well as a value for “deviation” will be given. If the percentage of deviation of the tag value from the set point exceeds this deviation, then the trigger condition becomes true.

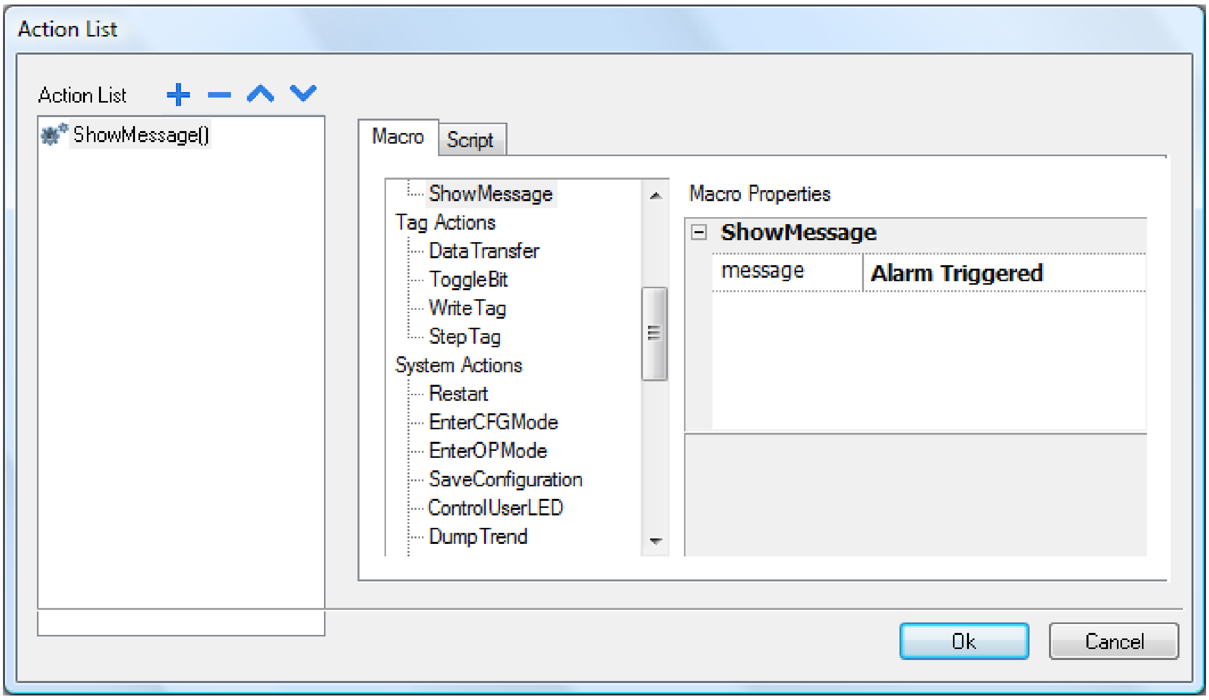
$$(Value_{now} - SetPoint > \left(\frac{deviation}{100}\right) * SetPoint$$

Tag

Attach the tag in which the alarm shall periodically check the tag value, so that the respective alarm(s) is triggered when this deviates from its limits. The alarm function will refer to the value of this tag, or to the state of a bit, in the case of bitmask, to determine when to trigger the alarm.

Action

Define the action(s) to be executed for the specific alarm. Actions are executed by default when the specified trigger condition becomes true. Additional conditions can be specified in the “Events” configuration (in the last column of the alarm editor, as explained in chapter “Action Enable").



ShowMessage()

ShowMessage

message	Alarm Triggered

Description

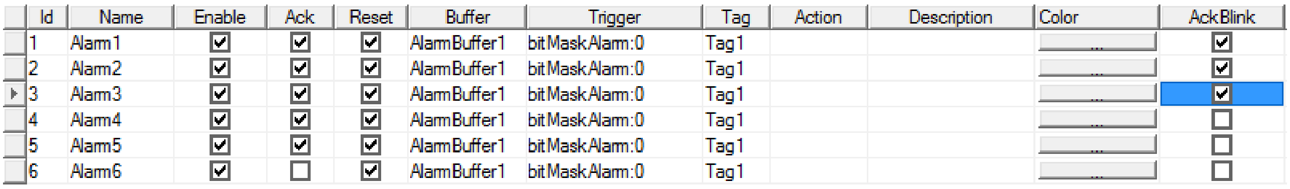
This is the description for the alarm condition. The alarm description is normally a text; this text supports the multiple language features. The text can be a combination between static and dynamic parts, where the dynamic ones are eventually some tag values. Please see the chapter “Live Data in Alarm Widget” for further information about this feature.

Colors

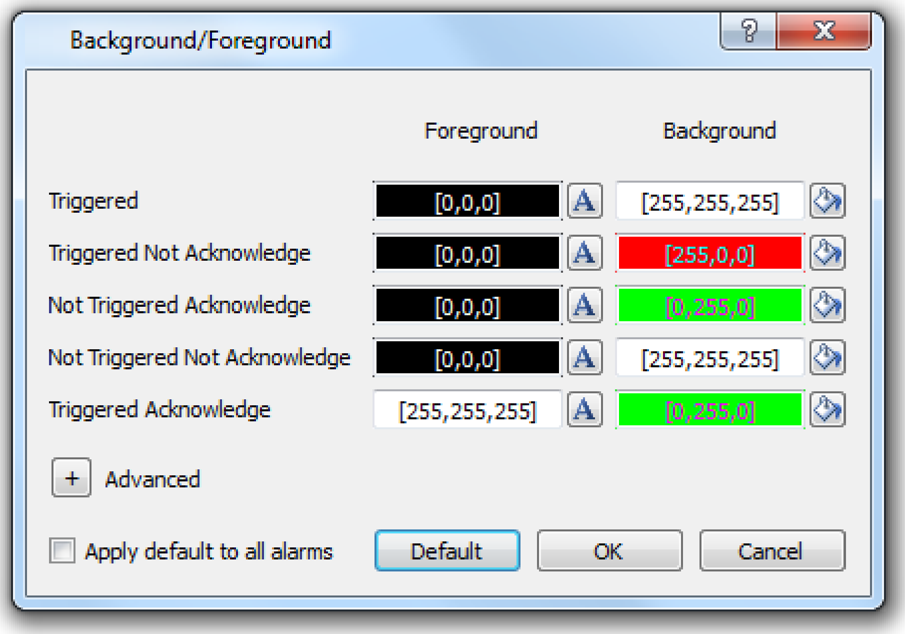
Foreground and Background colors of alarm rows (Active alarms widget) can be applied based on the status of alarm (ex. Triggered, Triggered Ack etc).

AckBlink

Make alarm row (of Active alarms widget) blink when an alarm is triggered. Stop blink when alarm has been Ack. Blink can be used only with alarms that have the Ack flag enabled.



Id	Name	Enable	Ack	Reset	Buffer	Trigger	Tag	Action	Description	Color	AckBlink
1	Alarm1	☑	☑	☑	AlarmBuffer1	bitMaskAlarm:0	Tag1			...	☑
2	Alarm2	☑	☑	☑	AlarmBuffer1	bitMaskAlarm:0	Tag1			...	☑
3	Alarm3	☑	☑	☑	AlarmBuffer1	bitMaskAlarm:0	Tag1			...	☑
4	Alarm4	☑	☑	☑	AlarmBuffer1	bitMaskAlarm:0	Tag1			...	☐
5	Alarm5	☑	☑	☑	AlarmBuffer1	bitMaskAlarm:0	Tag1			...	☐
6	Alarm6	☑	☐	☑	AlarmBuffer1	bitMaskAlarm:0	Tag1			...	☐



Background/Foreground

	Foreground	Background
Triggered	[0,0,0]	[255,255,255]
Triggered Not Acknowledge	[0,0,0]	[255,0,0]
Not Triggered Acknowledge	[0,0,0]	[0,255,0]
Not Triggered Not Acknowledge	[0,0,0]	[255,255,255]
Triggered Acknowledge	[255,255,255]	[0,255,0]

Severity

You can indicate the severity of each alarm. If multiple alarms are triggered simultaneously, the actions will be executed based on severity settings.

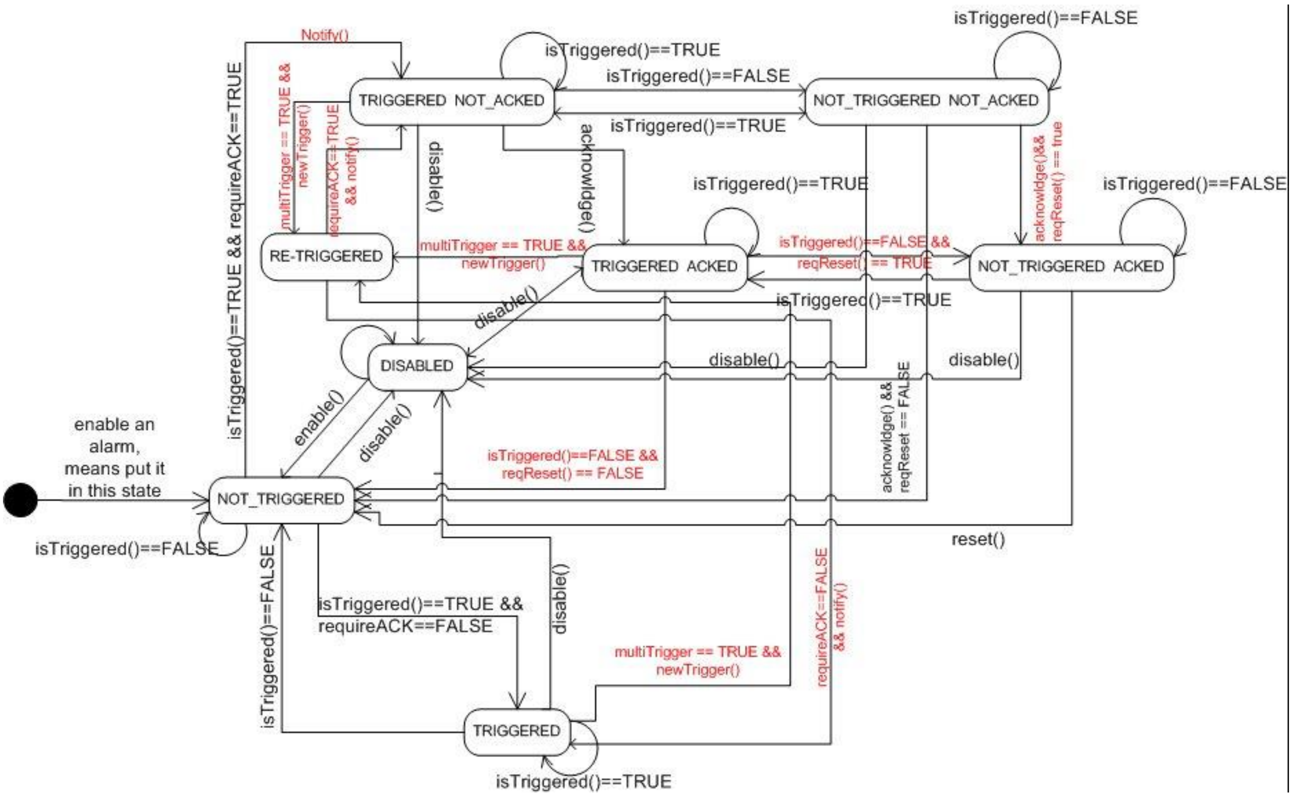
Events

These options allow you to specify conditions relating to the following matters: When the alarms events are to be logged, when the alarms widget view is to be refreshed or updated by the system, and some particular options for action execution.

Alarm's State Machine

The HMI system implements an alarm state machine which is described by the following figure.

The graph includes states and transitions between them according to the selected options and desired behavior.



isTriggered()==FALSE
Notify()
isTriggered()==TRUE
isTriggered()==FALSE
TRIGGERED NOT_ACKED
NOT_TRIGGERED NOT_ACKED
isTriggered()==TRUE
isTriggered()==TRUE && requireACK==TRUE
multiTrigger == TRUE && newTrigger()
requireACK==TRUE && notify()
disable()
acknowldge()
isTriggered()==TRUE
acknowldge()&& reqReset() == true
isTriggered()==FALSE
RE-TRIGGERED
multiTrigger == TRUE && newTrigger()
TRIGGERED ACKED
isTriggered()==FALSE && reqReset() == TRUE
NOT_TRIGGERED ACKED
isTriggered()==TRUE
disable()
DISABLED
disable()
disable()
acknowldge() && reqReset == FALSE
enable an alarm, means put it in this state
enable()
disable()
isTriggered()==FALSE && reqReset() == FALSE
NOT_TRIGGERED
isTriggered()==FALSE
reset()
isTriggered()==FALSE
isTriggered()==TRUE && requireACK==FALSE
disable()
multiTrigger == TRUE && newTrigger()
requireACK==FALSE && notify()
TRIGGERED
isTriggered()==TRUE

Setting Events

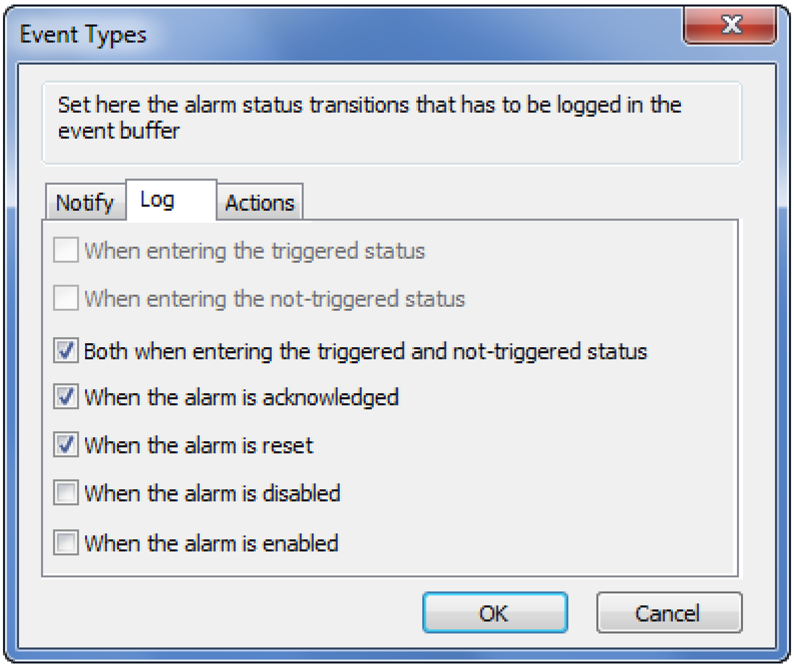
This chapter describes how to set events in the alarm configuration editor.

Log Events

1. Select the “Log" tab in the dialog box.

   ➤ The list below represents a set of conditions in which you may want to store the specific event in the alarm history buffer.

2. Click the check boxes corresponding to the application requirements.



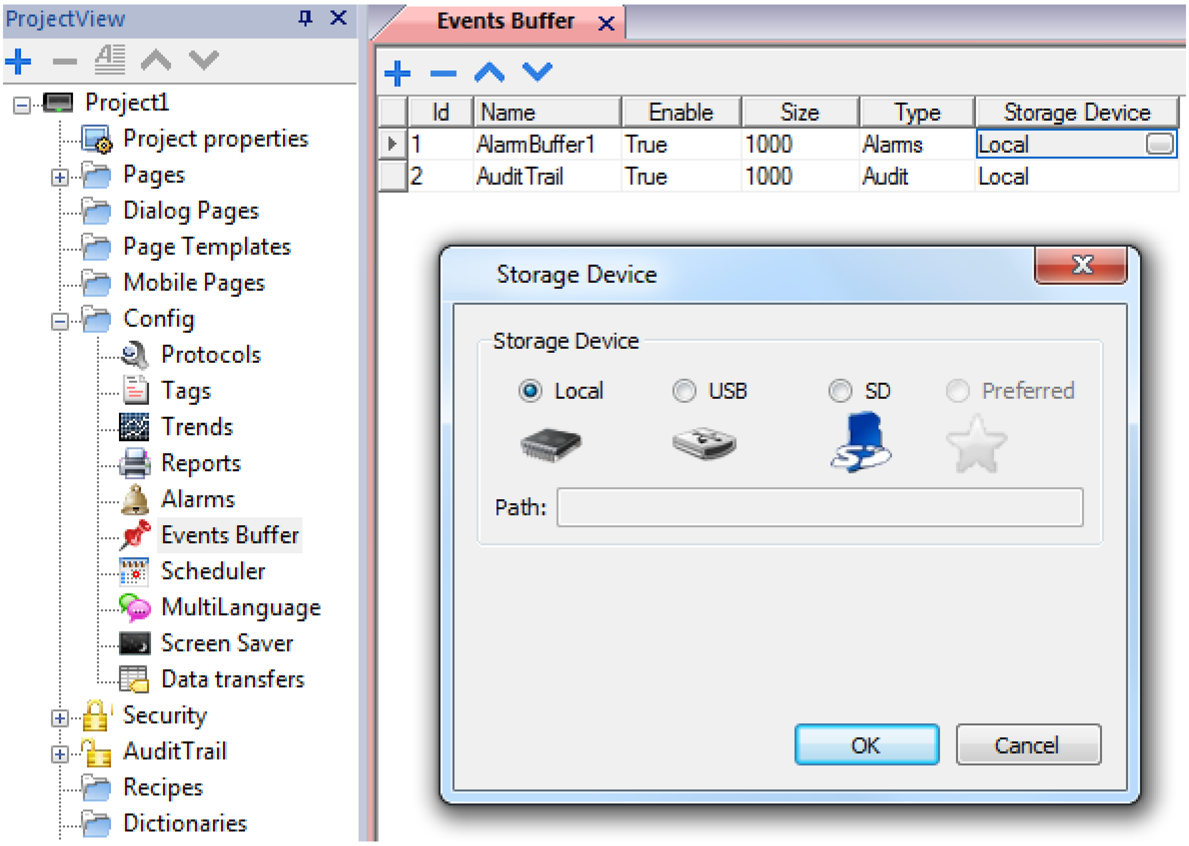
Set here the alarm status transitions that has to be logged in the event buffer

- □ When entering the triggered status
- □ When entering the not-triggered status
- ☑ Both when entering the triggered and not-triggered status
- ☑ When the alarm is acknowledged
- ☑ When the alarm is reset
- □ When the alarm is disabled
- □ When the alarm is enabled

The alarm events history can be accessed by logging in, in a dedicated buffer called “Event Buffer”.

3. To configure the event buffer, you have to double click on **Buffers** in the configuration editor.

Here there is an option for selecting the storage type. There are three options: “Flash”, “USBMemory”, “SD Card”.



Id	Name	Enable	Size	Type	Storage Device
1	AlarmBuffer1	True	1000	Alarms	Local
2	AuditTrail	True	1000	Audit	Local

Storage Device

Storage Device

● Local ○ USB ○ SD ○ Preferred

Path:

When the events’ buffers are stored in persistent storages (local, USB, SD, etc.), the system saves the file on disk every 5 minutes.

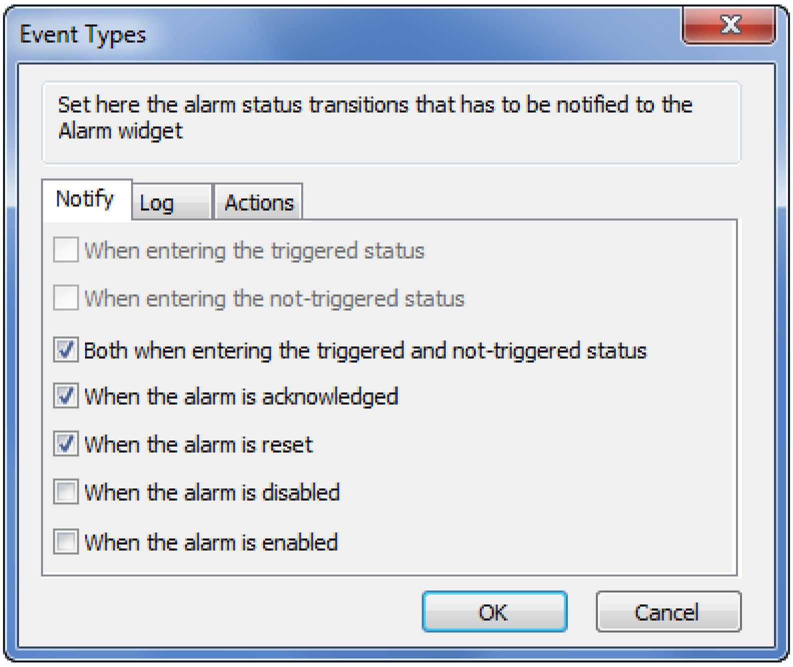
However, events of type alarms are saved immediately.

Notify

The user can choose the conditions under which the alarms should be notified in the alarms widget. This specifically refers to the default alarm widget, available in the widget gallery. The user can decide when the widget will be notified of a change to the alarm status.

TIP

We recommend leaving the default settings here, and changing only those necessary for specific application requirements.



Event Types

Set here the alarm status transitions that has to be notified to the Alarm widget

□ When entering the triggered status

□ When entering the not-triggered status

☑ Both when entering the triggered and not-triggered status

☑ When the alarm is acknowledged

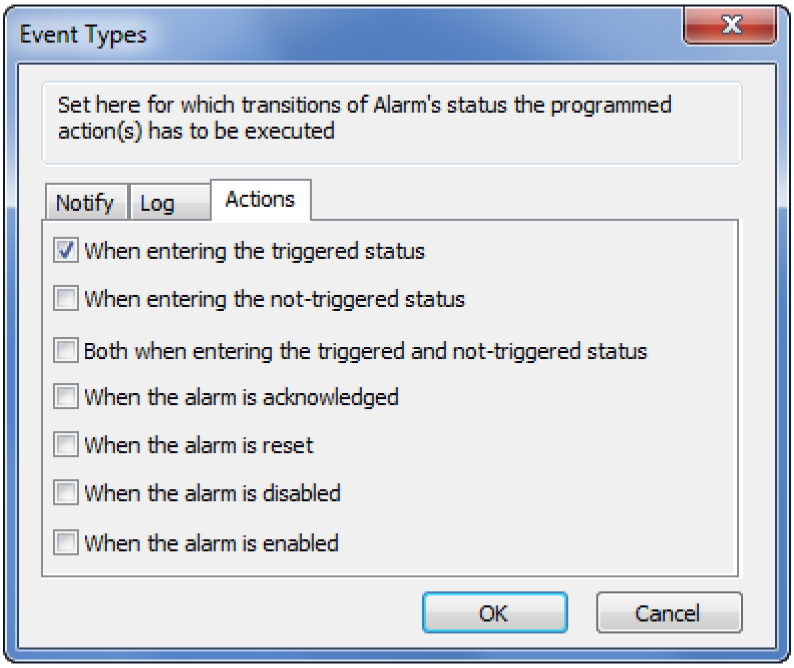
☑ When the alarm is reset

□ When the alarm is disabled

□ When the alarm is enabled

Actions

You can specify the conditions under which the action(s), configured for the specific alarm, must be executed.



Set here for which transitions of Alarm's status the programmed action(s) has to be executed

☑ When entering the triggered status

□ When entering the not-triggered status

□ Both when entering the triggered and not-triggered status

□ When the alarm is acknowledged

□ When the alarm is reset

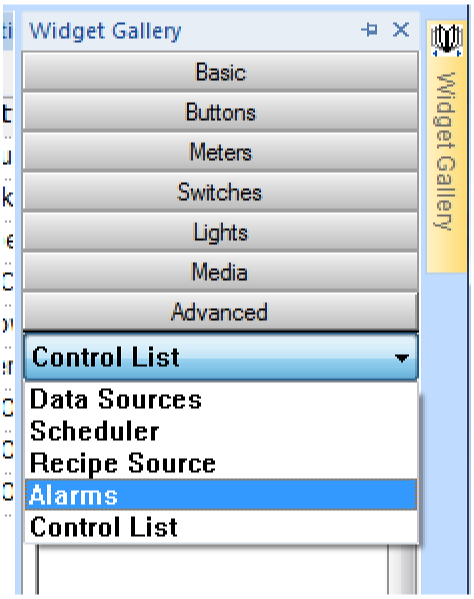
□ When the alarm is disabled

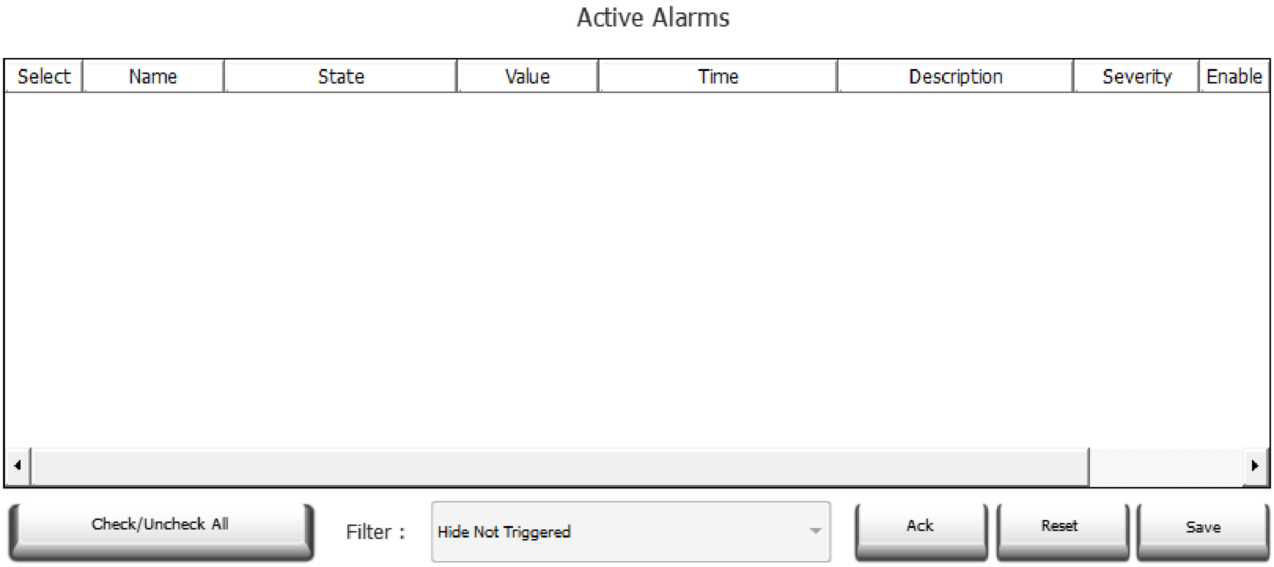
□ When the alarm is enabled

By default, the actions are executed only when the alarm enters the triggering condition. You may change this by configuring the system to execute the configured action also for the other alarms statuses available.

Active Alarms Widget

You can insert the active alarms widget in a page to see the status of alarms and to acknowledge or reset or enable/disable alarms. Simply drag and drop the active alarms widget from the advanced gallery page.

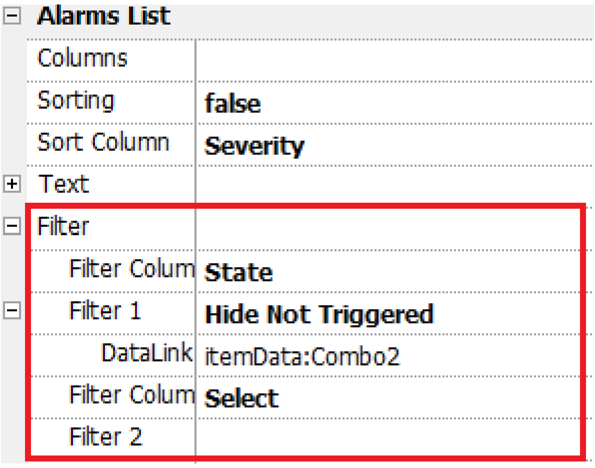




The alarm widget will display the alarms in Runtime Mode.

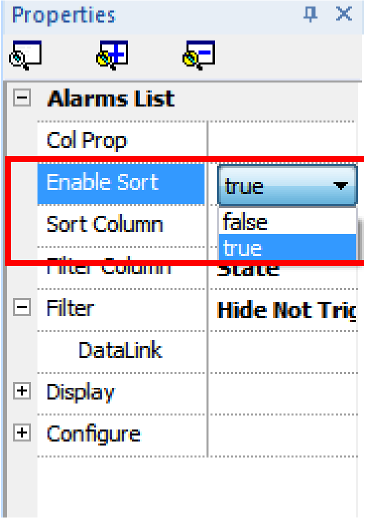
A Filter is available to show/hide just a subset of all configured alarms. Using the Combo box **Filter** it is possible for example to **Hide Not Triggered** alarms.

Another Filter (**Filter 2**) is available in widget properties and can be used to add a second filter based on another alarm field like Alarm name, or based on Severity or Description of alarm.



Alarms List	
Columns	
Sorting	**false**
Sort Column	**Severity**
Text	
Filter	
Filter Colum	**State**
Filter 1	**Hide Not Triggered**
DataLink	itemData:Combo2
Filter Colum	**Select**
Filter 2	

You can enable or disable the column sorting option, available at Runtime for the Alarms Widget, by clicking on the column header. The sorting order is based on the string sorting.



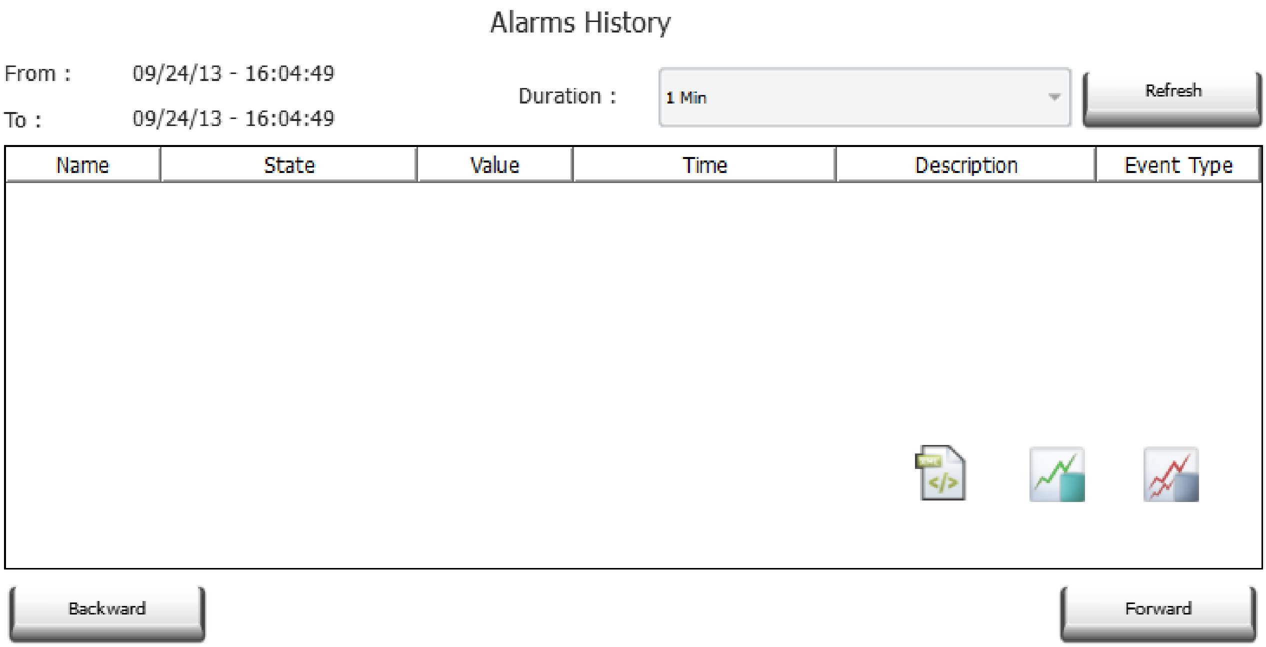
Alarms List	
Col Prop	
Enable Sort	true
Sort Column	false true
Filter Column	State
Filter	Hide Not Trig
DataLink	
Display	
Configure	

NOTE

Starting from version 1.80 the alarms' widget provided in the gallery no longer has the "Priority" column. The widget has a new column called "Severity" which comes by default next to the ID column. "Severity" column takes the values from the "Severity Settings" from the alarm editor.

Alarms History Widget

PB610 Panel Builder 600 automatically logs the alarm list based on the flag settings set in the alarms editor, under “Log Event Types”. To see the historical alarm list, you can use alarms history widget.



Alarms History

From : 09/24/13 - 16:04:49

To : 09/24/13 - 16:04:49

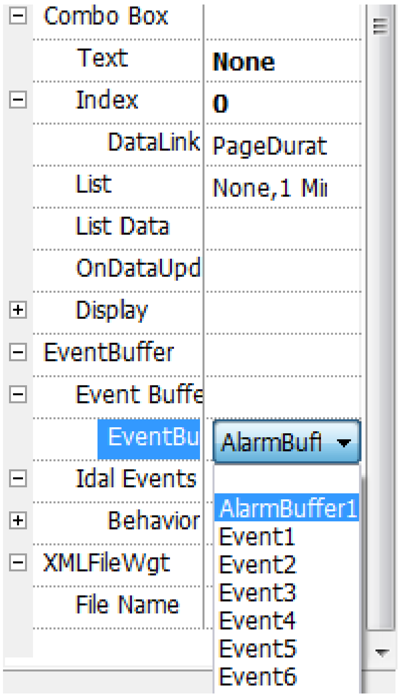
Duration : 1 Min

Refresh

Name	State	Value	Time	Description	Event Type

Backward

Forward



⊟ Combo Box	
Text	None
⊟ Index	0
DataLink	PageDurat
List	None,1 Mi
List Data	
OnDataUpd	
⊞ Display	
⊟ EventBuffer	
⊟ Event Buffe	
EventBu	AlarmBufl
⊟ Idal Events	
⊞ Behavior	
⊟ XMLFileWgt	
File Name	

The selection of the event buffer is available in the property panel.

> **NOTE**
>
> For each of the different alarm buffers, there is a specific event widget that must be configured for the project. The current version of the event list widget does not allow you to switch between buffers.

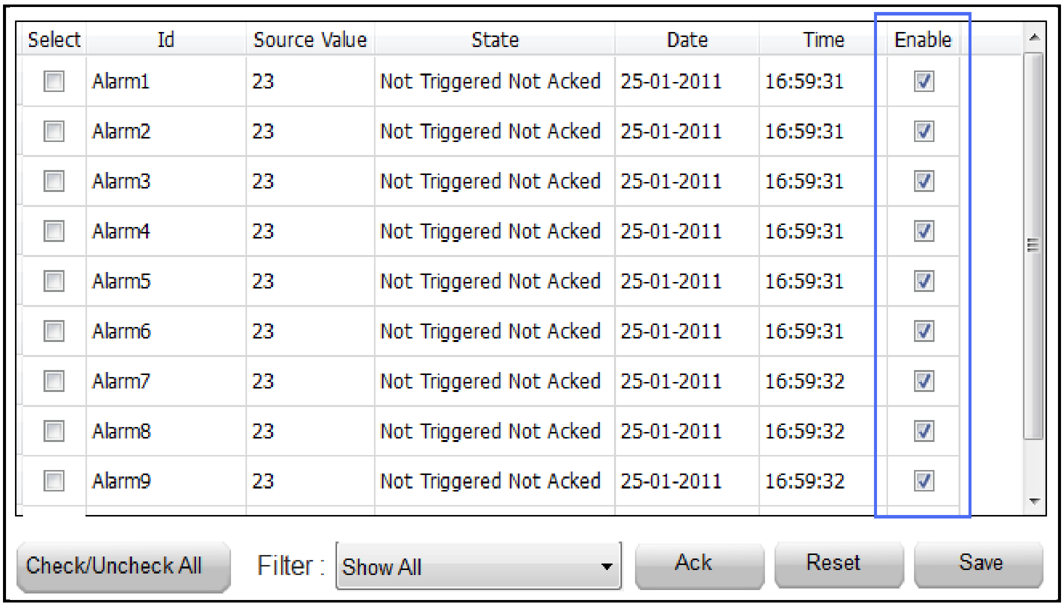
Managing Alarms at Runtime

When an alarm is triggered, the alarm will be displayed in the active alarms widget. The widget allows you to acknowledge and reset the alarm. The alarm display can be filtered by “hide not triggered”, “show all” and other custom filters.

The visualization of the alarm widget is not automatic. If the widget has been placed on a certain page, when an alarm is active, you must add a dedicated action that will go to the page showing the alarm widget.

Enable / Disable Alarms at Runtime

You can enable or disable the alarms in PB610 Panel Builder 600 Runtime. If you want to disable an alarm, just uncheck the alarms from the “Enable“ column in the alarm widget and execute the “Save**”** command. This way the alarm will not get triggered and the disabled alarm will not be displayed at runtime.



Select	Id	Source Value	State	Date	Time	Enable
□	Alarm1	23	Not Triggered Not Acked	25-01-2011	16:59:31	☑
□	Alarm2	23	Not Triggered Not Acked	25-01-2011	16:59:31	☑
□	Alarm3	23	Not Triggered Not Acked	25-01-2011	16:59:31	☑
□	Alarm4	23	Not Triggered Not Acked	25-01-2011	16:59:31	☑
□	Alarm5	23	Not Triggered Not Acked	25-01-2011	16:59:31	☑
□	Alarm6	23	Not Triggered Not Acked	25-01-2011	16:59:31	☑
□	Alarm7	23	Not Triggered Not Acked	25-01-2011	16:59:32	☑
□	Alarm8	23	Not Triggered Not Acked	25-01-2011	16:59:32	☑
□	Alarm9	23	Not Triggered Not Acked	25-01-2011	16:59:32	☑

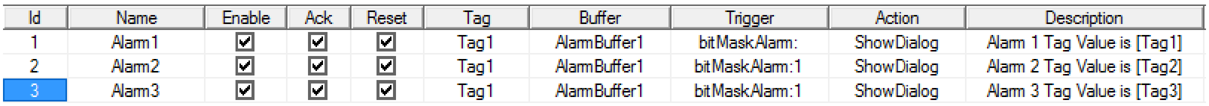
Filter : Show All

Later, if you want to re-enable the alarm, select the alarm and check the “Enable“ check box. Then click on **Save**. The alarm will now be subscribed and subject to being triggered.

Live Data in Alarms Widget

This feature is used to view the live tag data value inside the alarm description. It is applicable only for the "active alarms" widget.

To configure the live data visualisation in the alarm widget, follow a simple syntax rule. The tags to be included must be specified in the description, including the tag names in square brackets: [Tag name]



Id	Name	Enable	Ack	Reset	Tag	Buffer	Trigger	Action	Description
1	Alarm1	☑	☑	☑	Tag1	AlarmBuffer1	bitMaskAlarm:	ShowDialog	Alarm 1 Tag Value is [Tag1]
2	Alarm2	☑	☑	☑	Tag1	AlarmBuffer1	bitMaskAlarm:1	ShowDialog	Alarm 2 Tag Value is [Tag2]
3	Alarm3	☑	☑	☑	Tag1	AlarmBuffer1	bitMaskAlarm:1	ShowDialog	Alarm 3 Tag Value is [Tag3]

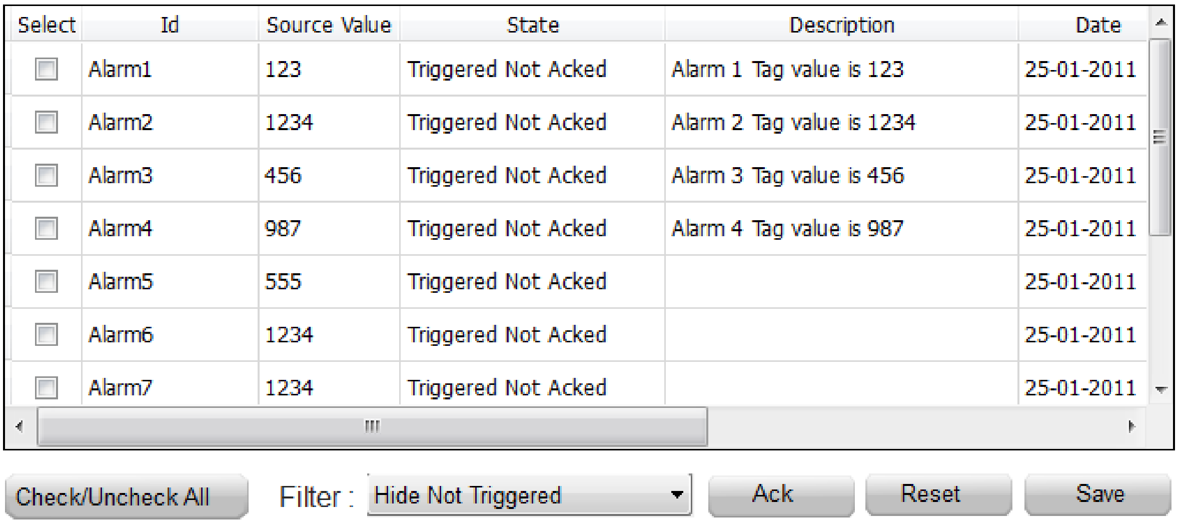
Example for live tag data.

During runtime in the alarm widget, the markers and tag name will be replaced, in the description column, by the actual value of the tag. The widget automatically refreshes and shows the current values of the tags in the widget.

In history alarm widget it will show the value of the tag at the moment the alarm was triggered.

Into the CSV file resulting from the dump of the alarms events list, the tag values can be seen in the description column.

Output will be displayed as shown in figure below.



Select	Id	Source Value	State	Description	Date
□	Alarm1	123	Triggered Not Acked	Alarm 1 Tag value is 123	25-01-2011
□	Alarm2	1234	Triggered Not Acked	Alarm 2 Tag value is 1234	25-01-2011
□	Alarm3	456	Triggered Not Acked	Alarm 3 Tag value is 456	25-01-2011
□	Alarm4	987	Triggered Not Acked	Alarm 4 Tag value is 987	25-01-2011
□	Alarm5	555	Triggered Not Acked		25-01-2011
□	Alarm6	1234	Triggered Not Acked		25-01-2011
□	Alarm7	1234	Triggered Not Acked		25-01-2011

NOTE

The ability to store the alarm description with tag values in the event buffer is a feature supported starting from version 1.80.

NOTE

Use '\' before '[ ]' where there is a need to show the '[ ]' in the description string. So, if the string to show is [Tag[1]], the correct syntax to use is [Tag\[1\]]

Exporting Alarm Buffers as CSV File

The historical alarm list (the event buffer) can be exported using the specific action called “DumpEventArchive”. The action is described in the chapter called “Dump Event Archive”. Please refer to that chapter for information about how to export the data.

NOTE	The tag values included in the alarms’ description are also included in the event description stored in the event buffer; the tags are sampled at the moment in which the alarm is triggered and that is the value recorded and included in the description; in the Alarm description, displayed by the alarm widget, the value may change because it is constantly updated, but no additional values are recorded. This feature is supported starting from version V1.80.

Recipes

Recipes are a feature for organizing data storage in the HMI device and include services for exchanging data with connected controller devices.

This data can be written to the controller, and, conversely, the data can be read from the controller and saved back on the HMI panel. This concept is normally referred as “Recipes”, and it offers you powerful way to extend the capabilities of the controller. This is especially true for controllers that have a limited amount of internal memory.

The recipe memory is the physical storage for the recipes. The “Recipe Tag” block basically identifies the “current recipe”: From the recipe memory, you take one recipe data record and designate it “current/active recipe”. Then, you operate transfers over this, to or from the controller. These tags can be displayed on the page.

Currently the recipe data is configured in PB610 Panel Builder 600, the user can specify default values for each element of the data records. On runtime, data can be edited; the new data is saved to a new and separated data file, different from the original one containing the default values. Any change to recipe data is file on disk. The use of a separate data file on PB610 Panel Builder 600 Runtime ensures that modified recipe values are retained throughout different project updates. In other words a subsequent project update does not influence the recipe data modified by the user on runtime.

NOTE

To reset the recipe data to the default values, there is a dedicated action called “Reset recipes”; see below in this chapter for further information.

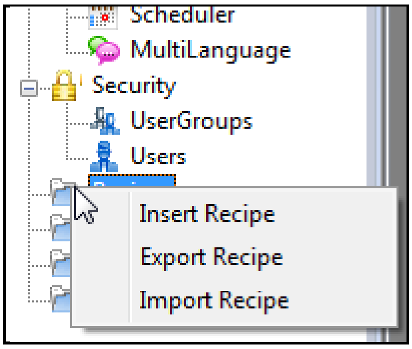
The User can also select where the Recipe needs to be stored. There are three options for this: FLASH, USB, and SD Card. The user can select any one.

You can configure recipes by adding the required controller data items to a page from the recipe widget. A recipe can be associated with a particular page and is composed of all the recipe data items on that page. Recipe data items contain all the information associated with normal controller data items. But, rather than the data being read and written directly to the controller during the course of normal operation, the data is instead read from and written to the Control Panel memory that is reserved for the data item.

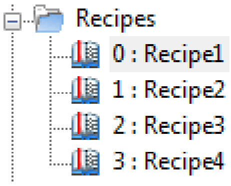
This chapter describes how to configure and use the recipes in PB610 Panel Builder 600 application.

Recipe Configuration Editor

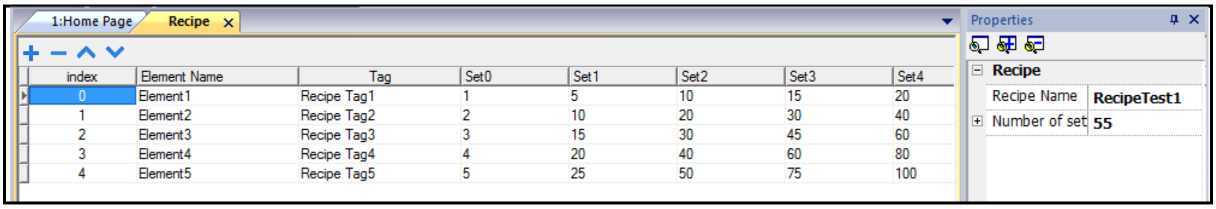
1. In the Project View pane, select **Recipes** and right click.
2. Choose **Insert Recipe** if you want to create a new recipe.



➤ The newly added recipe item will be added in the project workspace.



3. Double click on the recipes to open the recipe editor.

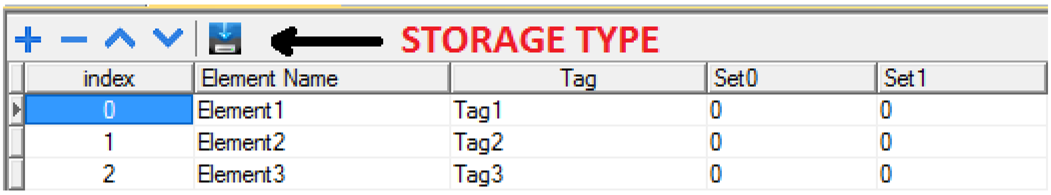


index	Element Name	Tag	Set0	Set1	Set2	Set3	Set4
0	Element1	Recipe Tag1	1	5	10	15	20
1	Element2	Recipe Tag2	2	10	20	30	40
2	Element3	Recipe Tag3	3	15	30	45	60
3	Element4	Recipe Tag4	4	20	40	60	80
4	Element5	Recipe Tag5	5	25	50	75	100

Recipe

Recipe Name	RecipeTest1
Number of set	55

4. Add the recipe elements by clicking the **+** button, and link the tags to the recipe element.
5. By clicking on the button for “Storage Type” you can select where to store recipe data.



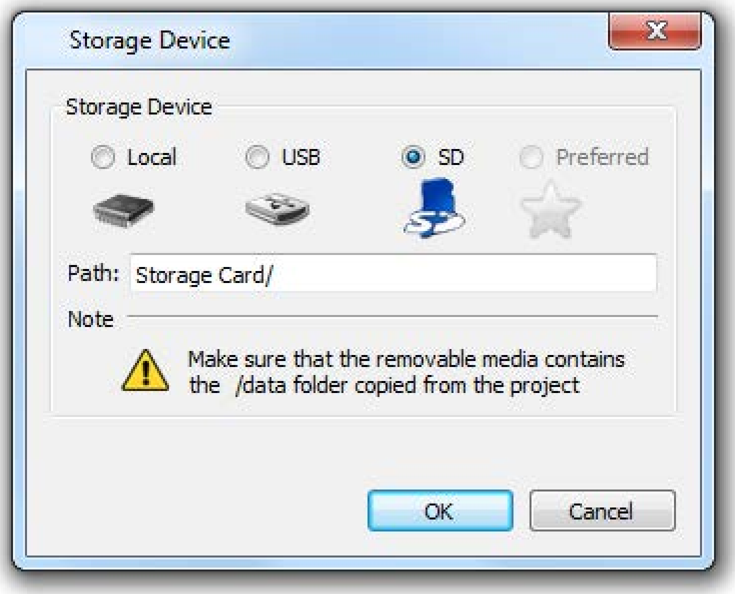
index	Element Name	Tag	Set0	Set1
0	Element1	Tag1	0	0
1	Element2	Tag2	0	0
2	Element3	Tag3	0	0

➤ A dialog in which the selection can be made will open.

6. For USB and SD card you can provide the folder location.

NOTE

Recipes configuration files are created automatically when the project is saved. Recipes files are saved into the subfolder data of the project folder into the PC by Panel Builder 600. When external storages are used, please copy this folder into the external storage selected. Default path is “/Storage Card/data” for SD or “/USBMemory/data” for USB storage. However, a subfolder of it can be used like “/USBMemory/MyRecipes/data”. The subfolder name “data” cannot be changed and is required for the recipes to work.


Storage Device

Storage Device

○ Local ○ USB ● SD ○ Preferred

Path: Storage Card/

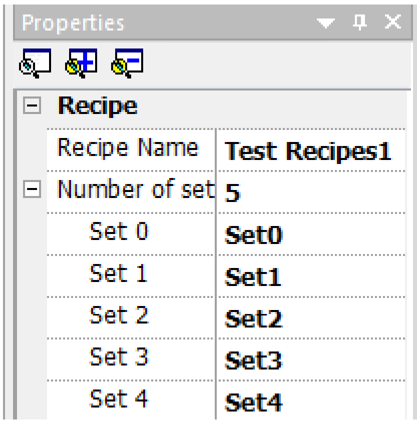
Note

Make sure that the removable media contains the /data folder copied from the project

Configuring Recipes Set on the Page

The number of parameter sets can be changed in the number of sets field in the property pane. From there you can also change the name of each recipe set.

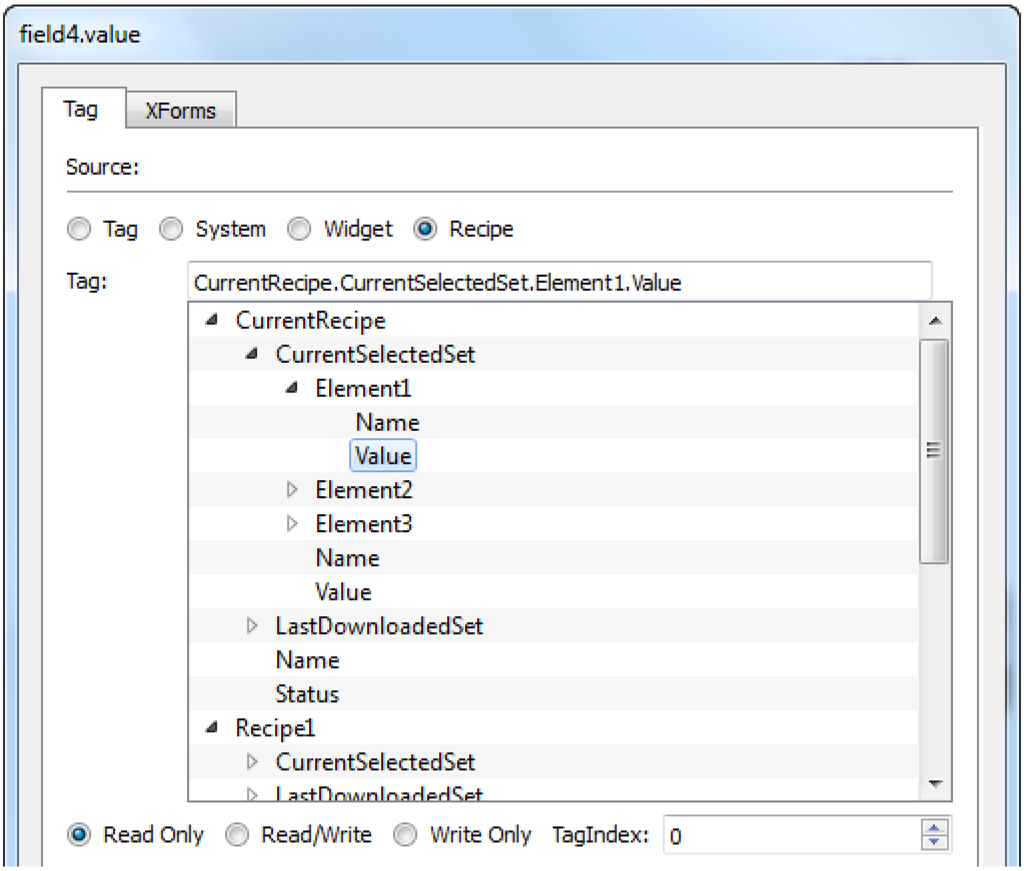
Recipe values for all the parameter sets can be entered into the recipe editor window.


Properties

Recipe	
Recipe Name	Test Recipes1
Number of set	5
Set 0	Set0
Set 1	Set1
Set 2	Set2
Set 3	Set3
Set 4	Set4

Defining Recipe Fields

The user can define the recipe field on the page by using the numeric field widget from the gallery and attach the tags from the recipe data source.



field4.value

Tag | XForms

Source:

○ Tag ○ System ○ Widget ● Recipe

Tag: CurrentRecipe.CurrentSelectedSet.Element1.Value

- CurrentRecipe
  - CurrentSelectedSet
    - Element1
      - Name
      - Value
    - Element2
    - Element3
    - Name
    - Value
  - LastDownloadedSet
  - Name
  - Status
- Recipe1
  - CurrentSelectedSet
  - LastDownloadedSet

● Read Only ○ Read/Write ○ Write Only TagIndex: 0

Example of a tag attached to a recipe field.

The *Attach To* dialog allows you to attach to the numeric field with all the different recipe variables, such as:

- Current Recipe > Current selected Recipe set > Element > Value (or) Name
- Selected Recipe > Selected Set0 > Element > Value (or) Name
- Selected Recipe list
- Currently selected Recipe list
- Recipe Status

When the numeric fields are defined as read/write, the default recipe data can be edited at Runtime. As explained in the introduction, these new values are stored in a separate file as modified recipe data.

Recipe Status

After every recipe upload or download, or recipe set modification, the recipe status parameters display a value. The following are the values and conditions for the recipe status system variable.

Code	Function	Description
0	Set modified	Current selected set changed.
1	Download triggered	Triggered a download request.
2	Download Done	Download action completed.
3	Download Error	Error occurred when doing download - errors like unknown set, unknown recipe, controller not ready, Tags write failed etc.
4	Upload triggered	Triggered an upload request.
5	Upload done	Upload action completed.
6	Upload Error	Error occurred when doing upload - errors similar to download errors.
7	General Error	Errors like data not available.

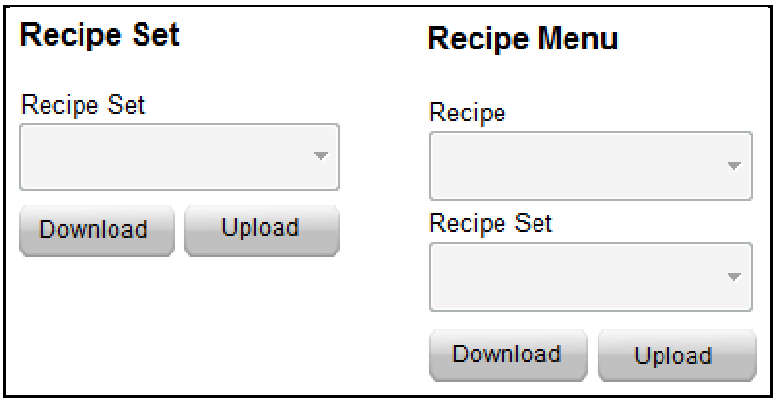
NOTE

When the panel starts up the value of Recipe Status is 0

Configuring Recipe Widget for Runtime Execution

Two default recipe widgets are available in the advanced widget gallery category.

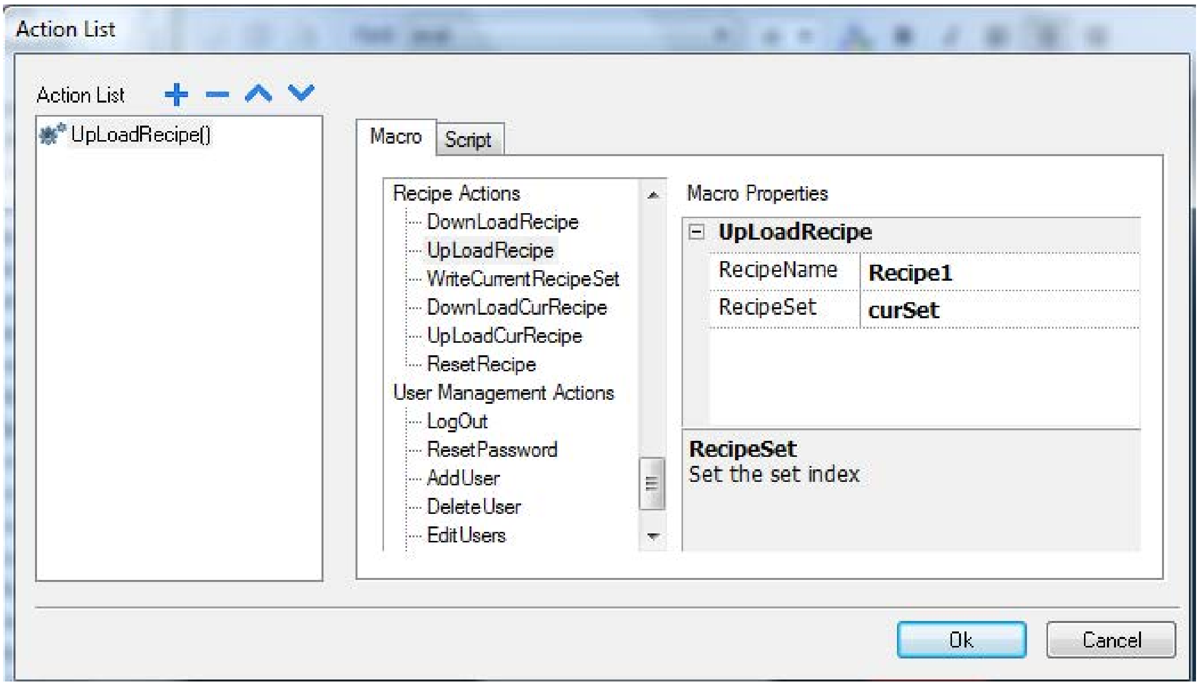
The “Recipe Set” widget allows you to select a recipe set for the upload and download operation. If you have more than one recipe in the project, then the “Recipe Menu” widget can be directly used to manage all the recipes in a single widget, listing recipes and selecting the sets for each recipe.



Configure Recipe Transfer Macros

The recipe transfer action can be completed through the action list dialog. The transfer of recipes can be achieved by any of the following methods:

- Attaching an action on an event for button or switches.
- Configuring the action from alarms' action list.
- Using the schedulers actions' list.
- Description of actions available for Recipes is included in the relevant chapter.



UpLoadRecipe

RecipeName	Recipe1
RecipeSet	curSet

RecipeSet
Set the set index

Upload or Download of Recipes during Runtime

Recipe Download through Recipe Widget in Runtime

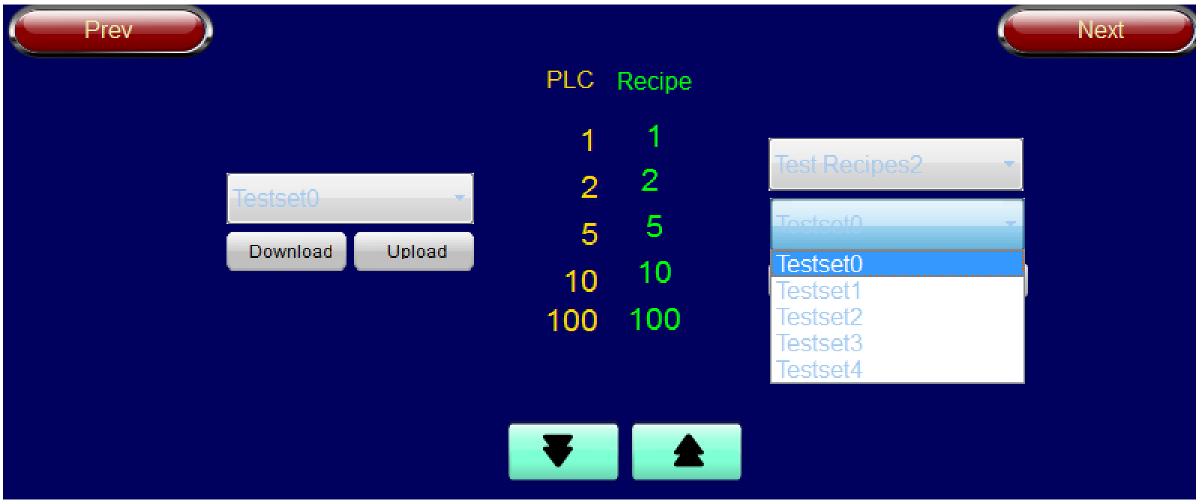
1. Drag and drop the recipe widget (as described in chapter "Configuring Recipe Widget for Runtime Execution") into the project to execute the recipe transfer in runtime.
2. Select the recipe from the drop down box, and select the recipe set from the set dropdown list.
3. Press the **Download** button to download the current selected recipe set, or press the **Upload** button to upload the current selected recipe set.

Recipe Download or Upload Through Recipe Transfer Macro in Runtime

The recipes can be downloaded or uploaded through the recipe transfer macro.

1. Define the macro button as described in chapter “Configure Recipe Transfer Macros”.
2. In runtime, execute the macro (if the macro is programmed with a push button, press the button).

➤ The recipes data will be transferred to the controller, or uploaded from the controller, depending on the action programmed.



Example of the recipe project.

Backup and Restore of Recipe Data

The recipe data stored on the HMI device can be exported for backup purposes and later restored to the system. Please refer to the chapter “Dump Recipe Data“ and “Restore Recipe Data” for further information.

Trends

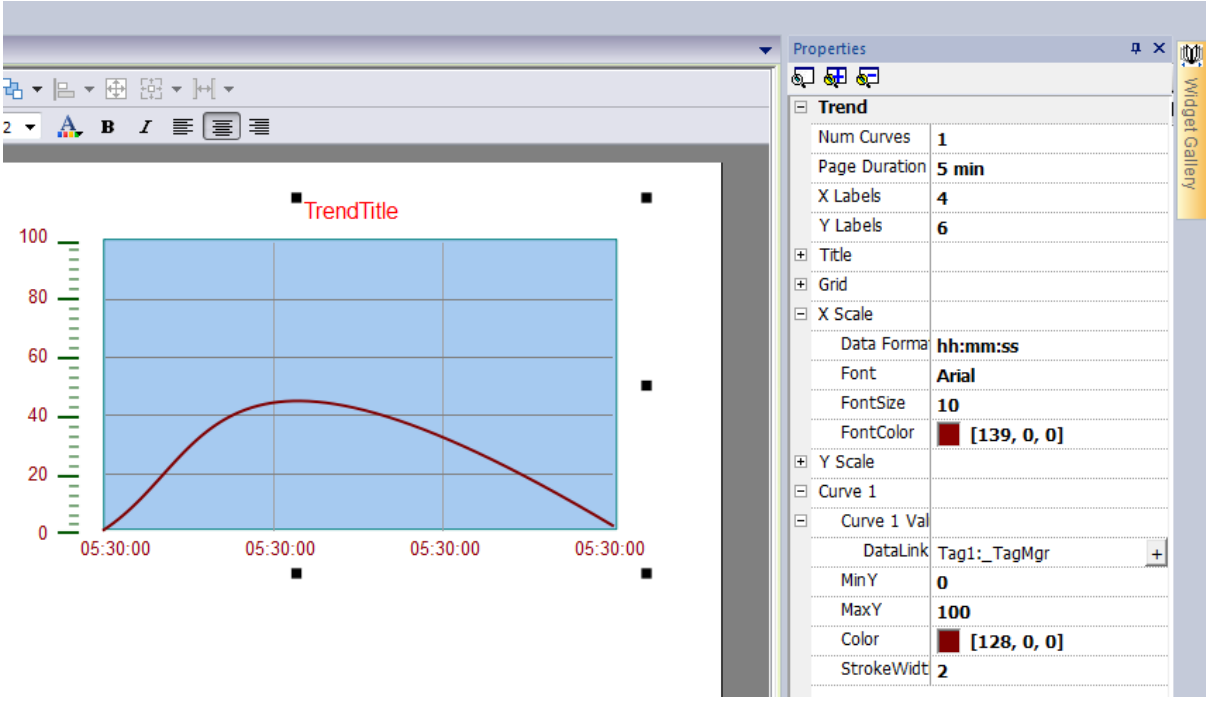
Trending is a method of sampling and recording the values of a specified tag according to sampling conditions (normally, the time).

Trending is divided into two main parts: Trend acquisition and trend viewing. Trend acquisition (trend editor) collects the data into a database. The trend viewer (trend widget) displays the data from this database in a graphical format.

Real-Time Trend

In Real-Time trend, the data will be presented directly in the trend window, and the changes to the live data can be seen directly in the format of a curve on the trend window. Users can manage the process by seeing the trend on the Control Panel. The real-time trend widget is just a viewer for a tag, and it does not refer to any saved data in any buffer. Any curve plotted is lost when the page containing the widget is changed.

1. To configure the Real-Time trend, just drag and drop the Real-Time trend widget from the *Basic* > *Trends* category of the gallery.



⊟ Trend	
Num Curves	1
Page Duration	5 min
X Labels	4
Y Labels	6
⊞ Title	
⊞ Grid	
⊟ X Scale	
Data Forma	hh:mm:ss
Font	Arial
FontSize	10
FontColor	🟥 [139, 0, 0]
⊞ Y Scale	
⊟ Curve 1	
⊟ Curve 1 Val	
DataLink	Tag1:_TagMgr
MinY	0
MaxY	100
Color	🟥 [128, 0, 0]
StrokeWidt	2

2. Select the trend widget and, in the properties pane, attach to the “Curve x Value" property the tag for which you want the data to be plotted. Data are always plotted against time.

NumCurves: Number of trend curves in the trend window. A maximum of 5 curves can be configured in a trend window.

Page duration: Time range of the X-axis. However, you can dynamically change the page duration in runtime with the Date Time combo widgets, thereby attaching to the trend window page duration properties.

X Labels: Number of labels in the X-axis scale.

Y Labels: Number of labels in the Y-axis scale.

Title: Trend title and font properties (like font size, label, etc.).

Curve x: Tag or trend buffer that will be plotted into the trend window.

Scaling can be applied to the tag values. To apply scaling, use the X forms attached to dialog.

You can set the minimum or maximum of the curves. You can also attach a tag to these minimum and maximum properties. This enhances the ability to change the min and max dynamically in the runtime.

Also you can modify the properties, such as “Colors”, “Update time”, “Number of samples”, etc. of the trend curves through the properties view.

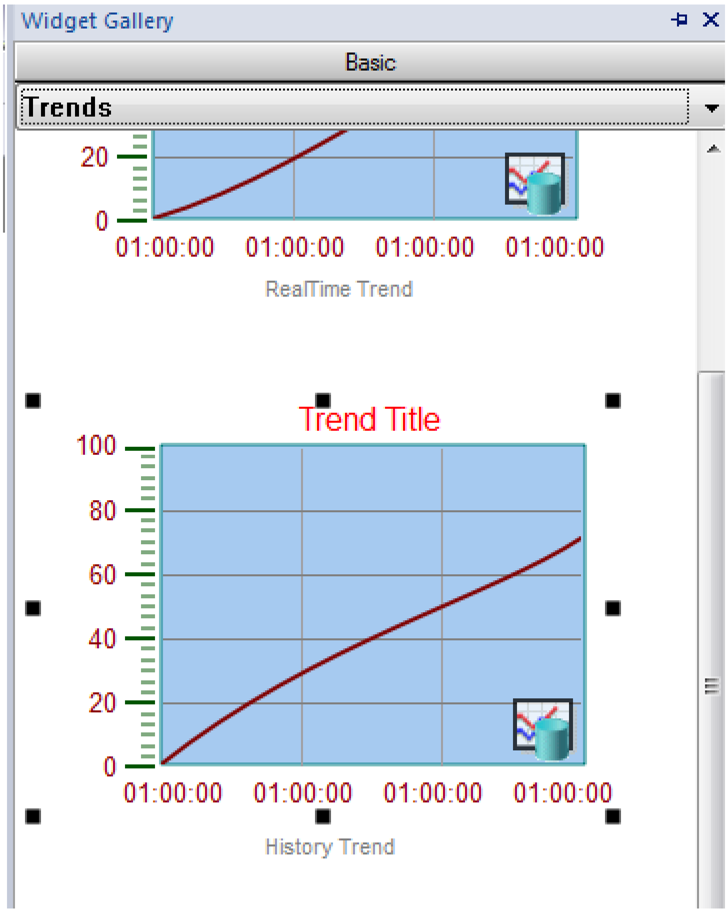
History Trend

If you want to analyze the data in the future, you will need to store the Trend data somewhere for later review. For this reason, we have introduced the History Trend. When you select History Trend, you can store date information with reference to time.

→ The first step in creating a history trend is to create a trend buffer.

The purpose of the trend buffer is to save a sequence of values of a specified tag in order to record the state of the tag while time changes. Once values are stored in the buffer a dedicated widget, called history trend viewer, can be used to represent the curve in a graphical format.

The history trend viewer is available in the widget gallery under the *Basic* >*Trends* category as shown by the following figure.

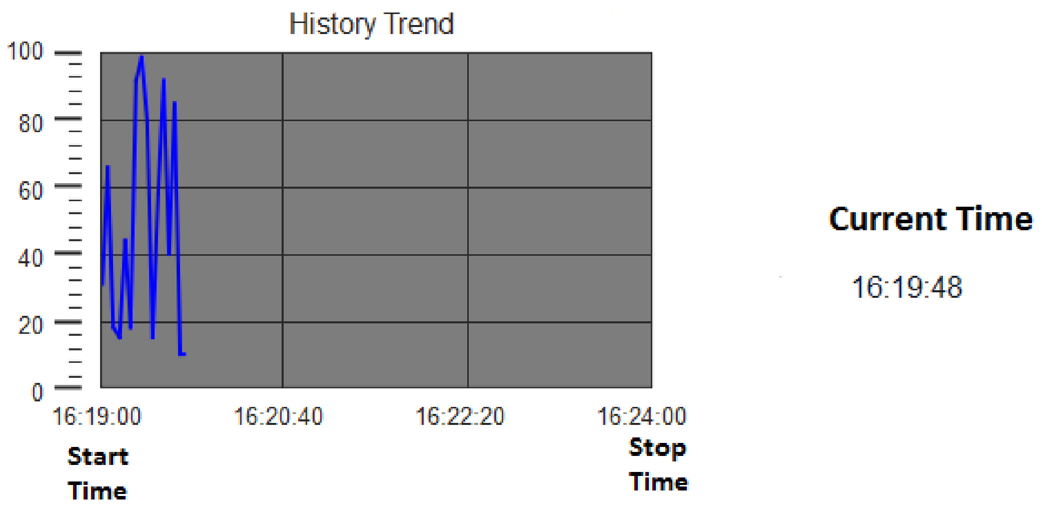


In the history trend widget the start time of the trend window will be the current time and stop time will be the current time + duration of the window.

The plot starts from the right end of the trend window as in the figure below. The graph will be automatically refreshed during a certain interval of time, till the stop time. When the curve reaches the stop time then the graph will scroll left and the update of the curve will continue until it reaches again the right side of the viewer; at that moment a new scroll is automatically done and the process repeats.

NOTE

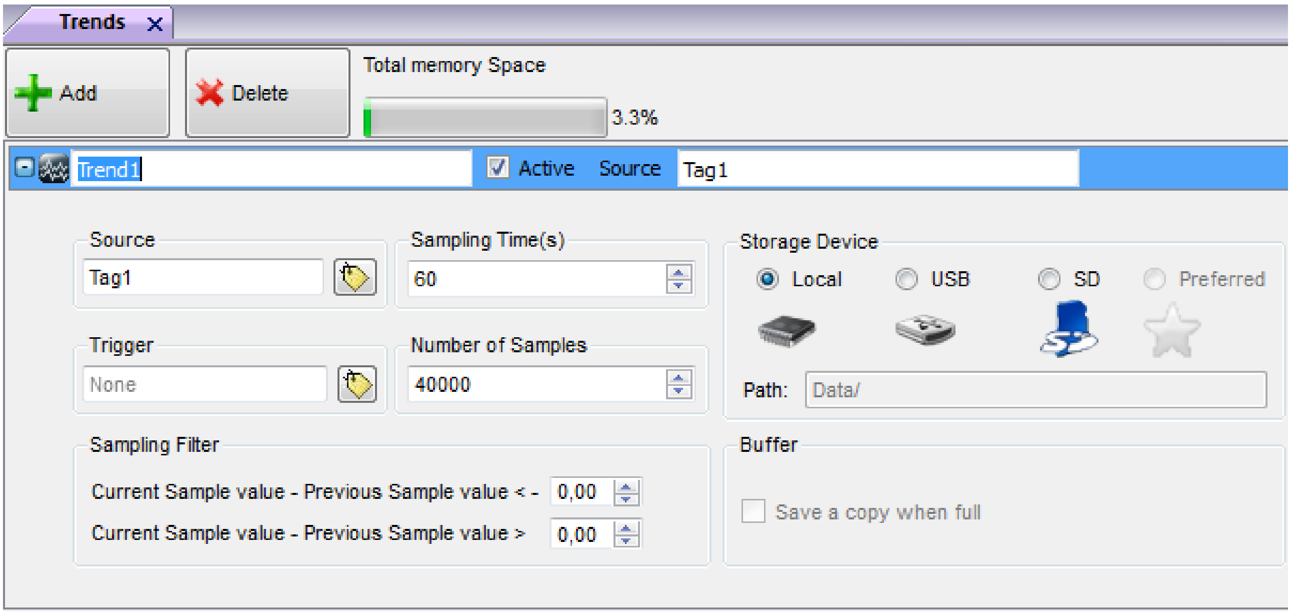
Automatic refresh is an option available starting from version 1.80.



Trend Editor

Historical trends require a proper configuration of trend data buffer. Trends' buffers are configured from the trend editor.

Trend buffers are stored in data files. There is an option to store these files on the internal storage (local), USBMemory or SD Card or custom folders based on target platform.



Total memory Space 3.3%

Trend1 ☑ Active Source Tag1

Source: Tag1

Sampling Time(s): 60

Storage Device: ● Local ○ USB ○ SD ○ Preferred

Trigger: None

Number of Samples: 40000

Path: Data/

Sampling Filter

Current Sample value - Previous Sample value < - 0,00

Current Sample value - Previous Sample value > 0,00

Buffer

□ Save a copy when full

1. In the project view pane, double click **Trends** to open the trend editor.
2. Add the trend buffer, by selecting the "**+** Add" button on the editor.
3. By clicking "**+**" near each trend buffer, the corresponding buffer configuration is expanded.

➤ The "Total memory Space" bar shows how much memory has been used by the trend buffers configured so far. The max number of samples allowed for a project is 1200000. The memory use is the percentage of this number. As in the Figure above, suppose the total number of samples used in the project is 40.000. Then the total memory used will be shown as 3%. This is calculated by the formula:

$$\text{Total Memory Space} = \left(\frac{Total\ Number\ of\ Samples\ used\ in\ the\ Project}{Max\ Number\ of\ Samples\ allowed\ for\ a\ Project}\right) * 100$$

As we increase the number of samples, the percentage of usage also increases and this will be shown in the bar.

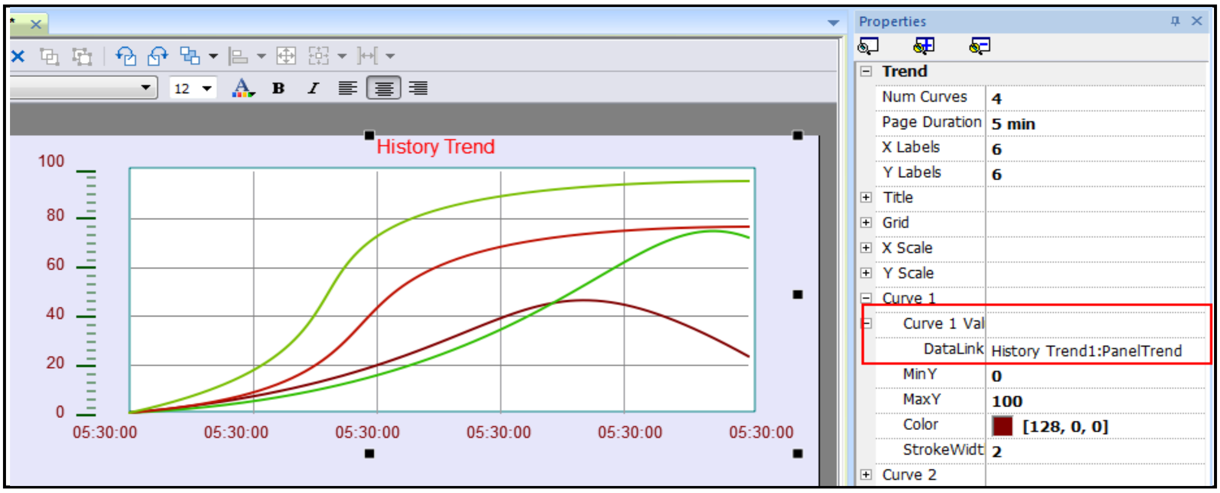
The following are the properties of each trend buffer in the trend editor:

Trend Name	Defines the trend buffer name, which will appear when you define the buffer to a trend window property pane. A default name is assigned by the system; the name can be modified by the user.
Active	Specifies if the trend runs by default when the system starts up. Note: The trend buffers cannot be activated during Runtime
Source	This combo list allows selecting the Tag which is sampled by the Trend manager system.
Sampling Time	Samples are collected and stored in the disk data file on a cyclical basis. Default sampling condition is the time; the sampling time specifies the sampling period in seconds.
Trigger	When the Trigger tag is specified, the source tag is not sampled on a cyclical basis but on the Trigger tag value change. In any case, the samples are plotted with respect to the time. The Trigger tag and source tag can be the same.
Number of Samples	This represents the buffer size expressed in samples.
Storage Device	This is an option to select where the trend buffer data file will be stored.
Buffer	Trend data is organized as a FIFO queue. Once the buffer gets full, the oldest values will be erased to create space for storing the new values. If **Save a copy when full** is selected, when the buffer gets full, before overwrite it, system create a backup copy of it into external storages.
Sampling Filter / Trigger Filter	When the triggering condition is the time, a new sample is considered significant (and then stored) only if its value, in comparison with the last saved value, goes out from the specified boundaries. In case the triggering condition is based on a trigger tag value change, the boundaries are applied to the trigger tag value.

Configuring Trend Window for History Trends

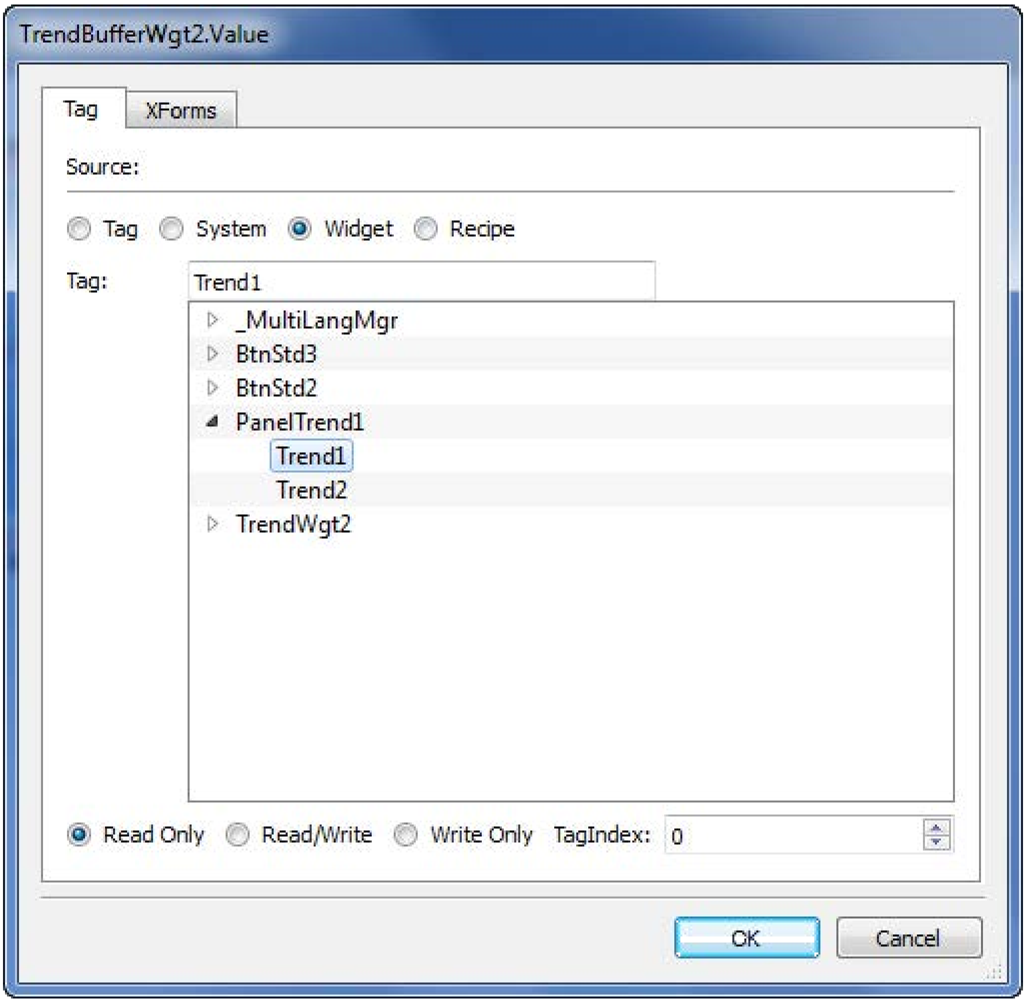
The history trend widget (trend window) is the area used to display the trend buffer in a curve format. After configuring the trend buffer in trend editor, you can use the historical trend viewer widget to plot the trend curve on the screen.

1. From the trend gallery, drag and drop the *History Trend* widget to the page.



Trend	
Num Curves	4
Page Duration	5 min
X Labels	6
Y Labels	6
Title	
Grid	
X Scale	
Y Scale	
Curve 1	
Curve 1 Val	
DataLink	History Trend1:PanelTrend
MinY	0
MaxY	100
Color	🟥 [128, 0, 0]
StrokeWidt	2
Curve 2	

2. In the property pane of the trend window, attach the history trend buffer to be plotted in the trend window.


TrendBufferWgt2.Value

Source:

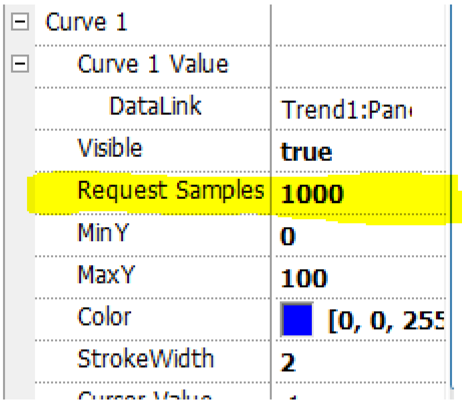
○ Tag ○ System ● Widget ○ Recipe

Tag: Trend1

● Read Only ○ Read/Write ○ Write Only TagIndex: 0

Trend Window Properties (Advanced View)

With the help of the property pane of the trend window, you can customize the trend window properties, such as X-axis time, Y-axis value, number of trend curves, changes to the labels, grids, number of samples etc. In the curve x category there is one property called *Request Samples*.



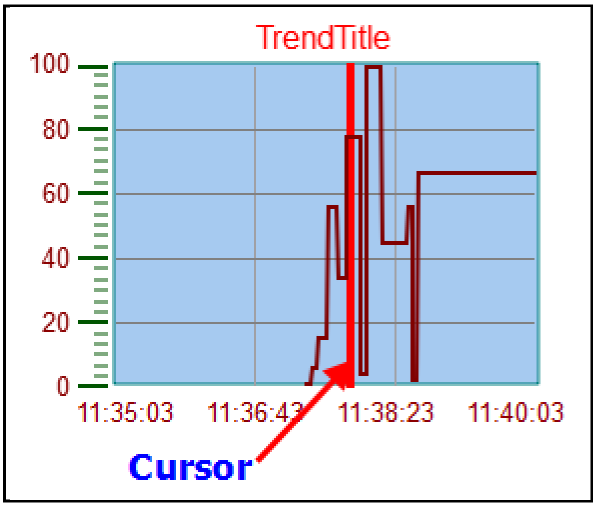
⊟ Curve 1	
⊟ Curve 1 Value	
DataLink	Trend1:Pan
Visible	true
Request Samples	1000
MinY	0
MaxY	100
Color	🟦 [0, 0, 255
StrokeWidth	2
Cursor Value	

The property represents the maximum numbers of samples read by the widget from the buffer data file; this block size can be adjusted to fine tune performances in trend viewer refresh especially when working with remote clients; the default value is normally a good compromise for most cases.

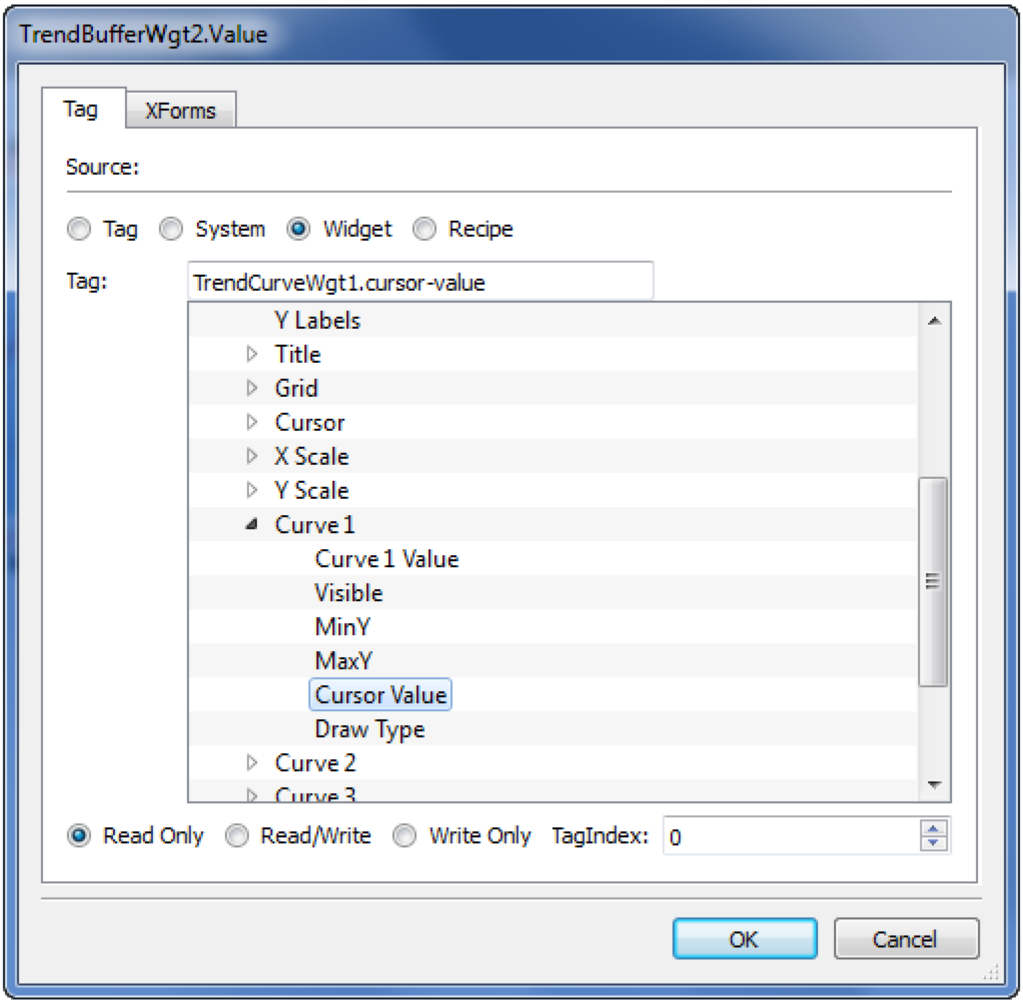
Trend Cursor

The trend cursor allows you to see the trend value at a point.

1. Use *Show Trend Cursor* macro and *Scroll Trend Cursor* macro to enable the trend cursor and move it to the required point to get the value of the curve at the particular instant in time.



2. To display the value of the trend cursor on the page, define a numeric field and attach the cursor value widget tag. This is the Y axis value of the cursor.


TrendBufferWgt2.Value

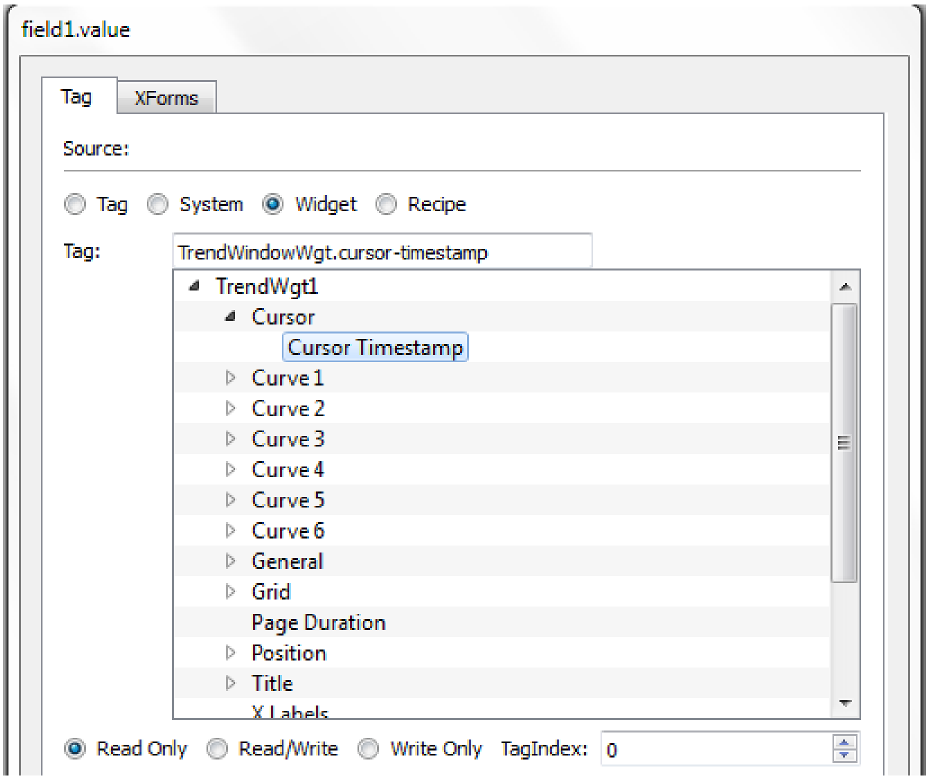
Source:

○ Tag ○ System ● Widget ○ Recipe

Tag: TrendCurveWgt1.cursor-value

● Read Only ○ Read/Write ○ Write Only TagIndex: 0

3. To get the time at the particular point where the cursor is placed, define a numeric field and attach to the “Widget” tag as shown in the figure below.



field1.value

Source: ○ Tag ○ System ● Widget ○ Recipe

Tag: TrendWindowWgt.cursor-timestamp

● Read Only ○ Read/Write ○ Write Only TagIndex: 0

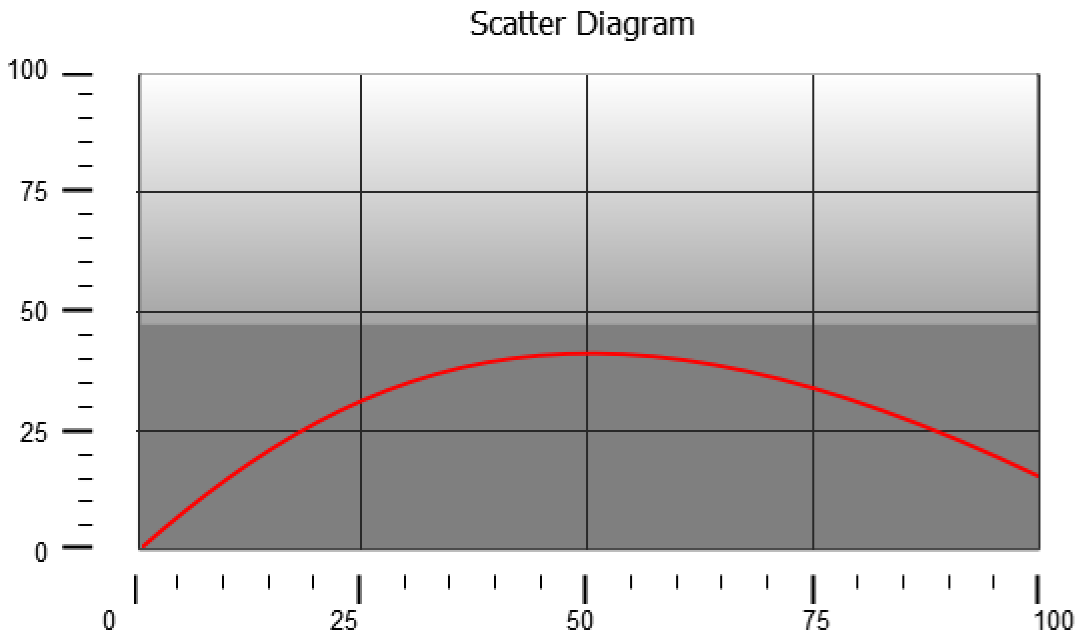
The “Widget” tag represents the X axis cursor value for the trend window.

Exporting Trend Buffer Data to CSV file

The trend buffers stored in the selected media can be exported to CSV file using dedicated actions. Please refer to the chapter “Dump Trend” for further information.

Scatter Diagram / XY Graph

A scatter diagram is a type of mathematical diagram using Cartesian coordinates to display values for two variables from a set of data. The data is displayed as a collection of points, each having the value of one variable determining the position on the horizontal axis and the value of the other variable determining the position on the vertical axis.



In Scatter Diagram a linear interpolation of points is done.

To create a new Scatter Diagram you have to proceed as follows:

1. Add Scatter Diagram widget into the page
2. Select the number of curves (Graph1...Graph5) to show
3. Customize the general graph properties as for Trends like X Min, X Max, Grid details
4. Define the max number of samples/values to consider (Max Samples) for each curve. This parameter set the max number of values to show in the graph starting from first element in the array. Ex. Tag1[20] and Max Samples = 10 will show just first 10 elements of the Tag1 array.
5. Define for each curve the two Tags of type Array to show (X-Tag and Y-Tag).

When the array tags change, it is possible to force a refresh using the dedicated macro RefreshTrend.

> **NOTE**
>
> The ScatterDiagram is considered as a different type of Trend Widget. However, only the RefreshTrend macro is supported for it.

Data Transfers

The Data Transfer feature allows the transfer of variable data from one device to another.

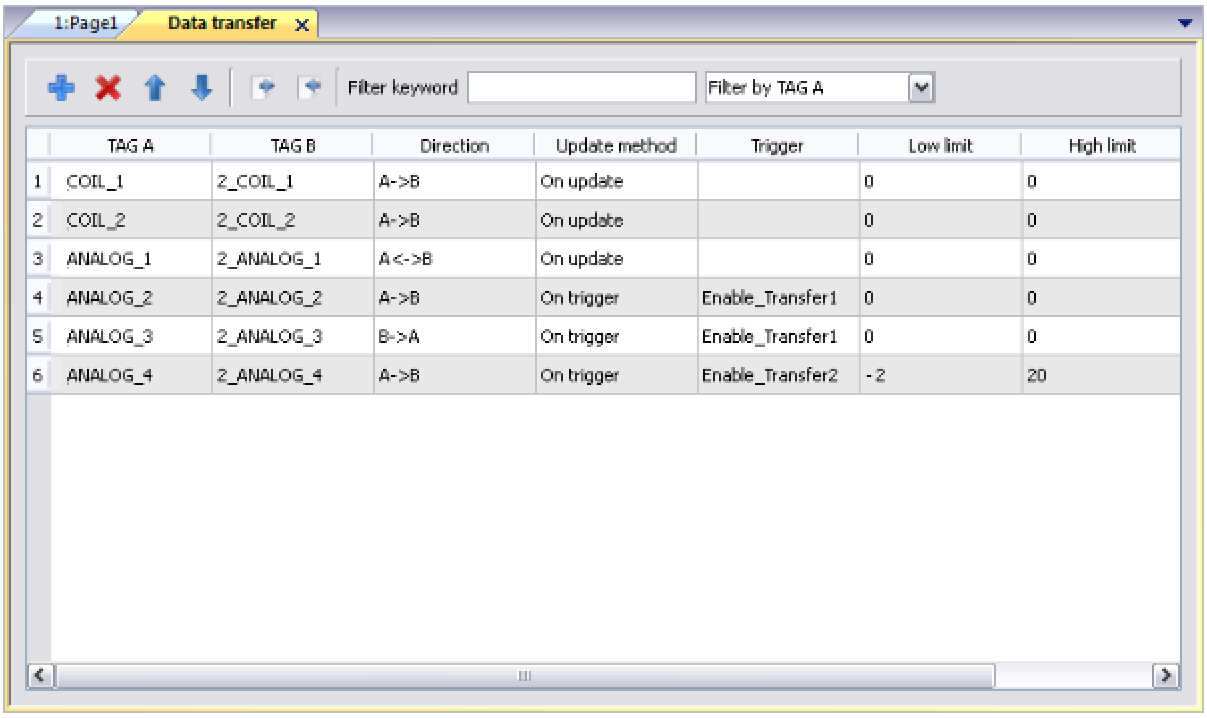
Using this feature an HMI panel can operate as a gateway between two devices, even if they do not use the same communication protocol.

The Data Transfer Editor

To configure each data transfer job, you need to correctly map the tags. This mapping is performed from the Data Transfer editor.

To configure the data transfer:

1. Double click on Config mode.
2. Double click on Data Transfer item.
3. To add a tag, click on the “+" icon: a new tag line is added.



	TAG A	TAG B	Direction	Update method	Trigger	Low limit	High limit
1	COIL_1	2_COIL_1	A->B	On update		0	0
2	COIL_2	2_COIL_2	A->B	On update		0	0
3	ANALOG_1	2_ANALOG_1	A<->B	On update		0	0
4	ANALOG_2	2_ANALOG_2	A->B	On trigger	Enable_Transfer1	0	0
5	ANALOG_3	2_ANALOG_3	B->A	On trigger	Enable_Transfer1	0	0
6	ANALOG_4	2_ANALOG_4	A->B	On trigger	Enable_Transfer2	-2	20

Each line in the Data Transfer editor defines a mapping rule for the alignment of the two tags.

You can define more mapping rules if you need different update methods or directions.

Data Transfer Toolbar Buttons

Import/Export

Data Transfer settings can be imported and exported in .csv format. This feature can be effectively used whenever it is more convenient to perform changes directly in the .csv file and then reimport the modified file.

Filter keyword

Sorts only rows containing that keyword. Click on the list box to select the column where you need to apply the filter.

Data Transfer Fields

Names of the Pair of Tags

Tag A / Tag B are names of the pair of tags to be mapped in order to be exchanged through the HMI panel.

Direction

A->B and B->A: Unidirectional transfers, values are always received by one tag and sent by the other tag in the specified direction.

A<->B: Bidirectional transfer, values are transferred to and from both tags.

Update Method

OnTrigger: Data transfer occurs when the value of the tag set as the trigger changes above or below the values set as boundaries of a tolerance range. Limits are recalculated on the previous tag value, the same that triggered the update.

> **NOTE**
>
> This method applies only to unidirectional transfers (A->B or B->A).

OnUpdate: Data transfer occurs whenever the value of the source tag changes.

> **NOTE**
>
> This method applies both to unidirectional and to bidirectional transfers (A->B, B->A and A<->B).

> **NOTE**
>
> When using the OnUpdate method, the first tags shown at start-up of the HMI panel may not correspond to current values. Content is synchronized the first time the source value changes

Trigger, High limit, Low limit

Tag values that trigger the data transfer process. When this tag changes its value outside the boundaries set as High limit and Low limit, data transfer is started. The range of tolerance is recalculated according to the specified limits on the tag value which triggered the previous update. No action is taken if the change falls within the set limits.

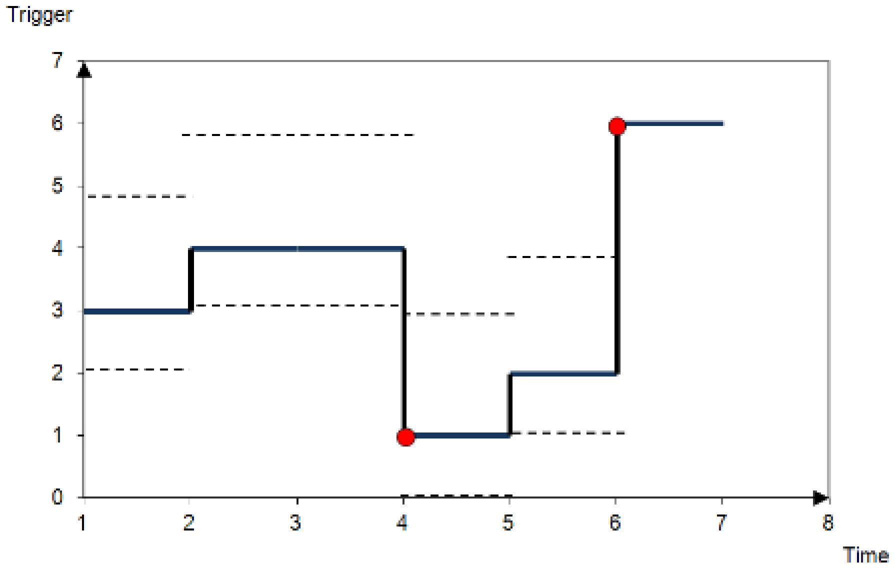
This mechanism allows triggering data transfers only when there are significan variations of the reference values.

> **NOTE**
>
> If both Low limit and High limit are set to "0", data transfer is triggered as soon as there is a change in the value of the trigger tag.

> **NOTE**
>
> Low limit is less or equal to zero.

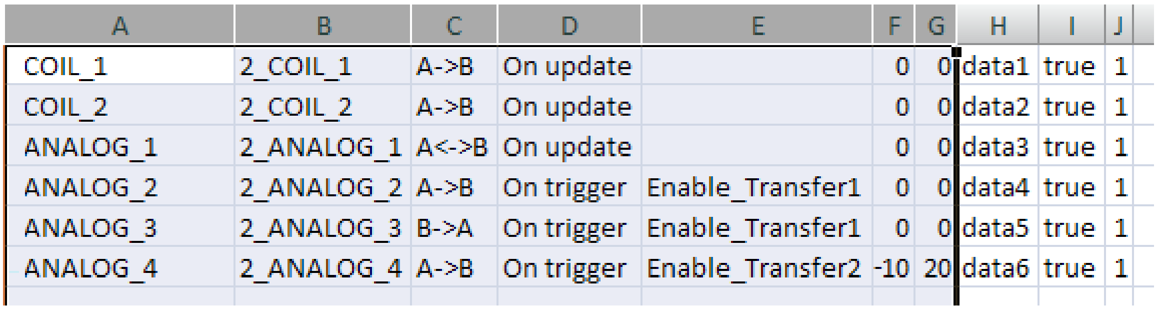
Below an example where:

- High limit=1,9
- Low limit=- 0,9
- • = points where the data transfer is triggered



Exporting Data to .csv Files

Configuration information for data transfers exported to a .csv file. Example is shown in figure.



A	B	C	D	E	F	G	H	I	J
COIL_1	2_COIL_1	A->B	On update		0	0	data1	true	1
COIL_2	2_COIL_2	A->B	On update		0	0	data2	true	1
ANALOG_1	2_ANALOG_1	A<->B	On update		0	0	data3	true	1
ANALOG_2	2_ANALOG_2	A->B	On trigger	Enable_Transfer1	0	0	data4	true	1
ANALOG_3	2_ANALOG_3	B->A	On trigger	Enable_Transfer1	0	0	data5	true	1
ANALOG_4	2_ANALOG_4	A->B	On trigger	Enable_Transfer2	-10	20	data6	true	1

Columns A through G contain the same data as in the Data Transfer editor. Some additional columns are present.

Column H: Unique identifier automatically associated by the Data Transfer to each line. When you edit the .csv file and you add one extra line, make sure you enter a unique identifier in this column.

Columns I – J: Reserved for future use,

Data Transfer Limitations and Recommendations

Correct definition of data transfer rules is critical for the good performance of the HMI panels.

To guarantee reliability of operation and performance keep in mind the following rules:

- The OnTrigger method allows only unidirectional transfers, (A->B or B->A)
- The OnUpdate method allows changing the values in accordance with the direction settings only when the source value changes. No action is performed at boot time of the HMI
- Panel Builder 600 is not a supervisory system. Its performance depends on:
    - number of data transfers defined in the Data Transfer editor,
    - number of data transfers eventually occurring at the same time,
    - number and size of features used in the project (i.e. tags, Alarms, Trends…).
    - Always test performance of operation during project development.
    - If inappropriately set, data transfer tasks can lead to conditions where the tags involved create a loop. Identify and avoid such conditions.

Offline Node Management

When one of the devices communicating with the HMI panel goes offline, this may reduce the overall communication performance of the system.

The offline node management feature recognizes offline devices and removes them from communication until they come back online.

Additionally if you know that any of the devices included in the installation is going to be offline for a certain time, you can manually disable it to maximize system performance.

NOTE

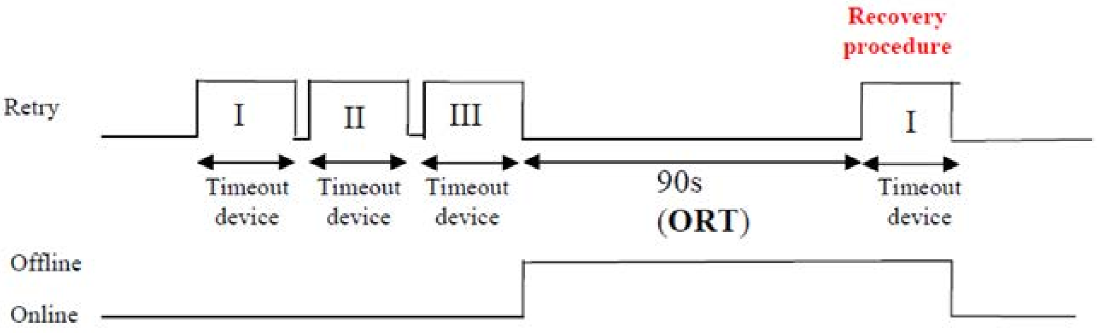
This feature is not supported by all communication protocols. Check protocol documentation to know if it is supported or not.

Offline Node Management Process

Steps of the process are:

- A certain device is online and it is regularly polled by the system: if the device does not answer to a poll, the system polls it again twice before declaring the device offline.
- When a device is offline, the system polls the device with a longer interval, called Offline Retry Time (ORT). If the device answers to the poll, the system declares it online and starts polling it at regular intervals.

The following schema shows the three polling attempts and the recovery procedure that starts when the Offline Retry Time is elapsed:



Manual Offline Node Management Process

Offline Node Management can be done manually.

- A specific device on a Node ID is online and it is regularly polled by the system.
- Using a macro action connected to a button the user can declare the device offline: the system stops polling it.
- Another macro action can be used to declare the device online: the system restarts polling it at regular intervals.

Manual Offline Configuration

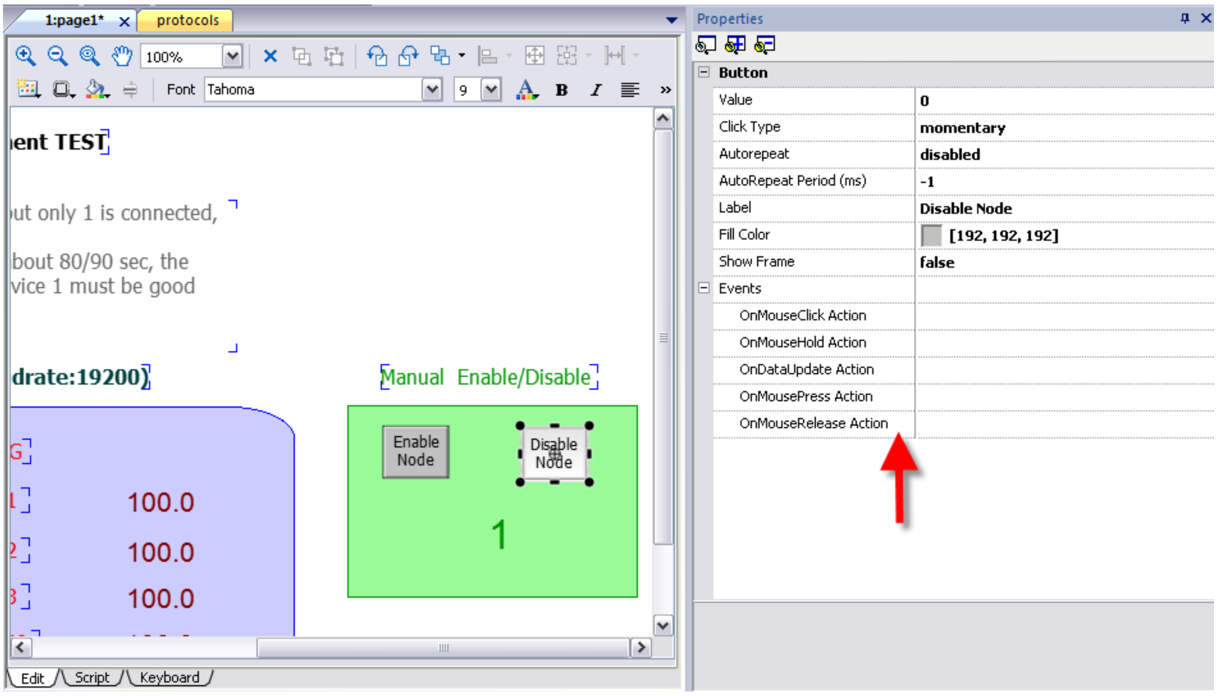
When you know that some devices in communication with the HMI are going to remain offline for a certain period of time, you can exclude them from data polling using the **Enable node** macro action.

For example, you can customize your page to contain a button and associate it to an action that will allow you to exclude and/or include a specific device node as needed.

The following example explains how to create a button that, when pressed, will disable an associated device.

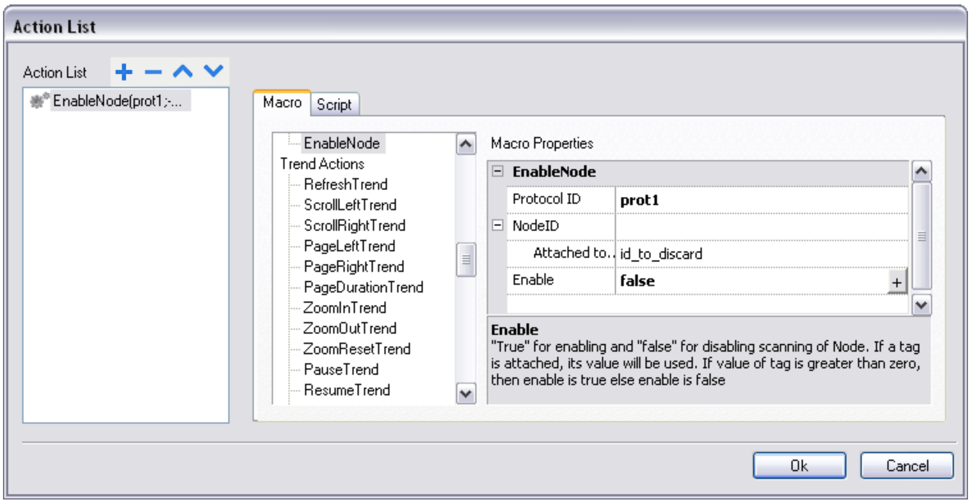
To do this:

1. In a page of your project add a button.
2. Associate an event to the button (for example **OnMouseRelease**)



Button	
Value	0
Click Type	momentary
Autorepeat	disabled
AutoRepeat Period (ms)	-1
Label	Disable Node
Fill Color	[192, 192, 192]
Show Frame	false
Events	
OnMouseClick Action	
OnMouseHold Action	
OnDataUpdate Action	
OnMousePress Action	
OnMouseRelease Action	

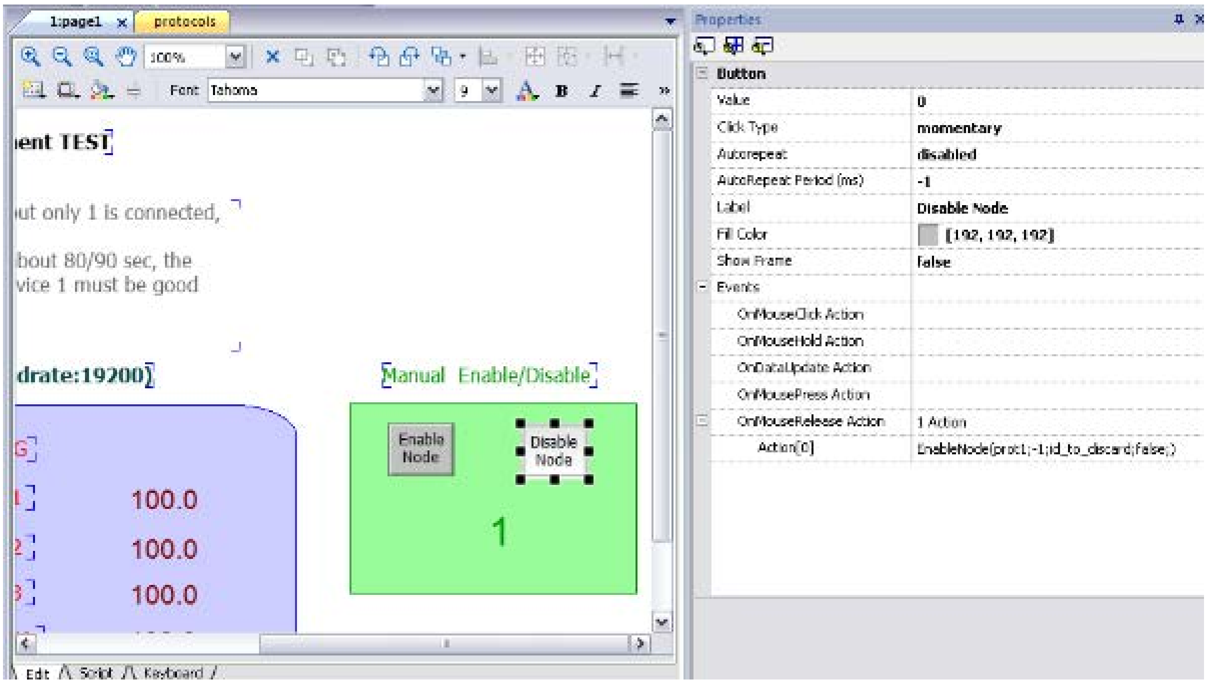
3. Click on the event row, click the '+' button and select Add action
4. Add the Tag Action EnableNode to the event (Tag Action -> EnableNode).
5. Make sure that the Enable field is set to “false”.



EnableNode	
Protocol ID	prot1
NodeID	
Attached to..	id_to_discard
Enable	false

Enable
"True" for enabling and "false" for disabling scanning of Node. If a tag is attached, its value will be used. If value of tag is greater than zero, then enable is true else enable is false

6. Enter the correct Protocol and Device ID and click Ok.



Button	
Value	0
Click Type	momentary
Autorepeat	disabled
AutoRepeat Period (ms)	-1
Label	Disable Node
Fill Color	▨ [192, 192, 192]
Show Frame	false
Events	
OnMouseClick Action	
OnMouseHold Action	
OnDataUpdate Action	
OnMousePress Action	
OnMouseRelease Action	1 Action
Action[0]	EnableNode(prot1;-1;id_to_discard;false;)

7. The event set is shown in the row. The associated device is indefinitely disabled and therefore no longer polled for data collection.

In the situation described above, you may want to create another button to re-enable the device when needed. In this case the Enable field will have to be set to “true”.

CAUTION! All disabled device nodes will remain disabled if the same project is downloaded on the panel, on the other hand, if a different project is downloaded, all disabled devices will be re-enabled. The same happens on package update.

TIP To make this feature more dynamic, you may decide not to indicate a specific NodeID but attach it to the value of a tag or to an internal variable created to identify different devices that might be installed in your network.

NOTE When using the action **Enable Node** described above to force a device node back online, data polling will start immediately.

Automatic Offline NodeDetection

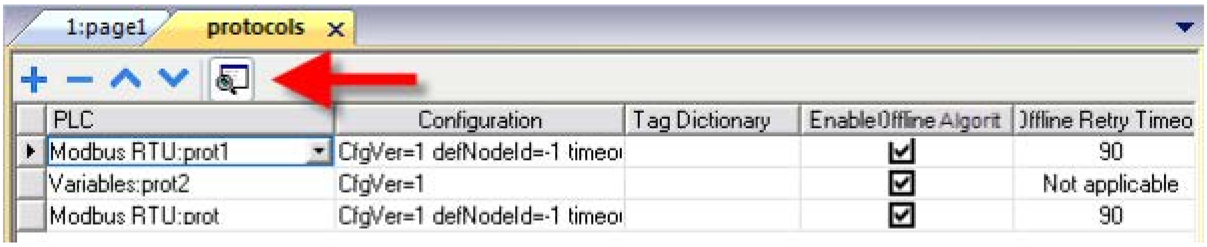
HMI panels can automatically disregard connected devices which are found to be offline.

When a device is found offline the first time, it is polled twice before being disregarded.

When it is declared offline it is polled at different intervals that can be set by the user.

To set the offline polling on one node ID:

1. Click the Config node and click Protocols.
2. Select the desired node ID.
3. Click on the Show Advanced Properties button: more columns are added to the table.
4. In the table set the Offline Retry Time parameter: the device on this node ID will be polled with this frequency when offline.



PLC	Configuration	Tag Dictionary	EnableOffline Algorit	)ffline Retry Timeo
Modbus RTU:prot1	CfgVer=1 defNodeId=-1 timeo		☑	90
Variables:prot2	CfgVer=1		☑	Not applicable
Modbus RTU:prot	CfgVer=1 defNodeId=-1 timeo		☑	90

Offline Management Toolbar Buttons

Advanced properties shows / hides the advanced properties columns.

Offline Management Fields

PLC	Protocol type and name
Configuration	Protocol settings
Tag dictionary	Tags imported for the protocol
Enable Offline Algorithm	Enable the Offline Management for the protocol
Offline Retry Time	Interval, expressed in seconds, between when the node was disregarded and when the recovery procedure started. Max value for ORT is 86400sec (24h).

Multi-Language

A true multilanguage feature has been implemented in PB610 Panel Builder 600 through code pages support from the Microsoft Windows systems. The multilanguage feature handles different code pages for the different languages. A code page (or a script file) is a collection of letter shapes used inside each language.

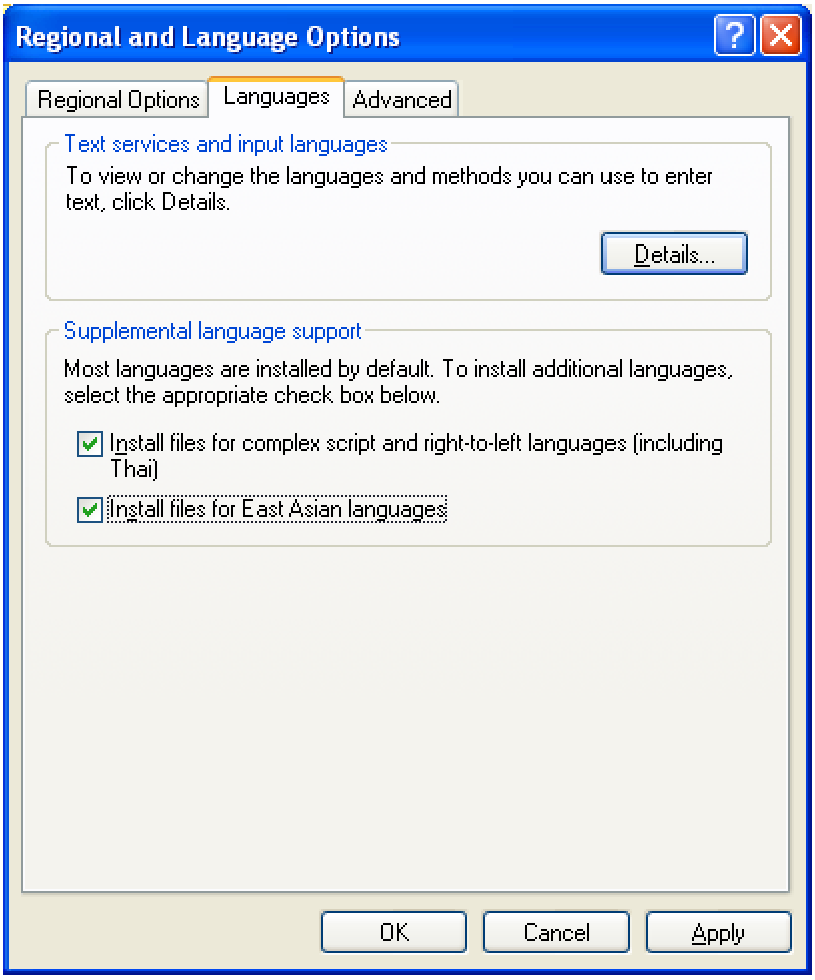
The multilanguage feature can be used for a project by defining languages and character sets. PB610 Panel Builder 600 also extends the truetype fonts (TTF) provided by Windows systems to provide different font faces associated with different character sets.

PB610 Panel Builder 600 has features to allow users to provide strings for each of the languages.

When in edit mode, PB610 Panel Builder 600 provides support to change the display language from language combo. This helps users see the page's look and feel at design time itself.

NOTE

In Windows XP operating systems, for the proper operation of the multiple language editor in PB610 Panel Builder 600, you will need to install the support for complex script and East Asian languages.



Regional and Language Options

Text services and input languages

To view or change the languages and methods you can use to enter text, click Details.

Supplemental language support

Most languages are installed by default. To install additional languages, select the appropriate check box below.

☑ Install files for complex script and right-to-left languages (including Thai)

☑ Install files for East Asian languages

PB610 Panel Builder 600 is actually supporting a restricted set of fonts for the Chinese languages.

For Simplified Chinese following fonts are supported:

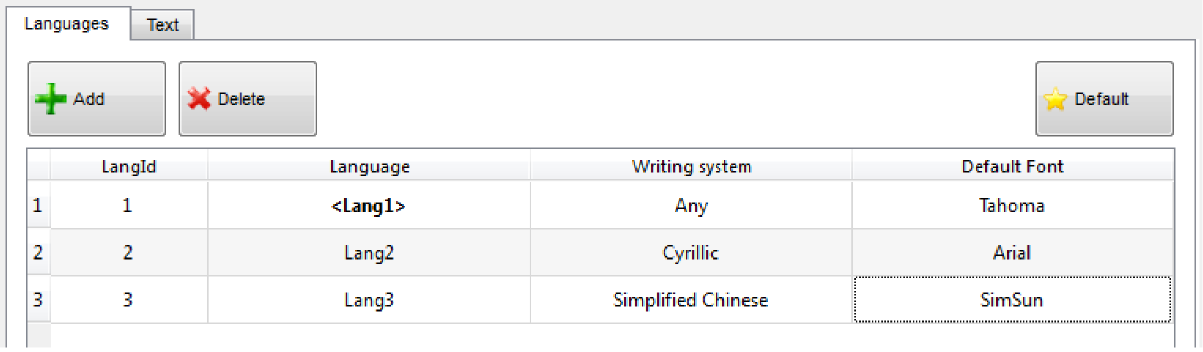
- Fangsong - simfang.ttf
- Arial unicode MS - ARIALUNI.TTF
- Kaiti - simkai.ttf
- Microsoft Yahei - msyh.ttf
- NSImsun - simsun.ttc
- SimHei - simhei.ttf
- Simsun - simsun.ttc

For the Traditional Chinese, following fonts are supported:

- DFKai-SB - kaiu.ttf
- Microsoft Sheng Hai - msjh.ttf
- Arial unicode MS - ARIALUNI.TTF
- MingLiU - mingliu.ttc
- PMingLiU - mingliu.ttc
- MingLiU_HKSCS - mingliu.ttc

Add a Language to Project

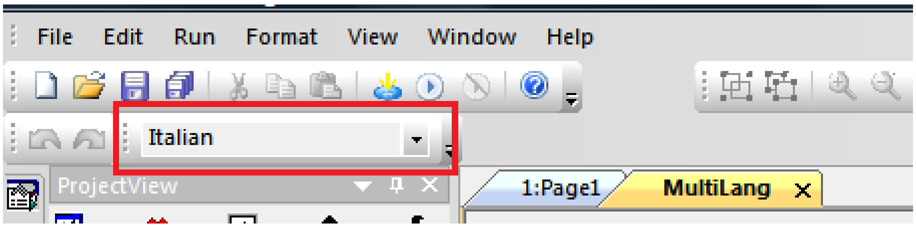
1. Launch multilanguage from the Project View pane.
2. Click the **Add** button to add the language.
3. Select the writing system and the default font used by all the “table like” widgets (such as alarms or events).
4. Use the **Default** button to set the default language used when PB610 Panel Builder 600 Runtime starts multilanguage.



	LangId	Language	Writing system	Default Font
1	1	<Lang1>	Any	Tahoma
2	2	Lang2	Cyrillic	Arial
3	3	Lang3	Simplified Chinese	SimSun

Language Display Combo

This combo can be used to change language at the design phase. This helps users to view the page in the different supported languages at design time itself.



Italian

Multi Language Widget

Multilanguage support is available for different objects, like push buttons, static text, message, alarm description and the pop-up messages.

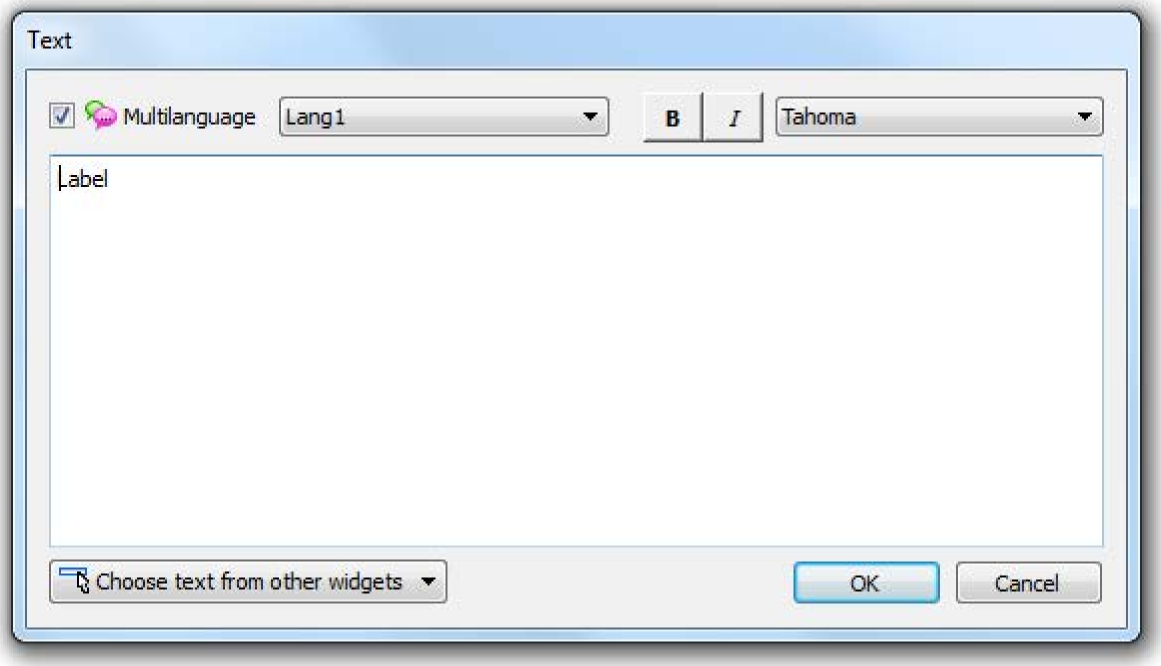
Multi Language for Static Text Widget

1. Double click a text widget on the page.

   ➤ The dialog shown below opens.

2. You can edit the text for the selected languages and select the font.

The bold, italic and color properties are set for all the languages globally for the widget. Text for each of the languages can be given, by selecting the language from combo. However, it is recommended to use the export and import features, as described in chapter "Export and Import of Multilanguage Strings".



Text

☑ Multilanguage: Lang1

Font: Tahoma

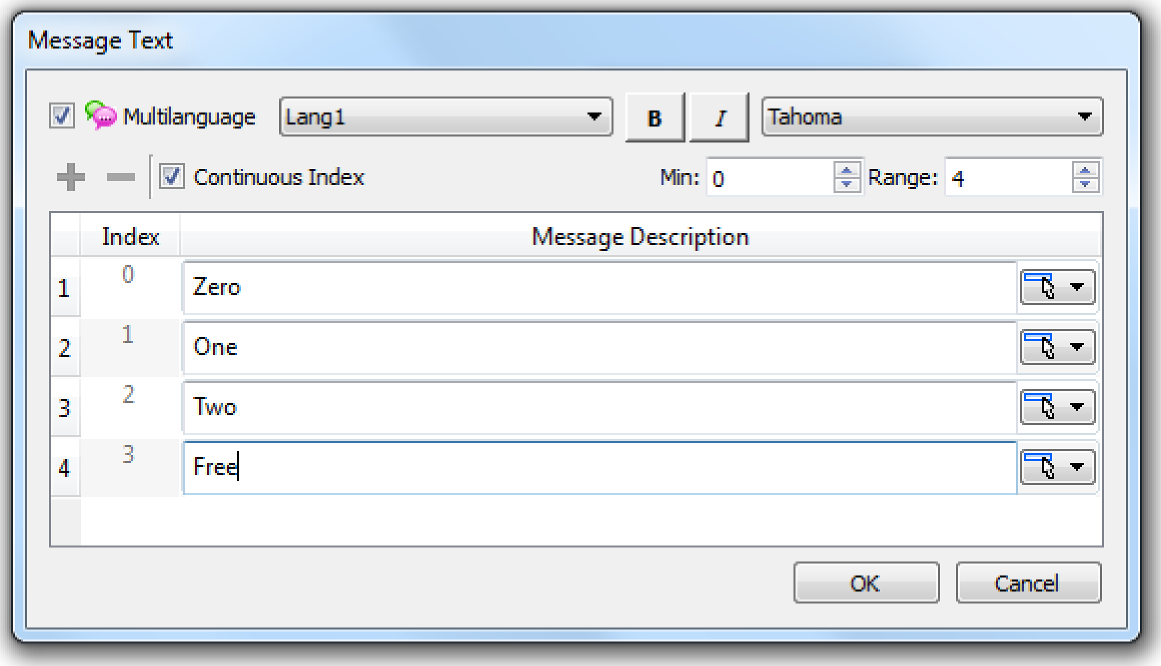
Label

Multi Language for Message Widget

PB610 Panel Builder 600 allows you to use the multilanguage in the message widget.

✓ You dragged and dropped a message widget.

1. Select the language from the language combo and enter the message description for the selected language.
2. Again, you can also use export and import strings as described in chapter "Export and Import of Multilanguage Strings.



☑ Multilanguage: Lang1

Font: Tahoma

☑ Continuous Index

Min: 0

Range: 4

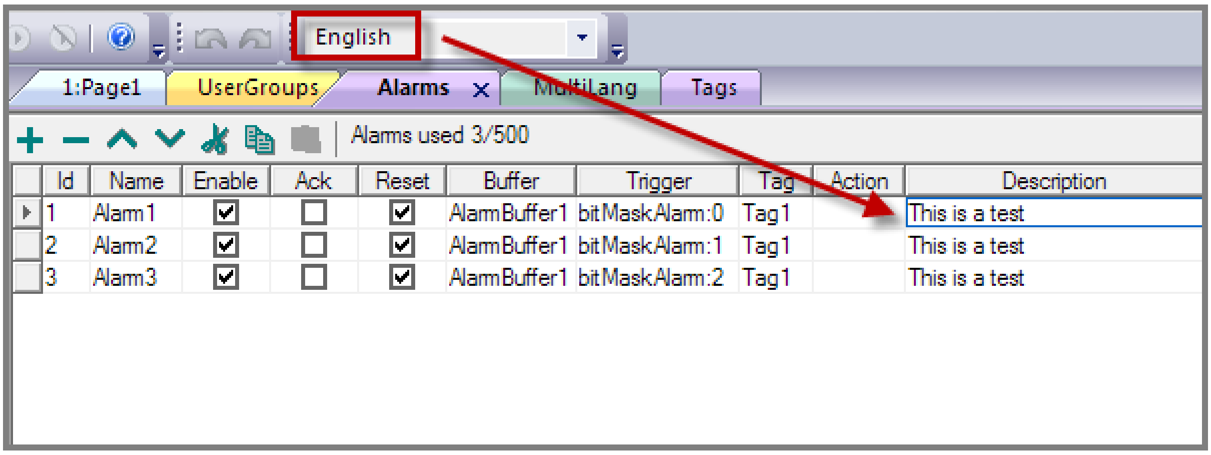
	Index	Message Description
1	0	Zero
2	1	One
3	2	Two
4	3	Free

Multi Language for Alarm Messages

PB610 Panel Builder 600 allows you to use multilanguage for alarm messages.

1. To add a multilanguage string for an alarm message, open the alarm editor and select the language list from the tool bar (language combo).
2. Add the alarm messages.

You can also use the export and import features, as described in chapter "Export and Import of Multilanguage Strings".



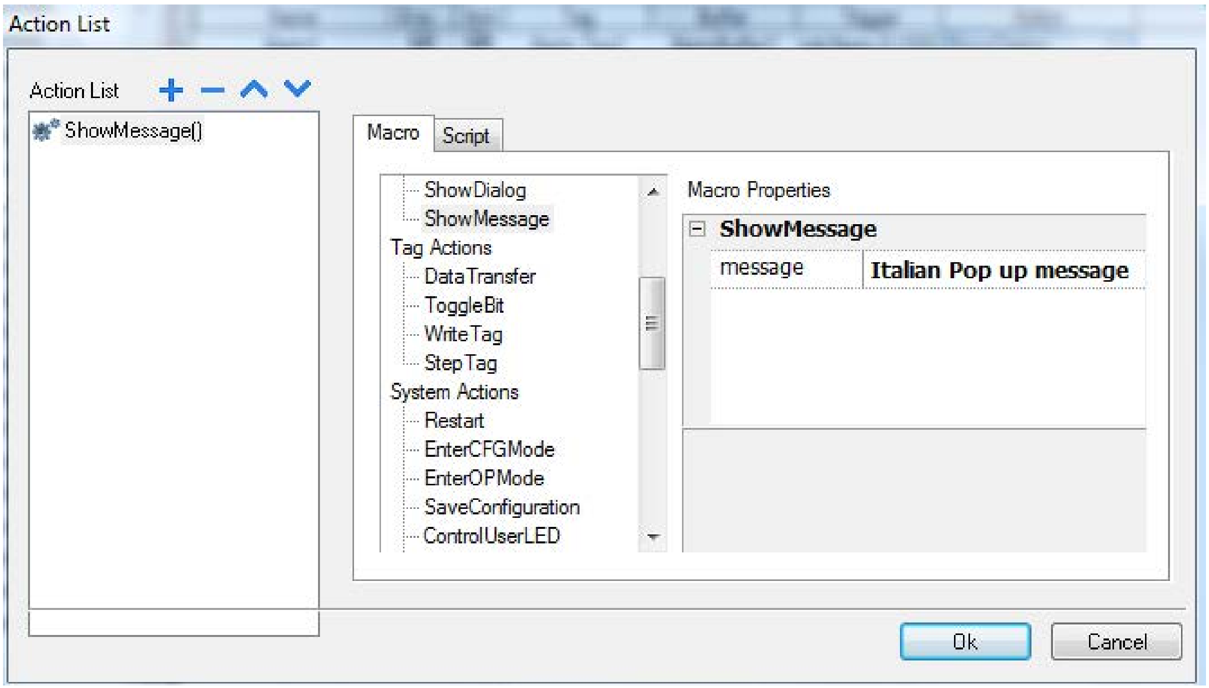
Alarms used 3/500

Id	Name	Enable	Ack	Reset	Buffer	Trigger	Tag	Action	Description
1	Alarm1	☑	□	☑	AlarmBuffer1	bitMaskAlarm:0	Tag1		This is a test
2	Alarm2	☑	□	☑	AlarmBuffer1	bitMaskAlarm:1	Tag1		This is a test
3	Alarm3	☑	□	☑	AlarmBuffer1	bitMaskAlarm:2	Tag1		This is a test

Multi Language for Pop-up Messages

For the pop-up message macro, you can define the multilanguages.

1. Select the language from language list combo.
2. Enter the message in the *show message* macro.



ShowMessage

message: Italian Pop up message

Export and Import of Multilanguage Strings

The easiest way to translate a project into multiple languages is to use the export feature, exporting all text to an external file. The translation can be executed in that documents, then using the import, bring all text for all languages back into the project.

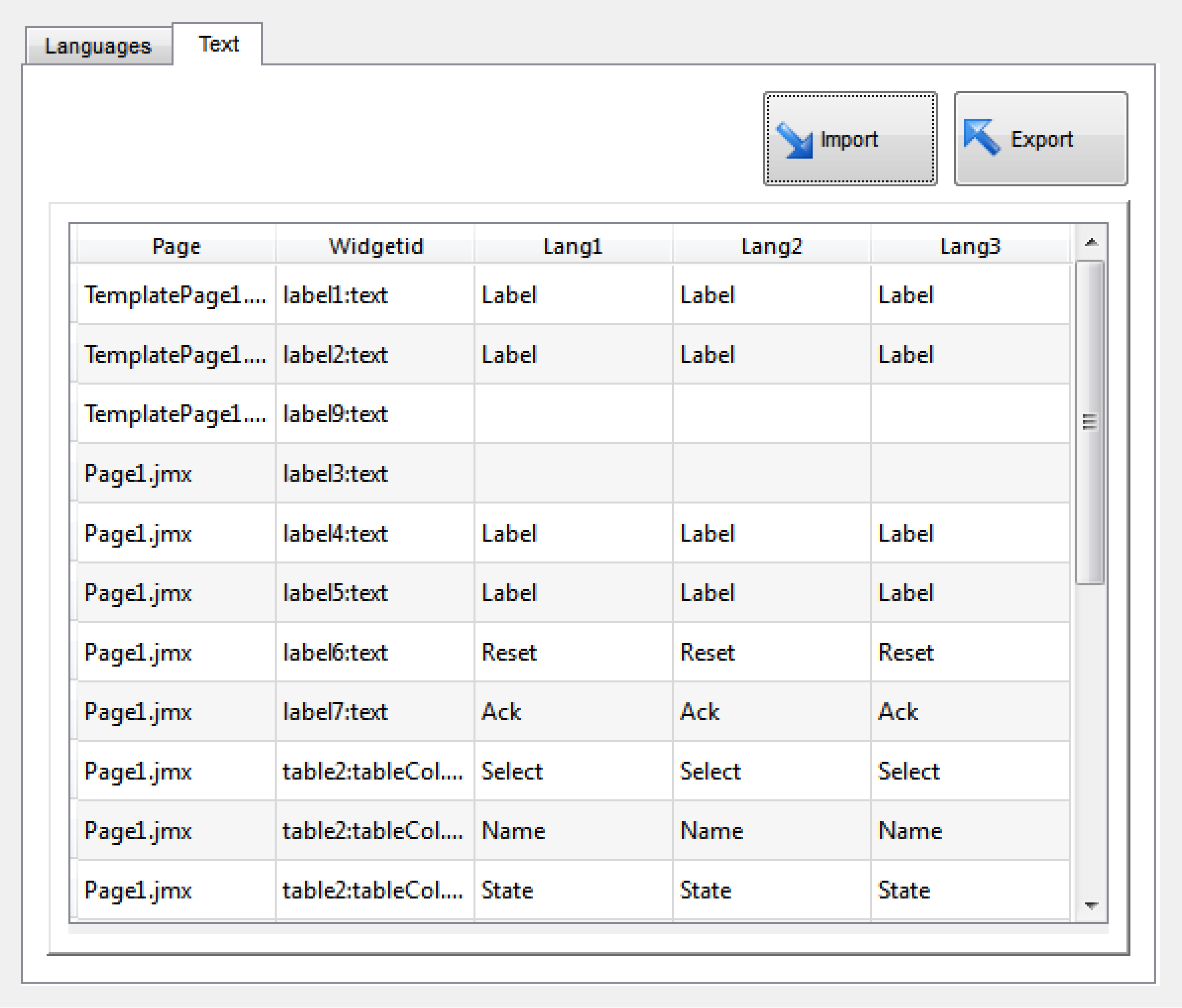
The multilanguage strings will be exported in CSV file format, then you can modify the strings with an external editor, and import it back to the PB610 Panel Builder 600.

The CSV file exported by PB610 Panel Builder 600 is coded in unicode. To edit it, you need a specific tool that supports a CSV file encoded in a unicode format.

1. To export the multilanguage string, open the multilanguage editor and switch to *Text* view.
2. Click the **Export** button and save the exported CSV file.
3. You can modify the exported CSV file and **Import** back to PB610 Panel Builder 600.
4. Click **Save** button to save the text.

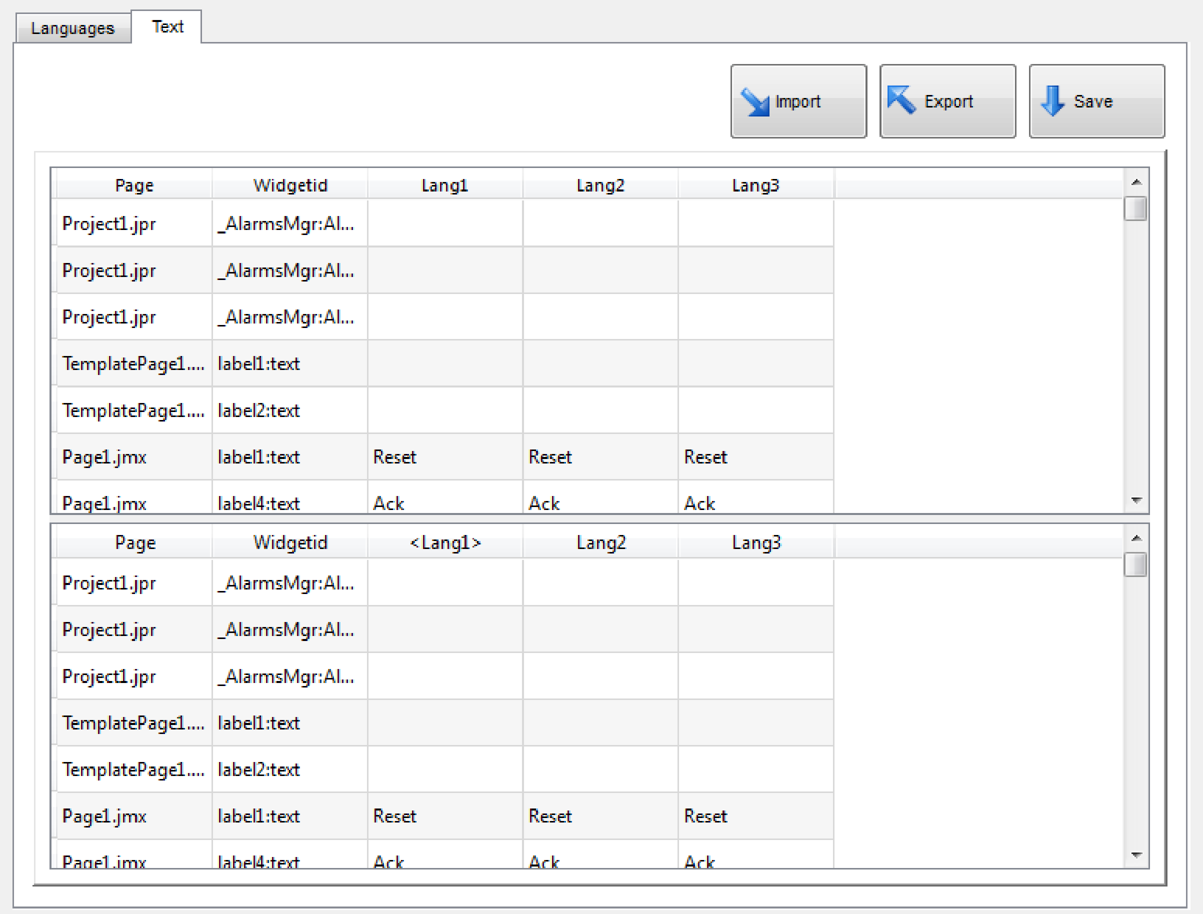
NOTE

It is recommended that you set all languages that will be used in the project before exporting the file. This will guarantee that the exported file will contain all columns and language definitions for that project.



Page	Widgetid	Lang1	Lang2	Lang3
TemplatePage1....	label1:text	Label	Label	Label
TemplatePage1....	label2:text	Label	Label	Label
TemplatePage1....	label9:text			
Page1.jmx	label3:text			
Page1.jmx	label4:text	Label	Label	Label
Page1.jmx	label5:text	Label	Label	Label
Page1.jmx	label6:text	Reset	Reset	Reset
Page1.jmx	label7:text	Ack	Ack	Ack
Page1.jmx	table2:tableCol....	Select	Select	Select
Page1.jmx	table2:tableCol....	Name	Name	Name
Page1.jmx	table2:tableCol....	State	State	State

The strings are imported matching the widget ID and the page number of each widget. To change the separator used in the exported file, please have the regional settings of your work PC changed. Upon importing, the separator information is retrieved from the file. If not found, the default of “,” is used. Immediately after the import, the modified strings will be displayed in the text tab. Once the user clicks the **Save** button, the changes are saved to the internal widgets.

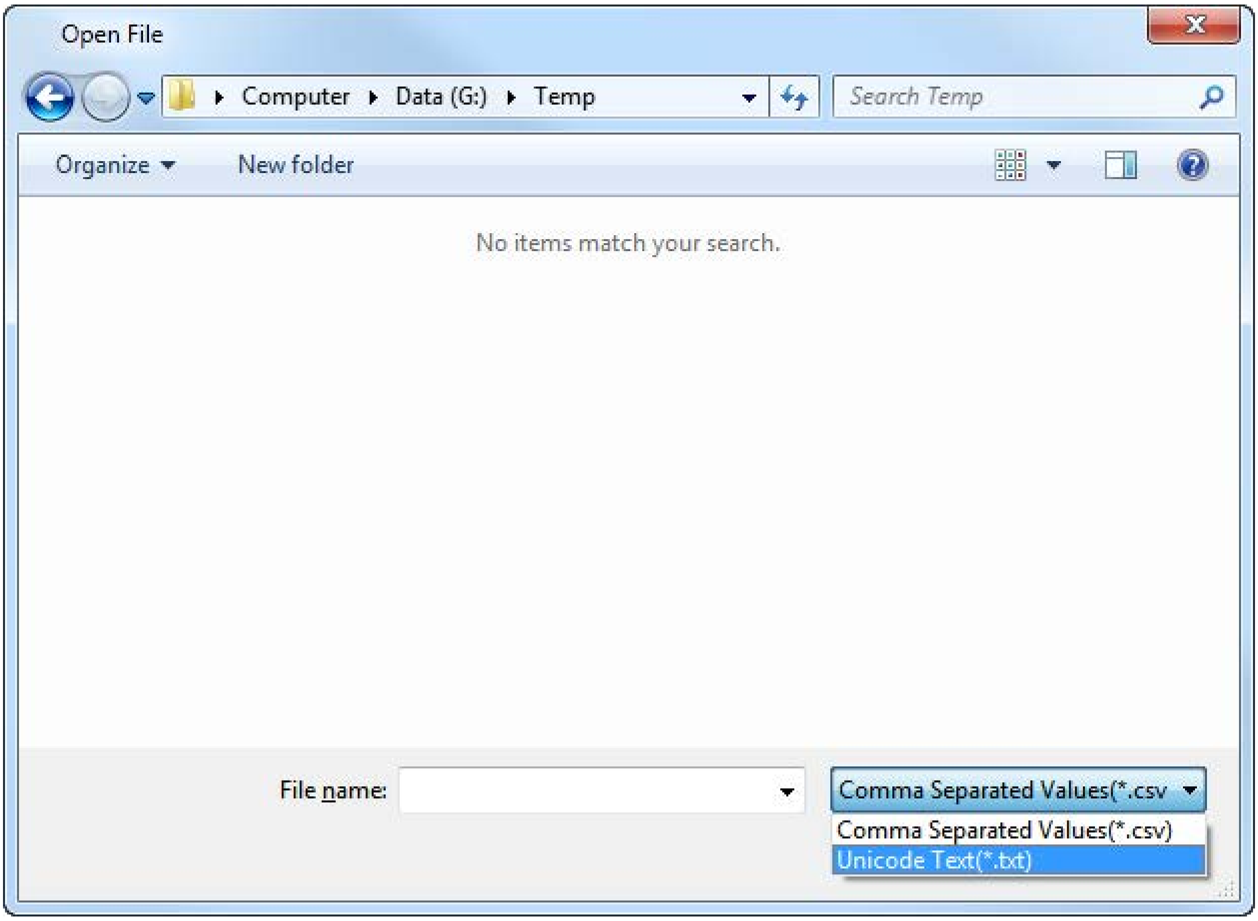


Page	Widgetid	Lang1	Lang2	Lang3
Project1.jpr	_AlarmsMgr:Al...			
Project1.jpr	_AlarmsMgr:Al...			
Project1.jpr	_AlarmsMgr:Al...			
TemplatePage1....	label1:text			
TemplatePage1....	label2:text			
Page1.jmx	label1:text	Reset	Reset	Reset
Page1.jmx	label4:text	Ack	Ack	Ack

Page	Widgetid	<Lang1>	Lang2	Lang3
Project1.jpr	_AlarmsMgr:Al...			
Project1.jpr	_AlarmsMgr:Al...			
Project1.jpr	_AlarmsMgr:Al...			
TemplatePage1....	label1:text			
TemplatePage1....	label2:text			
Page1.jmx	label1:text	Reset	Reset	Reset
Page1.jmx	label4:text	Ack	Ack	Ack

The feature Import supports two formats

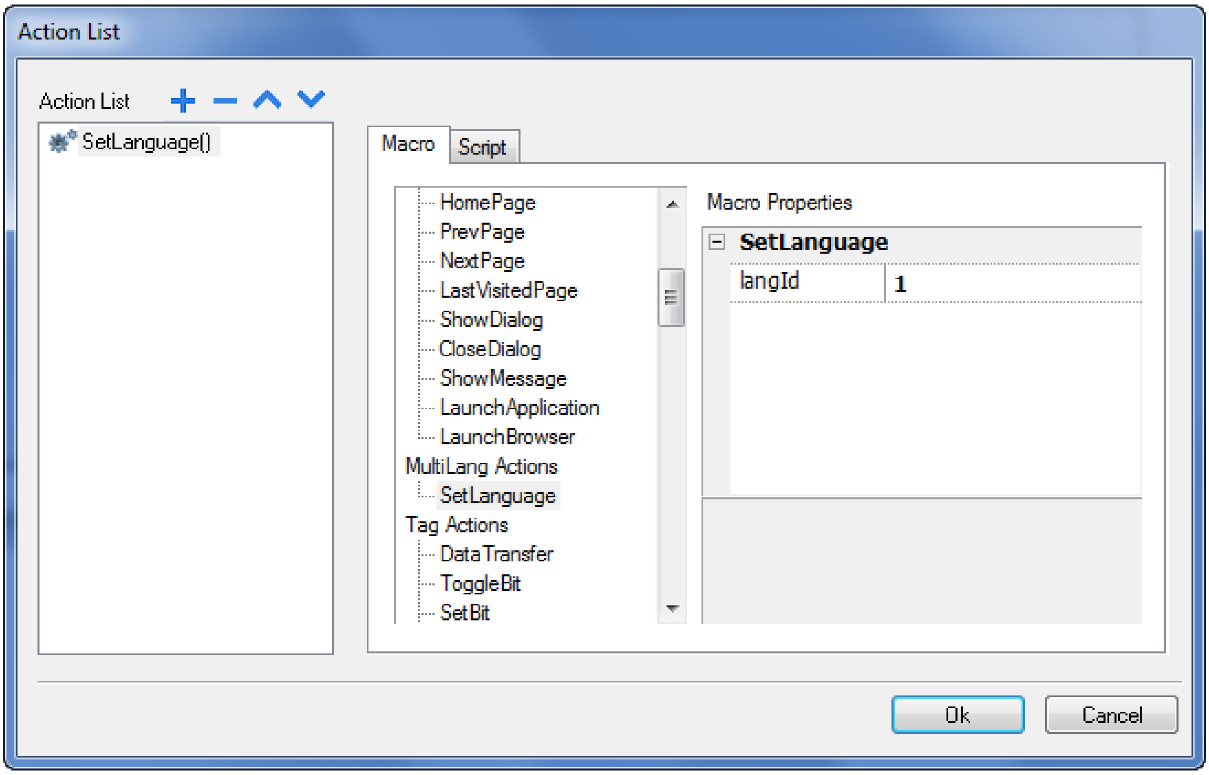
- Comma separated values (.csv)
- Unicode text (.txt)


Open File
Computer ▸ Data (G:) ▸ Temp
Search Temp
Organize ▾
New folder
No items match your search.
File name:
Comma Separated Values(*.csv
Comma Separated Values(*.csv)
Unicode Text(*.txt)

The Unicode Text file format must be used every time you import a file modified by Microsoft® Excel®. You can save your Excel® sheet in this format choosing File > Save As… and choose the option Unicode Text (*.txt) from the Save as type: combo.

Change Languages at Runtime

After the project download, PB610 Panel Builder 600 Runtime will start with the default language. You can change the language on runtime using the *Set language* macro.



langId 1

The language index corresponds to the language ID, as it can be read from the language configuration editor.

NOTE

After languages are changed on runtime with the macro execution, the changed language is saved and retained for the next run.

Limitations in UNICODE support

Panel Builder 600 has been designed for working with UNICODE text. However, for compatibility reasons with all platforms, UNICODE is supported only in a subset of field types.

Area	Field	Charset Accepted	Reserved Chars/Strings
Protocol Editor	Alias	ASCII [32..126]	(space) , ; : . < * >'
Tag Editor	Name	ASCII [32..126]	. &
	Group	ASCII [32..126]	\ / * ? : > < | " & # % ;
	Comment	Unicode	
Trends	Name	ASCII [32..126]	\ / * ? : > < | " & # % ;
Printing Reports	Name	ASCII [32..126]	\ / * ? : > < | " & # % ;
Alarms	Name	ASCII [36..126]	\ / * ? : > < | " & # % ;
	Description	Unicode	[] - for live tags, \ escape seq for [and \

Area	Field	Charset Accepted	Reserved Chars/Strings
Events	Buffer Name	ASCII [32..126]	\ / * ? : > < | " & # % ;
Scheduler	Name	ASCII [32..126]	\ / * ? : > < | " & # % ;
Languages	Language Name	ASCII [32..126]	\ / * ? : > < | " & # % ;
	Texts in widgets	Unicode	
	Texts from import files	Unicode	
User Group	Group Name	a-z A-Z _	admin, guest unauthorized
	Comments	Unicode	
User	Name	ASCII [32..126]	\ / * ? : > < | " & # % ;
	Password	Unicode	
	Comment	Unicode	
Recipes	Name	ASCII [32..126]	\ / * ? : > < | " & # % ; ! $'()+,=@[]{}~`
	Set Name	ASCII [32..126]	\ / * ? : > < | " & # % ; ! $'()+,=@[]{}~`
	Element name	ASCII [32..126]	\ / * ? : > < | " & # % ; ! $'()+,=@[]{}~`
General	Project Name	A-Z,a-z,0-9,-,_	“PUBLIC”, “readme”, “index.html”
	Page Name	A-Z,a-z,0-9,-,_	
	Dialog Page Name	A-Z,a-z,0-9,-,_	
	Template Page Name	A-Z,a-z,0-9,-,_	
	Keypad Name	A-Z,a-z,0-9,-,_	
	Files (Images/Video/etc..)	A-Z,a-z,0-9,-,_	
	Widgets ID	A-Z,a-z,0-9,-,_	
Runtime	PLC Communication	UTF-8, Latin1, UCS-2BE, UCS-2LE, UTF-16BE, UTF-16LE	

Scheduler

The PB610 Panel Builder 600 provides a scheduler engine that can be easily configured to program the execution of specific actions at repeated intervals, on time basis.

Depending on your application, creating a schedule is typically performed by a 2-step process.

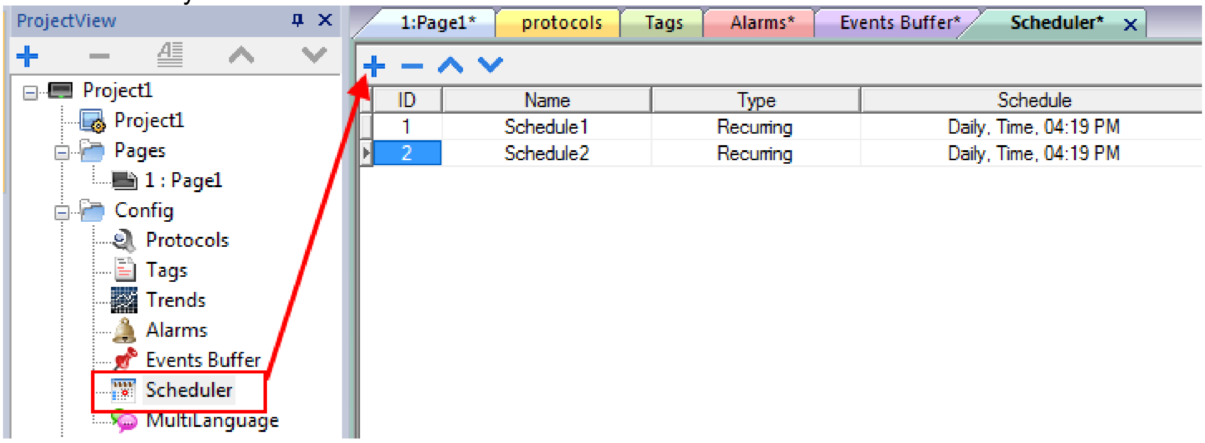
1. The first step is to define the parameters of the schedule that run on the panel. This includes selecting the actions to perform when the scheduled event is activated. The first step is performed using the Scheduler Editor.

2. The second step is to create a runtime user interface that allows the end-user to change settings per each defined user.
   For example the runtime user interface will allow the user to turn on a device at 5:00 pm, and turn the device off at 10:00 pm, every day. This can be done by dragging and dropping a predefined scheduler widget from the gallery, and placing it on the page. Once on the page, you can set the properties of the individual GUI elements to create the desired interface to be presented to the end-user.

Configuring the Scheduler Engine

The configuration of the scheduler engine is done using the scheduler editor. The scheduler editor is accessible from the “Scheduler” voice of the Project View pane.

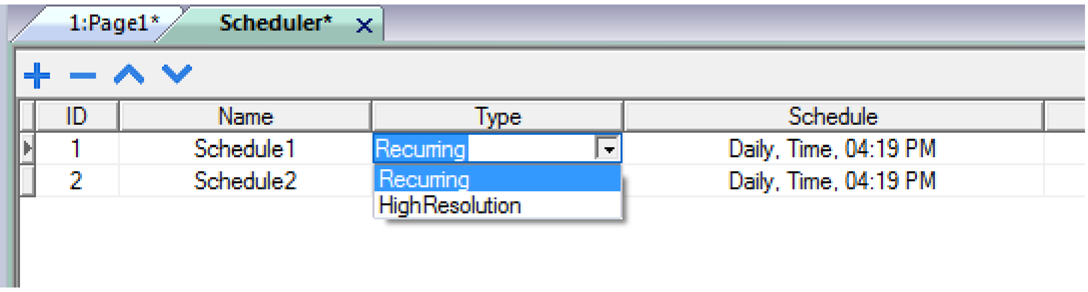
Click on the **+** symbol to add a schedule item.



ID	Name	Type	Schedule
1	Schedule1	Recurring	Daily, Time, 04:19 PM
2	Schedule2	Recurring	Daily, Time, 04:19 PM

Click on the “+" symbol to add a schedule item. Schedule items can be of two different types as listed below and shown in the figure below:

- Recurring Scheduler
- HighResolution Scheduler



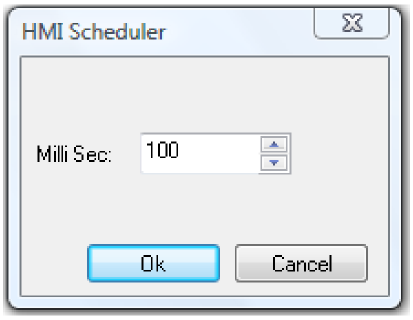
ID	Name	Type	Schedule
1	Schedule1	Recurring	Daily, Time, 04:19 PM
2	Schedule2		Daily, Time, 04:19 PM

Column	**Description**
Name	Scheduler name
Type	Type of scheduler (Recurring or High-Resolution)
Schedule	Different scheduler options, which are described in chapter “Recurrence Scheduler”, and in chapter “Type”.
Action	Defining macros, which have to be executed at the scheduled time
Priority	Setting a priority level for the event. This is used in case two distinct schedulers occur at the same time. The event with the higher priority will be executed before those of lower priority.

High Resolution Scheduler

This scheduler can be programmed to perform an action, or sequence of actions, repeatedly, at a specific duration. This scheduler can be set in milliseconds.

→ To configure the *high resolution* scheduler, select **High Resolution** from the type column and set the desired duration from the scheduler column.



HMI Scheduler

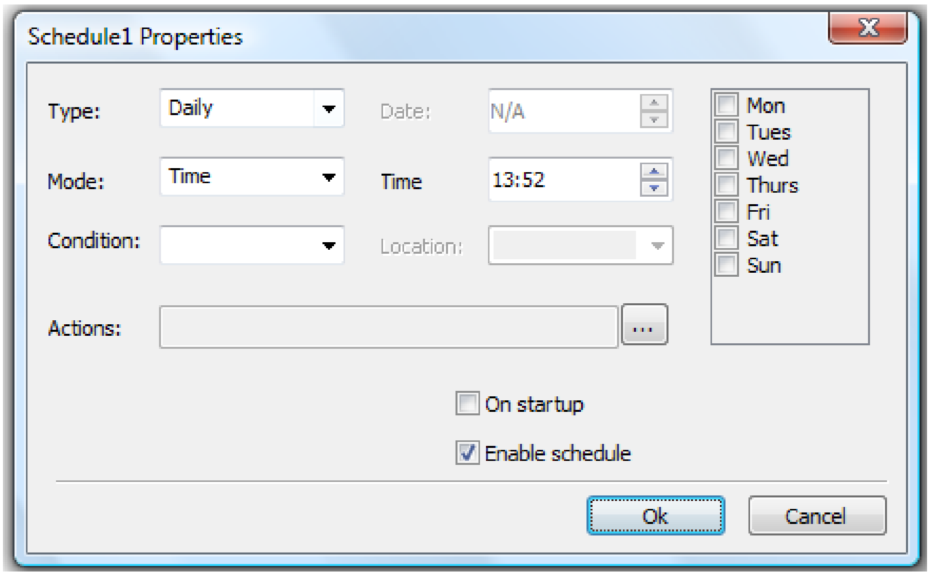
Milli Sec: 100

NOTE

This scheduler cannot be changed during runtime. If you want to change the schedule runtime, then use the recurrence scheduler by selecting **Every**, which is described in the following chapters. The minimum time resolution, when using a recurrence scheduler in **Every** mode, is one second.

Recurrence Scheduler

This scheduler can be programmed to perform an action, or sequence of actions, and the schedule can be modified during runtime.



Schedule1 Properties

Type: Daily
Date: N/A
Mode: Time
Time 13:52
Condition:
Location:
Actions:

□ Mon
□ Tues
□ Wed
□ Thurs
□ Fri
□ Sat
□ Sun

□ On startup
☑ Enable schedule

By default, when a schedule is added, the “Enable schedule” checkbox is marked. You have the option to keep a schedule in the project but disable it by unchecking the box.

Each scheduler can be configured to run once at startup (when the **On startup** check box is marked).

Additionally, you can specify by default, the scheduler to be enabled or not at first run by using the **Execute only at startup** check box.

Type

This combo allows you to select the type of the schedulers. You can change the type for the scheduler at any time during runtime, as described in chapter “Schedule the Events during Runtime”.

By Date

This schedule allows you to define the schedule for the specific date and time when the actions shall be executed.

Daily

This schedule defines the execution of a set of actions on a daily basis by specifying the time of day in which the actions are to be executed.

Every

This schedule is much like the *High Resolution* scheduler, with the possibility of changing it in the runtime. This schedule allows you to execute the macros with a specific time interval. The time interval can be set from 1 sec to 1 day.

Hourly

This schedule allows you to execute a set of actions on an hourly basis, by specifying the minute in which the actions have to be executed.

Monthly

This schedule allows you to execute a set of actions on a monthly basis, by specifying the day in which the actions have to be executed.

Weekly

This schedule allows you to execute a set of actions on a weekly basis by specifying the time and day(s) in which the actions have to be executed.

Yearly Schedule

This schedule allows you to execute a set of actions once a year, specifying the date and time in which the actions have to be executed.

Mode

Mode parameter is available for a subset of scheduler types. It’s not supported by scheduler of type Every, Hourly. This parameter allows choosing between following way of working:

Mode	Explanation
Time	This is the default. In this case is needed to specify details about time/date/week. Parameters depend on Type of scheduler selected.
Random10	Executed 10 minutes before/later the time specified. So, if time is 10:30, actions is executed in range 10:20...10:40 where 20...40 is random.
Random20	Executed 20 minutes before/later the time specified. So, if time is 10:30, actions is executed in range 10:10...10:50 where 10...50 is random.
Sunrise+	Executed n minutes/hours after sunrise time based on a specific location as explain in next chapter.
Sunrise-	Executed n minutes/hours before sunrise time based on a specific location as explain in next chapter.
Sunset+	Executed n minutes/hours after sunset time based on a specific location as explain in next chapter.
Sunset-	Executed n minutes/hours before sunset time based on a specific location as explain in next chapter.

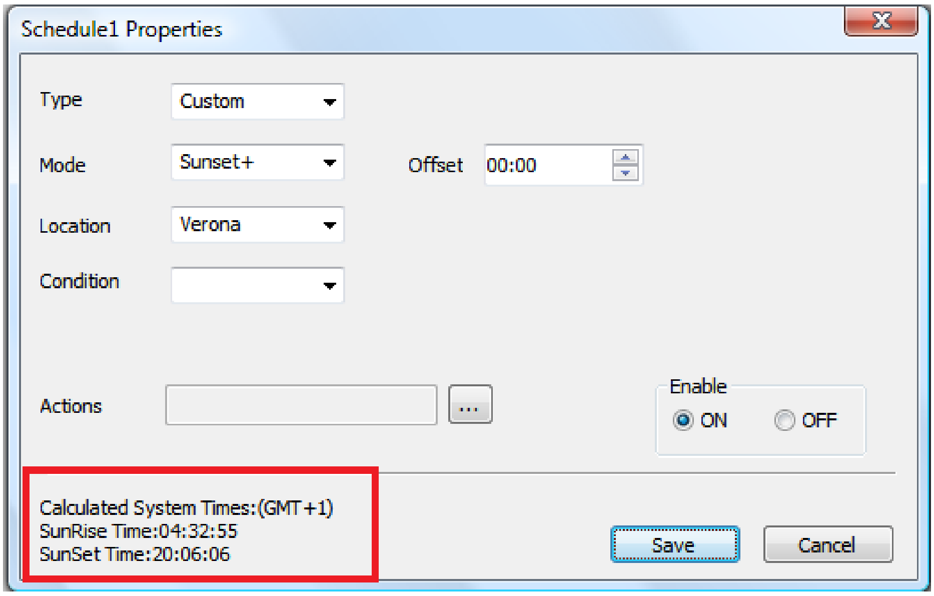
Configuring Location in PB610 Panel Builder 600

In PB610 Panel Builder 600 we have a unique feature that schedules based on sunrise and sunset. Before you start the sunrise or sunset scheduler, you need to define the location. Based on the UTC location, the system automatically calculates the sunrise and sunset time.

In the PB610 Panel Builder 600 installation, few locations are set by default. If your location does not show up under the list, you can add your location by entering the latitude, longitude and UTC information in the “Target_Location.xml” file under C:\Program Files\ABB\Panel Builder 600 Suite\Studio\config folder.

Example: The information for Verona City is as shown below: <file city="Verona" latitude="45.44" longitude="10.99" utc="1"/>

After entering the location information, the PB610 Panel Builder 600 displays the city name in the place combo list, and you can see the sunrise and sunset time on the dialog.



Schedule1 Properties

Type: Custom

Mode: Sunset+ Offset: 00:00

Location: Verona

Condition:

Actions:

Enable: ● ON ○ OFF

Calculated System Times:(GMT+1)
SunRise Time:04:32:55
SunSet Time:20:06:06

Condition

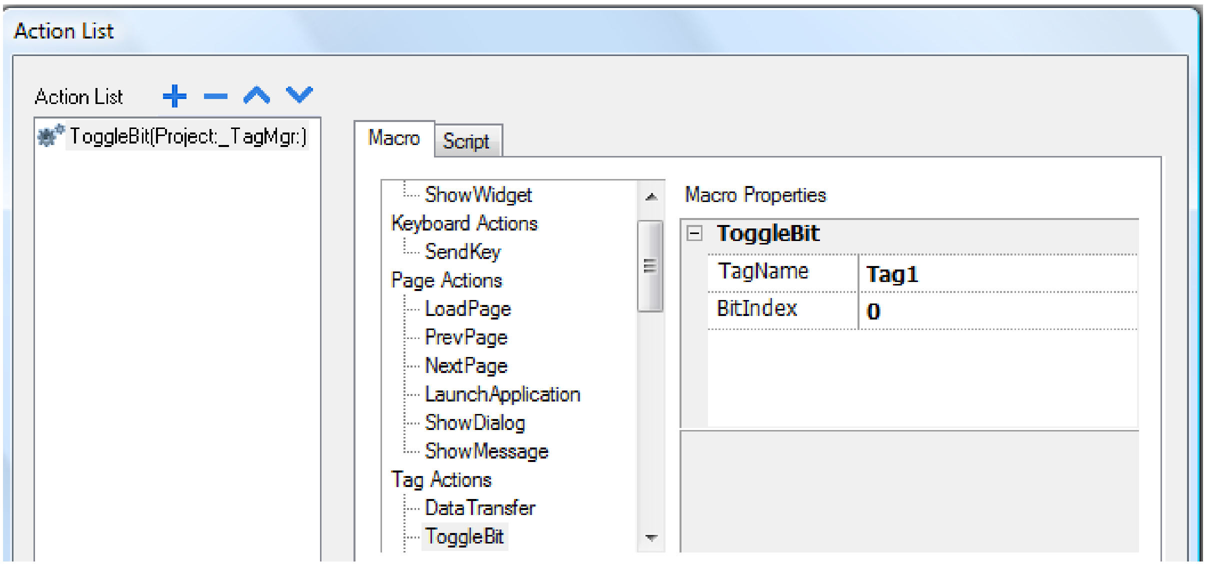
This combo allows you to select a boolean tag (Yes/No) to be evaluated, before activating the specified actions, at the moment the timer is triggered. If tag = true, the actions will be executed. If tag = false, the actions will not be executed.

Default value is “none”. The actions are executed when the timer is triggered.

NOTE The condition combo will list only the tag attached to the Boolean data type.

Actions

From the *Action List* dialog, you can add as many actions as desired. The actions will automatically be executed when the schedule time occurs.



ToggleBit

TagName	Tag1
BitIndex	0

NOTE

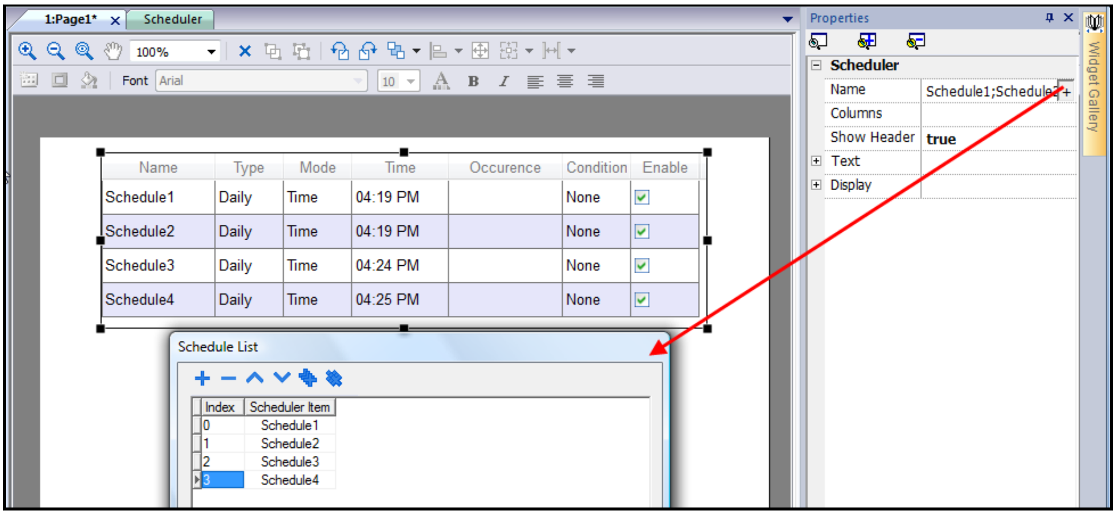
The actions should be programmed in the PB610 Panel Builder 600. Actions cannot be modified in runtime. All other scheduler parameters can be modified in runtime, such as type, mode, location etc.

Configuring the Schedule Interface for Runtime Interaction

The user interface for runtime is the widget called scheduler.

1. To add this to the project, just drag and drop it from the advanced section of the widget gallery.
2. In order to select the scheduler items to be displayed in the widget, click on the **+** button of the *Name* property that is part of the scheduler object.

   ➤ A dialog page opens where you can add the scheduler from the list at runtime.



Name	Type	Mode	Time	Occurence	Condition	Enable
Schedule1	Daily	Time	04:19 PM		None	☑
Schedule2	Daily	Time	04:19 PM		None	☑
Schedule3	Daily	Time	04:24 PM		None	☑
Schedule4	Daily	Time	04:25 PM		None	☑

Scheduler

Name	Schedule1;Schedule
Columns	
Show Header	true
Text	
Display	

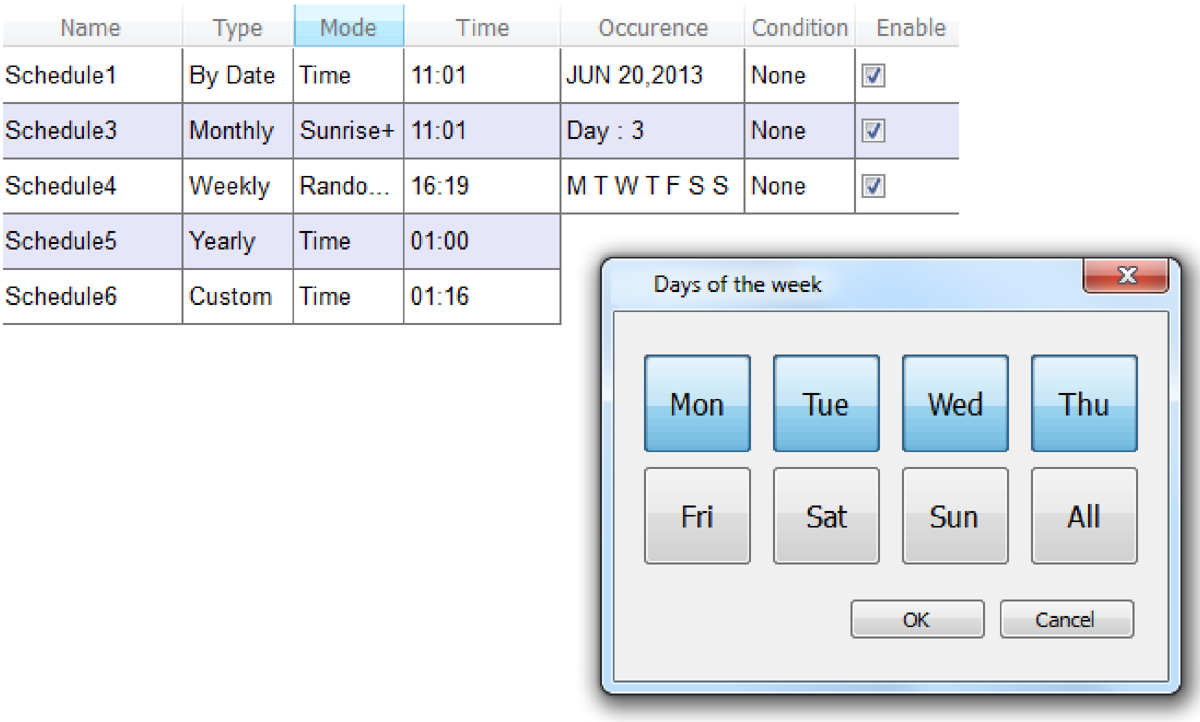
Index	Scheduler Item
0	Schedule1
1	Schedule2
2	Schedule3
3	Schedule4

Dialog page for adding the scheduler from the list

3. In the properties pane, you can customize the scheduler widget to adjust row colors, column width, show or hide column etc.

Schedule the Events during Runtime

If you defined the scheduler graphical user interface (GUI) on a page (as described in chapter "Configuring the Schedule Interface for Runtime Interaction", you can schedule the event and modify this schedule, during runtime on the server.
In runtime, the user has the flexibility to change all possible types and change the possibility to modes as described in the dedicated chapter.



Name	Type	Mode	Time	Occurence	Condition	Enable
Schedule1	By Date	Time	11:01	JUN 20,2013	None	☑
Schedule3	Monthly	Sunrise+	11:01	Day : 3	None	☑
Schedule4	Weekly	Rando...	16:19	M T W T F S S	None	☑
Schedule5	Yearly	Time	01:00			
Schedule6	Custom	Time	01:16			

Occurrence

This column specifies the date selected by the type of column, as shown in the figure.

Condition

This column lists the available boolean tags from the project. If a tag is selected as a condition, the scheduler will trigger only when the condition tag value is 1, otherwise the scheduler will not trigger.

Enable

This check box allows you to enable or disable the schedule. The scheduler will trigger when the enable check box is set. If you want to disable the scheduler temporarily, uncheck the Enable check box.

User Management and Passwords

This chapter describes the user management system. The main purpose of User management module is to restrict access to various objects/widgets and/or operations, by configuring user groups and their authorization level. Users, user groups and authorizations are the 3 entities used for users' handling.

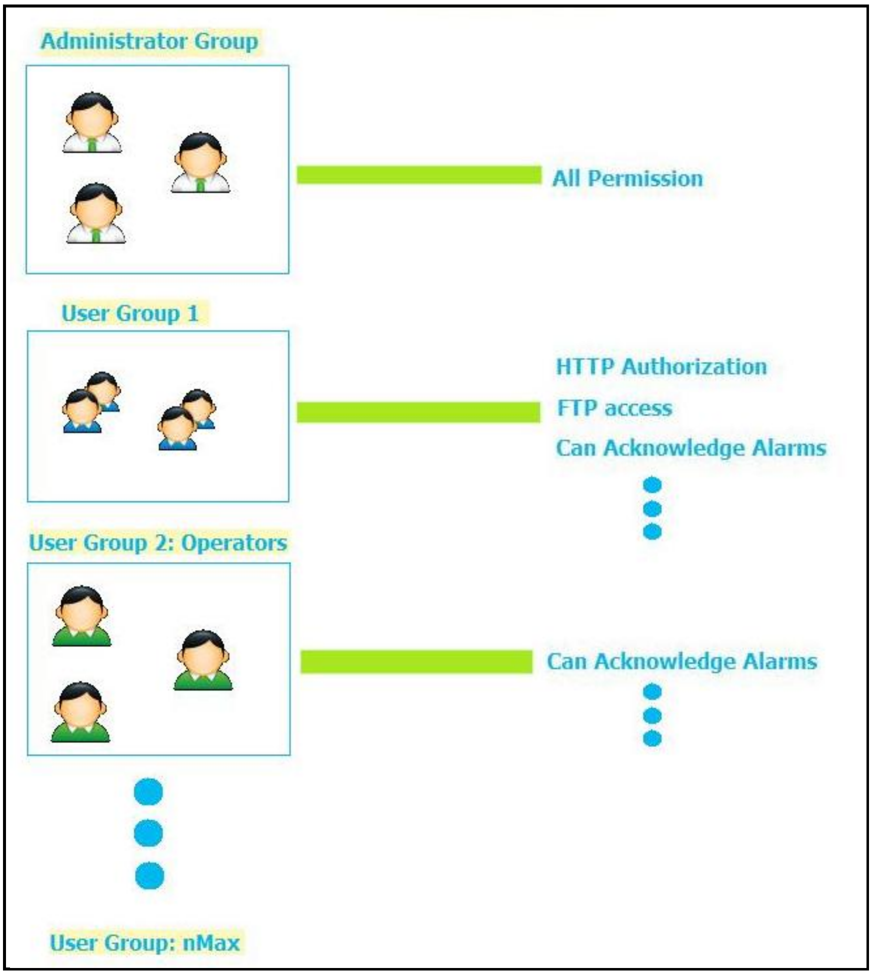
The basic entity is the user, representing an individual that has the need to work with the system.

Each user must be a member of a group. Users can be a member of just one group. Each group will have different types of authorizations and permissions assigned to them.

Authorizations and permissions for the groups are divided in two basic categories:

- Widgets' permissions (hide, read only, full access) and
- actions' permission (allowed or not allowed).

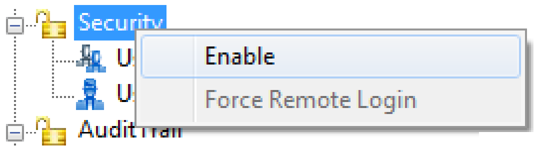
The proper combination of these will allow for the implementation of the necessary handling of security options for the application.



Configuring Security Options

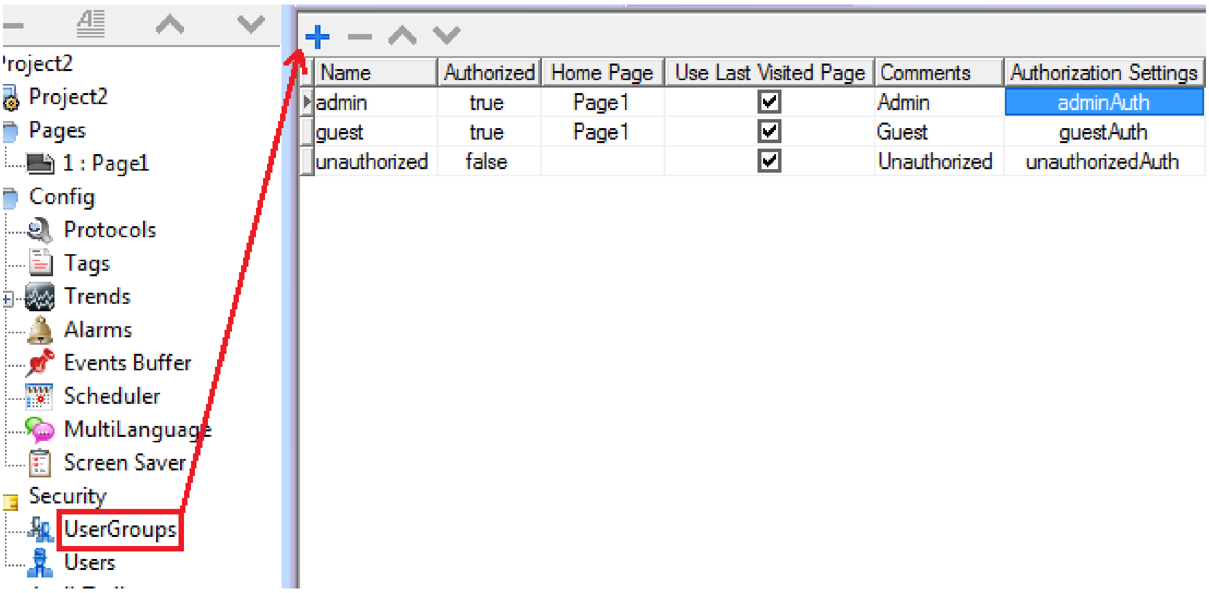
This section describes how to configure security-related settings in PB610 Panel Builder 600.

> **NOTE**
>
> To enable or disable the user management feature, right click on the *Security* folder in the Project View and set **Enable** or **Disable**.



Configuring Groups and Authorizations

1. Open *User Groups* to configure and assign their authorization in the ProjectView.



Name	Authorized	Home Page	Use Last Visited Page	Comments	Authorization Settings
admin	true	Page1	☑	Admin	adminAuth
guest	true	Page1	☑	Guest	guestAuth
unauthorized	false		☑	Unauthorized	unauthorizedAuth

2. Add new user groups by clicking the "**+**" Button.

   ➤ Three predefined groups are available by default.

The three predefined groups cannot be deleted and their names cannot be changed.

Predefined group's authorizations and comment fields can be instead changed according to application requirements.

For each group of users you can assign a home page. This means that, whenever a user from this user group is logged in, the selected home page for that group will appear.

There is one additional option called "Use Last Visited Page". If enabled, and a new user logs in, the page visited by the previous user will be displayed.

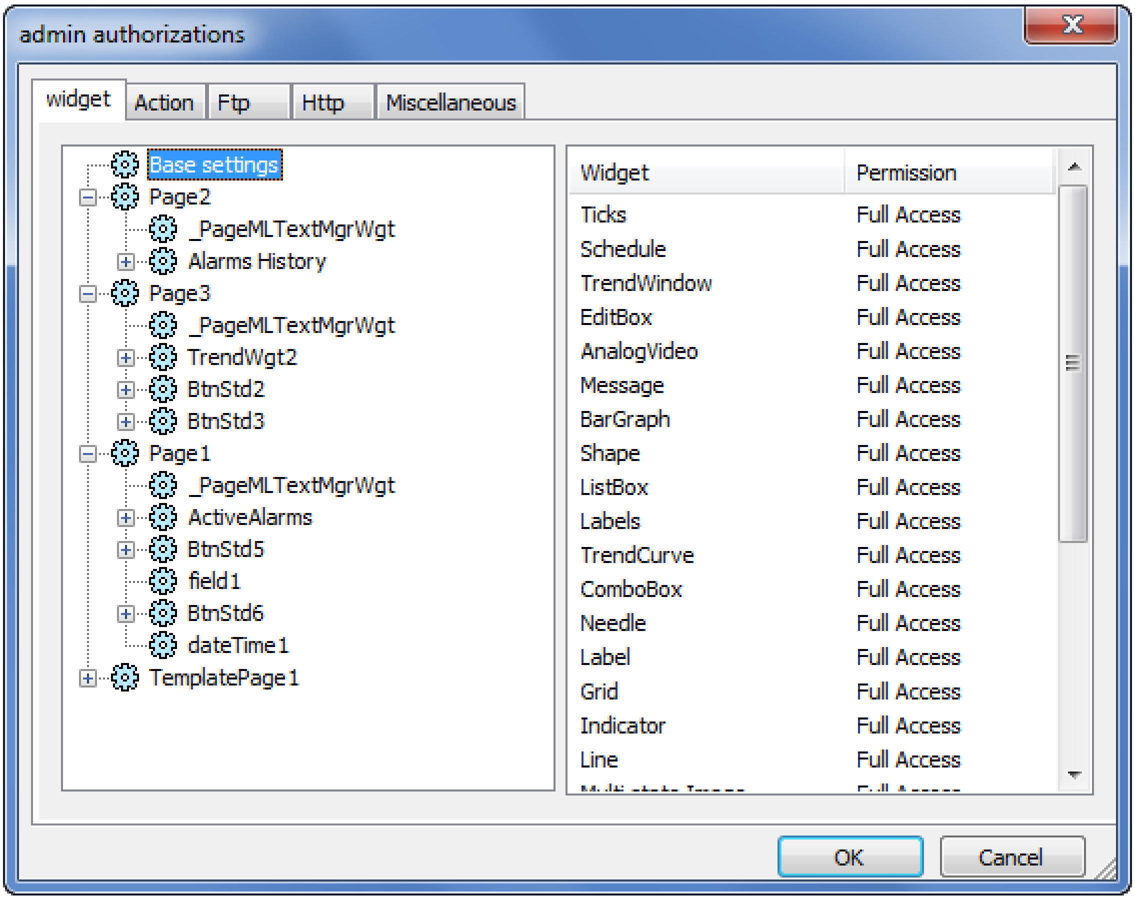
Modifying the Access Permission of Groups

→ To modify and assign the permissions, click the **Browse** button on the *Authorization Setting* column.

- You will see the “Admin Authorizations” dialog; here you have several tabs for the several available options.

Widget Permissions

The following figure shows the dialog where you can change the widgets’ permissions.

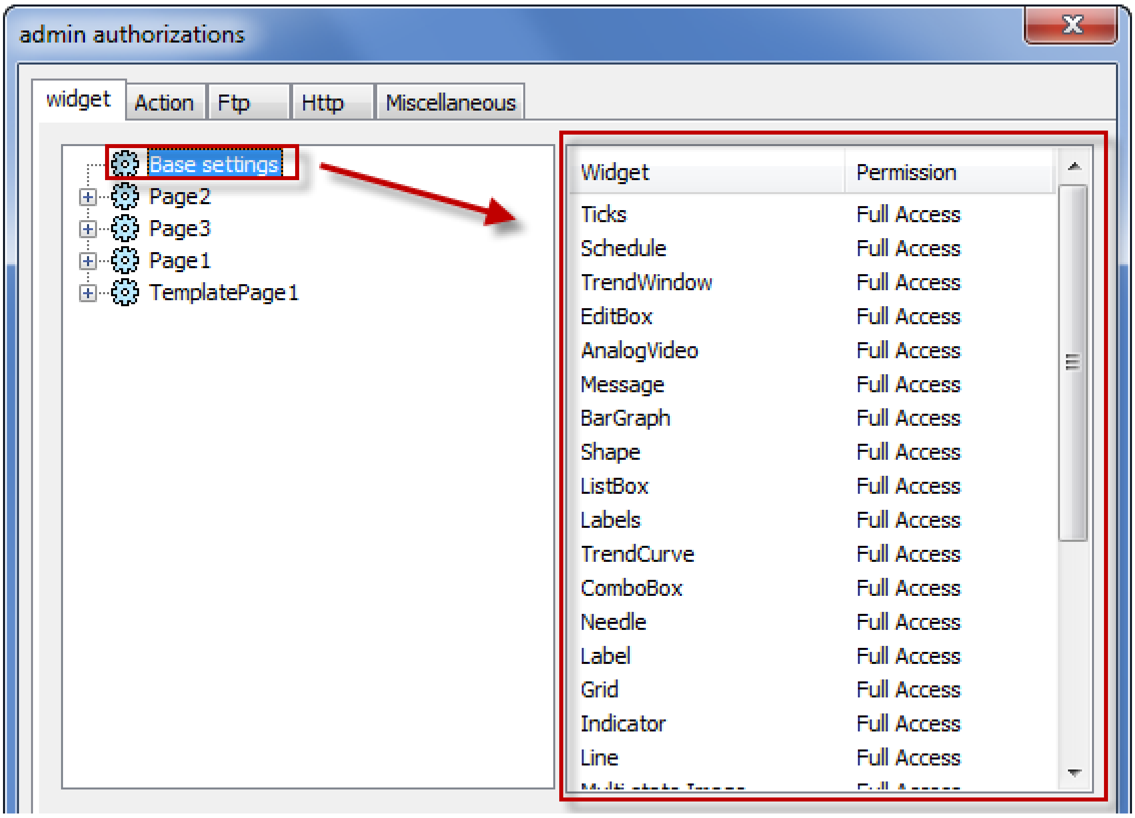


Widget	Permission
Ticks	Full Access
Schedule	Full Access
TrendWindow	Full Access
EditBox	Full Access
AnalogVideo	Full Access
Message	Full Access
BarGraph	Full Access
Shape	Full Access
ListBox	Full Access
Labels	Full Access
TrendCurve	Full Access
ComboBox	Full Access
Needle	Full Access
Label	Full Access
Grid	Full Access
Indicator	Full Access
Line	Full Access

For the widget, the possible options are:

- Full-Access
- Read-Only
- Hide

When you click on **Base settings** the right part of the dialog shows the permissions that will be valid as default and at project level.



Widget	Permission
Ticks	Full Access
Schedule	Full Access
TrendWindow	Full Access
EditBox	Full Access
AnalogVideo	Full Access
Message	Full Access
BarGraph	Full Access
Shape	Full Access
ListBox	Full Access
Labels	Full Access
TrendCurve	Full Access
ComboBox	Full Access
Needle	Full Access
Label	Full Access
Grid	Full Access
Indicator	Full Access
Line	Full Access

The widget´ security settings can be changed, not only globally, but also for each single widget present within the project; all the widgets can be reached from the tree structure on the left part of the widget tab.

Permissions can be given at three levels:

- “Project level”,
- “Page level” and
- “Widget level”.

In the tree structure the permission for a page can be set as

- “Full Access”,
- “Hide” or
- “Read Only”.

All the widgets in this page will take the settings that have been assigned to the page with a type of hierarchy logic.

Suppose the page permission is set as “Read Only”, then all the widgets in the page will have the permission as “Read Only”. On selecting a widget inside the page from the tree structure, you can see that the permission is given as “Use Base Settings”. It means that it takes the permission given to the page(Read Only).

The widget permission takes the priority as follows:

Low priority: Basic settings (widget settings in general for the project)

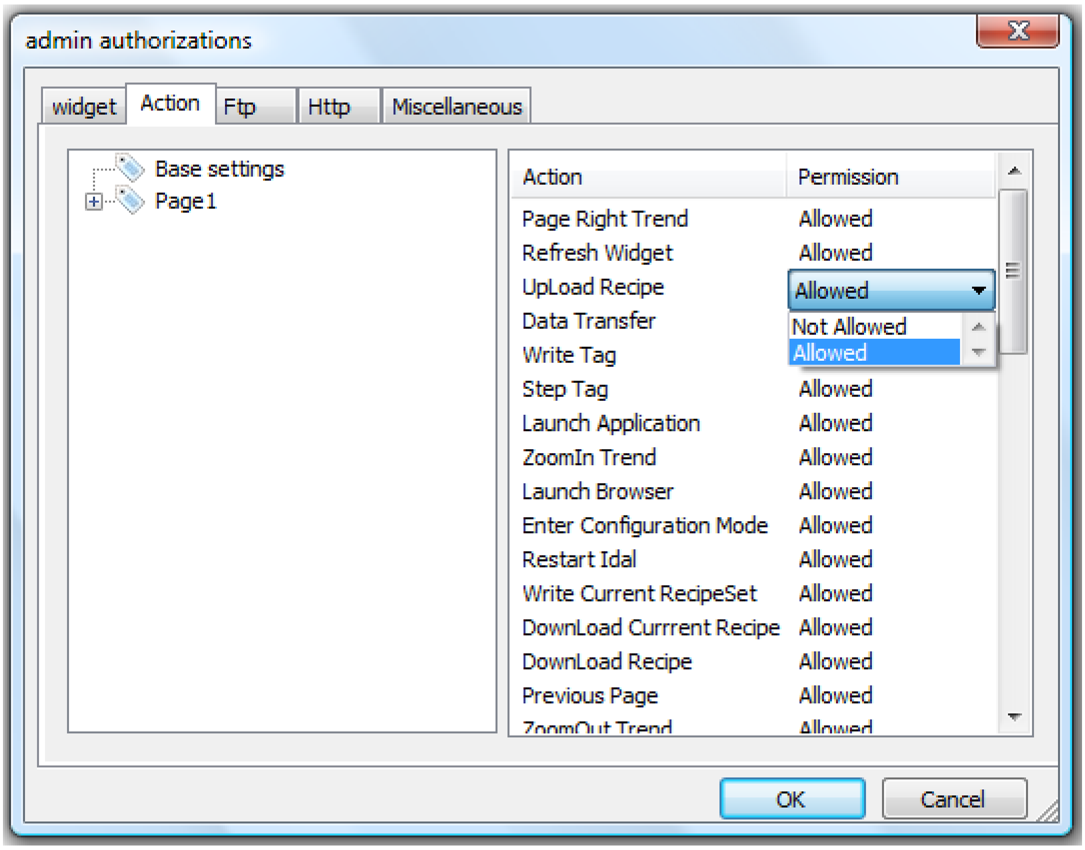
Medium priority: Page settings (settings for all the widgets of a particular page)

High priority: Widget settings (individual widget's or its group/parent widget's permission of any page).

For example suppose a widget is set “Read Only” permission at project level and it is given “Full Access” at page level then the page level settings will be taken.

Later in the chapter, we explain how to modify permissions for a specific widget directly from the page view (rather than locating the widget from the tree view shown in the authorizations’ dialog).

Action Permissions



Action	Permission
Page Right Trend	Allowed
Refresh Widget	Allowed
UpLoad Recipe	Allowed Not Allowed Allowed
Data Transfer	
Write Tag	
Step Tag	Allowed
Launch Application	Allowed
ZoomIn Trend	Allowed
Launch Browser	Allowed
Enter Configuration Mode	Allowed
Restart Idal	Allowed
Write Current RecipeSet	Allowed
DownLoad Currrent Recipe	Allowed
DownLoad Recipe	Allowed
Previous Page	Allowed
ZoomOut Trend	Allowed

With this dialog, it is possible to assign the authorizations for the actions with respect to a project. The access is either **Allowed** or **Not Allowed**.

As for the widgets, the authorizations can be assigned globally, but also for each single page and widget programmed into the project.

Later in the chapter, we will explain how to modify permissions for a specific action directly from the page view (rather than locating the action from the tree view shown in the authorizations' dialog).

FTP Authorizations

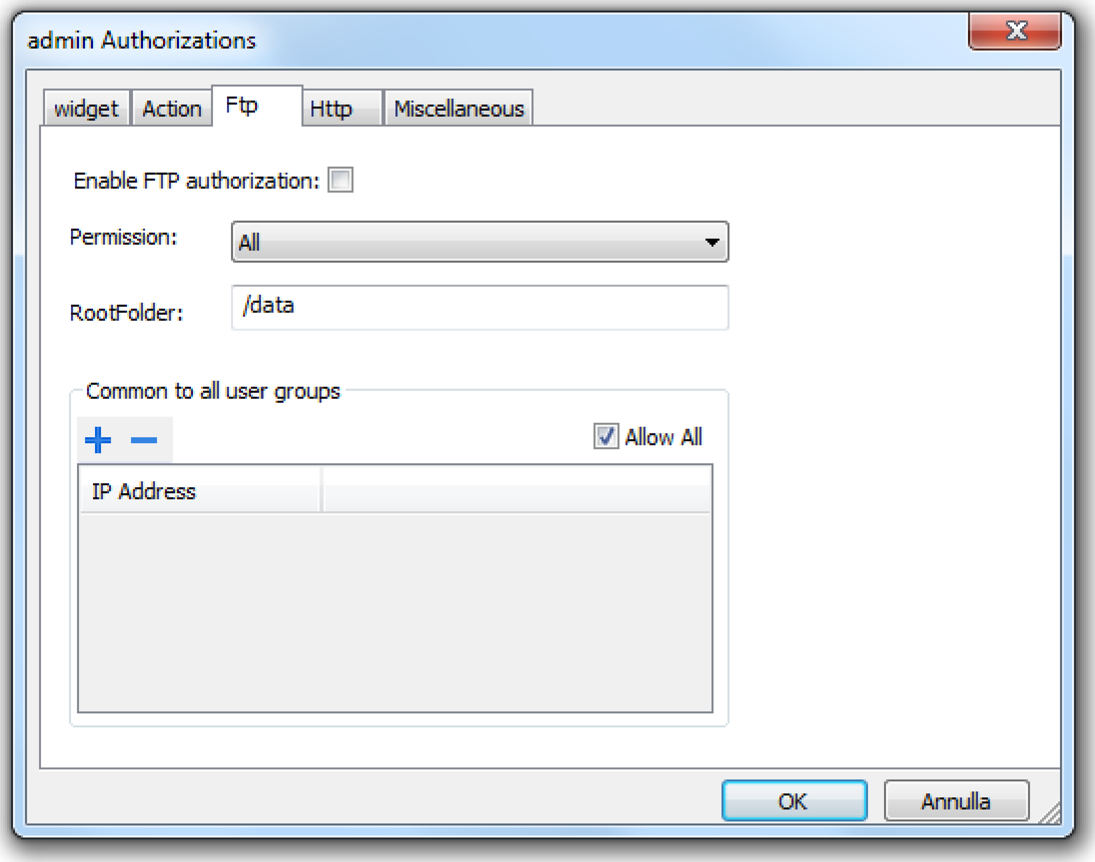
Per each group you can set specific authorizations related to the use of the FTP server as shown in the following picture.

FTP permissions can be enabled or disabled. If enabled, you can specify from the **Permission** combo box the access level selecting between “All“, „Write“, „Read“, “Browse”, and “None”.

The IP address list access allows to specify from which IP an incoming FTP connection should be accepted.

NOTE

IP access list configuration is common to all groups.



admin Authorizations

Enable FTP authorization: □

Permission: All

RootFolder: /data

Common to all user groups

☑ Allow All

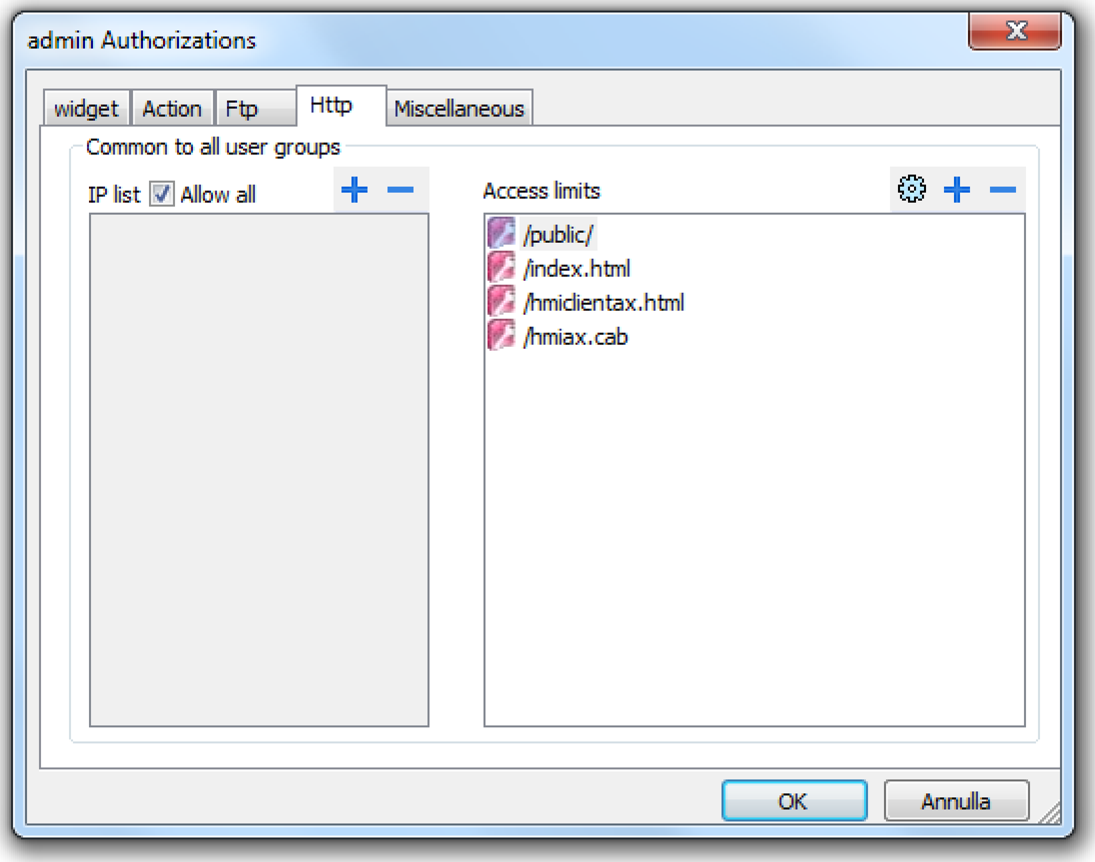
HTTP Authorizations

The HTTP authorization dialog allows to configure the restrictions related to http access to the web server integrated in the runtime. HTTP settings are common to all groups and are valid just if security flag is enabled.

IP list can be used to list allowed ip addresses. Default is **Allow all.** Only IP listed in **IP list** will be authorized to access to http server embedded into the runtime.

Access limits is used to allow or restrict access to particular files and folders into the workspace. Based on **Force Remote Login** flag default workspace access change and as consequence using **Access limits** is possible to open or close access to specific resources.

Force Remote Login	Default Access to workspace	Access limits
-	FULL	-
Disable	FULL	Can be used to block access to some files/folders or to require auth for it
Enable	No Access	Can be used to open access to files/folders



admin Authorizations

Common to all user groups

IP list ☑ Allow all

Access limits

/public/
/index.html
/hmiclientax.html
/hmiax.cab

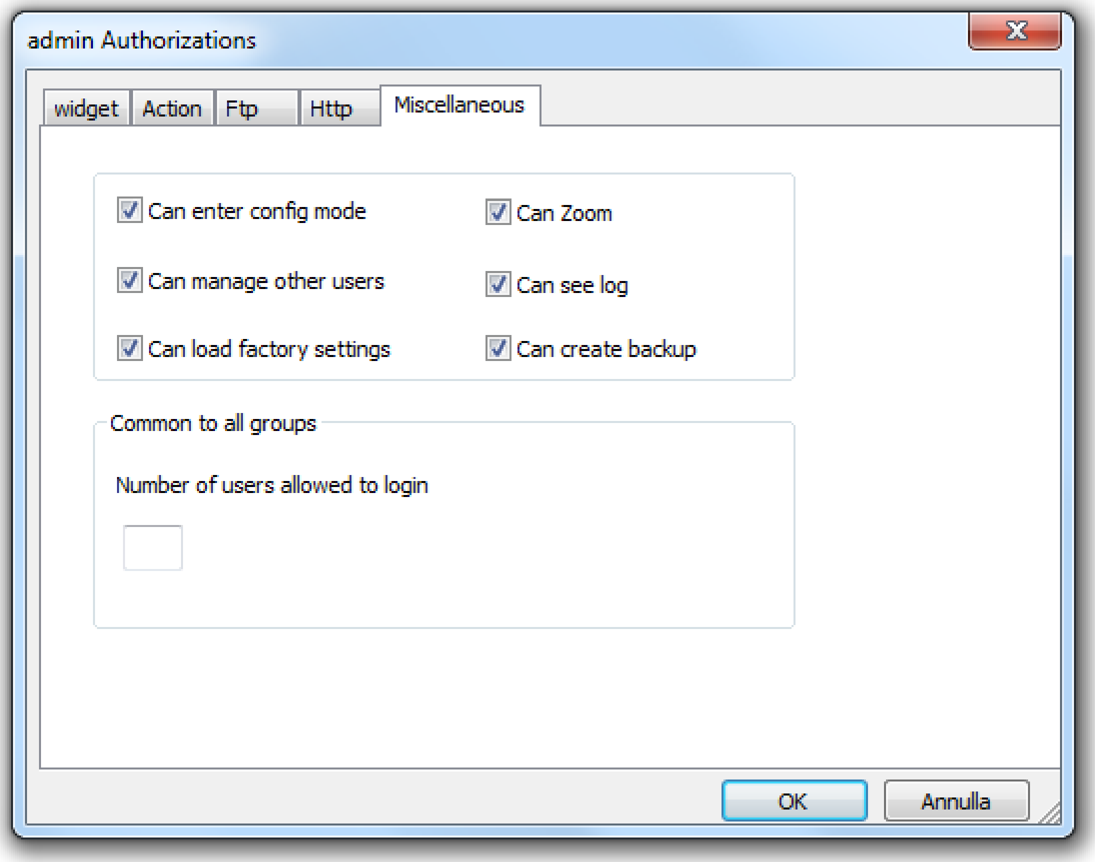
Set default access limits icon on the left of **“+ - “** can be used to restore default configuration removing user customizations. Default is allow following public resources:

- **PUBLIC** folder and Index.html, that contain web console and public resources
- **ActiveX** files (hmiclientax.html, hmiax.cab)

Miscellaneous

The *Miscellaneous* tab contains different settings related to the several options as indicated in the following picture.

Please note that as indicated in the picture, some settings are related to the group, but some settings are global to all groups.



☑ Can enter config mode
☑ Can Zoom
☑ Can manage other users
☑ Can see log
☑ Can load factory settings
☑ Can create backup

Common to all groups

Number of users allowed to login

Setting	Explanation
Can enter config mode	Allow users of group to move runtime to configuration mode (for maintenance usually).
Can manage other users	Allow users of group to manage other users like a superuser at runtime. A user with this permission can add new users, remove users or change user permissions.
Can load factory settings	Users setting can be changed at runtime by authorized users and are saved into internal storage usually. A user with this authorization can execute a macro to clear these dynamic files and restore user management setting as was at beginning after first project download.
Can zoom	Allow user to zoom in/out using context menu at runtime
Can see log	Allow user to see logs at runtime
Can create backup	Allow user to backup project.
Number of users allowed to login	Max number of users that can be connected to runtime in the same time. Default is 3.

Access Priority

If the access control is applied to a widget, page or even the Global Access, then the top priority goes to the widget access.

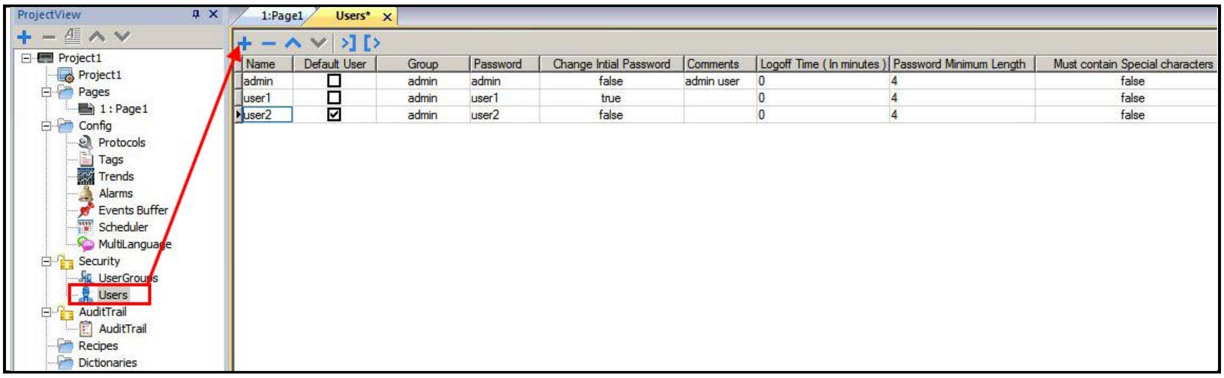
- Top Priority: Control from widget
- Medium Priority: Page access or its parent access
- Low Priority: Global access

These "exceptions" configured for an action or a widget directly from the page view, have priority over the base settings.

Configuring Users

1. To configure users, double click on **Users** in the Project View.
2. Click on **+** to add a new user.

A user named admin is already present by default and this user cannot be deleted.



Name	Default User	Group	Password	Change Intial Password	Comments	Logoff Time (In minutes)	Password Minimum Length	Must contain Special characters
admin	□	admin	admin	false	admin user	0	4	false
user1	□	admin	user1	true		0	4	false
user2	☑	admin	user2	false		0	4	false

Column	Explanation
Name	User Name
Default User	Identifies the user which is automatically logged-in by the system when starting, re-starting or after a logout. Only one default user is allowed.
Group	Groups of the user. Groups are used to assign authorizations to users.
Password	Password for the user
Change Initial Password	If true, the user is forced to change his password on first logon.
Comments	Comments for the user
Logoff time (In minutes)	The user will be automatically logged off after the specific time with no actions on the panel. After log off, the PB610 Panel Builder 600 Server goes to default user.
Minimum Length	In numbers, the minimum length of the password must be equal or greater than the set value.
Must Contain Special Characters	If true, the password must contain at least one special character.
Must Contain Numbers	If true, the password must contain at least one numeric digit.

Default User

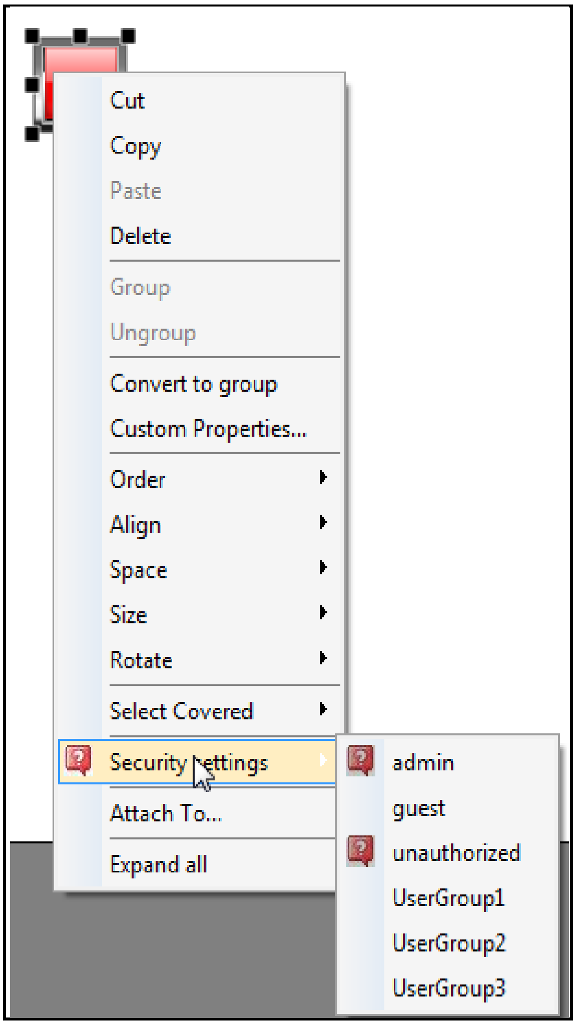
You can program a default user for a project. When the server starts or reboots, the runtime is logged in with the default user. All the privilege settings of the default user will be activated in the system. If you want to log in as different user in runtime, you can use either the *Switch User* macro or the *Log Off* macro.

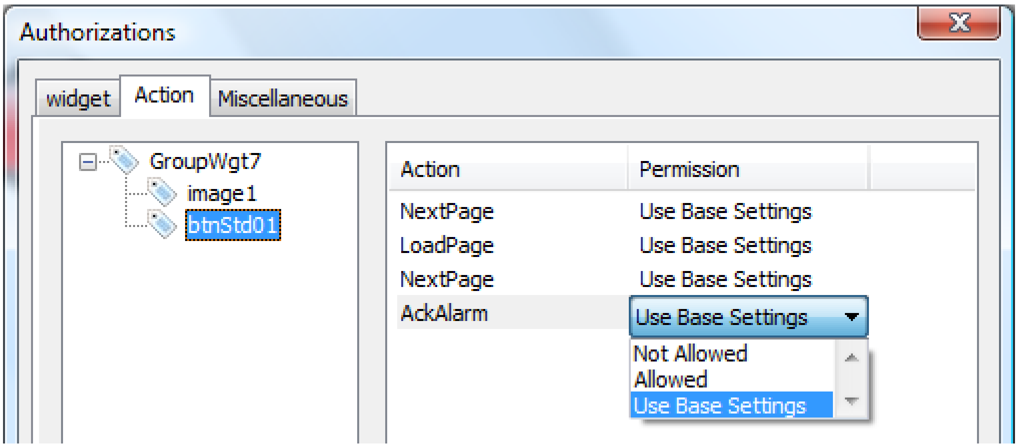
The default user will automatically get logged in if any user (other than default user) logs off.

Assigning Widget Permissions from Page View

You can assign different levels of security to different user groups, on a single widget, directly from the project pages.

1. Select the widget.
2. Right click and select **Security settings** from the context menu. Next, choose the group and assign the security properties to access the widget.





Action	Permission
NextPage	Use Base Settings
LoadPage	Use Base Settings
NextPage	Use Base Settings
AckAlarm	Use Base Settings

Operation on Runtime

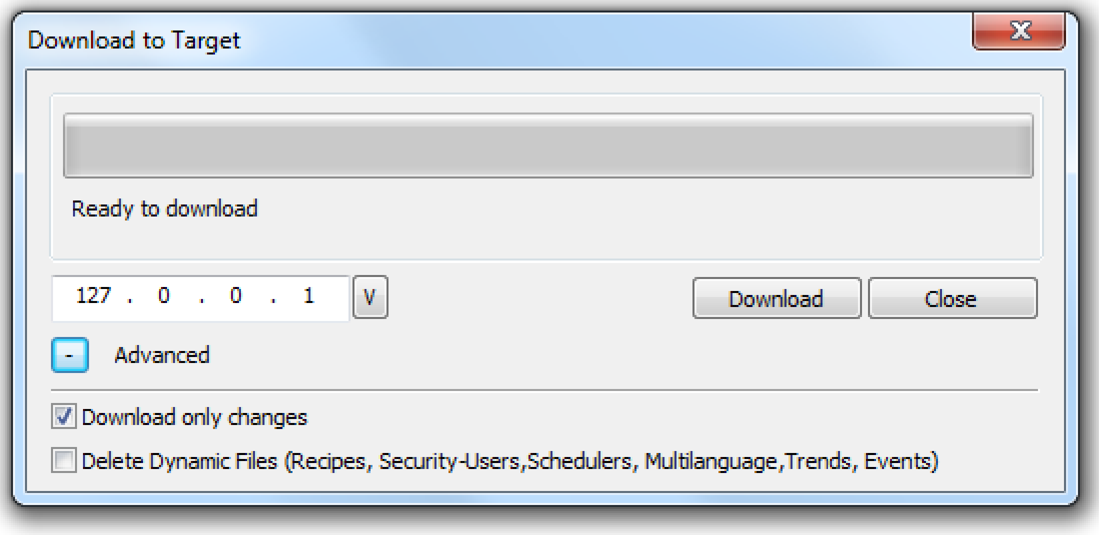
After starting the runtime, if **a default user** is specified within the project, the system will provide automatic login of that user without prompting for a user login. If no default user is configured, the system will ask for a User name and Password, and based on the user, the Runtime will allow only the configured permissions for that logged user.

There are specific actions for user logout, user edit, user add, remove user and switch user. Users can be edited, added or removed on runtime as explained in chapter "User Management Actions".

All the users' information modified in runtime is stored in a separated file, thereby preventing loss of the users' configurations in case of a new project download.

To remove dynamic files and changes applied to user's configuration during runtime there're two ways:

- Runtime side: **DeleteUMDynamicFile** action
- Panel Builder 600 side: **Delete Dynamic Files** flag available in download dialog

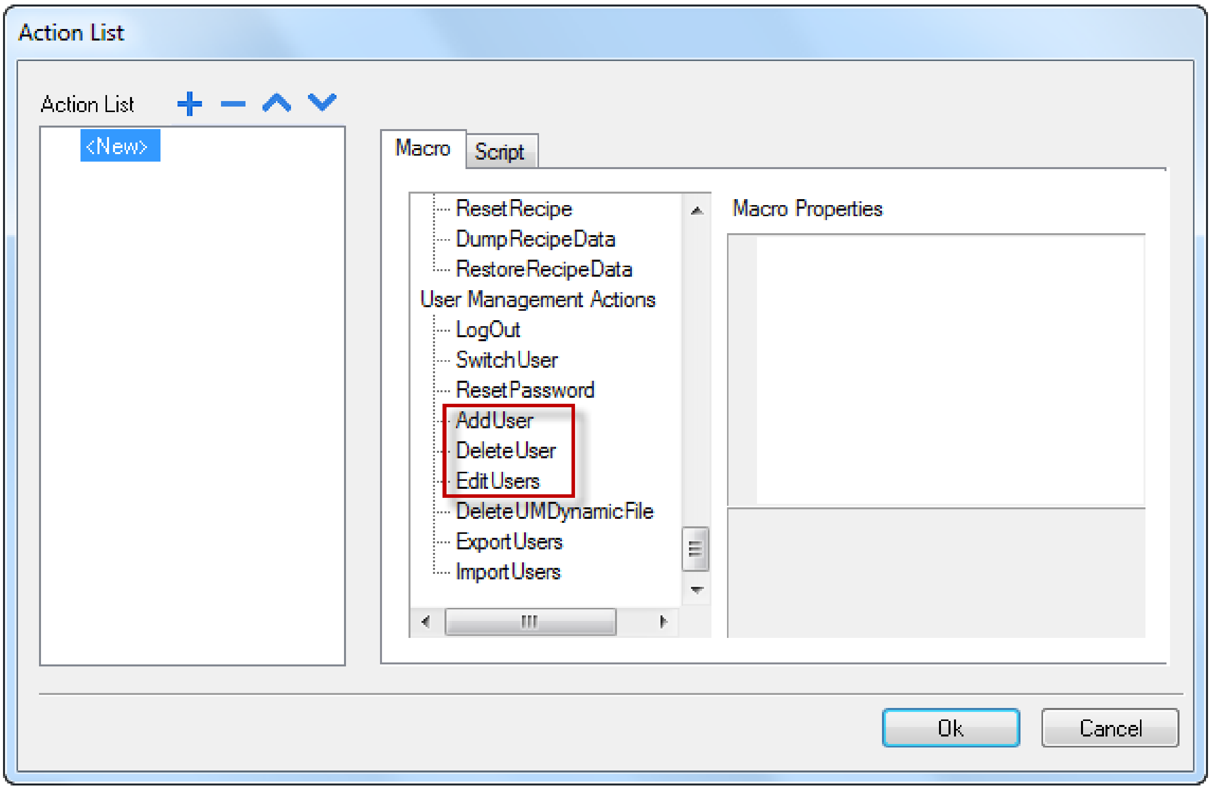


Ready to download

127 . 0 . 0 . 1

☑ Download only changes

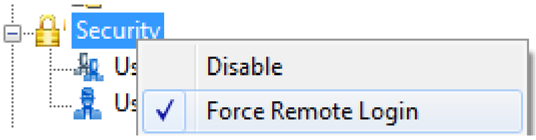
☐ Delete Dynamic Files (Recipes, Security-Users,Schedulers, Multilanguage,Trends, Events)



Force Remote Login

Starting from v1.90 of Panel Builder 600, a new flag is available to force user to LogIn when using remote access (via Activex or Remote Client), this is working when user management is enabled. If **Force Remote Login** is not enable remote access will use same level of protection of local access.

Force Remote Login is useful in particular when a default user is configured in runtime to automatically login without having to enter a login and password at startup, but a remote access protection is required.



Force Remote Login, when enabled, blocks all access from web to workspace folder in runtime. The only files/folders still accessible when this flag is enabled are as default:

- **PUBLIC** folder and **Index.html**, that contain web console and public resources
- **ActiveX** files (**hmiclientax.html**, **hmiax.cab**)

Please check **Security** -> **UserGroups** -> **Authorization Settings** -> **HTTP** tab for more details related to HTTP access limits or chapter **HTTP Authorizations.**

Audit Trails

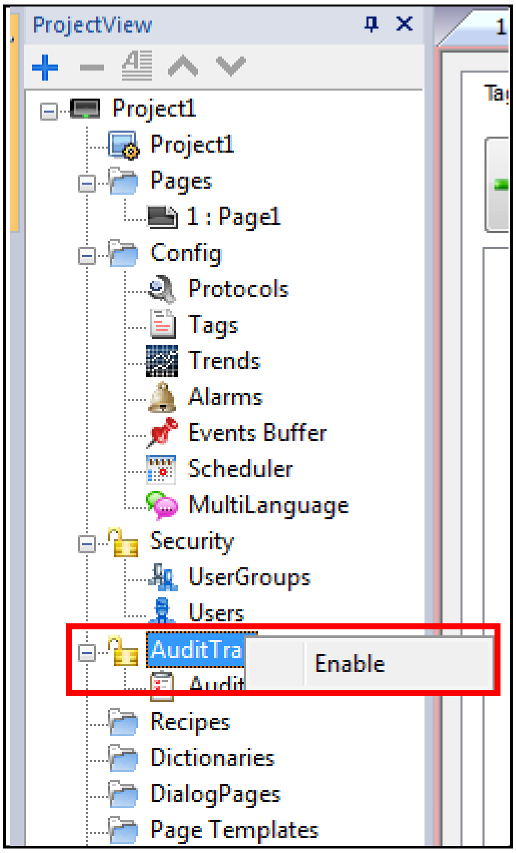
PB610 Panel Builder 600 supports *Audit Trail* functionality, which provides essential process tracking, user identification linked to time and the date of events logged facilitating recalls and/or rationalizing of your production processes.
The *Audit Trail* function provides flexible, tailor-made and easy-to-review event logs.

The *Audit Trail* (or audit log) is a chronological sequence of audit records, each containing information on the actions executed and the user that did them.
The *Audit Trail* can be enabled with or without user management. So it can access and supervise all actions from all users, and a normal user could not stop or change this.

Enable or Disable the Audit Trail

→ In the Project View pane, right click on the *Audit Trail* and click either **Enable** or **Disable** for the *Audit Trail* recording on runtime.

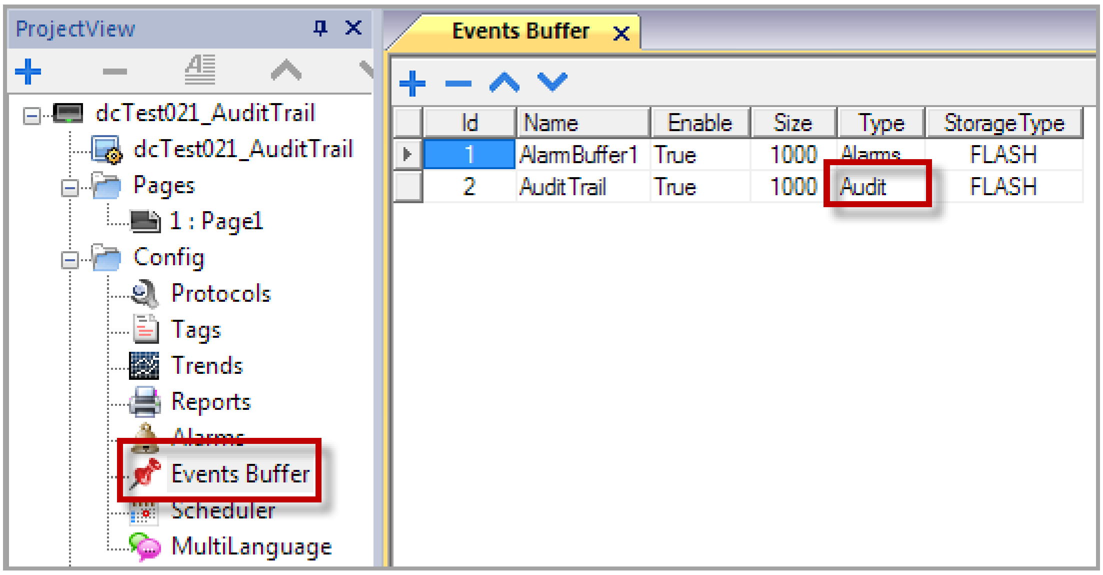
- ➤ The padlock symbol in the tree informs you if the *Audit Trail* is enabled or disabled. When the *Audit Trail* is enabled, the padlock symbol shows locked, otherwise it stays open.



Configure Audit Events

You can have more than one set of *Audit Records*. To add to the audit files, you need to configure the events buffer.

1. Double click the *Events Buffer* from the project workspace.
2. Add the events buffer and set the file size.
3. Select the log type **Audit**.



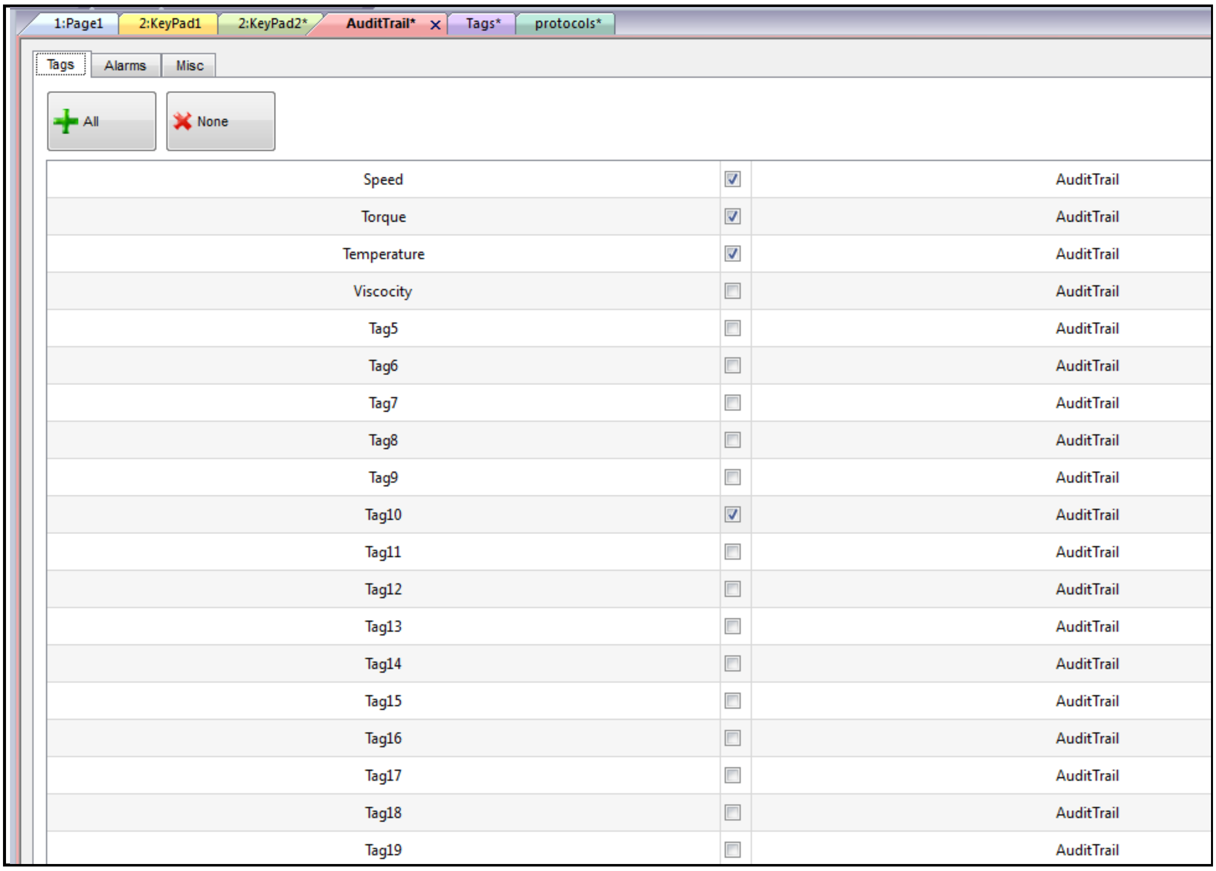
Id	Name	Enable	Size	Type	Storage Type
1	AlarmBuffer1	True	1000	Alarms	FLASH
2	Audit Trail	True	1000	Audit	FLASH

Here there is an option for selecting the storage where the dumped audit files have to be stored

The systems provides a save to file on the disk every 5 minutes.

Configure Tags in the Audit Trail

For most of the cases, all the tags specified in the project do not necessarily need to be monitored. You can customize the tags to be monitored by *Audit Trail*.



Speed	☑	AuditTrail
Torque	☑	AuditTrail
Temperature	☑	AuditTrail
Viscocity	□	AuditTrail
Tag5	□	AuditTrail
Tag6	□	AuditTrail
Tag7	□	AuditTrail
Tag8	□	AuditTrail
Tag9	□	AuditTrail
Tag10	☑	AuditTrail
Tag11	□	AuditTrail
Tag12	□	AuditTrail
Tag13	□	AuditTrail
Tag14	□	AuditTrail
Tag15	□	AuditTrail
Tag16	□	AuditTrail
Tag17	□	AuditTrail
Tag18	□	AuditTrail
Tag19	□	AuditTrail

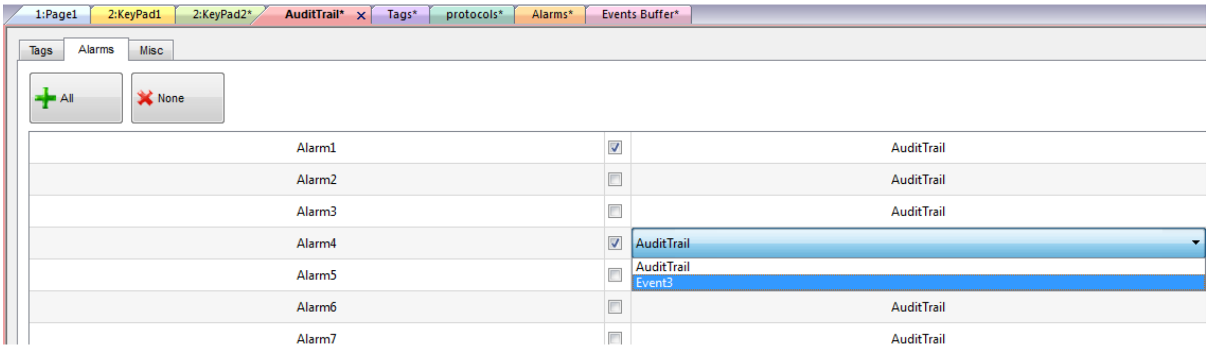
In the *Audit Trail* editor, all the tags are available for selection. You can select only the tags to be monitored by *Audit Trail*. For each selected tag, the *Audit Trail* will record the write operation to that tag, together with the time stamp and user that executed the write operation.

Configure Alarms in the Audit Trail

You can specify the alarms to be monitored by the *Audit Trail*.

1. Double click *Audit Trail* from the project workspace.
2. Click on the *Alarms* tab.
3. Select the alarms you want to be logged in the *Audit Trail*.

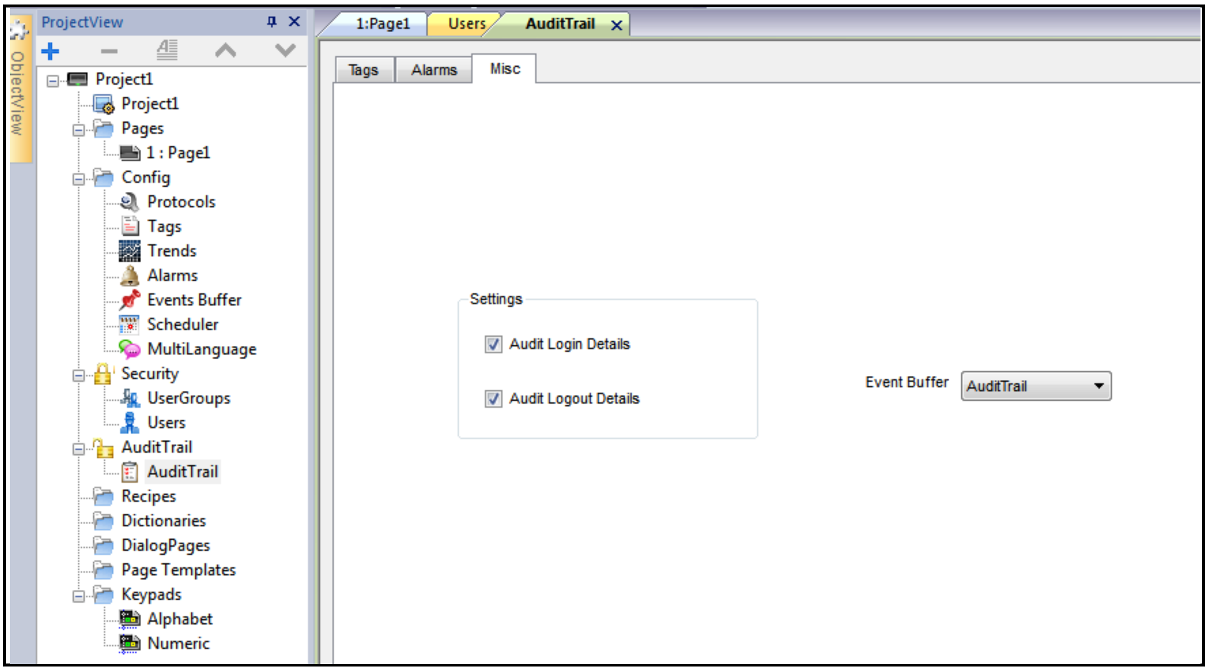
   ➤ The *Audit Trail* for alarms will also record and acknowledge the operation done by the logged-in user.



Alarm1	☑	AuditTrail
Alarm2	□	AuditTrail
Alarm3	□	AuditTrail
Alarm4	☑	AuditTrail
Alarm5	□	AuditTrail Event3
Alarm6	□	AuditTrail
Alarm7	□	AuditTrail

Configure Login or Logout Details in Audit Trail.

Audit Trail can record information about user login and user logout events. These settings are available in the *Misc* tab of the *Audit Trail*.



Settings

☑ Audit Login Details

☑ Audit Logout Details

Event Buffer: AuditTrail

Viewing Audit Trails in Runtime

Audit Trail data cannot be displayed in runtime, they are only available in the exported data file.

Exporting Audit Trail as CSV File

You can convert the audit data to a *.csv* file.

For detailed description look the explanation provided for the Dump Archive macro action, chapter "Dump Archive".

Reports

A report is a collection of information that will be printed when triggered by an event.

The Panel Builder 600 programming software provides an editor to configure reports, their content, the printer and the trigger conditions.

The report comes as a special collection of pages with header, footer and body, including options for cover page. When configuring reports, Panel Builder 600 provides access to a dedicated widget gallery featuring only the widgets available for reports.

When the programmed event is triggered the report printout is started and the entire printing activity is carried out in the background.

Adding a Report

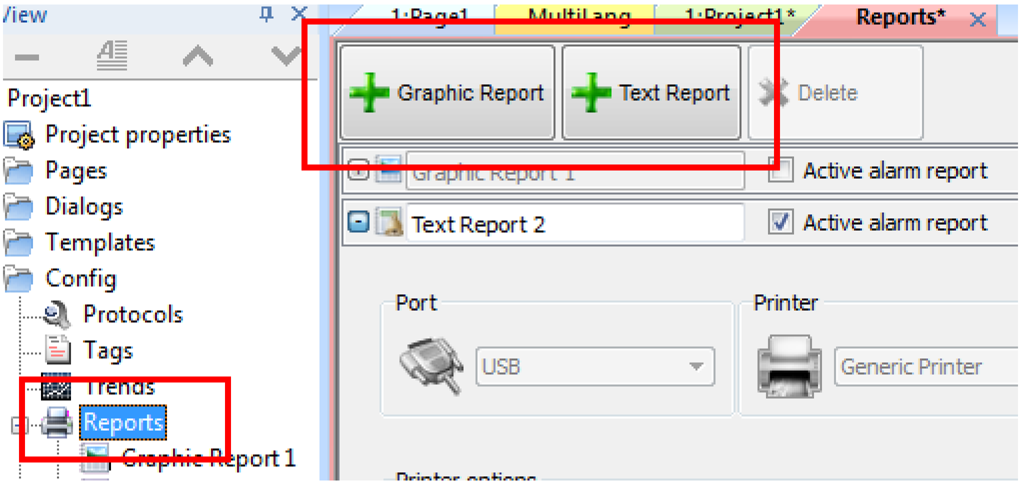
In the Project Workspace, double click on Reports to open the Editor. Then add the report by clicking the “+" button.

Two types of reports are available:

- Text report
- Graphic report

Text reports are used to configure the line by line printing of alarms. Text reports are designed to work with line printers. Text is sent directly to printer´s port without using any special driver. Not all printers support this operation mode. This printing mode only works in WinCE platforms and requires to use a physical port.

Graphic reports contain graphical elements and may include complex widgets such as screen shots, or alarms. A specific printer driver for each printer is required for printing graphic reports: The list of supported drivers is part of following chapters.



Graphic Report 1 □ Active alarm report

Text Report 2 ☑ Active alarm report

Port: USB

Printer: Generic Printer

Printer options

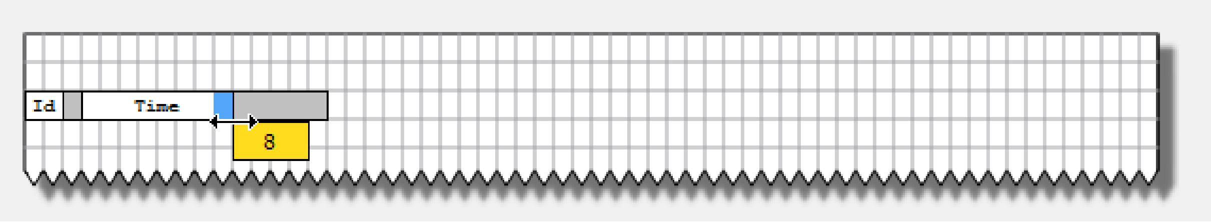
Text Reports

To add a text report for line by line alarm printing click on the “Text Report” button in the reports toolbar.

The format of the report can be freely defined using the report editor; the paper size can be defined in number of characters, while the available fields are listed in the box on the right side.

To include a field in the line to be printed, just drag and drop it from the list to the page layout.

The field can be resized using the mouse, the tooltip shows the dimension in “chars”.



In case the text cannot fit in the dedicated space the auto wrap is applied.

Printer options can be used to control flush of pages in printer. Depending on the printer, text can be printed immediately or after a timeout (from few seconds to minutes). However, it is always possible to force flush when one of following conditions happens: after n events, after n lines or after n seconds. A temporary buffer is used by runtime software. Flush conditions are in OR, so, as soon as a condition is met, the page will be flushed out of printer.

NOTE Only line printers support text report operation mode (see list of supported printers).

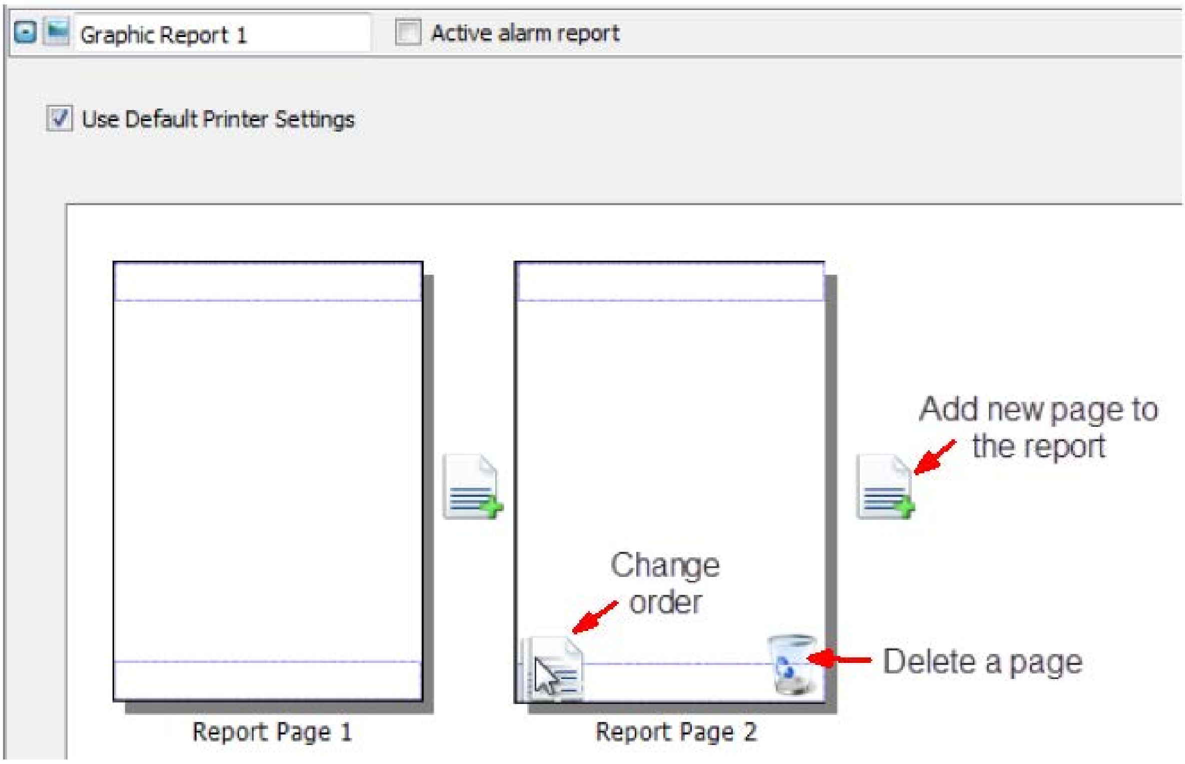
NOTE Text Reports only work in WinCE platforms and require to specify a physical port (PDF format is not supported by text report).

NOTE In line printing, text is printed immediately line by line or after a timeout. This timeout may depend on printer model (could also take minutes for some models not designed for line printing).

Graphic Report

To add a graphic report, click on the “Graphic Report” button in Reports toolbar.

The following figure shows report configuration editor.



This part of the editor is used to set the number of pages and their order.

→ Use the icon with the “+” symbol to add a new page to the report layout.

When the mouse goes over a page already configured, two icons appear to allow reordering or deleting pages.

→ Double click on a page to edit the page report content using the page editor.

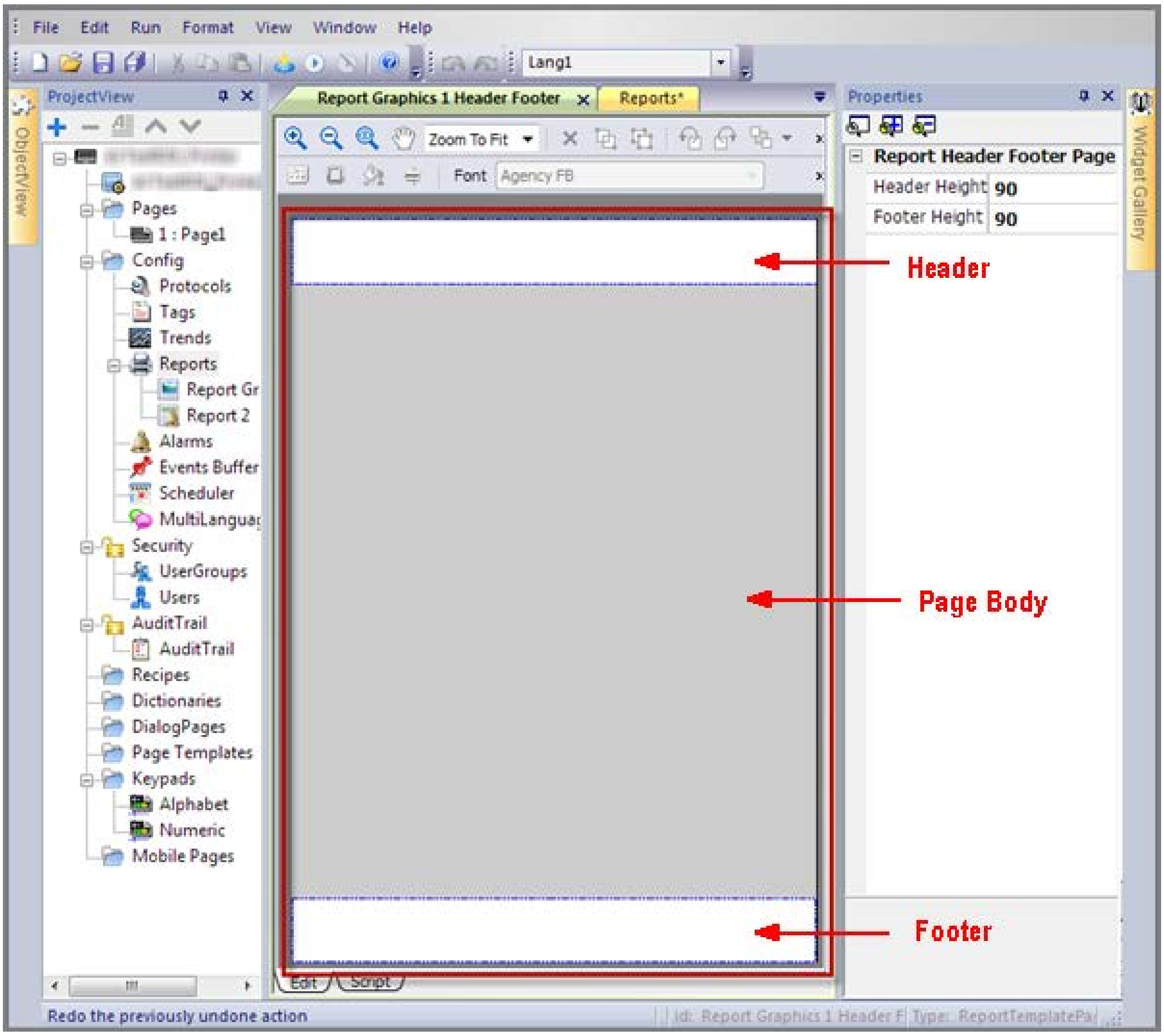
Each page is divided in three sections: the header, the footer and the page body.

In the page editor the area under editing is shown in white, the others are grayed out.

→ To edit a different section, just double click over the grayed out area.

→ Check “Active alarm report” for that one you select as active.

Page Body



Report Header Footer Page

Header Height	90
Footer Height	90

The page body is the central part of the page

The widget gallery accessible from the right side sliding tab is context-sensitive and includes only the widgets available for the area under editing.

Header and Footer

The header and footer are respectively the top and bottom parts of the page.

The widget gallery accessible from the right side sliding tab is contextual and includes only the widgets available for the area under editing.

The Context Widget Gallery

The widget gallery which can be normally recalled from the right side sliding pane is always adapting itself to the context.

The available widgets are:

Page Number Widget: automatic page numbering

Screenshot Widget: used to take a print screen of the current page HMI is showing. When you drag & drop the widget on the page it will automatically get the page dimensions of the HMI

Alarm widget: used to print the entire contents of the event buffer (the Default buffer is Alarm Buffer1)

The **"Text"** category collects the typical widgets used to compose reports with labels and numeric fields.

Printer Configuration

A default printer can be configured from PrinterSetting menu for all reports. Each report can be configured to use it or to use a different type of printer

For PDF printer (supported only by graphic reports) you can define the folder where files are saved by using “Printed Files Location”.

Supported Printers

List of print languages supported by WCE driver **printCE.dll** (driver in use in WCE platform):

Print Language	**Comments**
HP PCL 3, HP PCL 5e, HP PCL3GUI	printers compatible with **HP PCL3/PCL5e/PCL3GUI**, including models many **DeskJet**, **LaserJet, DesignJet**
Epson ESC/P2	printers compatible with **ESC/P2**, **LQ**
Epson Stylus Color	printers compatible with **Epson Stylus Color**
Epson LX (9-pin)	9-pin printers compatible with **Epson LX, FX, PocketJet (line printers)**
Cannon iP100, iP90, BubbleJet	printers compatible with **BubbleJet, iP90, iP100**
PocketJet II, 200, 3	printers compatible with **Pocket Jet**
MTE Mobile Pro Spectrum	printers compatible with **MTE Mobile Pro Spectrum**
Adobe PDF File	**Adobe PDF** file
SPT-8	printers compatible with **SPT-8**
M1POS	printers compatible with **M1POS**
MP300	printers compatible with **MP300**
Zebra	printers compatible with **Zebra** CPCL language.
Intermec PB42, PB50, PB51, PB2, PB3	printers compatible with **Intermec PB42/50/51/2/3**
Datamax Apex	printers compatible with **Datamax Apex**

Supported ports:

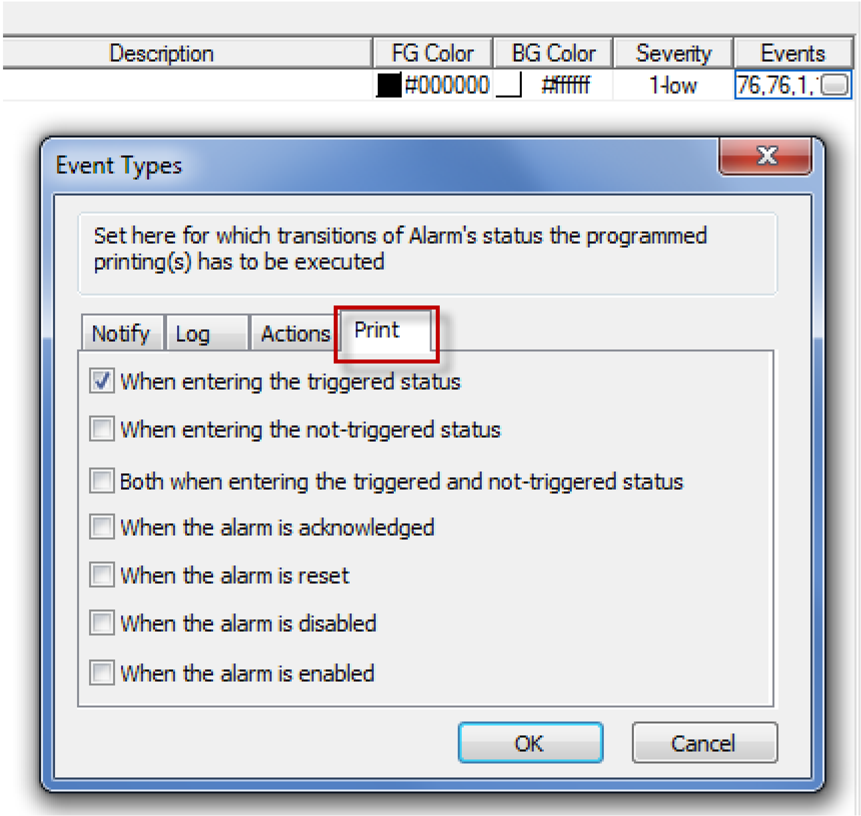
- LPT1 (USB printers)
- File (PDF)

NOTE

In Win32 platform the only supported printers are **PDF** and **Default.** Default is used to indicate default OS printer configured in target PC. Any printer (not just USB printers) can be used in Win32 platform.

Print Events

The configured reports can be triggered by specific events.
For Alarms, the configuration of the events can be done directly in the alarm editor from the Events dialog by clicking on the **Print** tab as shown in figure.



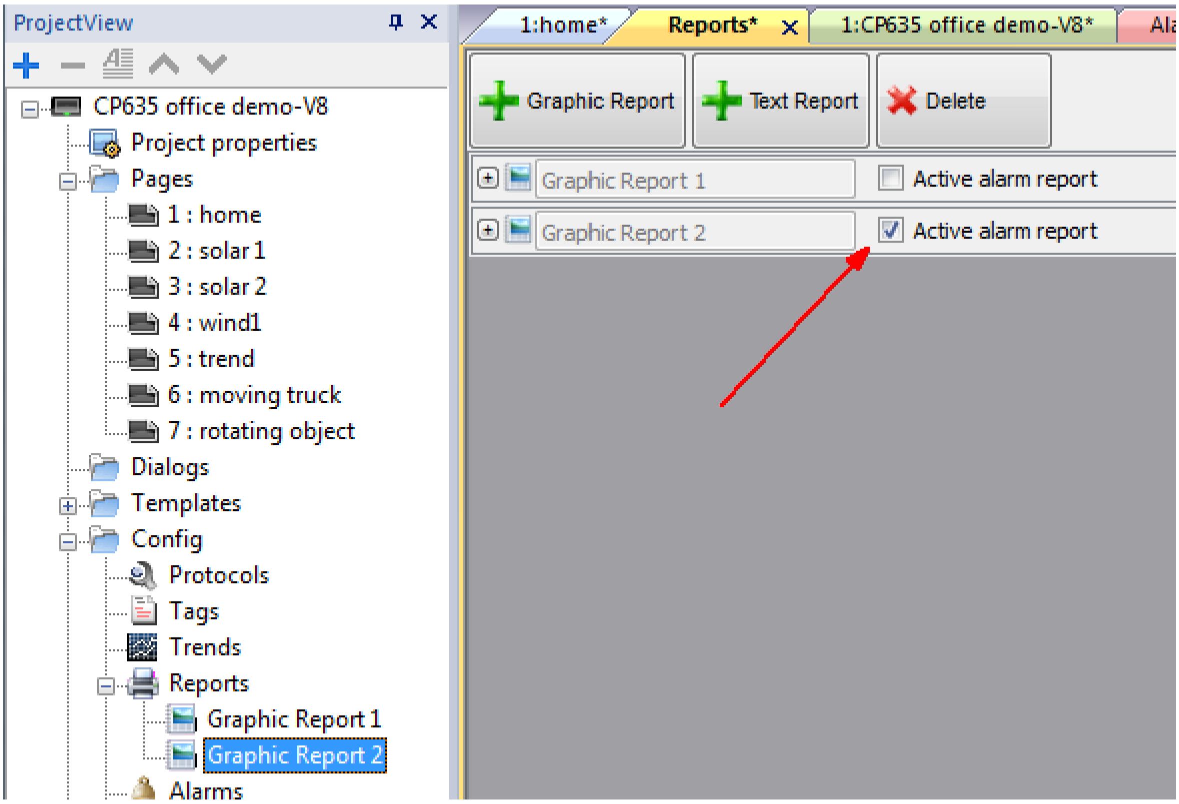
Description	FG Color	BG Color	Severity	Events
	#000000	#ffffff	1-low	76,76,1,

Event Types

Set here for which transitions of Alarm's status the programmed printing(s) has to be executed

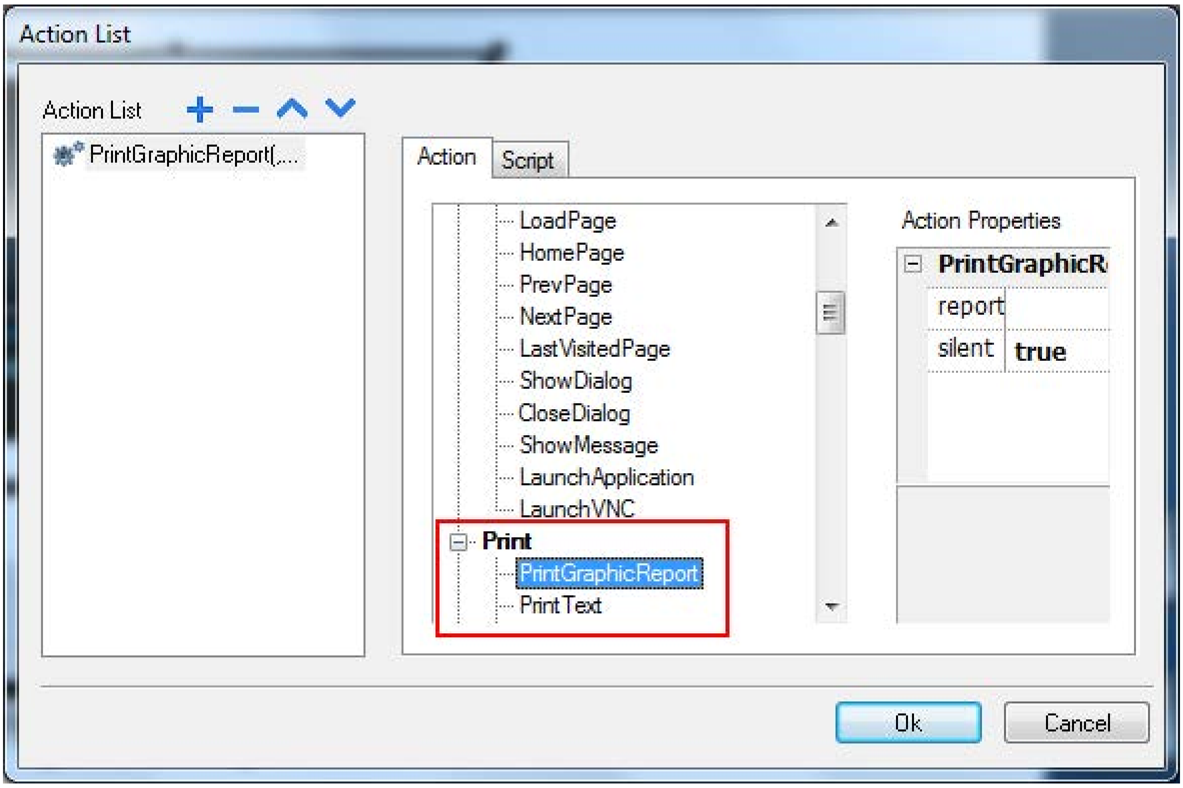
☑ When entering the triggered status
□ When entering the not-triggered status
□ Both when entering the triggered and not-triggered status
□ When the alarm is acknowledged
□ When the alarm is reset
□ When the alarm is disabled
□ When the alarm is enabled

Only one report can be set as **Active alarm report** in a project. An alarm report can be a *Text Report* or a *Graphic Report*.



Graphic Report 1 □ Active alarm report
Graphic Report 2 ☑ Active alarm report

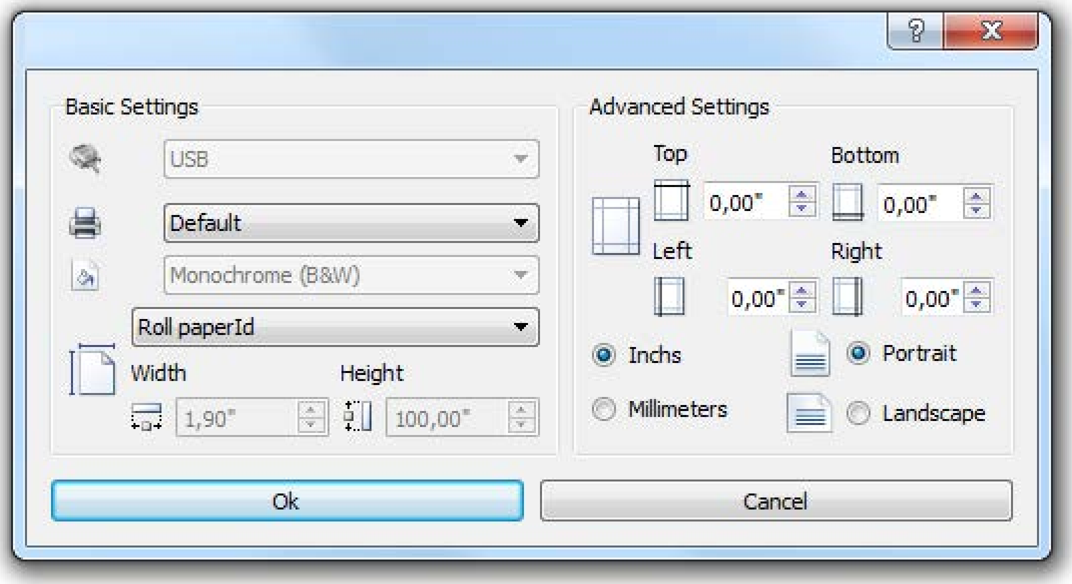
A Graphic report printing can be started also using the dedicated action call **PrintGraphicReport** from widgets´ properties pane.



PrintGraphicReport(....

PrintGraphicR	
report	
silent	true

The **Silent** option (**true** by default in action settings) allows, when set to false, a dialog to pop-up at runtime asking the user to adjust printer settings as shown in figure.



USB

Default

Monochrome (B&W)

Roll paperId

Width 1,90"

Height 100,00"

Top 0,00"

Bottom 0,00"

Left 0,00"

Right 0,00"

● Inchs

○ Millimeters

● Portrait

○ Landscape

Minimum Requirements

Report printing requires operating system (BSP) V1.54 or above for Windows CE devices.

Screen Saver

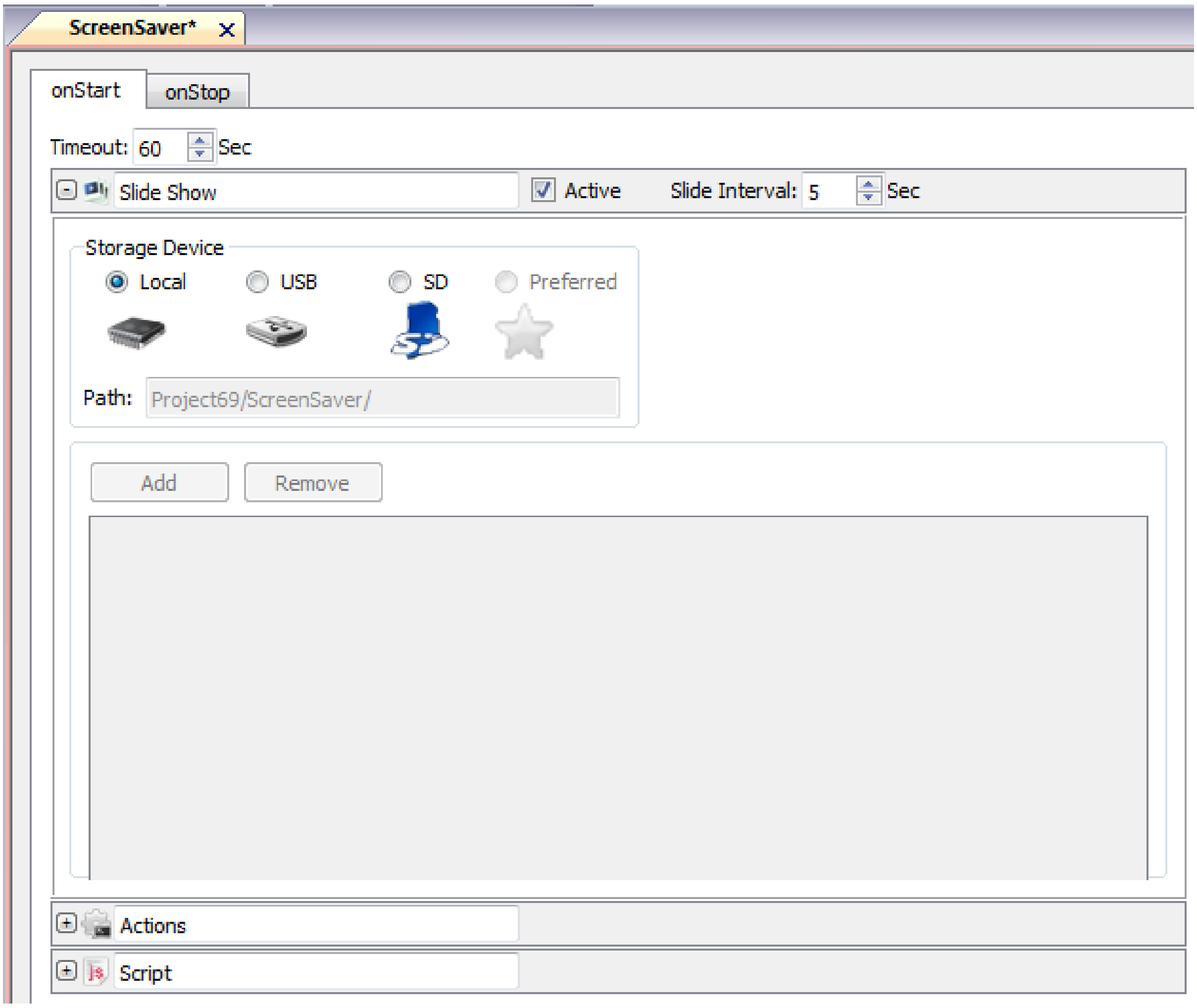
A screensaver is used to turn off the screen backlight or show a slideshow when the HMI is not in use.

A screensaver start when one of following events does not happen for a certain time range (**Timeout**):

- Touch of display
- Mouse move
- External keyboard key pressed

Screensaver configuration is available in Panel Builder 600 in **Config -> ScreenSaver** section.

→ Enable the screensaver via right mouse click on it in the menu **Config**.



Timeout: 60 Sec

Slide Show ☑ Active Slide Interval: 5 Sec

Storage Device

● Local ○ USB ○ SD ○ Preferred

Path: Project69/ScreenSaver/

To configure a slideshow, proceed as follow:

- Select **Timeout** value (number of seconds before screensaver start when there's no user interaction)
- Select **Slide Interval** (the number of seconds before switch slide)
- Select **Storage Device** used for reading images used by slide show (Internal Storage, USB or SD).

For internal storage (Local), it is possible to select and import images that later will be downloaded into the device at project download. Images are downloaded into the folder workspace\projectname\screensaver.

When an external storage is used, images are located in the folder "screensaver" available in USB devices or SD Card.

The supported image formats are: JPEG/PNG.

When the screensaver starts/stops, it is possible to execute some actions (macros or javascript functions). In Tab **onStart** actions can be configured to execute when the screensaver starts, in Tab **onStop** actionswill be executeed when the screensaver stops.

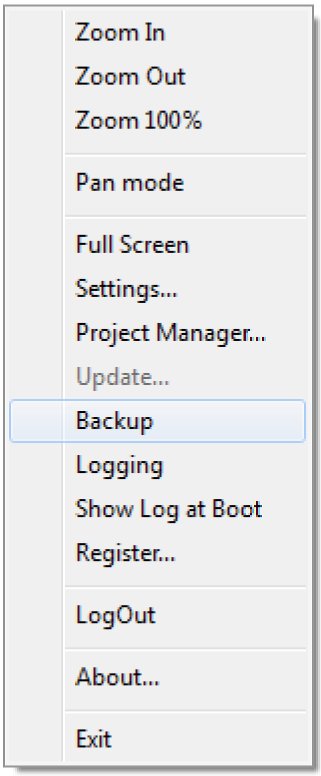
Backup/Restore

Backup/Restore of the HMI runtime and project is available.

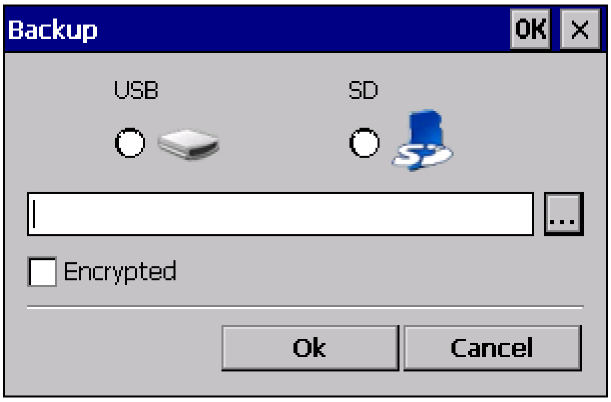
Backup operation is working as follows:

1. Automatically unload current project to unlock opened files in use
2. Archive in a .zip file (standard or encrypted) the content of qthmi folder that contain runtime, projects, dynamic files like recipes / alarms / trends etc.
3. Reload project

Backup can be executed from the context menu in runtime -> **Backup**.



When Backup is called from the context menu, a dialog appears to guide the user in backup operation selecting the path where to save the .zip file with backup.



Backup files can be saved in all available storages like USB devices, SD card, network folders etc.

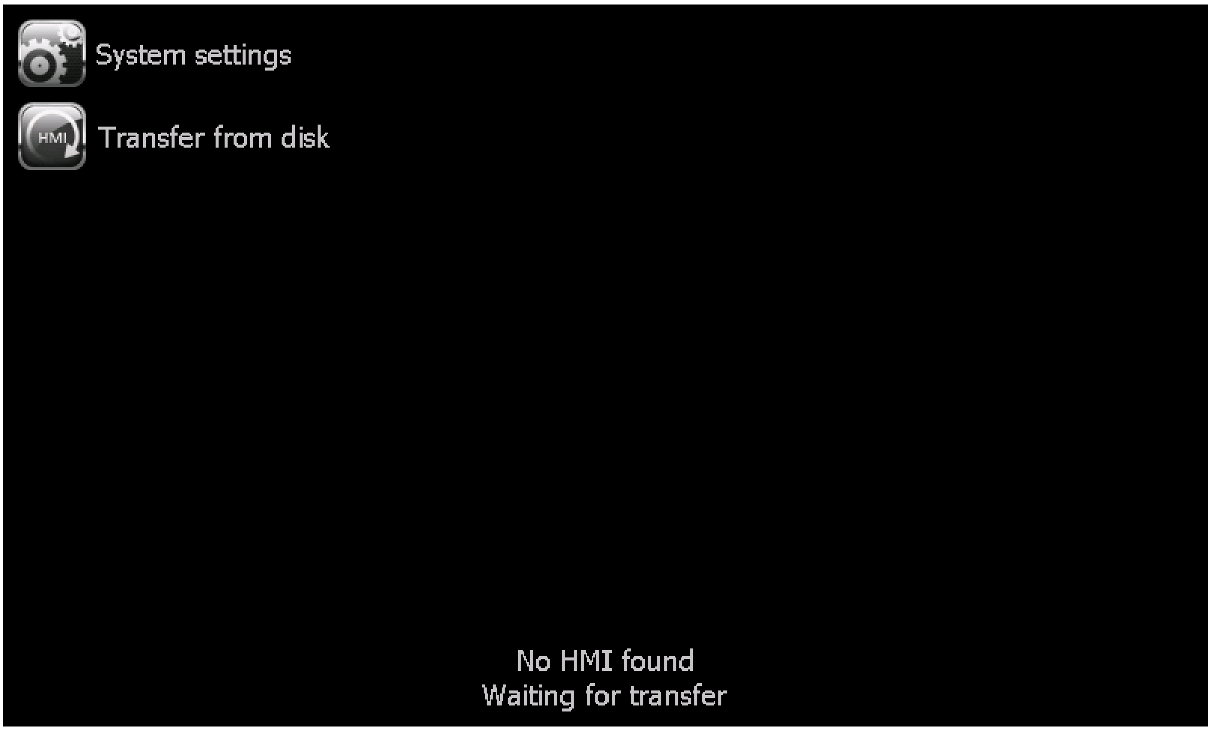
NOTE

Backup is available in WCE platform only. It´s not supported in Win32 / Windows client.

NOTE

Backup ignores external storages like USB and SD cards. So, if dynamic data like recipes, trends, events are archived there the backup will ignore it.

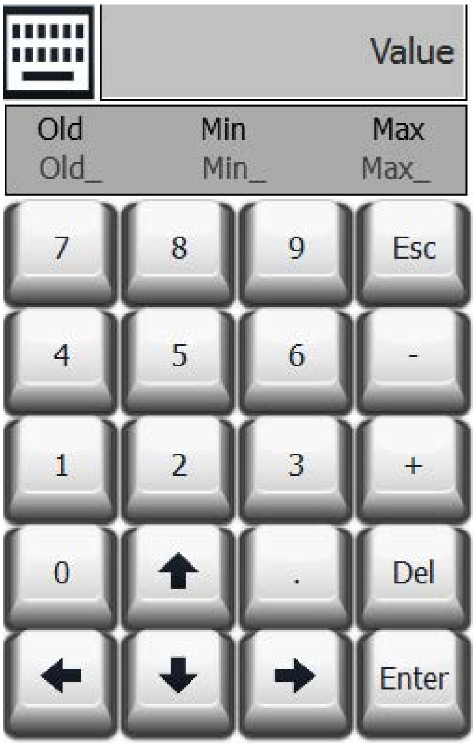
Backup package can be restored from a formatted HMI panel using **Transfer from disk** option in the BSP Loader menu. Just select backup file and the system will automatically check the package to confirm its compatibility with the current platform and install it.



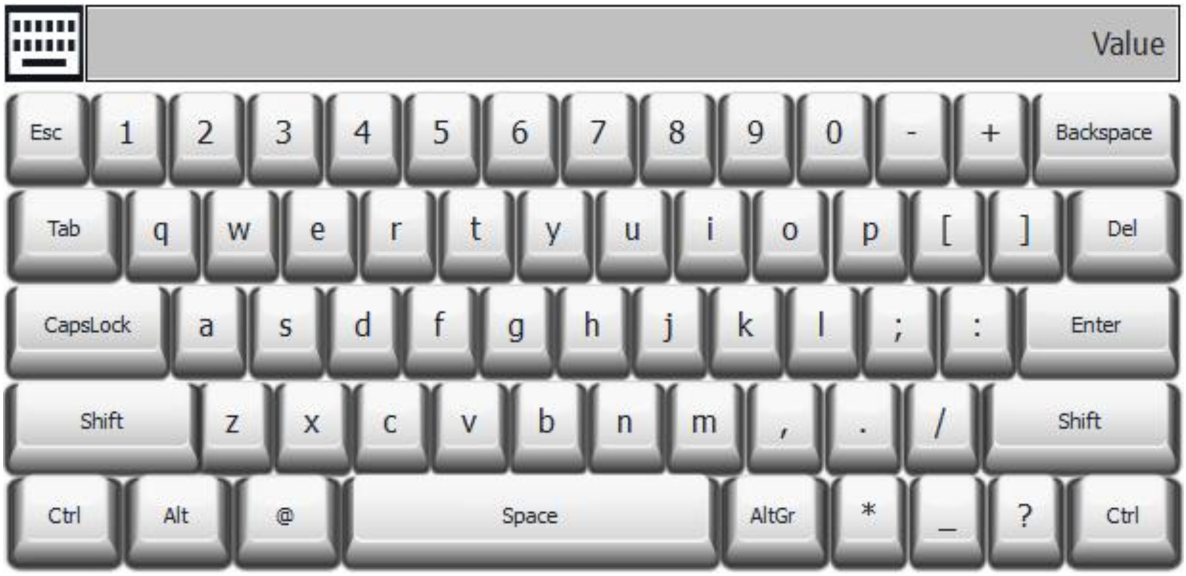
No HMI found
Waiting for transfer

Keypads

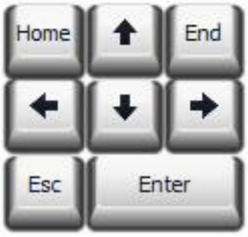
Keypads are used for data entry operations. In PB610 Panel Builder 600 there are several keypads provided by default: Numeric, alphabet, alphabet small and up-down keypads.



Numeric keypad


Value
Esc 1 2 3 4 5 6 7 8 9 0 - + Backspace
Tab q w e r t y u i o p [ ] Del
CapsLock a s d f g h j k l ; : Enter
Shift z x c v b n m , . / Shift
Ctrl Alt @ Space AltGr * _ ? Ctrl

Alphabet keypad



Up-down keypad

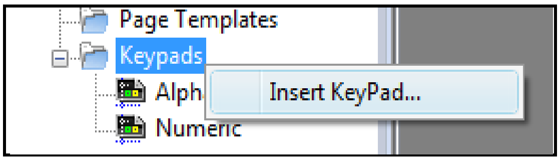
Creating and Using Custom Keypads

Keypads can be created from scratch following the described procedure.

NOTE You can also change the existing ones directly applying modifications to them.

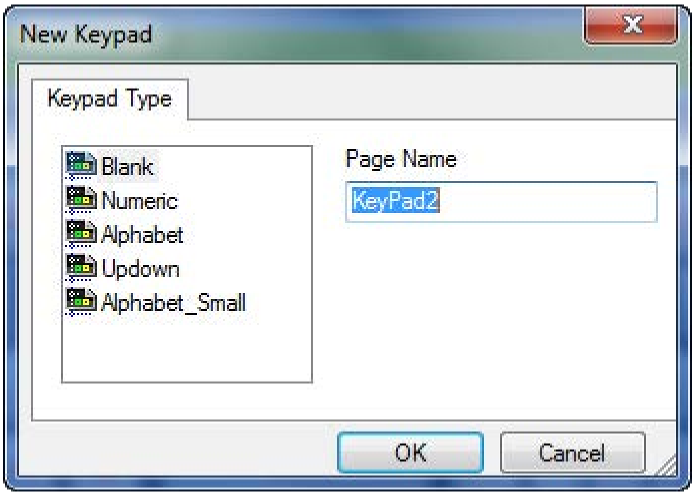
1. Right click on the **Keypad** folder from the Project View pane.

   ➤ The following context menu is displayed.



2. Click on **Insert KeyPad**.

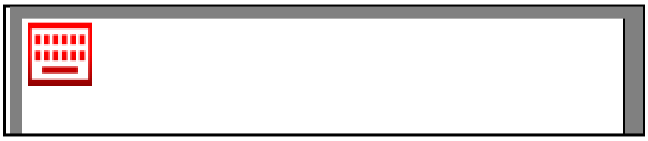
   ➤ The following pop-up with the *New Keypad* dialog is generated.



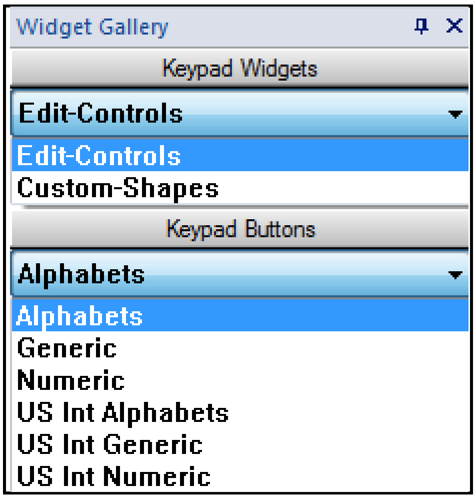
KeyPad2

3. You can select any of the available keypads that are provided in the project template (the list shown on the left side) to create a custom keypad. If you need to create a keypad from scratch, select the **Blank** option.

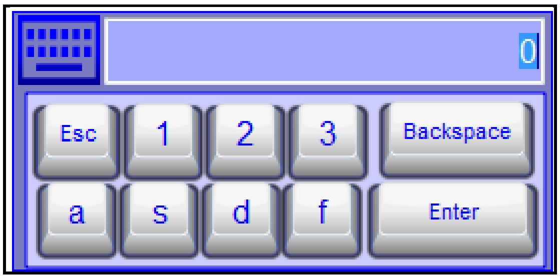
   ➤ The following blank keypad is inserted.



4. You can use the widgets available from the keypad widgets gallery to create the custom keypad.

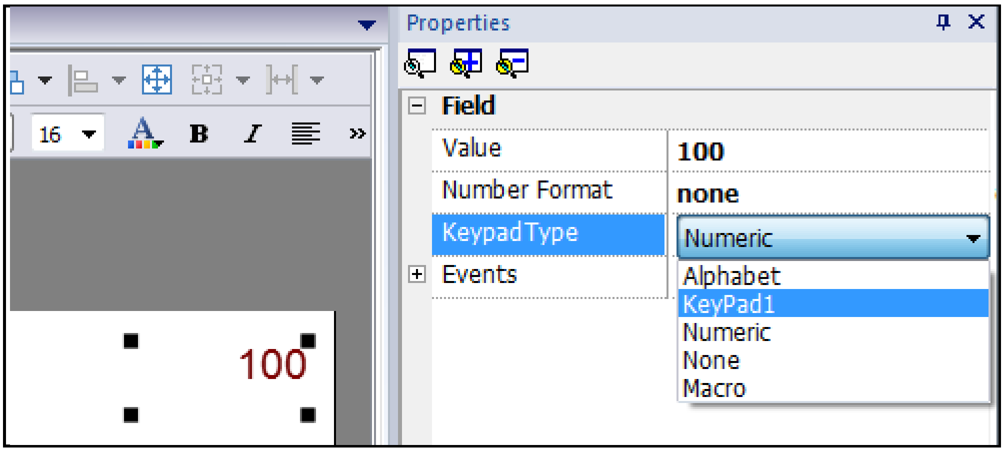


- ➤ Newly created keypads will be saved in the project folder, together with all other project files.



Sample of a custom-created keypad

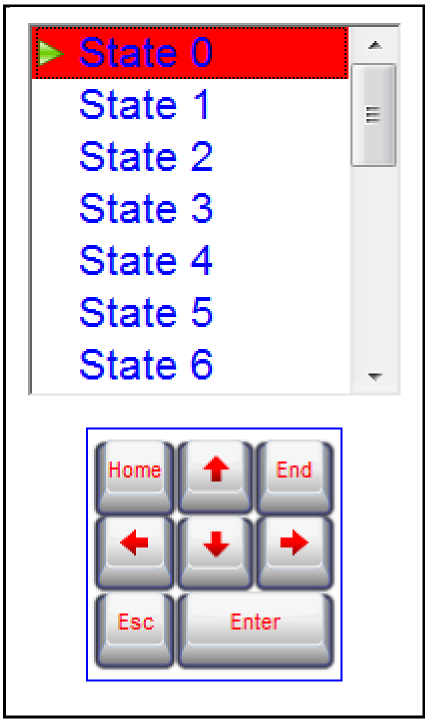
5. The custom-created keypad may be used for any specific field where the keyboard type property has been properly set by selecting the corresponding keypad from the property *Keypad Type* in the Property View.



Field

Value	100
Number Format	none
KeypadType	Numeric
Events	

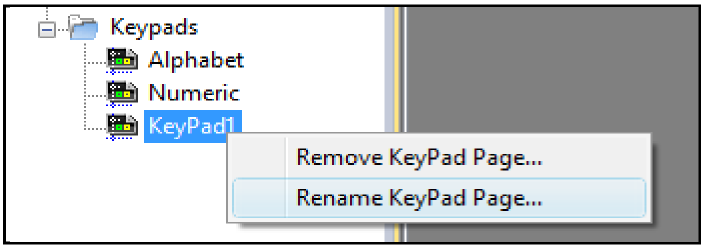
The *Up-down* keypad is mainly used for moving cursors in widgets and supporting this function. An example is the following control list.



Deleting or Renaming Custom Keypads

1. Right click on the *Keypad* folder in the Project View.

   ➤ The following context menu is displayed.



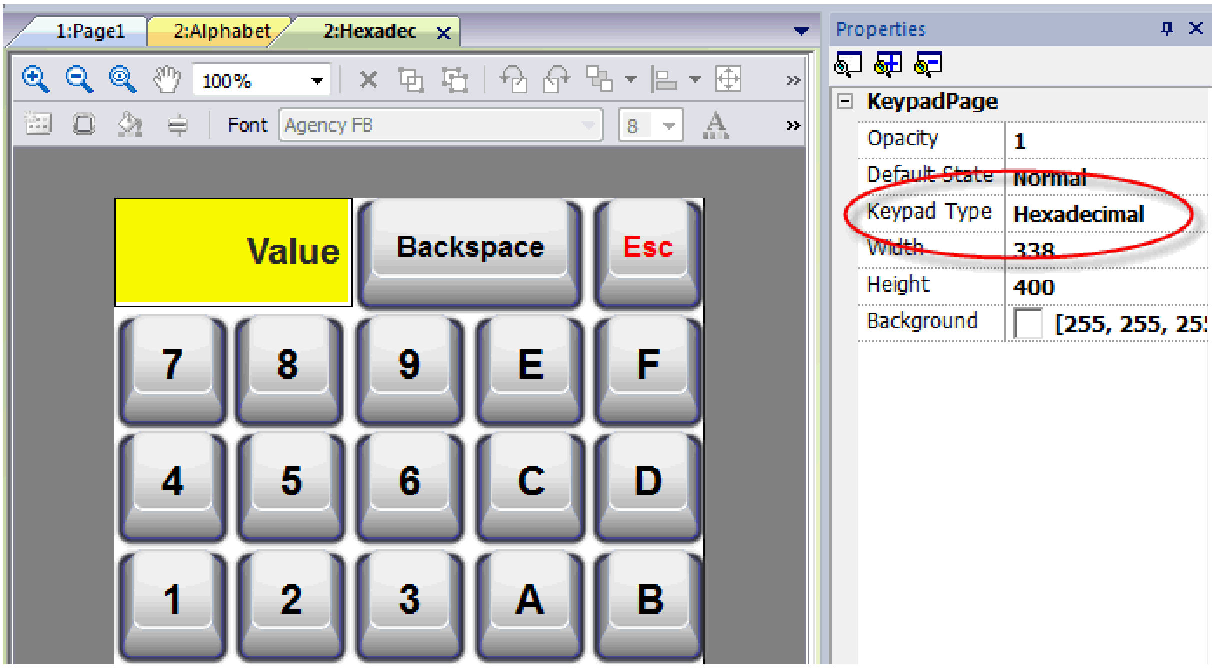
2. Choose the **Remove Keypad Page** option to remove the keypad page from the project, or the **Rename Keypad Page** option to rename the keypad.

> **NOTE**
>
> By default, any numeric widget (read/write numeric field) will be assigned the numeric keypad. If you decide to modify the default numeric keypad that will be used throughout the project, the following procedure is recommended, so you won't need to assign that new keypad to all numeric entry widgets. First, create a new keypad, using the numeric keypad as the keypad type and save it with a different name. This will be a backup of the numeric keypad. Then open and modify the default numeric keypad, and save it with its original name. The now modified numeric keypad will be assigned by default to all numeric fields in the project.

Keypad Type

The **Keypad Type** is one of the parameters available in properties window of keypads. Use this parameter to define what type of data entry is needed. Follow the list of options available:

Option	**Explanation**
Auto	This is the default.
Decimal	Only numeric keys are accepted. Entering 10, the keypad return back value 10 that will be display as 10 if the attached field is numeric or ASCII, as A if the attached filed is hexadecimal.
Hexadecimal	Only hexadecimal keys are accepted. Entering 10, the keypad return back value 16 that will be display as 16 if the attached field is numeric or ASCII, as 10 if the attached field is hexadecimal.
Ascii	All keys are enabled. Entering 1A keypad return back value 1A that will be display as 1 if the attached field is numeric, as 1A if the attached field is ASCII or as 1A if the attached field is hexadecimal.



KeypadPage

Opacity	1
Default State	Normal
Keypad Type	Hexadecimal
Width	338
Height	400
Background	□ [255, 255, 25

External Keyboards

Runtime has been designed to work with external keyboards connected via USB.

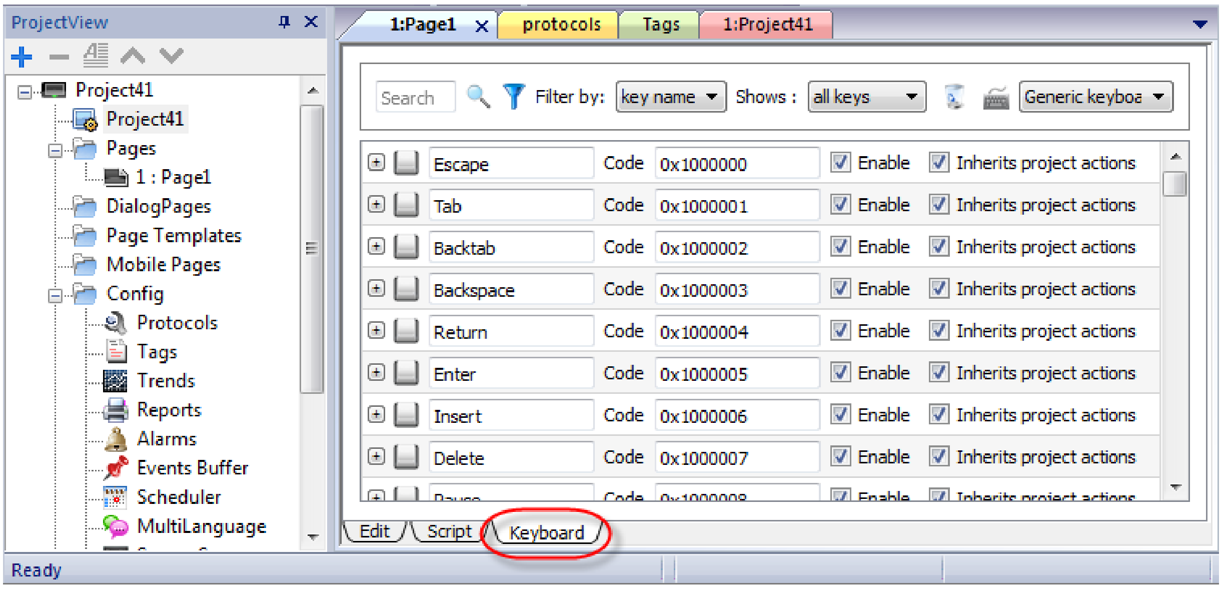
Keyboards can be used for

- Data Entry (default)
- Actions map on specific keys.

You can map for example, you can map the “right arrow” key event "pressOnClick" to the LoadPage action.

You can configure your keyboard at project level so that the setting you create will be inherited by all the page. In each page you can then choose which key setting will be inherited from the project and which one you will customize for the specific page.

The Keyboard Editor can be opened using the tab **Keyboard** at the bottom of the project or page workspace.



Escape	Code	0x1000000	☑ Enable	☑ Inherits project actions
Tab	Code	0x1000001	☑ Enable	☑ Inherits project actions
Backtab	Code	0x1000002	☑ Enable	☑ Inherits project actions
Backspace	Code	0x1000003	☑ Enable	☑ Inherits project actions
Return	Code	0x1000004	☑ Enable	☑ Inherits project actions
Enter	Code	0x1000005	☑ Enable	☑ Inherits project actions
Insert	Code	0x1000006	☑ Enable	☑ Inherits project actions
Delete	Code	0x1000007	☑ Enable	☑ Inherits project actions
Pause	Code	0x1000008	☑ Enable	☑ Inherits project actions

Each row in the Keyboard Editor corresponds to a Key. For each key, the following information is available:

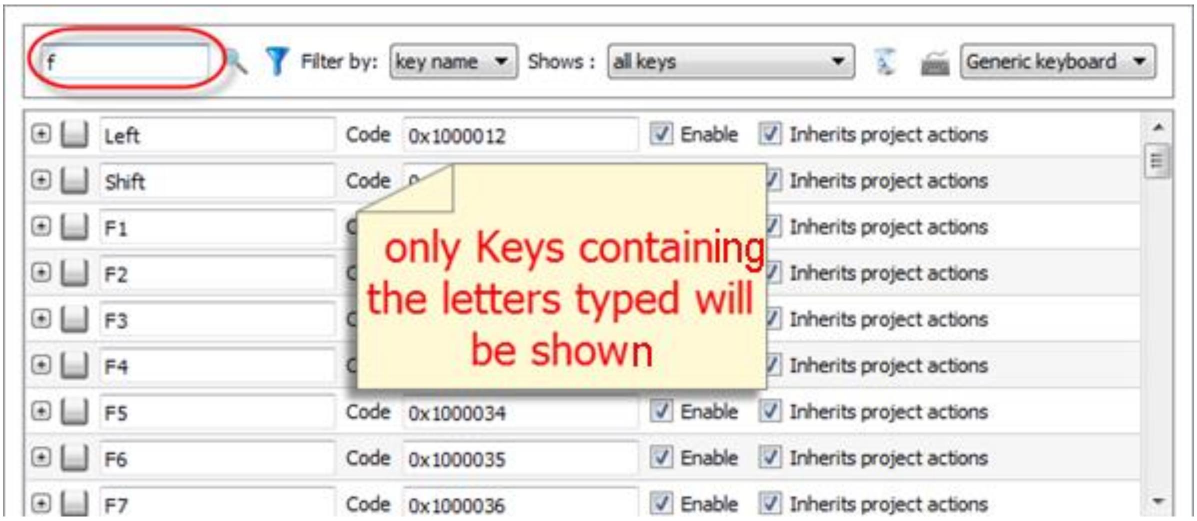
Information	**Explanation**
Item	Description
Label	The name of the key
Code	The code of the key
Enable	The individual enable status of the key
Inherits project actions	Defines whether the key is inheriting the action programmed at the project level

The table shows the possible configurations:

Enable	Inherits project actions	Editor appearance	Runtime behavior
Checked	Unchecked	Action lists show the page actions (or nothing if the list is empty)	Only the page actions (if any) will be executed.
Checked	Checked	Action lists show the project actions only and cannot be edited	Only the configured project actions (if any) will be executed.
Unchecked	Checked	Inherits project actions checkbox and all action lists are disabled. Action lists show the project actions only.	No page or project action will be executed.
Unchecked	Unchecked	Inherits project actions checkbox and all action lists are disabled. Action lists show the project actions only.	No page or project action will be executed.

Search and Filter

After selecting Filter by to key name, you can start typing the name of the Key in the box at the left of the toolbar and the Keyboard Editor will list only the keys whose Label contains the text you have entered.

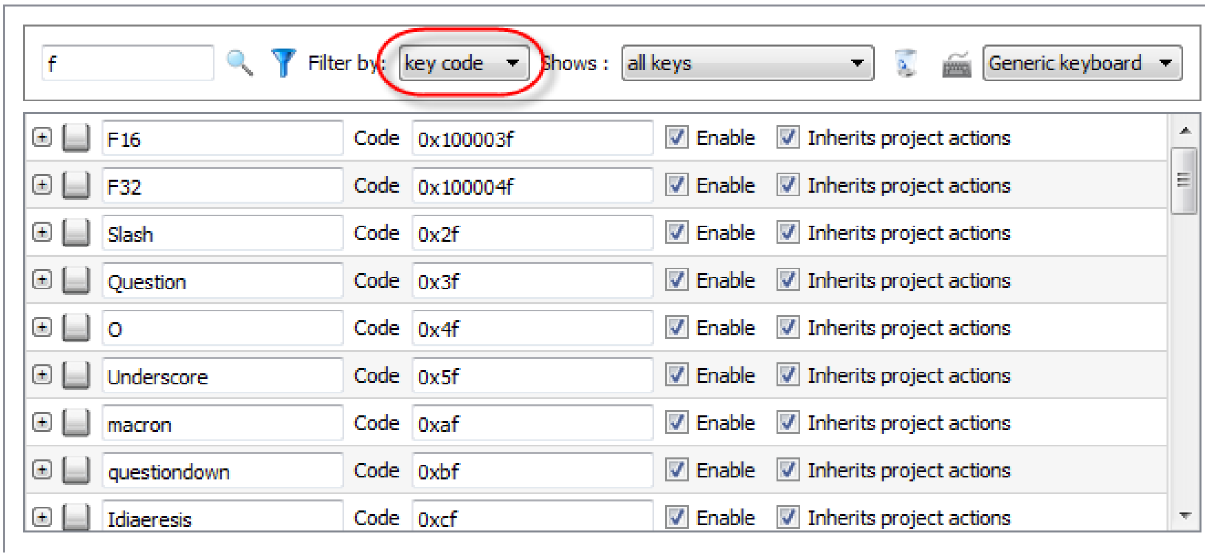


f | Filter by: key name | Shows : all keys | Generic keyboard

Left	Code	0x1000012	☑ Enable	☑ Inherits project actions
Shift	Code	0		☑ Inherits project actions
F1				☑ Inherits project actions
F2				☑ Inherits project actions
F3				☑ Inherits project actions
F4				☑ Inherits project actions
F5	Code	0x1000034	☑ Enable	☑ Inherits project actions
F6	Code	0x1000035	☑ Enable	☑ Inherits project actions
F7	Code	0x1000036	☑ Enable	☑ Inherits project actions

only Keys containing the letters typed will be shown

Alternatively, if **Filter by** has been set to **key code** only the Keys containing the text in their Code column will appear in the list.



f | Filter by: key code | Shows : all keys | Generic keyboard

F16	Code	0x100003f	☑ Enable	☑ Inherits project actions
F32	Code	0x100004f	☑ Enable	☑ Inherits project actions
Slash	Code	0x2f	☑ Enable	☑ Inherits project actions
Question	Code	0x3f	☑ Enable	☑ Inherits project actions
O	Code	0x4f	☑ Enable	☑ Inherits project actions
Underscore	Code	0x5f	☑ Enable	☑ Inherits project actions
macron	Code	0xaf	☑ Enable	☑ Inherits project actions
questiondown	Code	0xbf	☑ Enable	☑ Inherits project actions
Idiaeresis	Code	0xcf	☑ Enable	☑ Inherits project actions

Shows

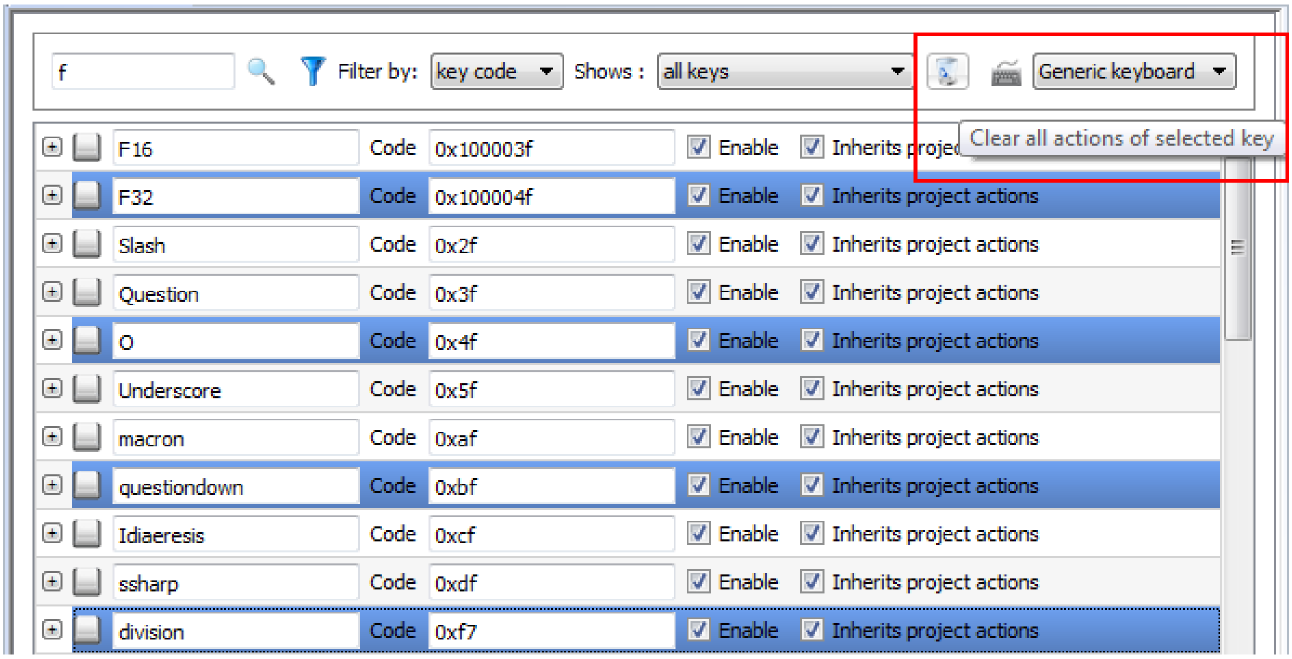
You can easily select what keys will be listed in the keyboard editor window.

Code	Description
All Keys	The editor will show all keys available in the keyboard layout
Modified Keys	The editor will show only keys that have been configured with some actions at the page level
Modified Keys in project	The editor will show only keys that have been configured with actions at the project level

Clear Actions

The **Clear Actions** button is available to delete actions configured for one or more keys in the Keyboard Editor. To Clear Action, select one or more keys and then press the Clear Actions button.

You will clear all action configured for the selected keys either in page or in project will be removed depending on what we are currently configuring, either the project actions in the project view or the page actions in the page view. A confirmation dialog will appear to request confirmation of the requested command.



f Filter by: key code Shows : all keys Generic keyboard

Clear all actions of selected key

Key		Code				
F16	Code	0x100003f	☑	Enable	☑	Inherits projec
F32	Code	0x100004f	☑	Enable	☑	Inherits project actions
Slash	Code	0x2f	☑	Enable	☑	Inherits project actions
Question	Code	0x3f	☑	Enable	☑	Inherits project actions
O	Code	0x4f	☑	Enable	☑	Inherits project actions
Underscore	Code	0x5f	☑	Enable	☑	Inherits project actions
macron	Code	0xaf	☑	Enable	☑	Inherits project actions
questiondown	Code	0xbf	☑	Enable	☑	Inherits project actions
Idiaeresis	Code	0xcf	☑	Enable	☑	Inherits project actions
ssharp	Code	0xdf	☑	Enable	☑	Inherits project actions
division	Code	0xf7	☑	Enable	☑	Inherits project actions

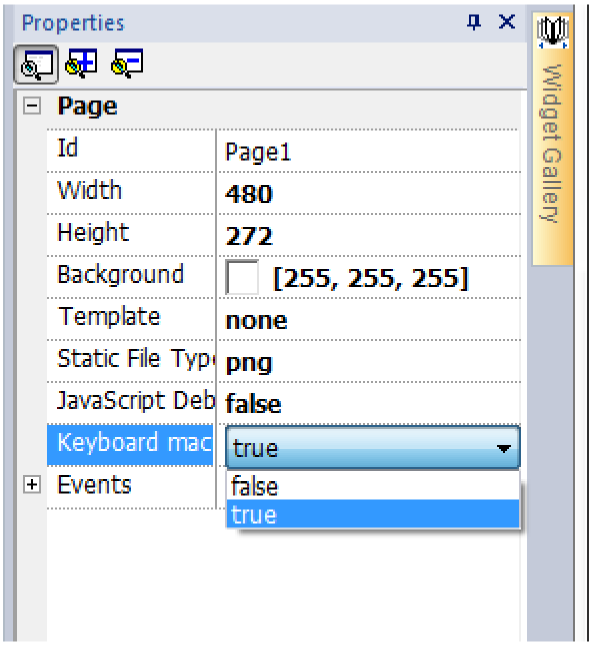
Keyboard Layout

The keyboard layout combo box allows the user to select the layout of the keyboard.

Generic Keyboard corresponds to a generic International Keyboard layout.

Enable Keyboard

You can enable/disable keyboard actions both at the project level and at the page level. A dedicated property is available in the project property sheet and in the page property sheet.



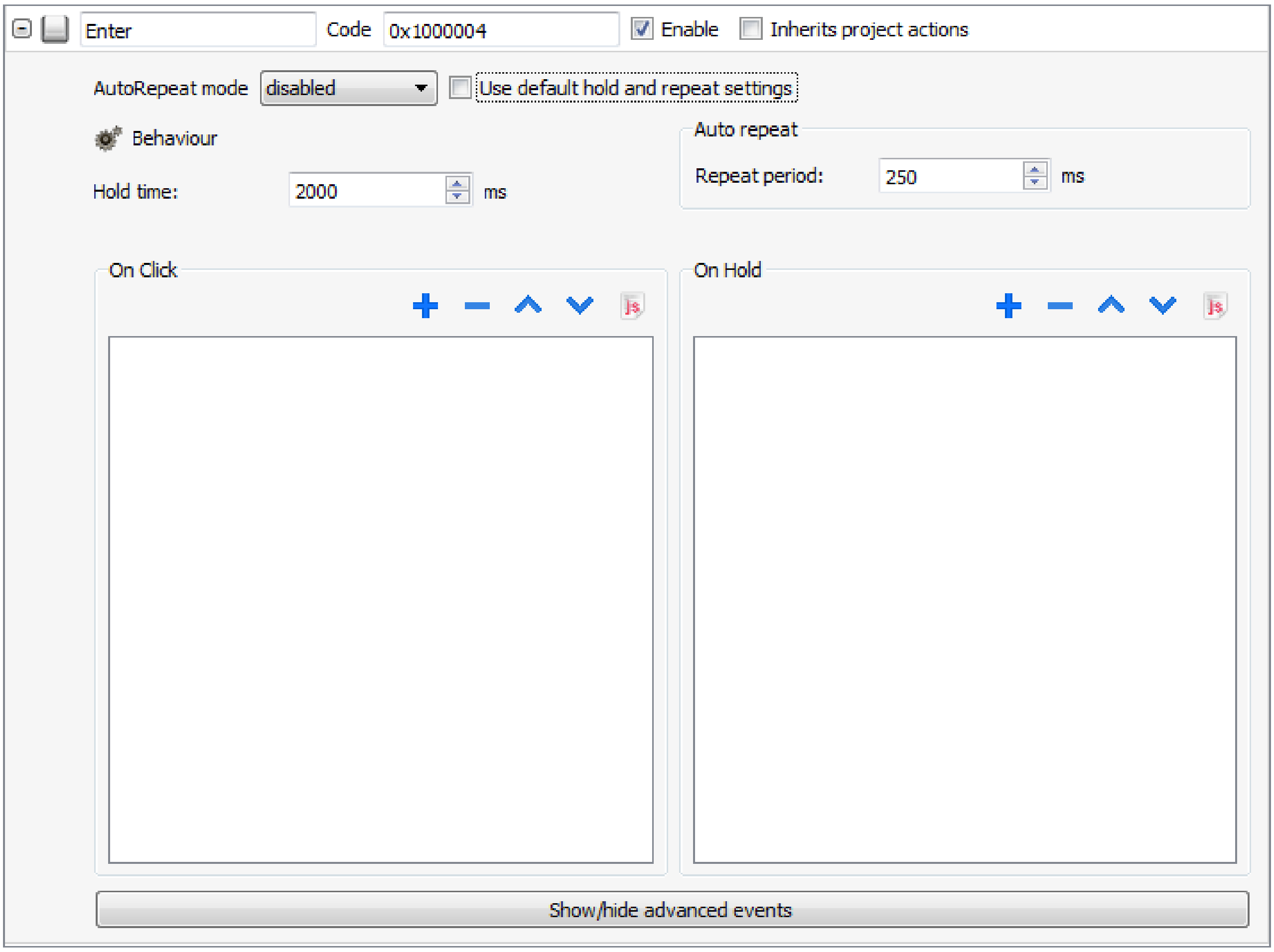
Page	
Id	Page1
Width	480
Height	272
Background	□ [255, 255, 255]
Template	none
Static File Typ	png
JavaScript Deb	false
Keyboard mac	true false true
Events	

The Keyboard can also be disabled at runtime by using a dedicated macro command **KeyboardMacros**.

Configure Macro Actions for Keys

To configure actions for keys in the Keyboard Editor just click on **+** on the key you want to program and you will obtain the expanded view for key configuration.

Press the + or JS buttons to add macro commands or Javascript functions to the key event you want to configure.



Enter Code 0x1000004 ☑ Enable □ Inherits project actions

AutoRepeat mode: disabled □ Use default hold and repeat settings

Behaviour

Hold time: 2000 ms

Auto repeat

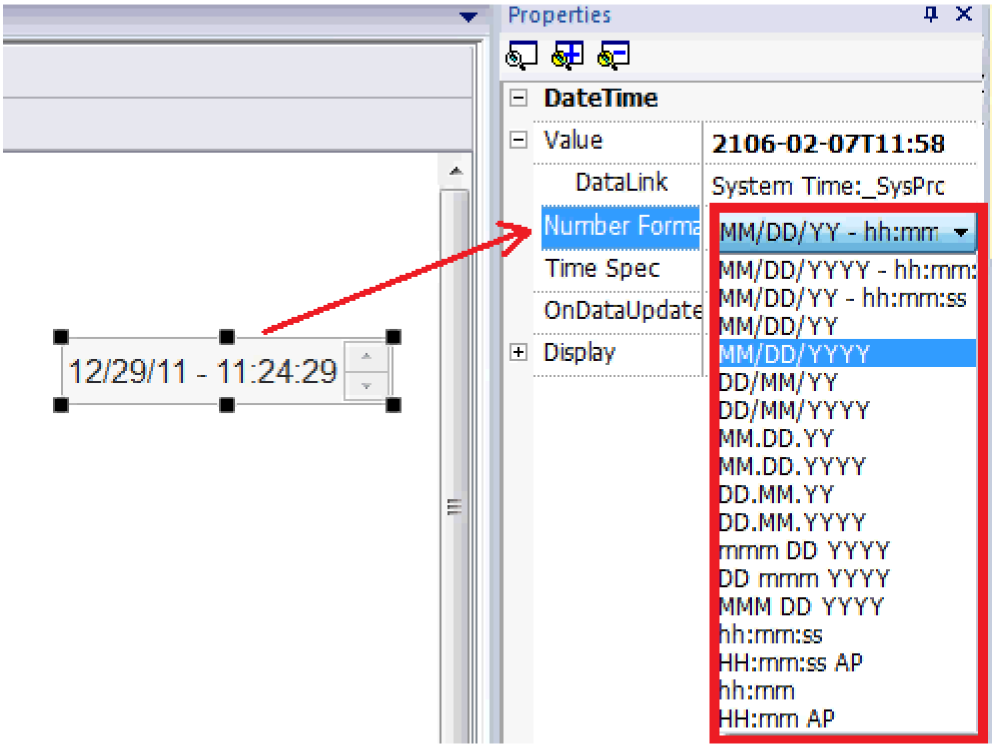
Repeat period: 250 ms

Special Widgets

DateTime Widget

DateTime widget can be used to view and edit the current time and date at runtime. The widget can be found in the widget gallery in *Basic* > *Edit-Controls*.

In the Property View pane of the widget you can set the format of date and time as “Date only”, “Time only”, “Date and Time”; different formats for representing date and time are available as shown in figure below.



DateTime
Value 2106-02-07T11:58
DataLink System Time:_SysPrc
Number Forma MM/DD/YY - hh:mm
Time Spec
OnDataUpdate
Display

“Time Spec” option allows selecting which time the widget has to show during runtime, three options are available for this property:

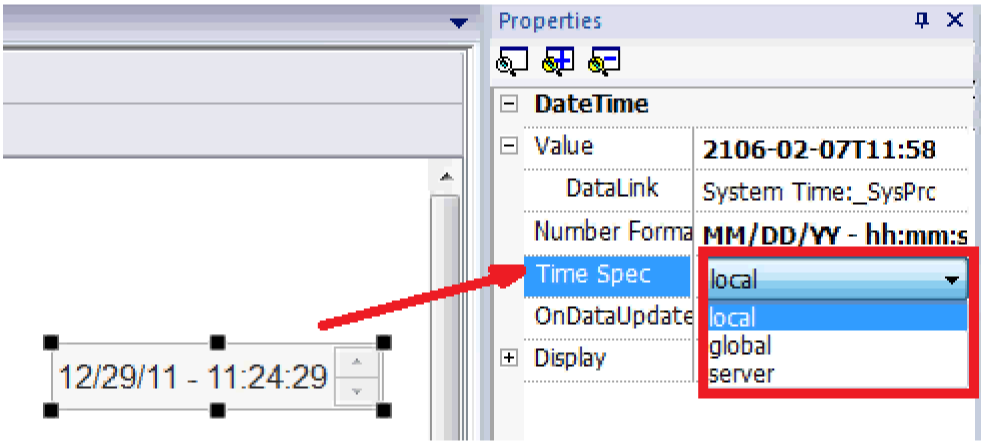
- “server”,
- “local” and
- “global”.

To understand the difference between the options available for the “Time Spec” property, you need to recall the basic concepts behind the HMI system architecture. Please read the chapter “Server Runtime Modes” to get familiar with the HMI software architecture first.

If we select “server” as “Time Spec”, the widget will show the time information as handled by the server side of the HMI system.

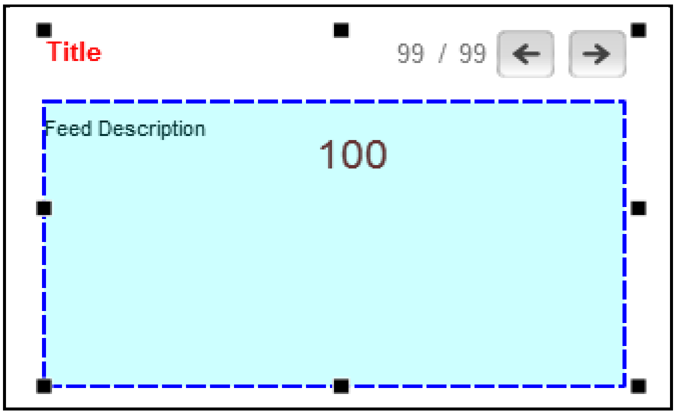
If we select “global” as “Time Spec”, it will show the Global Time (GMT).

If we select “local” as “Time Spec”, it will show the Local Time in the widget (the time of the target where the project is running).



RSS Feed Widget

The RSS (**R**eally **S**imple **S**yndication) Feed widget allows to display on the screen your favorite RSS Feeds directly from the internet. The widget is available in the widget gallery under the media category.



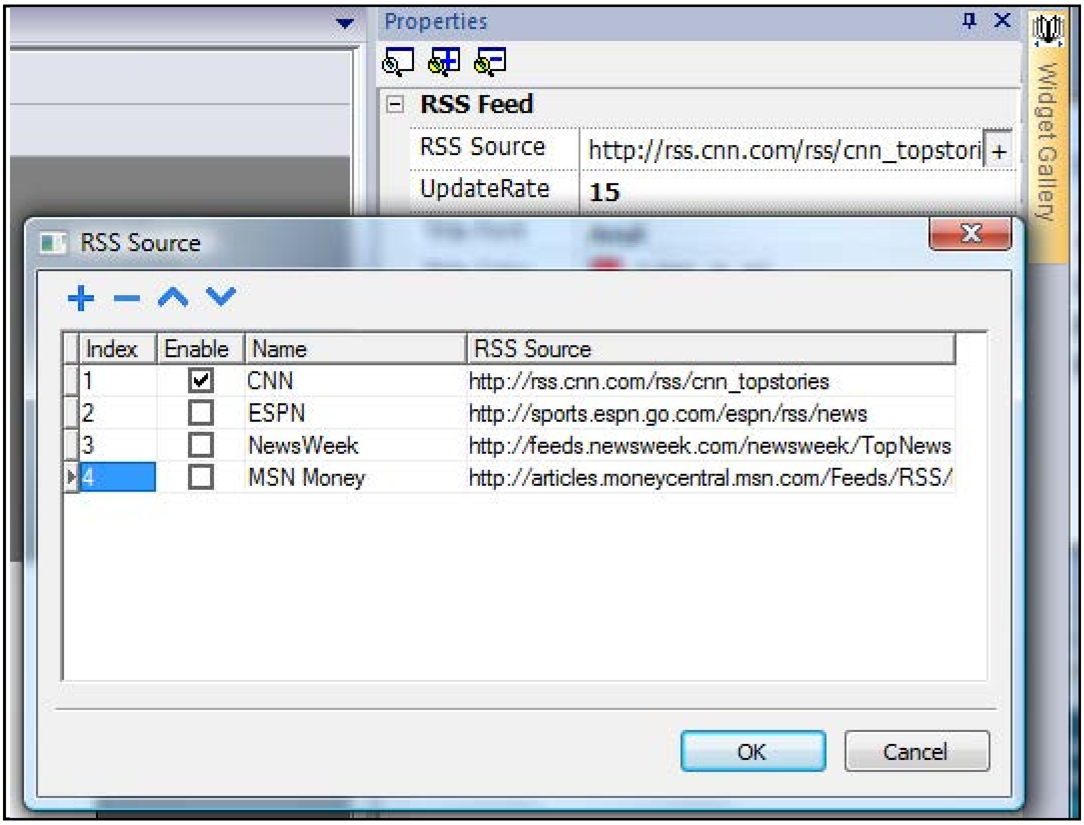
RSS Feed widget

The RSS Feed widget main properties are:

- *RSS Source* allows you to specify the feed URL,
- *UpdateRate* allows to specify the refresh time.

NOTE

Feeds sources are fixed and cannot be changed in runtime.



RSS Feed	
RSS Source	http://rss.cnn.com/rss/cnn_topstori
UpdateRate	15

Index	Enable	Name	RSS Source
1	☑	CNN	http://rss.cnn.com/rss/cnn_topstories
2	□	ESPN	http://sports.espn.go.com/espn/rss/news
3	□	NewsWeek	http://feeds.newsweek.com/newsweek/TopNews
4	□	MSN Money	http://articles.moneycentral.msn.com/Feeds/RSS/

Properties

NOTE

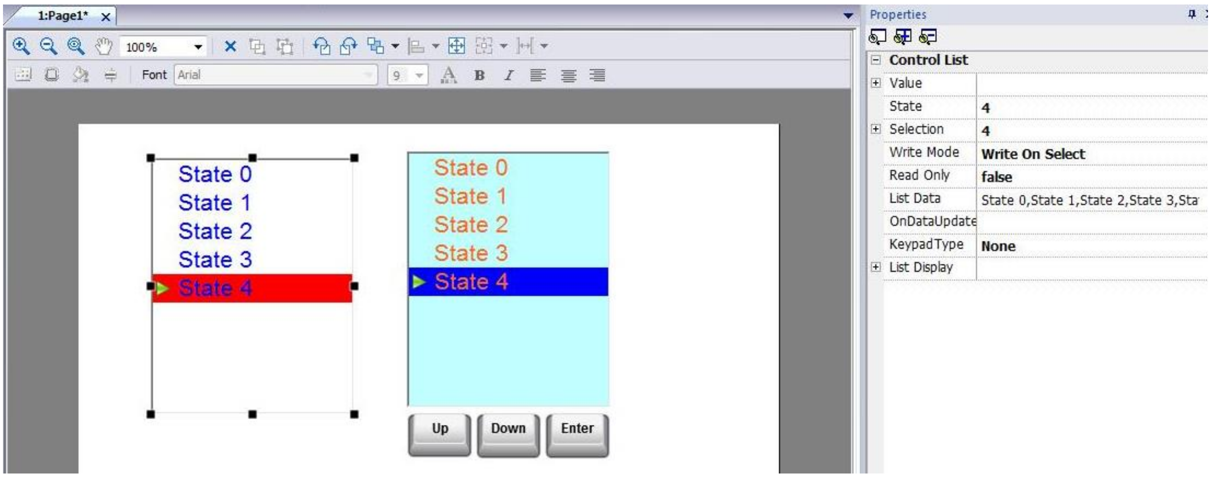
The RSS widget is specifically designed to work on units where the Internet Explorer browser (Pocket Internet Explorer) is part of the operating system.

Control List Widget

PB610 Panel Builder 600 provides control list widgets, a convenient way to represent the status associated to a particular process but also a way to control that process from the same widget.

The control list widget is available in the widget gallery under the advanced category.

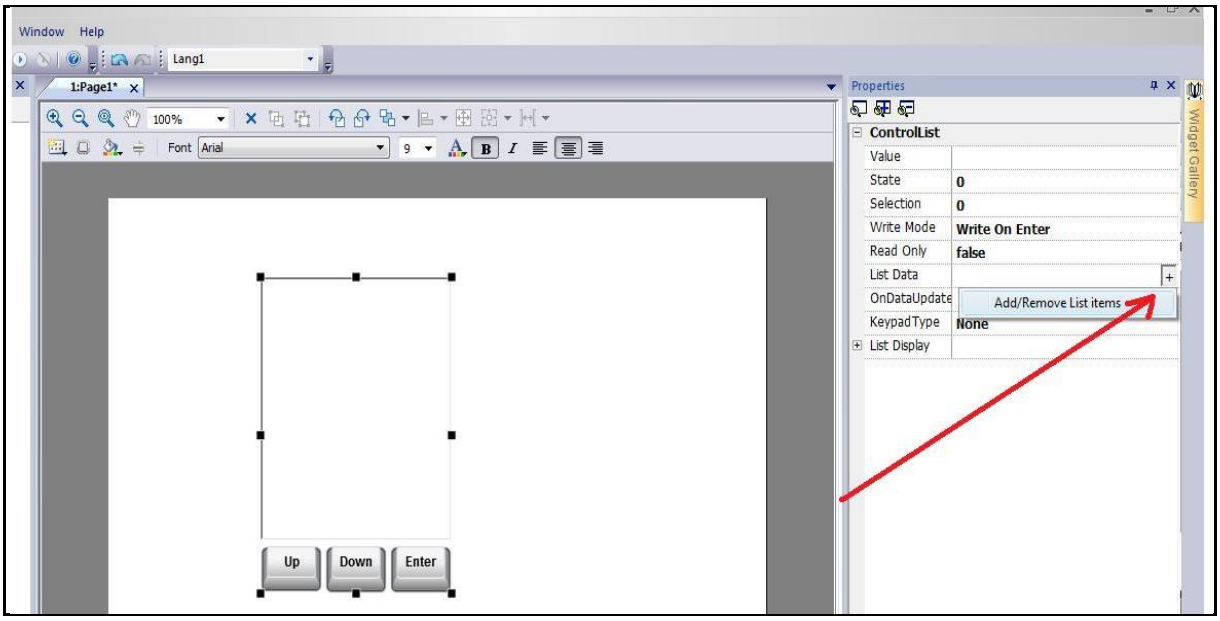
There are two types of control lists. One is a control list group, in which the up and down buttons are present on the control list itself. The state can be selected with the up and down buttons. The other type of control list has no pre-configured buttons in the group. In that case, the state can be selected by pressing on the screen.



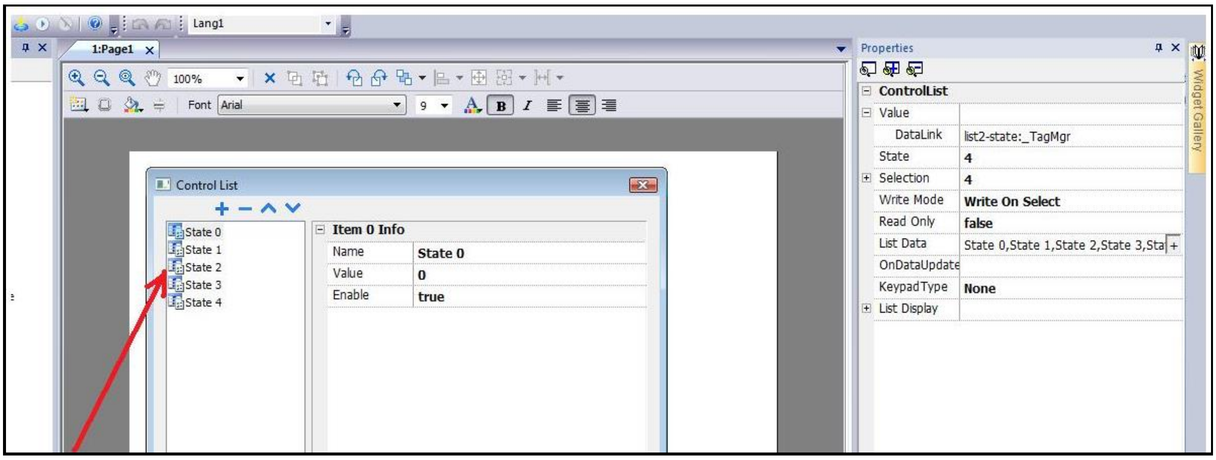
Control List	
Value	
State	4
Selection	4
Write Mode	Write On Select
Read Only	false
List Data	State 0,State 1,State 2,State 3,Sta
OnDataUpdate	
KeypadType	None
List Display	

State

States are added by selecting **Add/Remove** list items from the “List Data” voice in the Property View pane. Any value can be assigned to a state. Activating the state will result in a write operation to the tag, which has been linked to the value property of the control list widget.

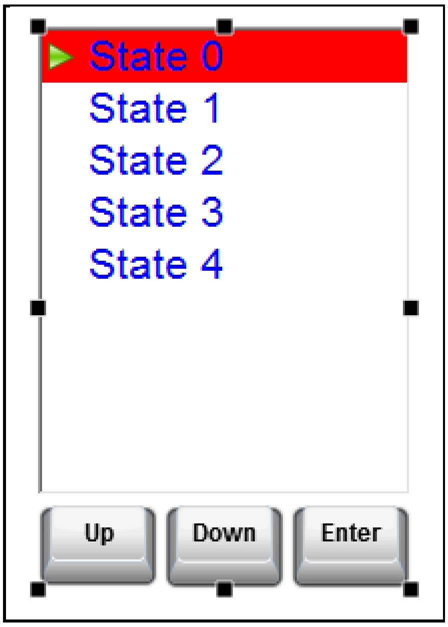


ControlList	
Value	
State	0
Selection	0
Write Mode	Write On Enter
Read Only	false
List Data	
OnDataUpdate	
KeypadType	None
List Display	



Item 0 Info	
Name	State 0
Value	0
Enable	true

ControlList	
Value	
DataLink	list2-state:_TagMgr
State	4
Selection	4
Write Mode	Write On Select
Read Only	false
List Data	State 0,State 1,State 2,State 3,Sta
OnDataUpdate	
KeypadType	None
List Display	



Selection

Selection shows which status is currently selected, and will appear as a highlight cursor moving up and down (according to the use of the defined keys). The *Selection* property can be attached, as well, to a tag. The small triangle on the left side of the list tells you what the current status is.

There are two write modes for the control list:

Write on Select:

→ .The value will automatically be written when the status is selected.

Write on Enter:

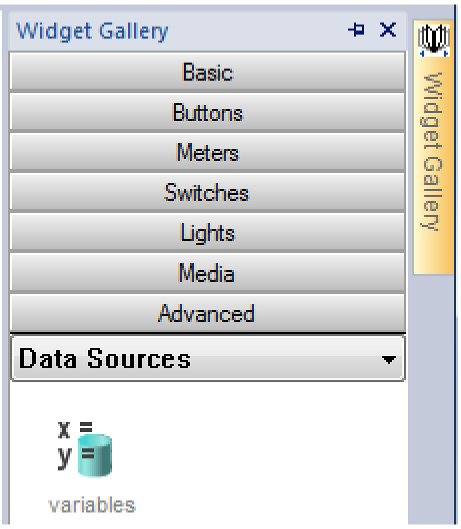
→ First select the state, and then press the enter key to write the status value to the tag.

Read Only:

The Read only property of the widget can be attached to a tag and will control whether the control list will be just an indicator, or a combination of both. For example, with a machine in Manual mode, the Control list will let the operator select which state should be active and while in Auto mode, the list is an indication of the active step.

Variables Widgets:

The variables widget is available in the advanced category under the “Data Sources” sub-category.

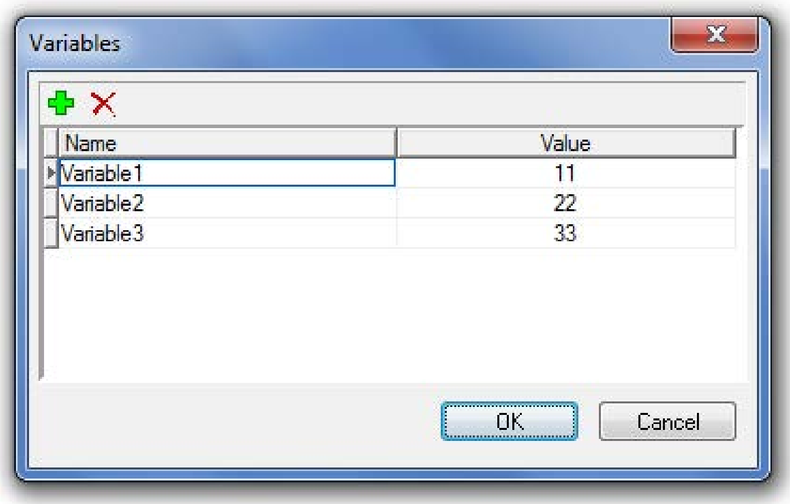


The purpose of variable widgets is to have some internal variables that can be used for certain operations such as data transfer or use in JavaScript. The variables are local to the page where the widget is inserted.

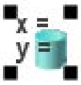
1. Drag and drop the widget to any position on the page.

   ➤ This will display a place holder to remember the widget is there, but it will not be visible at runtime.

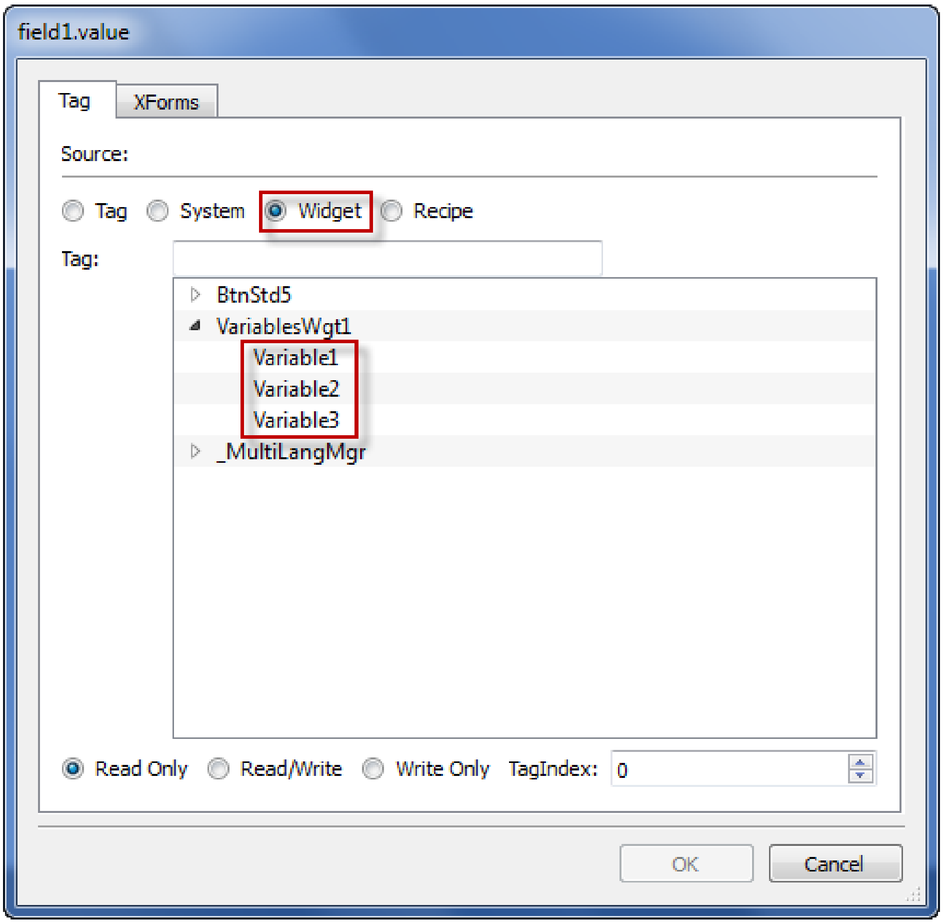
→ You can create some variables and assign values.



Name	Value
Variable1	11
Variable2	22
Variable3	33



The configured variables can be referenced from the “Attach” tag dialog once you click on the widget source.



field1.value

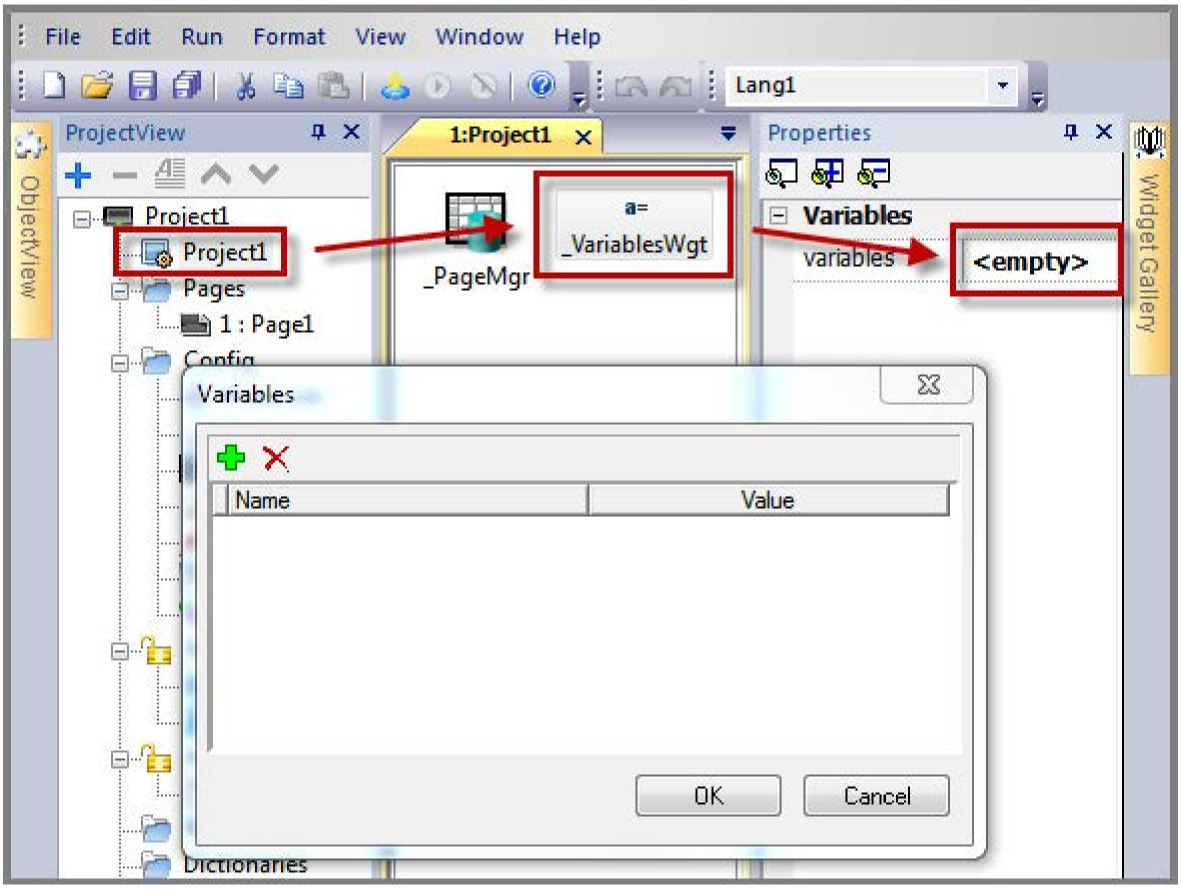
Source: ○ Tag ○ System ● Widget ○ Recipe

Tag:

- BtnStd5
- VariablesWgt1
  - Variable1
  - Variable2
  - Variable3
- _MultiLangMgr

● Read Only ○ Read/Write ○ Write Only TagIndex: 0

In case you need global variables, they can be configured from the project widget adding the desired variables to the global variable widget.



Variables

variables <empty>

Name	Value

Using Variables in JavaScript

The variables defined according to the explanation above can be also referenced in JavaScript with the following syntax:

- **For local variables:**

```
var varWgt = page.getWidget("_VariablesWgt");
```

```
var compVar = varWgt.getProperty("VariableName");
```

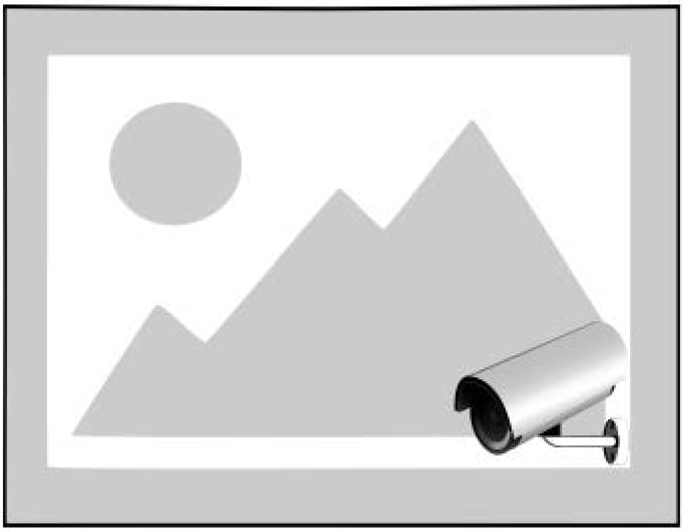
- **For global variables:**

```
var varWgt = project.getWidget("_VariablesWgt");
```

```
var compVar = varWgt.getProperty("VariableName");
```

IPCamera Widget

An IPCamera widget is available in **Widget Gallery.** Using this widget is possible to show images captured from an IPCamera to show a video stream.



Follow the list of main parameters available for IP Camera widget:

Parameter	**Explanation**
Camera IP	IP/URL of the IPCamera when used in JPEG format.
Refresh Rate	Number of JPEG images for second allowed. The max frame rate is 1fps.
User Name	Useful when IPCamera device is protected by a username & password
Password	Useful when IPCamera device is protected by a username & password
MJPEG Camera IP	IPCamera widget can be used also with streaming HTTP MJPEG. In this case parameters “Camera IP” and “Refresh Rate” are ignored. “MJPEG Camera IP” is the IP of URL of MJPEG streaming. Ex. http://192.168.0.1/video.cgi

The only supported protocol is HTTP.

For showing single frames the only supported format is JPEG while for streaming the only supported format is Motion JPEG.

Performance of streaming are not fixed and depend on many factors like:

- Frame size

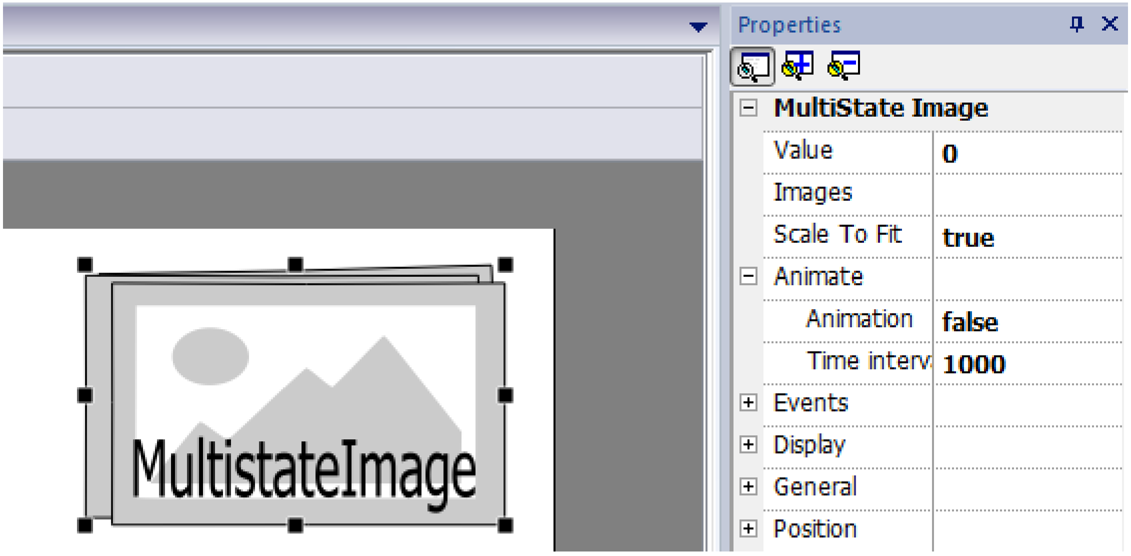
- Frame compression level
- CPU of HMI Panel
- Quality of IPCamera

Based on these factors the widget can reach up to 25 fps.

Multiple widgets are supported, however in this case performance could reduce frame rate of each single widget.

Multistate Image Widget

Multistate Image is a widget designed to show an image from a collection based on the value of a tag used as Index. This widget may be used also for simple animations.



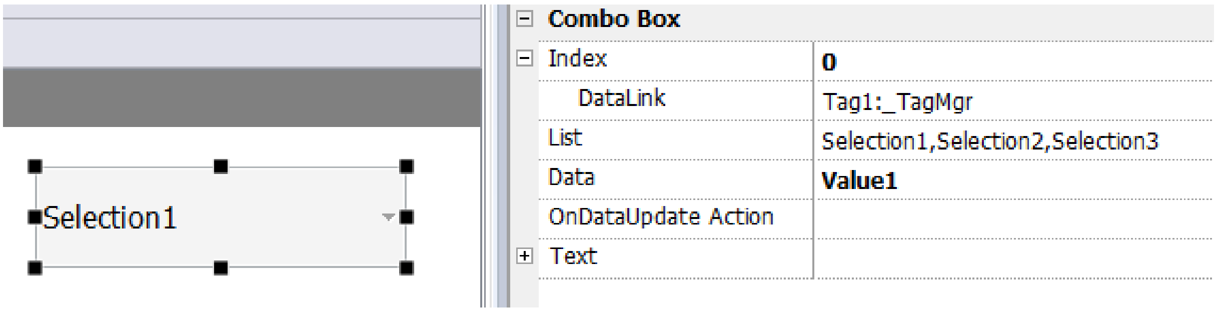
MultiState Image
Value 0
Images
Scale To Fit true
Animate
Animation false
Time interv 1000
Events
Display
General
Position

Follow the list of parameters of MultistateImage widget:

Parameter	Explanation
Value	Index of Image to show in Widget. Attaching this property to a tag is possible to use this tag as Index, as a selector. Ex. value=0 means show Image with Index 0 in “Images” collection.
Images	Images collection. Add/Remove Images is used to add/remove images that later can be shown using related Index.
Animate	When Animation is true, a slideshow is shown and images change every 1000 ms (default Time interval).

Combo Box Widget

The Combo Box widget is available in widget gallery and is already used by many widgets as a selector widget or as a way to filter rows shown in a table (like alarms or trends) based on values selected on combo.

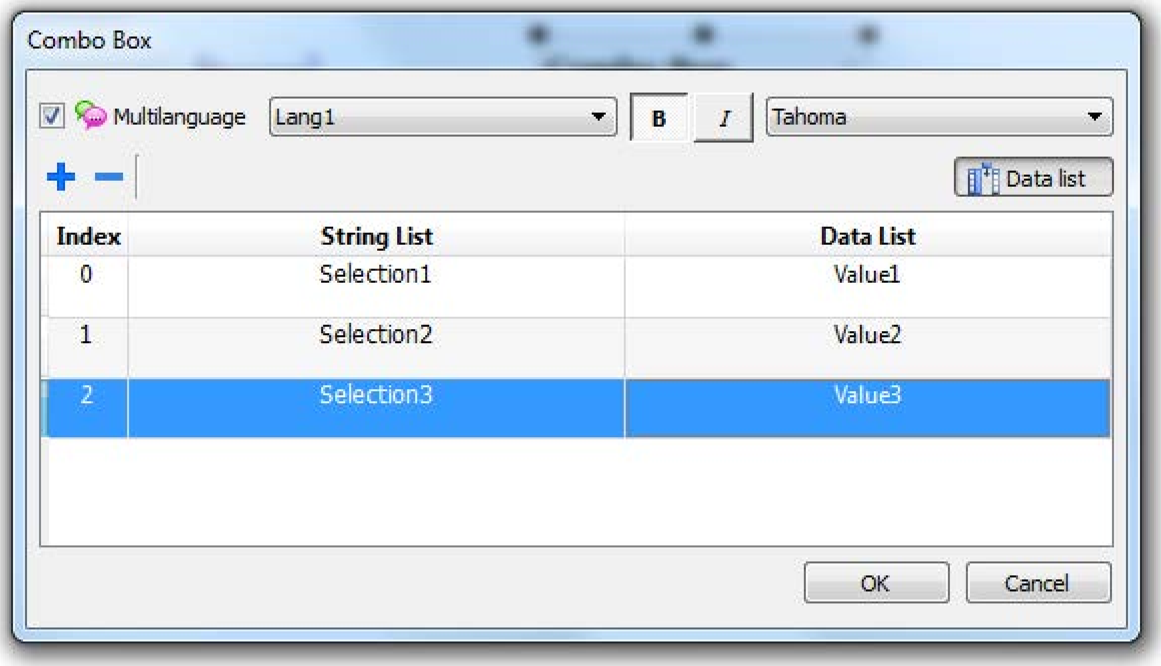


Selection1

Combo Box	
Index	0
DataLink	Tag1:_TagMgr
List	Selection1,Selection2,Selection3
Data	Value1
OnDataUpdate Action	
Text	

Follow the list of parameters of Combo Box Widget:

Parameter	Explanation
Index	Each item listed in a combo has an index 0...n. This field returns the index (integer) of selected item in combo.
List / String List	Strings to show as items into combobox widget. This field is multilanguage.
Data / Data List	Optional parameter available in “advanced” mode that allows returning in field “Data” of the widget the related value reported in “Data List” (as string). Usually “Index” is enough for 90% of applications, however sometime is useful to return a custom value based on item selected in combo box.



Combo Box

☑ Multilanguage: Lang1; Tahoma

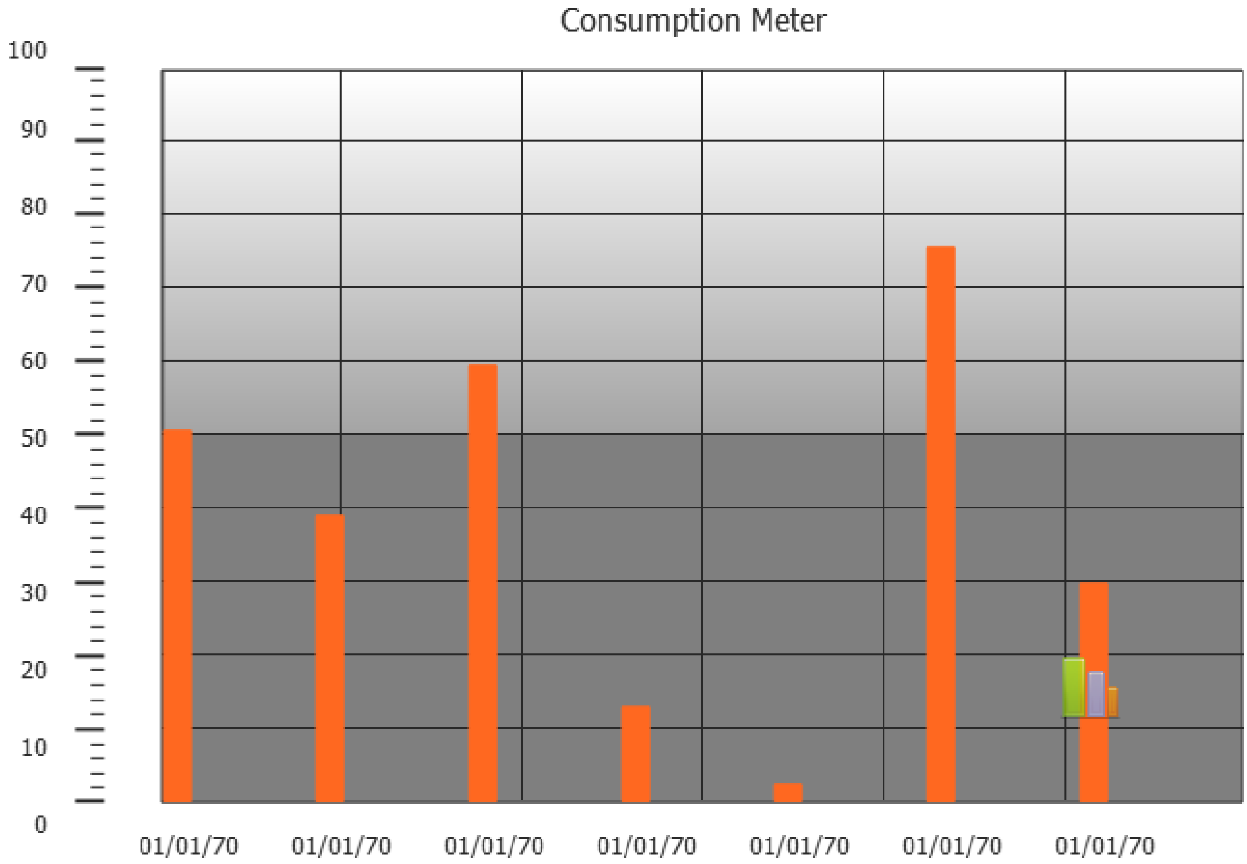
Index	String List	Data List
0	Selection1	Value1
1	Selection2	Value2
2	Selection3	Value3

It is useful in many applications to attach fields like **Index** or **Data** to tags to know the values related to the selected item in the combo box. When attached to **Index**, the tag will contain the index (integer) of the selected item (0...n), when attached to **Data**, it will contain the data value (string) specified in the **Data List.**

Consumption Meter Widget

Consumption meter widget is available in the widgets gallery. This widget has been intended specifically to monitor a resource which is continuously increasing. The system reads the value of the resource and calculate the increment in a predefined a range of time, the increment is then represented in a trend-like window, using bar-graphs. A typical application for this widget is the calculation of the power consumption of a system.

Representation is done using a bar graph where it is also possible (when range is weekly) to assign different colors based on the range of time.



Below you can find a description of the main properties of this widget:

Property	Explanation
Value	This is the resource monitored by the system
Graph Duration / Graph Duration Units	These properties determines the time period that will be represented in the trend window
Bar Duration/Bar Duration Units	These properties determines the time period represented by each bar composing the graph
Time Periods	This property allows to highlight, with a minimum resolution of 1 hour, the increment of the monitored resource in a determined time period by using a specific color. Each bar composing the graph will then be represented using different colors, each showing the increment of the monitored resource in the corresponding time period.

Example: design a monitor for the consumption of energy with a weekly scale and a daily unit. Follow these steps to configure the widget:

1. Add protocol
2. Add Tag and link it to the physical variable to monitor (total energy consumed – ex. KW/h), we can call this Tag KWh. This tag contain an incremental number that summarize how many KW/h has been consumed from when energy started.

3. Add a Trend and link it to the KWh Tag to monitor

4. Add Consumption meter widget into a Page

5. Attach **Value** property of Consumption Meter to the KWh Tag.

6. Set **Graph Duration/Units** to 1 Week (range of time considered by Widget). This allow us to have a weekly graph of consumed energy.

7. Set Bar Duration/Units to 1 day (range of time where calculate consume of energy)

8. In Properties -> **Consumption Meter, you can** change the number of labels to show in the bar graph (ex. **X Labels** = 7 if we need a weekly graph).

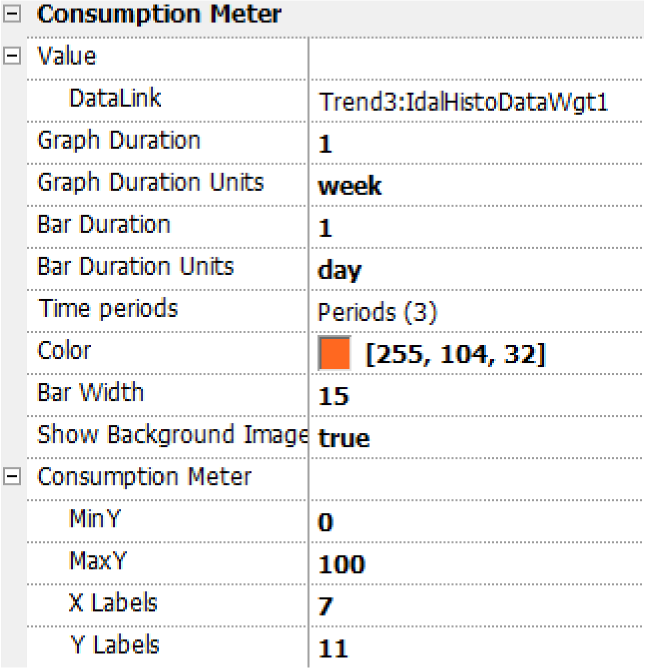
9. **Open Time Periods** to access a configuration dialog that allow the setting of different colors for different values of the Tag KWh monitored in each bar.

10. Add as many color bands as you need, in this example we've added 3 color bands.

11. Assign to each hour in the weekly table the related band. In this example a red band (E1) was used to indicate the range of time in the day/week where the cost of energy is the highest.

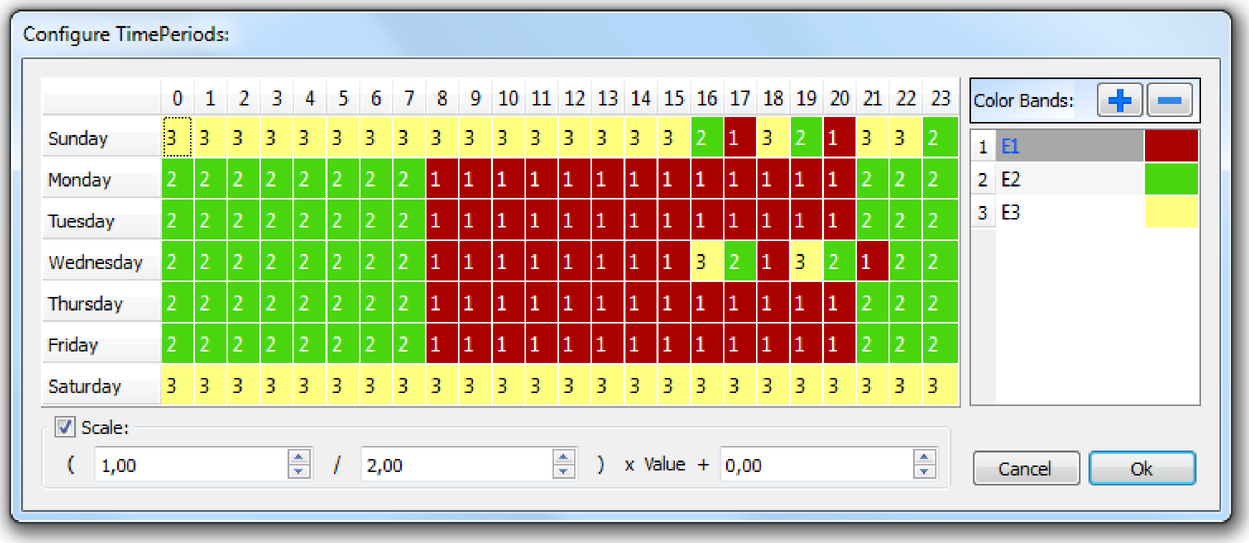
12. For each band, if needed, a scale factor can be assigned.

The result is a consumption meter as a bar graph that shows daily consumption of energy (KW/h in this example) where the colors indicate the different energy cost that have been consumed. The height of each bar represents the amount of energy in the range of time considered (1 day in this example).



⊟ **Consumption Meter**	
⊟ Value	
DataLink	Trend3:IdalHistoDataWgt1
Graph Duration	**1**
Graph Duration Units	**week**
Bar Duration	**1**
Bar Duration Units	**day**
Time periods	Periods (3)
Color	🟧 **[255, 104, 32]**
Bar Width	**15**
Show Background Image	**true**
⊟ Consumption Meter	
MinY	**0**
MaxY	**100**
X Labels	**7**
Y Labels	**11**

In **TimePeriods dialog**, to assign the color to the cells of the table, you can select the cells and click on the related band/color. Another way is to enter the index value of the band (1, 2, 3) into the cell to color it.



Configure TimePeriods:

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
Sunday	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	2	1	3	2	1	3	3	2
Monday	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	1	1	1	1	1	2	2	2
Tuesday	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	1	1	1	1	1	2	2	2
Wednesday	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	3	2	1	3	2	1	2	2
Thursday	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	1	1	1	1	1	2	2	2
Friday	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	1	1	1	1	1	2	2	2
Saturday	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3

Color Bands:

1	E1	🟥
2	E2	🟩
3	E3	🟨

☑ Scale:

(1,00 / 2,00) x Value + 0,00

The Macro **ConsumptionMeterPageScroll** can be used to scroll the bar graph back and forth. The macro **RefreshTrend** is necessary to refresh the bar graph because it’s not automatically refreshed. All other Macro of Trends are not supported today by the consumptionMeter.

Custom Widgets

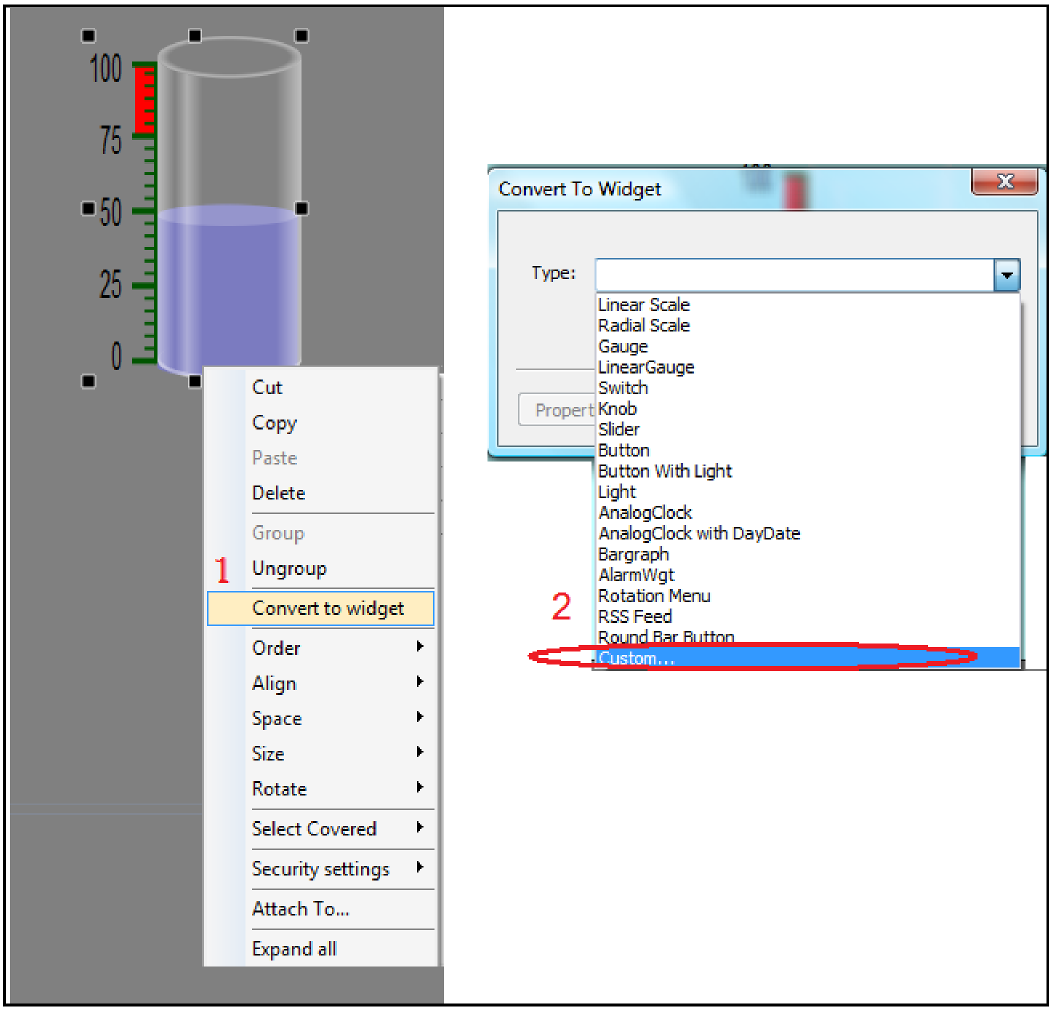
PB610 Panel Builder 600 includes a large library that includes predefined dynamic widgets (such as buttons, lights, gauges, switches, trends, recipes, and dialog items), as well as static images (such as shapes, pipes, tanks, motors etc.). With the widget library, you can simply drag and drop an object on to the page, and then size it, move it, rotate it or transform it any way you want. All widgets in the gallery are vector based and allow lossless enlargement.

Custom widgets are widgets created by the user and based on the various existing widgets in the gallery. This chapter describes how to create a customized widget and assign its properties.

The advantage of the custom widget is that it builds a widget out of any complexity, including several elements, and can decipher which properties have to be published and made available in the “custom widget” Property pane.

Creating Custom Widget

1. Select all the widgets you want as part of the custom widget and make them one group.
2. Right click on the group, and select **Convert to widget** from the context menu.



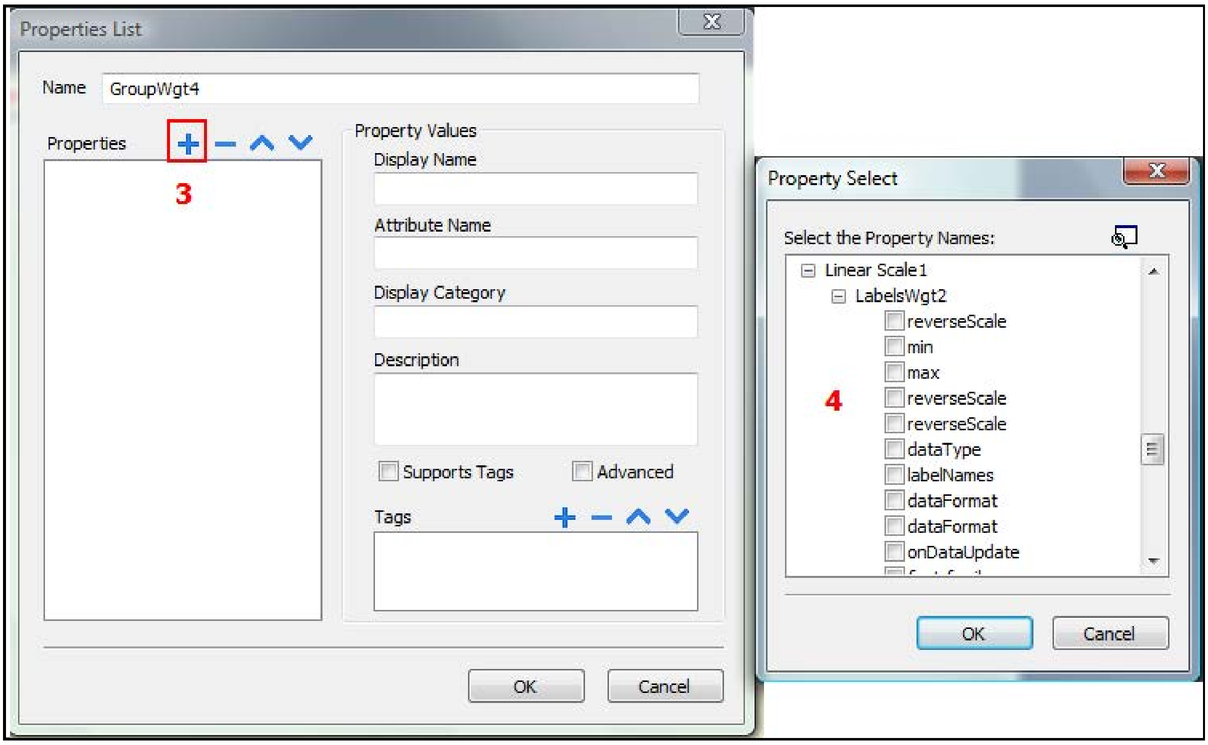
3. Select existing custom widget types (such as knobs, button with light etc.) or select **Custom…** to create a new custom widget type.

Adding the Properties

After creating the custom widget, the next step is to add the properties that will be published in the custom widget property pane. The *Property Select* dialog shows all the applicable properties for the grouped widget. This is basically a list of all the properties of all widgets grouped together.

✓ A custom widget is created.

1. Click the corresponding check box to select the properties.



2. Enter the name of the custom widget.

   ➤ This is the name that will appear in the Property View.

3. The next step is to select the properties that will be displayed in the Property View. Click on the **+** button above the properties list box.

   ➤ The *Property Select* dialog will be displayed.

NOTE The *ConvertToWidget* dialog shows “standard” custom widget types. These types are defined in the gallery. The dialog, however, does not show types that are specifically created for a project.

Display Name

The *Display Name* and *Description* are names that will be shown in the Property View. You can change this value to set the information for each custom widget property.

Attribute Name

The *Attribute Name* is the name exposed by PB610 Panel Builder 600, to JavaScript functions and *Attach Tag* dialog. The default property name has the form *WidgetType.name*. *WidgetType* is the type of widget and *name* is the attribute name. If you have more than one widget of the same type, the widget type name will be *WidgetType01*, *WidgetType02* etc.

Display Category

The *Display Category* is the category or group of the property in the Property View. All properties in the same category are shown together in the Property View. This allows you to organize the properties in the Property View. The *Display Category* allows you to view by category group, by clicking on either the **Collapse** or **Expand** button. For example, you can declare position properties, like the X-coordinate, height, width properties in a single display category called *Position*.

Description

The *Description* property allows you to define the description and comments within the property. The information will be displayed in the Property pane.

Advanced

The properties are shown in standard or advanced mode. The *Advanced* check box allows you to specify whether each property should appear in the advanced, or in the simple Property pane.

Supports Tag

The *Supports Tags* check box must be marked if the property supports the “Attach To” attribute.

Tags

The *Tags* list box indicates the internal tag name for the widget. This internal tag name is typically the same as the attribute name. But you can assign a different attribute name for your custom widget.

The *Tags* list is also used to combine tags.

1. If you want to combine two or more properties into one, select the primary property in the property list and click on the **+** button above the *Tags* list box.

   ➤ The *Property Select* dialog is displayed.

2. Select the properties that should be combined.

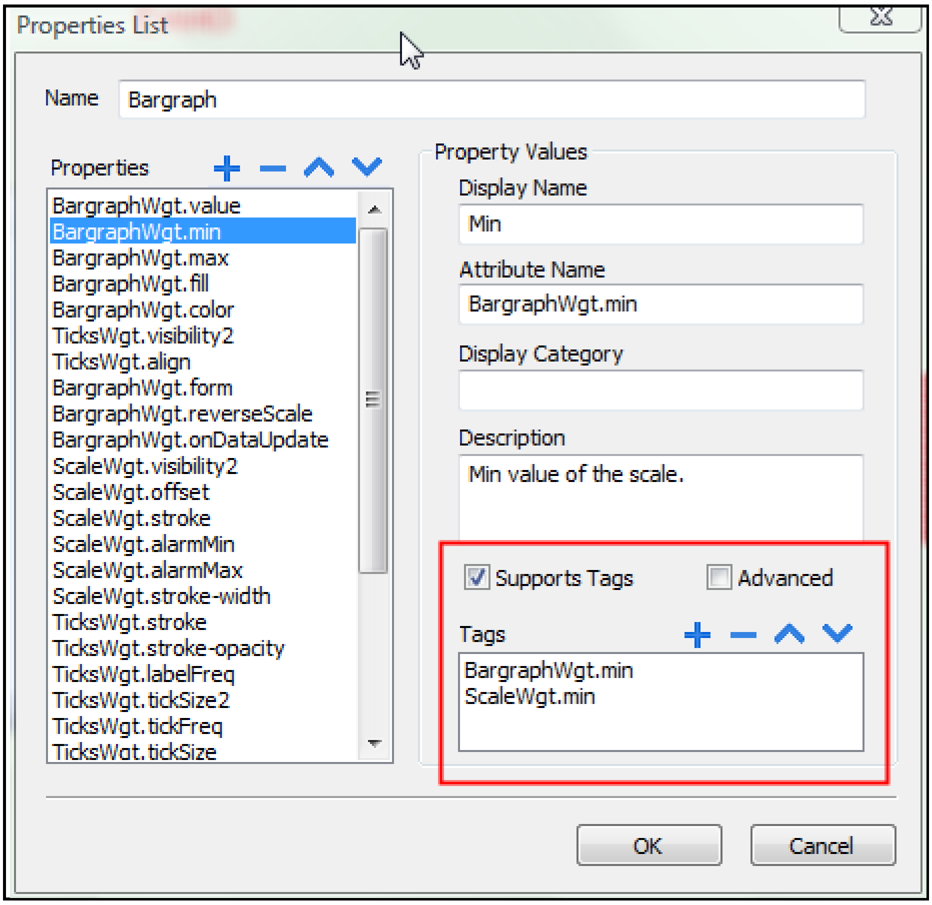
NOTE

This dialog box only shows the properties that should be combined. Not all properties are shown in the properties list.

Example:

1. To combine the *min* property of the scale widget and bar graph widget, click on the *NeedleWgt* min property and click on the *BargraphWgt* min property from the property select dialog.
2. Click the **OK** button.

   ➤ Both attributes are shown in the *Tags* list box.



Properties List

Name: Bargraph

Properties:
BargraphWgt.value
BargraphWgt.min
BargraphWgt.max
BargraphWgt.fill
BargraphWgt.color
TicksWgt.visibility2
TicksWgt.align
BargraphWgt.form
BargraphWgt.reverseScale
BargraphWgt.onDataUpdate
ScaleWgt.visibility2
ScaleWgt.offset
ScaleWgt.stroke
ScaleWgt.alarmMin
ScaleWgt.alarmMax
ScaleWgt.stroke-width
TicksWgt.stroke
TicksWgt.stroke-opacity
TicksWgt.labelFreq
TicksWgt.tickSize2
TicksWgt.tickFreq
TicksWgt.tickSize

Property Values
Display Name: Min
Attribute Name: BargraphWgt.min
Display Category:
Description: Min value of the scale.

☑ Supports Tags ☐ Advanced

Tags:
BargraphWgt.min
ScaleWgt.min

→ You can arrange the order of the properties by clicking on the **up** or **down** button in the property list.

→ To remove a property, select the property name and click on the **delete** button.

   ➤ When a property is selected in the property list, the property information is shown in the dialog box.

NOTE

Custom widgets usually are composed by many sub widgets. For example a button is a complex widget composed by two Image widgets, a button widget and label. This is clearly visibile in the ObjectView when the widget is selected. To select a sub widget like the label in a button, use ObjectView or Shift + leftClick of mouse. In this way sub widget can be changed without ungroup all widget.

Editing Custom Properties

You can change the properties of a custom widget after it has been created.

1. Simply right click on the widget in the page editor and select the **Custom Properties** menu item from the context menu.

   ➤ The *Custom Properties* dialog is displayed.

2. Change the properties.

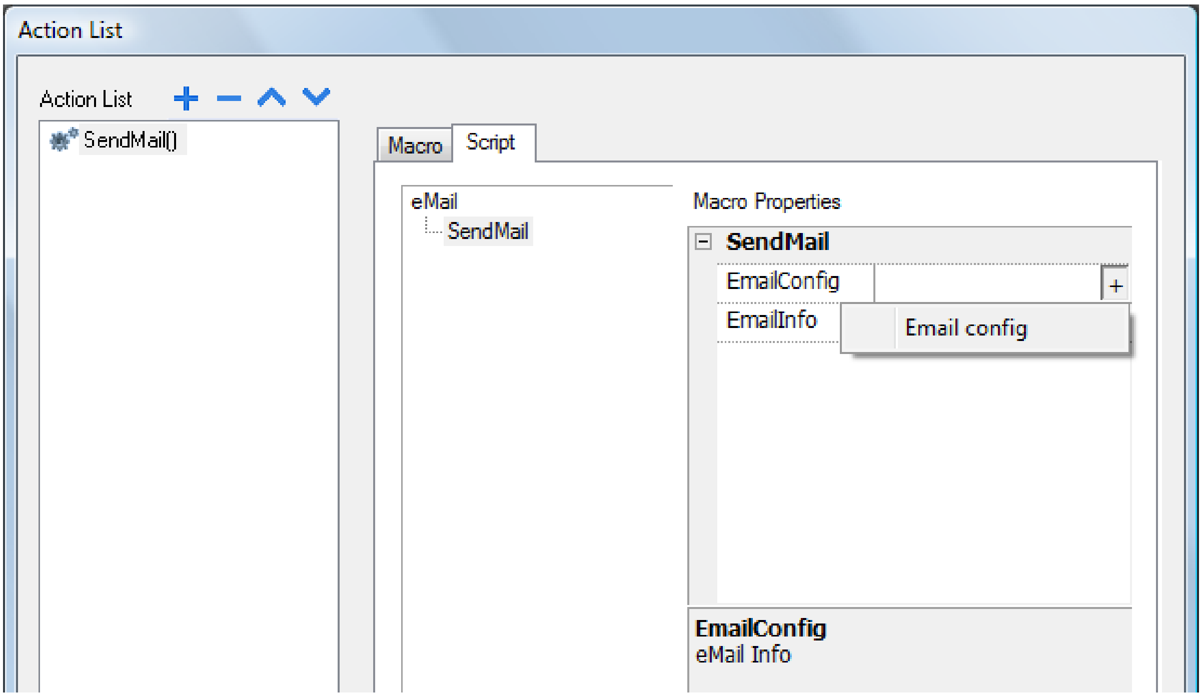
Sending E-Mail

The *SendMail* action can be programmed as trigger action for an alarm or for a timed scheduled action.

You can include tags in the e-mail body. Upon executing the action, tags value, instantaneously, will be detected by the system and included in the message body.

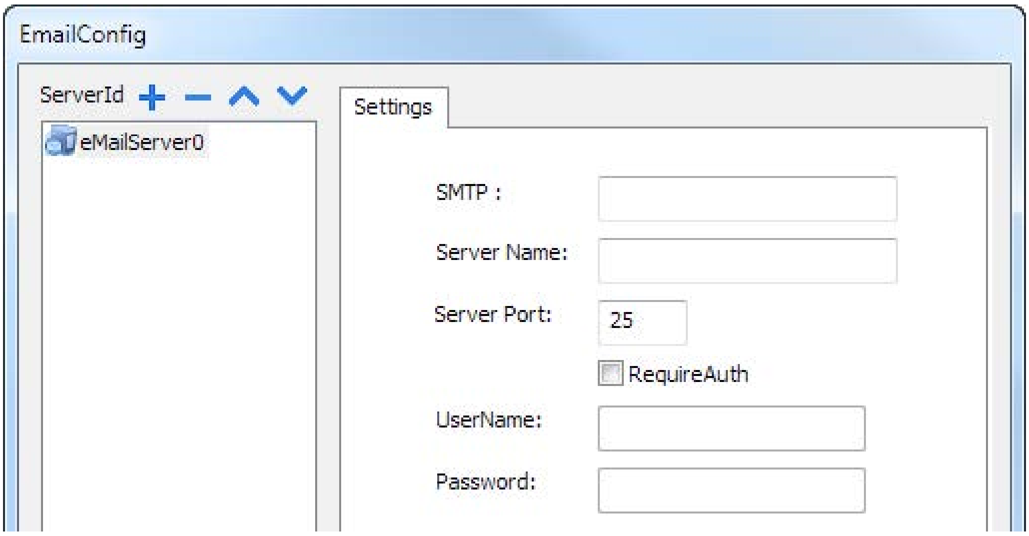
Sending E-Mail Script

The *SendMail* action is available under the script tab of the action list for alarms and scheduler actions.



Configure E-Mail Server

To configure the e-mail server, you need to provide the following information: SMTP Server Address (**SMTP**), **Server Name** (optional – it can be used for information purposes), **Server Port** and, if authentication is required, **UserName** and **Password**.



SMTP :

Server Name:

Server Port: 25

□ RequireAuth

UserName:

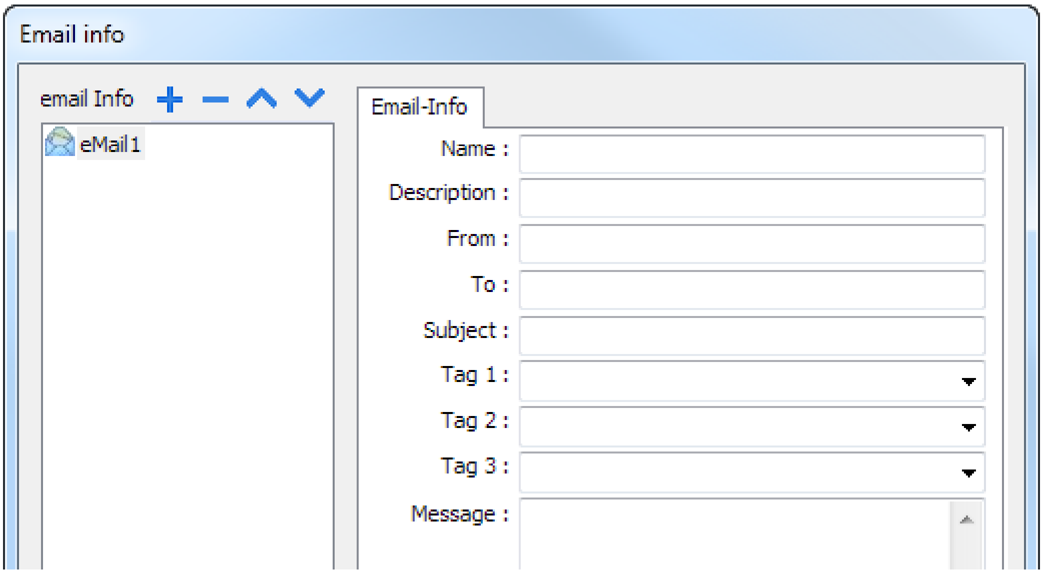
Password:

NOTE

You can add more E-mail servers by clicking the “+" button on the left hand side.
When you are writing, the system does not yet support any security login mechanism, such as SSL.

Configure E-Mail Accounts

In the *e-mail Info* the recipient e-mail addresses are set. If you want to send to more recipients, separate the e-mail address with a semi-colon “;”.

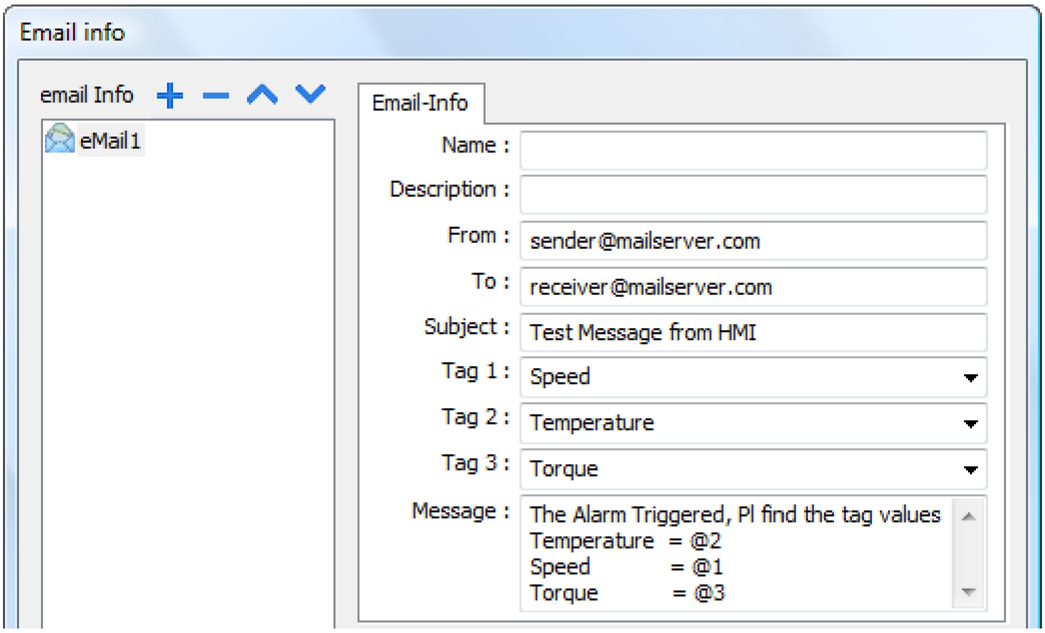


Sending Live Tag Data through E-Mail

You can send live tag data to the recipients within the e-mail body.

1. In *e-mail Info*, select the tags you want to send from the Tag1, Tag2, Tag3 dialog boxes.
2. In the e-mail body, use the keyword “@TagIndex" to display the tag data.

The “Tag” information that follows the “@” symbol represents the index of the tag as per the configuration made. In the example below, the “Temperature” tag has index 2; to insert the value of the “Temperature” tag in the mail message body, use the “@2” syntax.



Name :

Description :

From : sender@mailserver.com

To : receiver@mailserver.com

Subject : Test Message from HMI

Tag 1 : Speed

Tag 2 : Temperature

Tag 3 : Torque

Message :

```
The Alarm Triggered, Pl find the tag values
Temperature  = @2
Speed        = @1
Torque       = @3
```

Example

Limitation

A maximum number of 3 tags is supported in each e-mail message body.

JavaScript

The purpose of this chapter is to describe the JavaScript interface implemented in PB610 Panel Builder 600. PB610 Panel Builder 600 JavaScript is based on the ECMA Script programming language, http://www.ecmascript.org, as defined in standard ECMA-262. Microsoft's Cakra and Firefox Spider Monkey JavaScript engines support the ECMA Script standard. If you are familiar with JavaScript, you can use the same type of commands in PB610 Panel Builder 600 as you do in a web browser. If you are not familiar with the ECMA Script language, there are several existing tutorials and books that cover this subject, such as:

https://developer.mozilla.org/en/JavaScript

The purpose of this document is not to explain JavaScript language, but rather to describe how JavaScript is used in the PB610 Panel Builder 600 HMI Runtime.

Execution

A JavaScript function is executed when an event occurs. For example, a user can define a script for the *OnMousePress* event and the JavaScript script will be executed when the button is pressed on the panel.

It is important to note that JavaScript functions are not executed in the same manner as certain other controller programming scripts, such as Ladder Logic. JavaScript functions are not executed at a given scan rate the whole time, but they are only executed when the given event occurs. This approach minimizes the overhead required to execute logic on the Control Panel.

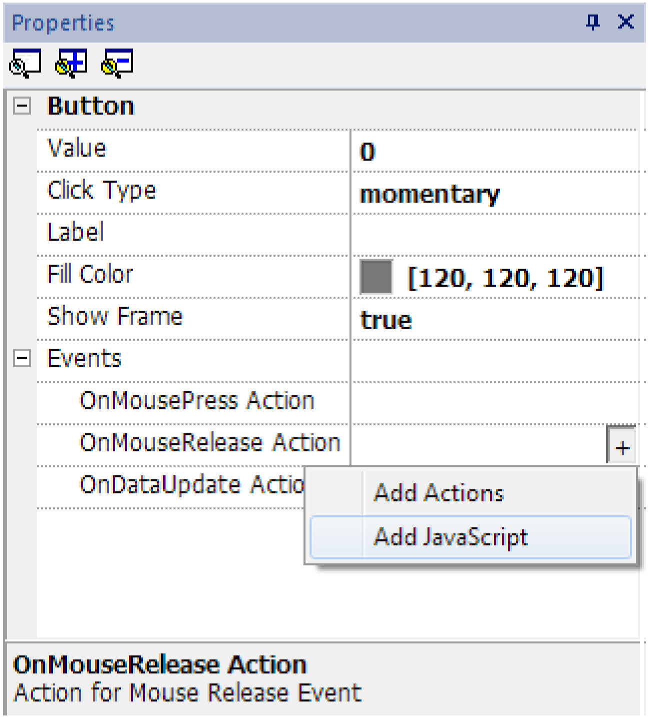
PB610 Panel Builder 600 provides a JavaScript engine running at the client side. Each project page can contain scripts with scope local to the page where they are programmed. The project can also contain global scripts that can be executed by scheduler events or alarm events, but it is important to understand that the scripts are still executed at the client side. In other words, having more than one client connected to the Control Panel (for instance, an external PC running the Windows Client) means each client will run the same script, providing output results that depend on the input. Inputs provided to the different clients may be different.

This can be clarified, for instance, considering a situation in which the script acts based on a slider position, which can be different for the different clients.

Events

You can add JavaScript in the following events:

- Widget Events
- Page Events
- System Events



Button	
Value	0
Click Type	momentary
Label	
Fill Color	[120, 120, 120]
Show Frame	true
Events	
OnMousePress Action	
OnMouseRelease Action	
OnDataUpdate Actio	

OnMouseRelease Action
Action for Mouse Release Event

Widget Events

onMouseClick

void onMouseClick (me, eventInfo)

This event is available only for buttons and it occurs when the button is pressed and released quickly.

Parameters

- me: The object that triggers the event.
- eventInfo: It is reserved for future enhancements.

```
function buttonStd1_onMouseClick(me, eventInfo) {
    //do something…
}
```

onMouseHold

void onMouseHold (me, eventInfo)

This event is available only for buttons and it occurs when the button is pressed and released after n seconds where n=**Hold Time** seconds specify in widget properties.

Parameters

- me: The object that triggers the event.
- eventInfo: It is reserved for future enhancements.

```
function buttonStd1_onMouseHold(me, eventInfo) {
    //do something…
}
```

onMousePress

void onMousePress (me, eventInfo)

This event is available only for buttons and it occurs when the button is pressed.

Parameters:

- me: The object that triggers the event.
- eventInfo: It is reserved for future enhancements.

```
function buttonStd1_onMousePress(me, eventInfo) {
     //do something…
}
```

onMouseRelease

void onMouseRelease (me, eventInfo)

This event is available only for buttons and it occurs when the button is released.

Parameters:

- me: The object that triggers the event.
- eventInfo: It is reserved for future enhancements.

```
function buttonStd1_onMouseRelease(me, eventInfo) {
     //do something…
}
```

onDataUpdate

Boolean onDataUpdate (me, eventInfo)

This occurs when the data attached to the widget changes.

Parameters:

- me: The object that triggers the event.
- eventInfo: An object with these fields (you can refer fields using "." - dot notation):
- oldValue: The old value that is the widget value before the change.
- newValue: The new value that is the value which will be updated to the widget.
- attrName: The attribute on which the event is generated.
- index: An integer attribute index if any, default = 0.
- mode: "W" when user is writing to the widget, "R" otherwise.

This event is triggered by the system before the value is passed to the widget; this means the code programmed here can modify or alter the value before it is actually passed to the widget.

The code can terminate with a "Return TRUE" or "Return FALSE".

After terminating the code with "Return FALSE", the control is returned to the calling widget that may launch other actions.

After terminating the code with TRUE, the control is NOT returned to the widget and this makes sure that no additional actions are executed following the calling event.

```
function buttonStd1_onDataUpdate(me, eventInfo) {
  if ( eventInfo.oldValue < 0) {
       //do something…
  }
  return false;
}
```

Page Events

onActivate

void onActivate (me, eventInfo)

This event occurs each time the page is shown.

Parameters:

- me: The object that triggers the event.
- eventInfo: It is reserved for future enhancements.

This JavaScript will execute when the page is active. It means that, when the page is loaded, the script will execute.

```
function Page1_onActivate(me, eventInfo) {
    //do something…
}
```

onDeactivate

void onDeactivate (me, eventInfo)

This occurs when leaving the page.

Parameters:

- me: The object that triggers the event.
- eventInfo: It is reserved for future enhancements.

```
function Page1_onDeactivate(me, eventInfo) {
    //do something…
}
```

onWheel

void onMouseWheelClock(me, eventInfo)

This occurs when a wheel device is moving (ex. Mouse wheel).

Parameters

- me: The object that triggers the event.
- eventInfo: It is reserved for future enhancements.

```
function Page1_onMouseWheelClock(me, eventInfo) {
    //do something…
}
```

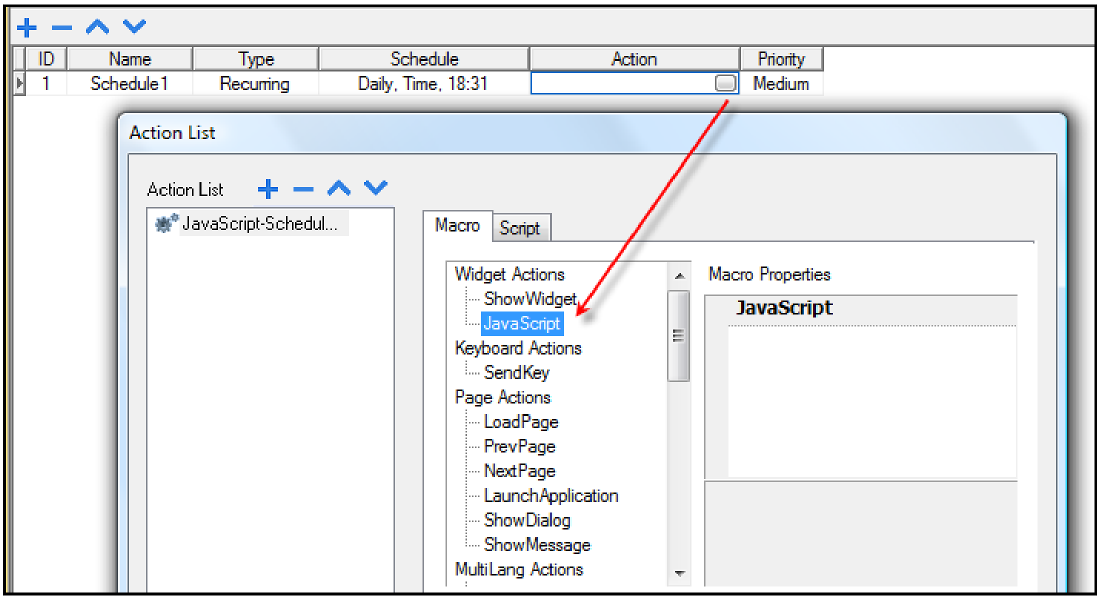
System Events

There are three types of system events:

- related to the scheduler,
- related to the alarms
- Related to wheel device

Scheduler Event

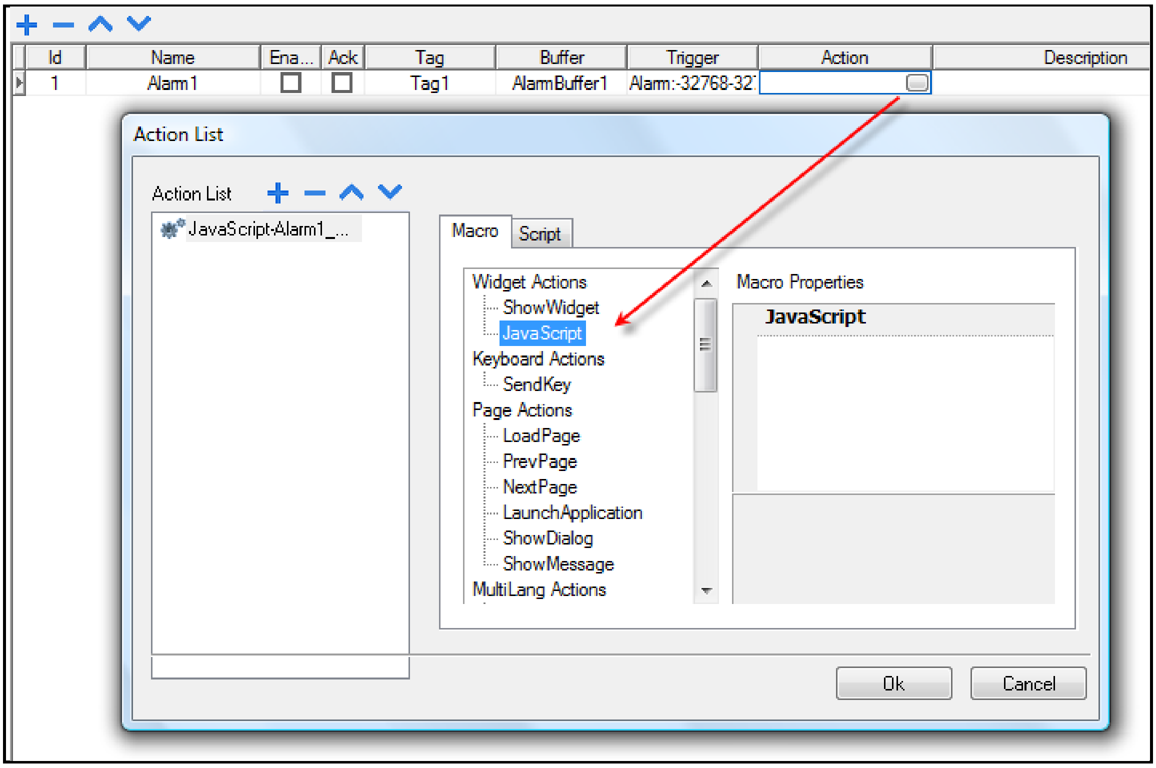
This event occurs when triggered by the proper action available in the scheduler system.



ID	Name	Type	Schedule	Action	Priority
1	Schedule1	Recurring	Daily, Time, 18:31		Medium

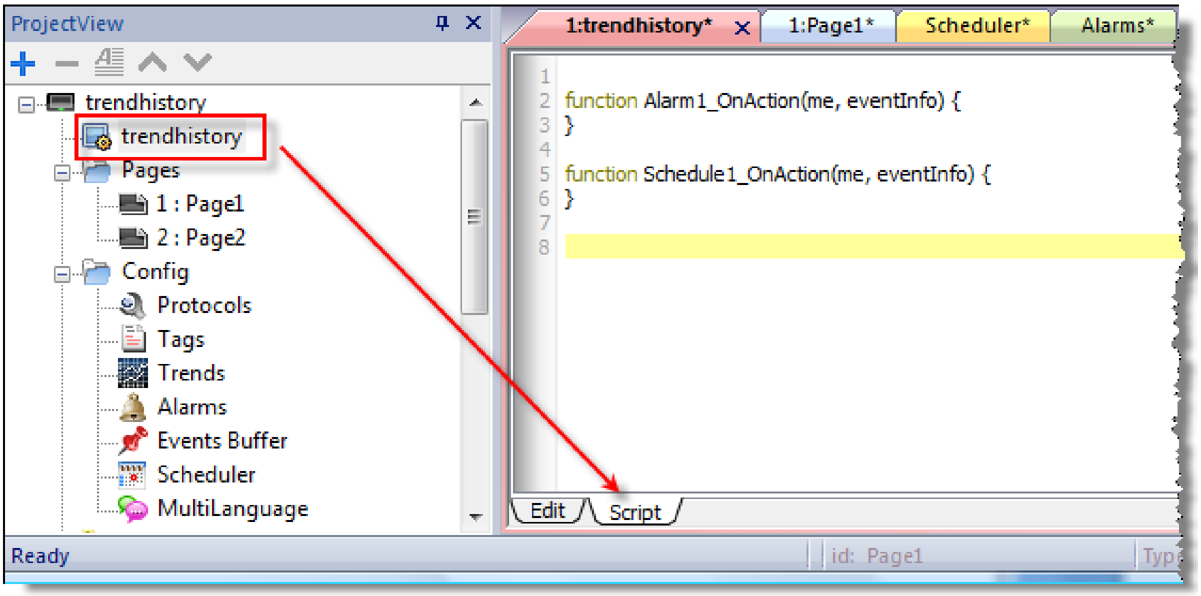
Alarm Event

This event occurs when triggered by a specific alarm condition and programmed in the proper action.



Id	Name	Ena...	Ack	Tag	Buffer	Trigger	Action	Description
1	Alarm1	□	□	Tag1	AlarmBuffer1	Alarm:-32768-32		

Once the system events are configured, the custom code for them can be edited from the global JavaScript editor interface, which is available from the Project View (double click on the project name icon).



```
function Alarm1_OnAction(me, eventInfo) {
}

function Schedule1_OnAction(me, eventInfo) {
}
```

onWheel

void onMouseWheelClock(me, eventInfo)

This occurs when a wheel device is moving (ex. Mouse wheel).

Parameters

- me: The object that triggers the event.
- eventInfo: It is reserved for future enhancements.

```
function Project1_onMouseWheelClock(me, eventInfo) {
    //do something…
```

Objects

PB610 Panel Builder 600 uses JavaScript objects to access the elements of the page. Each object is composed of properties and methods that are used to define the operation and appearance of the page element. The following objects are used to interact with elements of the HMI page.

Object	**Explanation**
Widget	The Widget class is the base class for all elements on the page including the page element
Page	This object references the current HMI page. The page is the top-level object of the screen
Group	A group is a basic logical element that is associated with a set of logical tags. It provides an interface to enable the uniform operation on a set of logically connected tags
Project	This object defines the project widget. The project widget is used to retrieve data about the project such as tags, alarms, recipes, schedules, tags and so on. There is only one widget for the project and it can be referenced through the project variable
State	Class for holding state of a variable acquired from the controlled environment. Beside value itself, it contains the timestamp indicating when the value is collected together with flags marking quality of the value.

Widget

This widget class is the base class for all elements on the page including the page element.

NOTE

Widget is not a specific element but a JavaScript class.

NOTE

Important!
When you change the properties of widgets with JavaScript you have to set the widget “Static Optimization“ to “Dynamic“, otherwise changes to properties will be ignored. You can find the option “Static Optimization“ in the “Advance Properties“.

The following properties are common among all widgets.

objectName

string objectName

It gets the name of the widget. The name is a unique ID for the widget.

```
function btnStd04_onMouseRelease(me) {
    var wgt = page.getWidget("rect1");
    var name = wgt.objectName;
}
```

x

number x

It gets or sets the widget “x” position in pixels.

```
function btnStd1_onMouseRelease(me) {
    var wgt = page.getWidget("rect1");
    wgt.x = 10;
}
```

y

number y

It gets or sets the widget “y” position in pixels.

```
function btnStd1_onMouseRelease(me) {
    var wgt = page.getWidget("rect1");
    wgt.y = 10;
}
```

width

number width

It gets or sets the widget width in pixels.

```
function btnStd1_onMouseRelease(me) {
    var wgt = page.getWidget("rect1");
    wgt.width = 10;
}
```

height

number height

It gets or sets the widget height in pixels.

```
function btnStd1_onMouseRelease(me) {
    var wgt = page.getWidget("rect1");
    wgt.height = 10;
}
```

visible

boolean visible

It gets or sets the widget visible state.

```
function btnStd4_onMouseRelease(me) {
    var wgt = page.getWidget("rect1");
    wgt.visible = false;
}
function btnStd5_onMouseRelease(me) {
    var wgt = page.getWidget("rect1");
    wgt.visible = true;
}
```

value

number value

It gets or sets the widget value.

```
function btnStd6_onMouseRelease(me) {
    var wgt = page.getWidget("field1");
    wgt.value = 100;
}
```

opacity

number opacity (range from 0 to 1)

It gets or sets the widget opacity. Values are decimals from 0 to 1, where 1 is 100% opaque.

```
function btnStd8_onMouseRelease(me) {
    var wgt = page.getWidget("rect1");
    wgt.opacity = 0.5;
}
```

rotation

number rotation (in degrees)

It gets or sets the rotation angle for the widget. The rotation is done by degree and makes a clockwise rotation, starting at the East position.

```
function btnStd9_onMouseRelease(me) {
    var wgt = page.getWidget("rect1");
    wgt.rotation = 45;
}
```

userValue

string userValue

It gets or sets a user-defined value for the widget. This field can be used by JavaScript functions to store additional data with the widget.

```
function btnStd9_onMouseRelease(me) {
    var wgt = page.getWidget("rect1");
    wgt.userValue = "Here I can store custom data";
}
```

Every widget has some specific properties that you can access using dot notation. For an up-to-date and detailed list of properties you can use the Qt Script Debugger inspecting the widget methods and properties.

The following methods are common among all widgets:

getProperty

object getProperty(propertyName, [index])

Returns a property.

Parameters:

- propertyName: A string containing the name of property to get.
- index: The index of the element to get from the array. Default is “0”.

Almost all properties that are shown in the PB610 Panel Builder 600 Property View can be retrieved from the getProperty method. The index value is optional and only used for widgets that support arrays.

```
function buttonStd1_onMouseRelease(me, eventInfo) {
    var shape = page.getWidget("rect2");
    var y_position = shape.getProperty("y");
}
```

```
function buttonStd2_onMouseRelease(me, eventInfo) {
    var image = page.getWidget("multistate1");
    var image3 = image.getProperty("imageList", 2);
    //…
}
```

setProperty

boolean setProperty(propertyName, value, [index])

Sets a property for the widget

Parameters:

- propertyName: A string containing the name of property to set.
- value: A string containing the value to set the property.
- index: The index of the element to set in the array. Default is “0”.

Almost all properties that are shown in the PB610 Panel Builder 600 Property View can be set by this method. The index value is optional and only used for widgets that support arrays (for example a MultiState image widget). The setProperty method returns a boolean value TRUE or FALSE to indicate if the property was set or not.

```
function buttonStd1_onMouseRelease(me, eventInfo) {
    var setting_result = shape.setProperty("y", 128);
    if (setting_result)
            alert("Shape returned to start position");
}
```

```
function buttonStd2_onMouseRelease(me, eventInfo) {
    var image = page.getWidget("multistate1");
    var result = image.setProperty("imageList", "Fract004.png", 2);
    //…
}
```

Page

This object references the current HMI page. The page is the top-level object of the screen.

Follow the list of Page Object properties

backgroundColor

string backgroundColor (in format rgb(xxx, xxx, xxx) where xxx range from 0 to 255)

The page background color

```
function btnStd11_onMouseRelease(me) {
    page.backgroundColor = "rgb(128,0,0)";
}
```

width

number width

The page width in pixels.

```
function btnStd05_onMouseRelease(me) {
    var middle_x = page.width / 2;
}
```

height

number height

The page height in pixels.

```
function btnStd05_onMouseRelease(me) {
    var middle_y = page.height / 2;
}
```

userValue

string userValue

It gets or sets a user-defined value for the widget. This field can be used by JavaScript functions to store additional data with the page.

```
function btnStd9_onMouseRelease(me) {
    page.userValue = "Here I can store custom data";
}
```

Follow the list of Page Object methods

getWidget

object getWidget(wgtName)

It returns the widget with the given name.

Parameters:

- wgtName: A string containing the name of widget.

Return value

An object representing the widget. If the widget does not exist, null is returned.

```
function btnStd1_onMouseRelease(me) {
    var my_button = page.getWidget("btnStd1");
}
```

setTimeout

number setTimeout(functionName, delay)

It starts a timer to execute a given function after a given delay once.

Parameters:

- functionName: A string containing the name of function to call.
- delay: The delay in milliseconds.

Return value

It returns a number corresponding to the timerID.

```
var duration = 3000;
var myTimer = page.setTimeout("innerChangeWidth()", duration);
```

clearTimeout

void clearTimeout(timerID)

It stops and clear the timeout timer with the given timer.

Parameters:

- timerID: The timer to be cleared and stopped.

```
var duration = 3000;
var myTimer = page.setTimeout("innerChangeWidth()", duration);
// do something
page.clearTimeout(myTimer);
```

setInterval

number setInterval(functionName, interval)

It starts a timer that executes the given function at the given interval.

Parameters:

- functionName: A string containing the name of function to call.
- interval: The interval in milliseconds.

Return value

It returns a number corresponding to the timerID.

```
var interval = 3000;
var myTimer = page.setInterval("innerChangeWidth()", interval);
```

clearInterval

void clearInterval(timerID)

It stops and clears the interval timer with the given timer.

Parameters:

- timerID: The timer to be cleared and stopped.

```
var interval = 3000;
var myTimer = page.setInterval("innerChangeWidth()", interval);
// do something
page.clearInterval(myTimer);
```

clearAllTimeouts

void clearAllTimeouts()

It clears all the timers started.

```
Page.clearAllTimeouts();
```

Group

A group is a basic logical element that is associated with a set of logical tags. It provides an interface to enable the uniform operation on a set of logically connected tags.

Follow the list of methods supported by Group Object

getTag

object getTag(TagName)

Gets the tag specified by TagName from the group object.

Parameters:

- TagName: A string representing the tag name.

Return value

An object that is the value of the tag or if tag value is an array it returns the complete array. If you need to retrieve an element of the array, check the method getTag available in object Project. undefined is returned if tag is invalid.

```
var group = new Group();
project.getGroup("GroupName", group);
var value = group.getTag("Tag1");
```

getCount

number getCount()

Returns total number of tags in this group.

Return value

The number of tags.

```
var group = new Group();
project.getGroup("GroupName", group);
var value = group.getCount();
```

getTags

object getTags()

Returns the list of all tags in group.

Return value

An array of all tags in the group.

```
var group = new Group();
project.getGroup("enginesettings", group);
var tagList = group.getTags();
for(var i = 0; i < tagList.length; i++){
      var tagName = tagList[i];
      //do something…
}
```

Project

This object defines the project widget. The project widget is used to retrieve data about the project such as tags, alarms, recipes, schedules, tags and so on. There is only one widget for the project and it can be referenced through the project variable.

Follow the list of properties of Project Object

startPage

string startPage

The page shown when the application is started

```
var startPage = project.startPage;
project.startPage = "Page2.jmx";
```

Follow the list of methods of Project Object

nextPage

void nextPage()

The script executes the next page macro.

```
project.nextPage();
```

prevPage

void prevPage()

The script executes the previous page macro.

```
project.prevPage();
```

homepage

void homePage()

The script executes the Home page macro.

```
project.homePage();
```

loadPage

void loadPage(pageName)

The script executes to load the set page defined in the script.

```
project.loadPage();
```

showDialog

void showDialog(pageName)

The script executes to show the dialog page.

```
project.showDialog();
```

closeDialog

void closeDialog()

The script executes to close the currently-opened dialog page.

```
project.closeDialog();
```

showMessage

void showMessage(message)

The script executes to display the message popup.

```
project.showMessage("Hi This is test message");
```

getGroup

number getGroup(groupName, groupInstance, [callback])

Fast read method which gets the values of all tags in a group.

Parameters:

- groupName: A string containing the name of the group.
- groupInstance: The group element to be filled.
- callback: A string containing the name of the function to be called when the group is ready.

Return value

A number value that is the status: 1 for success, 0 for fail.

```
var group = new Group();
var status = project.getGroup ("enginesettings", group);
if (status == 1) {
    var value = group.getTag("Tag1");
    if (value!=undefined) {
        // do something with the value
    }
}
```

```
var g = new Group();
var status = project.getGroup ("enginesettings", g, "fnGroupReady");
function fnGroupReady(groupName, group) {
    var val = group.getTag("Tag1");
    if (val!=undefined) {
        // do something with the value
    }
}
```

getTag

object getTag(tagName, state, index)

void getTag(tagName, state, index, callback)

It returns the tag value or the complete array if index value is “-1” of the given tagName.

Parameters:

- tagName: A string of the tag name.
- state: The state element to be filled.
- index: An index if the tag is array type. “-1” returns the complete array. Default is “0”.
- callback: Function name if an asynchronous read is required. Default = “”.

Return value

Tags value is returned. If tag is array type and index = -1 then the complete array is returned.

Remarks

For non array tags provide index as 0.

```
var state = new State();
var value = project.getTag("Tag1", state, 0);
//
//for non array type
//tags index is not considered, so can be left as 0
//
if (value!=undefined) {
    //...do something with s
}
```

```
var state = new State();
var value = project.getTag("Tag1", state, -1, "fnTagReady");

function fnTagReady(tagName, tagState) {
    if (tagName=="Tag1") {
            var myValue = tagState.getValue();
    }
}
```

setTag

number setTag(tagName, tagValue, [index], [forceWrite])

Sets the given tag in the project. Name and value are in a string.

Parameters:

- tagName: A string of the tag name.
- tagValue: An object containing the value to write.
- index: An index if tag is array type. Set “-1” to pass complete array. Default is “0”.
- forceWrite: A boolean value for enabling force write of tags, the function will wait for the value to be written before it returns back. Default is false.

Return value

Integer value for denoting success and failure of action when forceWrite is true. A “0” means success and “-1” means failure. If forceWrite is false, returned value will be undefined.

```
var val = [1,2,3,4,5];
var status = project.setTag("Tag1", val, -1, true);
if (status == 0) {
    // Success
} else {
    // Failure
}
```

```
var val = "value";
project.setTag("Tag1", val);
```

getRecipeItem

object getRecipeItem (recipeName, recipeSet, recipeElement)

Gets the value of the given recipe set element.

Parameters:

- recipeName: A string representing the recipe name.
- recipeSet: A string representing the recipe set can be either the recipe set name or “0” based set index.

recipeElement: A string representing the recipe element can be either the element name or “0” based element index.

Return value

An object with the value of the recipe undefined is returned if invalid. If of type array, an array object type is returned.

```
var value = project.getRecipeItem("recipeName", "Set", "Element");
```

setRecipeItem

number setRecipeItem (recipeName, recipeSet, recipeElement, value)

Gets the value of the given recipe set element.

Parameters:

- recipeName: A string representing the recipe name.
- recipeSet: A string representing the recipe set, can be either the recipe set name or “0” based set index.
- recipeElement: A string representing the recipe Element, can be either the element name or “0” based element index.
- value: An object containing the value to store in the recipe. It can be an array type too.

Return value

Integer value for denoting success and failure of action. A “0” means success and “-1” means failure.

```
var val = [2,3,4];
project.setRecipeItem("recipeName", "Set", "Element", val);
if (status == 0) {
    // Success
} else {
    // Failure
}
```

downloadRecipe

void downloadRecipe (recipeName, recipeSet)

Downloads the recipe set to corresponding tag.

Parameters:

- recipeName: A string representing the recipe name.
- recipeSet: A string representing the recipe set can be either the recipe set name or “0” based set index.

```
project.downloadRecipe("recipeName", "Set");
```

uploadRecipe

void uploadRecipe (recipeName, recipeSet)

Uploads the value of tags into the provided recipe set.

Parameters:

- recipeName: A string representing the recipe name.
- recipeSet: A string representing the recipe set, can be either the recipe set name or “0” based set index.

```
project.uploadRecipe("recipeName", "Set");
```

launchApp

void launchApp (appName, appPath, arguments, singleInstance)

Execute an external application.

Parameters:

- appName: A string contains the application name
- appPath: A string contains the application path, it must be an absolute path.
- Arguments: A string contains the arguments to send to application executed.
- singleInstance: true=single instance allowed, false allow multiple instance

```
project.launchApp("PDF.exe","\\Flash\\QTHMI\\PDF","\\USBMemory\\file.pdf   ","true");
```

printGfxReport

void printGfxReport(reportName, silentMode)

Prints the graphic report specified by reportName.

Parameters:

- reportName: A string containing the report name
- silentMode: true = silent mode (avoids to show printer settings dialog)

```
project.printGfxReport("Report Graphics 1", true);
```

printText

void printText(text, silentMode)

Print a fixed text.

Parameters:

- text: A string to print
- silentMode: true = silent mode (avoids to show printer settings dialog)

```
project.printText("Hello I Am Text Printing",true);
```

emptyPrintQueue

void emptyPrintQueue()

Empties the print queue. Current job will not be aborted.

```
project.emptyPrintQueue();
```

pausePrinting

void pausePrinting();

Suspends printing operations. Will not suspend the print of a page already sent to the printer.

```
project.pausePrinting();
```

resumePrinting

void resumePrinting();

Resumes previously suspended printing.

```
project.resumePrinting();
```

abortPrinting

void abortPrinting();

Aborts current print operation and proceed with the next one in queue. This command will not abort the print of a page already sent to the printer.

```
project.abortPrinting();
```

printStatus

project.printStatus;

Returns a string representing current printing status:

- error: an error occurred during printing
- printing: ongoing printing
- idle: system is ready to accept new jobs
- paused: printing has be suspended

```
var status = project.printStatus;
project.setTag("PrintStatus",status);
```

printGfxJobQueueSize

project.printGfxJobQueueSize;

Returns the number of graphic reports in queue for printing.

```
var gfxqueuesize = project.printGfxJobQueueSize;
project.setTag("printGfxJobQueueSize",gfxqueuesize);
```

printTextJobQueueSize

project.printTextJobQueueSize;

Returns the number of text reports in queue for printing.

```
var textjobqueuesize = project.printTextJobQueueSize;
project.setTag("printTextJobQueueSize",textjobqueuesize);
```

printCurrentJob

project.printCurrentJob;

Returns a string representing current job being printed

```
var currentjob = project.printCurrentJob;
project.setTag("printCurrentJob",currentjob);
```

printActualRAMUsage

project.printActualRAMUsage;

Returns an estimate of RAM usage for printing queues

```
var myVar = project.printActualRAMUsage;
alert(" actual ram usage is   "+ myVar);
```

printRAMQuota

project.printRAMQuota;

Returns the maximum allowed RAM usage for printing queues

```
var ramquota = project.printRAMQuota;
project.setTag("printRAMQuota",ramquota);
```

printActualDiskUsage

project.printActualDiskUsage;

Returns the spool folder disk usage (for PDF printouts)

```
var myVar1 = project.printActualDiskUsage;
alert(" actual disk usage is   "+ myVar1);
```

printDiskQuota

project.printDiskQuota;

Returns the maximum allowed size of spool folder (for PDF printouts)

```
var diskquota = project.printDiskQuota;
project.setTag("printDiskQuota",diskquota);
```

printSpoolFolder

project.printSpoolFolder;

Returns current spool folder path (for PDF printouts)

```
var spoolfolder = project.printSpoolFolder;
project.setTag("printSpoolFolder",spoolfolder);
```

printPercentage

project.printPercentage;

Returns current job completion percentage (meaningful only for multipage graphic reports)

```
var percentage = project.printPercentage;
project.setTag("printPercentage",percentage);
```

State

Class for holding state of a variable acquired from the controlled environment. Beside value itself, it contains the timestamp indicating when the value is collected together with flags marking quality of the value.

Follow the list of methods for State Object

getQualityBits

number getQualityBits()

Returns an integer - a combination of bits indicating tag value quality.

Return value

A number containing the quality bits.

```
var state = new State();
var value = project.getTag("Tag1", state, 0);
var qbits = state.getQualityBits();
```

getTimestamp

number getTimestamp()

Returns time the value was sampled.

Return value

A number containing the timestamp (for example 1315570524492).

Remarks

Date is a native JavaScript data type.

```
var state = new State();
var value = project.getTag("Tag1", state, 0);
var ts = state.getTimestamp();
```

isQualityGood

boolean isQualityGood()

It returns whether value contained within this state object is reliable.

Return value

boolean true if quality is good, false otherwise.

```
var state = new State();
var value = project.getTag("Tag1", state, 0);
if (state.isQualityGood()) {
    // do something…
}
```

Keywords

Global objects are predefined and always available objects that can be referenced by the names listed below.

page

object page

It references the page object for the current page.

```
function btnStd04_onMouseRelease(me) {
    var wgt = page.getWidget("rect1");
    var name = wgt.objectName;
}
```

project

object project

It references the project widget.

```
var group = new Group();
project.getGroup("GroupName", group);
var value = group.getCount("Tag1");
```

Global Functions

print

void print(message)

It prints a message to the HMI logger window.

Parameters:

- message: A string containing the message to display.

```
print("Test message");
```

alert

void alert(message)

It displays a popup dialog with the given message. The user must press the **OK** button in the dialog to continue with the execution of the script.

Parameters:

- message: A string containing the message to display.

NOTE The alert function is often used for debugging JavaScript routines.

```
alert("Test message");
```

Limitations

Widgets cannot be instantiated from JavaScript. The Widgets can only be accessed and changed. If you need additional Widgets on the page, you can add hidden Widgets on the page, and show or position them from JavaScript.

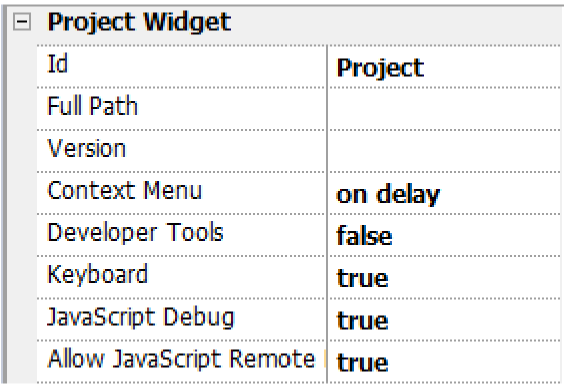
Debugging of JavaScript

Panel Builder 600 and Runtime include a JavaScript debugger to allow user to debug problems.

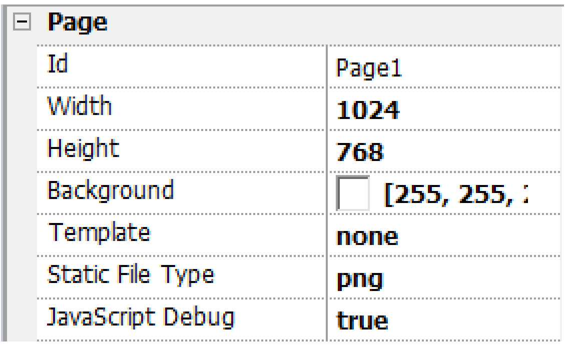
There are two types of debuggers:

- **Runtime debugger**: a debugger running directly in target device (HMI panel)
- **Remote debugger**: a debugger running on a remote PC connected to target device via Ethernet (usually PC with Panel Builder 600)

To enable the debugging mode, in the “Advanced Properties” of a page, set *JavaScript Debug* to **true**.



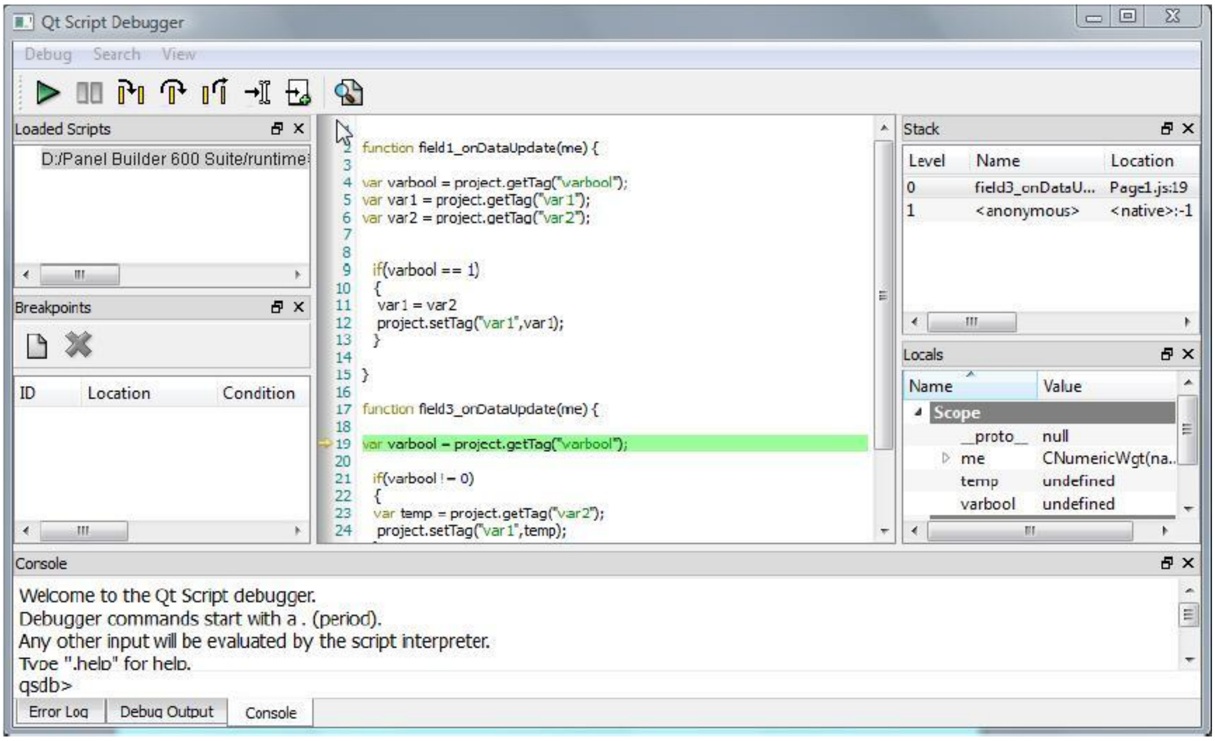
⊟ Project Widget	
Id	Project
Full Path	
Version	
Context Menu	on delay
Developer Tools	false
Keyboard	true
JavaScript Debug	true
Allow JavaScript Remote	true



⊟ Page	
Id	Page1
Width	1024
Height	768
Background	□ [255, 255,
Template	none
Static File Type	png
JavaScript Debug	true

For schedulers and alarms debugging, enable JavaScript Debug in Project properties.

In runtime, when the events are called, the script debugger will show the debug information. In the box **Locals** you can inspect all available variables and elements.



```
function field1_onDataUpdate(me) {

var varbool = project.getTag("varbool");
var var1 = project.getTag("var1");
var var2 = project.getTag("var2");


  if(varbool == 1)
  {
   var1 = var2
   project.setTag("var1",var1);
  }

}

function field3_onDataUpdate(me) {

var varbool = project.getTag("varbool");

  if(varbool != 0)
  {
  var temp = project.getTag("var2");
   project.setTag("var1",temp);
```

Level	Name	Location
0	field3_onDataU...	Page1.js:19
1	<anonymous>	<native>:-1

Name	Value
Scope	
__proto__	null
me	CNumericWgt(na..
temp	undefined
varbool	undefined

```
Welcome to the Qt Script debugger.
Debugger commands start with a . (period).
Any other input will be evaluated by the script interpreter.
Type ".help" for help.
qsdb>
```

For a complete reference guide about Qt Script Debugger you can open the following link in your browser: http://qt-project.org/doc4.8/qtscriptdebugger-manual.html.

NOTE

For UN20 target (WCE MIPS hmi panels), local debugger has been disabled. However, remote debugger is available to debug JS from a PC connected to HMI panel via Ethernet.

NOTE

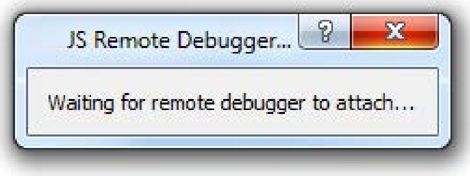
Remote debugger not supported in Windows Client and ActiveX.

Remote JavaScript Debugger

Remote JS debugger can be opened directly from Panel Builder 600 **Run** -> **Start JS Remote Debugger** or from related icon in toolbar.

To start remote debugging, proceed as follow:

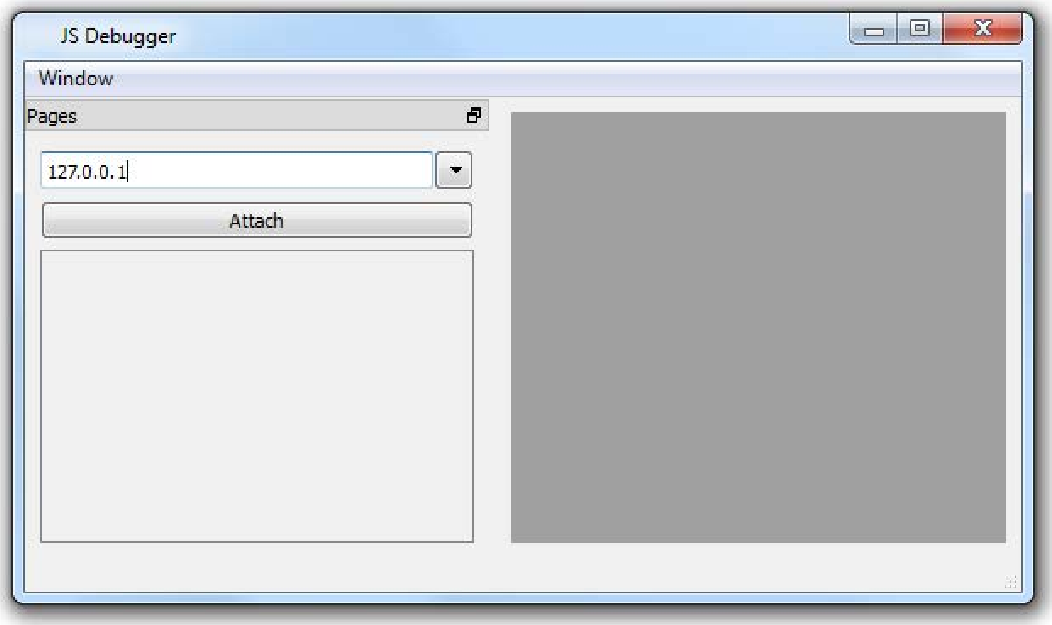
1. Download project with **Allow JavaScript Remote** enabled in project properties and **JavaScript Debug** enable in all pages where debugging is required.
2. Once started, runtime shows waiting for remote debugger as shown below:



JS Remote Debugger...

Waiting for remote debugger to attach...

3. In JS Debugger window, select IP of the target and click **Attach** to connect debugger to the target.



127.0.0.1

NOTE

Remote JS debugger requires port 5100/TCP in the runtime side.

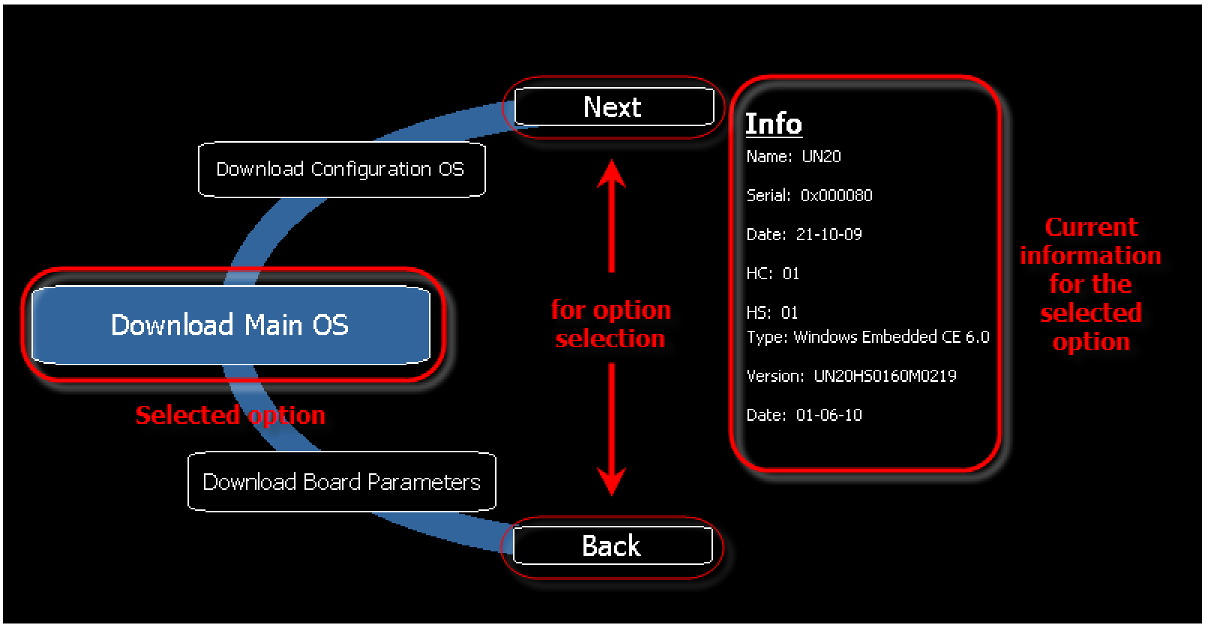
NOTE

For UN20 target (WCE MIPS hmi panels), local debugger has been disabled. However, remote debugger is available to debug JS from a PC connected to HMI panel via Ethernet.

NOTE

Remote debugger not supported in Windows Client and ActiveX.

System Settings Tool



Info

Name: UN20

Serial: 0x000080

Date: 21-10-09

HC: 01

HS: 01

Type: Windows Embedded CE 6.0

Version: UN20HS0160M0219

Date: 01-06-10

Selected option

for option selection

Current information for the selected option

System settings tool

The system settings tool is a rotating menu.

→ Use the navigation buttons **Next** and **Back** to scroll between the available options.

- On the left side the selected component and function are highlighted.
- On the right side, on the “Info” pane, the information about the selected option is shown. For example the version of the Main OS component.

The system settings tool has two operating modes:

- User Mode and
- System Mode.

The difference between them is the number of available options.

User Mode

User Mode is a simple interface where users can get access to the basic settings of the HMI panel: The System Settings tool is accessible at Runtime from the **context menu** by selecting the item **Show system settings**.

When activated in this way, the System Settings tool always starts in **User Mode**.

The context menu can be activated by pressing and holding down a screen area without buttons or other touch sensitive elements, until the menu is displayed.

Main Items available in User Mode are:

Item	**Explanation**
Calibrate Touch	To calibrate the touch screen if needed
Display settings	Backlight and Brightness control
Time	Internal RTC settings

BSP Settings	Operating system version, Unit operating timers: power up and activated backlight timers, Buzzer control, Battery LED control
Network	IP address settings
Plug-in List	Provides a list of the plug-in modules installed and recognized by the system; this option may not be supported by all platforms and all versions.

System Mode

System Mode is the interface of the System Settings tool with all the options enabled.

The HMI products support a special procedure for accessing the System Settings tool; the special procedure is required to start the System Settings in **System Mode**, or when the standard access procedure is not accessible for some reason.

When activated by this special procedure, the System Settings tool always starts in System Mode.

The special access to the System Settings tool can be activated with a **tap-tap sequence** over the touch screen during the power-up phase. Tap-tap consists of a high frequency sequence of touch activations, done by the simple means of finger tapping the touch screen, performed during the power up and started immediately after the device has been powered.

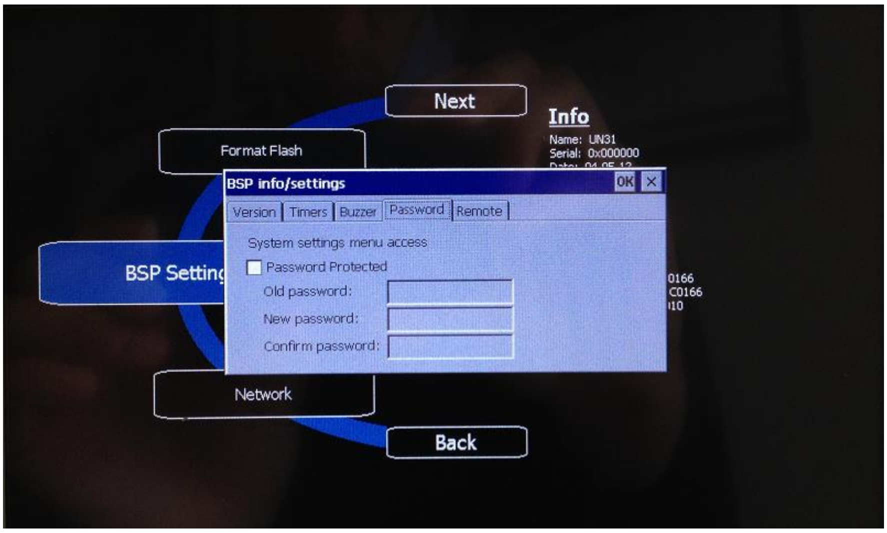
In addition to the options available in User Mode, the following important features are available:

Feature	**Explanation**
Format Flash	To format the internal panel flash disk. All projects and the runtime will be erased, returning the panel to a factory new condition
Restore Factory Settings	Restore factory settings is used as an alternative option to Format Flash (that´s a slow operation) to restore device factory settings. Options available are: Uninstall HMI: removes the HMI runtime from the unit (if present); at the next start the panel will behave as a brand new unit. This command does not reset settings like IP, brightness or RTC. Clear System Settings: allows you to reset the system parameters (registry settings) Files deleted are: \\Flash\\Documents and Settings\\system.hv \\Flash\\Documents and Settings\\default\\user.hv \\Flash\\Documents and Settings\\default.mky \\Flash\\Documents and Settings\\default.vol Also System Mode password is reset. Clear internal Ctrl App: clear current folders used by CODESYS V2.3 and CODESYS V3 internal controllers for applications • \Flash\QtHmi\RTS\APP\*.* • \Flash\QtHmi\RTS\VISU\*.* • \Flash\QtHmi\codesys\* • \Flash\$SysData$\codesys* Clear sysdata settings: clear \Flash\$SysData$ folder (used by technique support only for problems related to display settings) Note: Not all targets and BSPs contain all these options.
Resize Image Area	Resizes the Flash portion reserved to store the splash screen image that is displayed at power up. Default settings are normally ok for all units.
Download Configuration OS	checks the actual version and upgrades the back-up operating system (see relevant chapter, for additional details)
Download Main OS	checks actual version and upgrades the main operating system (see relevant chapter for additional details)

Feature	Explanation
Download Splash Image	Loads a new file for the splash screen image displayed by the unit at power up; the image must be supplied in a specific format. We suggest that you update the splash screen image directly from the Panel Builder 600 programming software.
Download Bootloader	Checks the actual version of the system boot loader and upgrades the system boot loader.
Download Main FPGA	Checks the actual version and upgrades the main FPGA file; this command may not be available in all platforms and versions.
Download Safe FPGA	Checks the actual version and upgrades the back-up (safe) copy of the FPGA file; this command may not be available in all platforms and versions.
Download System Supervisor	Checks the actual version and upgrades the system supervisor firmware (used for the RTC and power supply handling).

CAUTION! Important: Operation with the System Settings Tool is critical and, when not performed correctly, may cause product damages requiring service of the product. Ask Technical Support for further details.

When executed in “System Mode” the System settings also provides the “BSP Settings”. Only when recalled from System Mode, the BSP settings shows an additional tab called “**Password**” as shown in the figure below.



Name: UN31
Serial: 0x000000

BSP info/settings

System settings menu access

□ Password Protected

Old password:
New password:
Confirm password:

This function allows you to protect with password the access to the System settings in “System Mode” so all the advanced and critical functions are not easily accessible to anyone.

To activate the protection, simply mark the check box “Password Protected” and specify the desired password as shown in the following figure.

The password must be at least 5 characters long.

If you are changing a password previously inserted or disabling the protection, you are asked to provide the old password first.

NOTE Please keep a note of the configured password in a safe place. There is no way to reset the password protection. In case it is lost the unit must be returned to the factory for proper reconditioning.

When the System settings menu is protected by a password, for each critical function that may compromise the proper system operation you try to execute, the HMI will prompt you to enter the password. If correct, the operation will proceed, if wrong, the operation will be aborted.

Updating System Components in HMI Panels

Most of the system software components can be easily upgraded by the end users; this ensures a high degree of flexibility in providing updates and fixes to existing and running systems.

This upgrade can be done using USB flash drives, loaded with the new software modules, and by running the procedure, described in detail in this chapter.

Each unit comes from the manufacturing with a "product code" label, which includes all the information related to the factory settings (in terms of hardware, software and firmware components).

Product labeling is the first reference for checking the factory settings and version of the components installed at time of manufacturing.

The update tool on the HMI panel also provides the user with detailed information on the components actually running in the system.

NOTE

Files required for upgrades depend on the product code. Using the wrong files for upgrade may result in system malfunctions, and may even render the system unusable.

NOTE

Files for upgrades are distributed on demand as a technical support activity.

WARNING!

The downgrade of components is a very dangerous operation that could block machine and make it not more usable for HW/FW compatibility problems. Downgrade operations are not allowed and reserved to tech. support only.

List of Upgradable Components

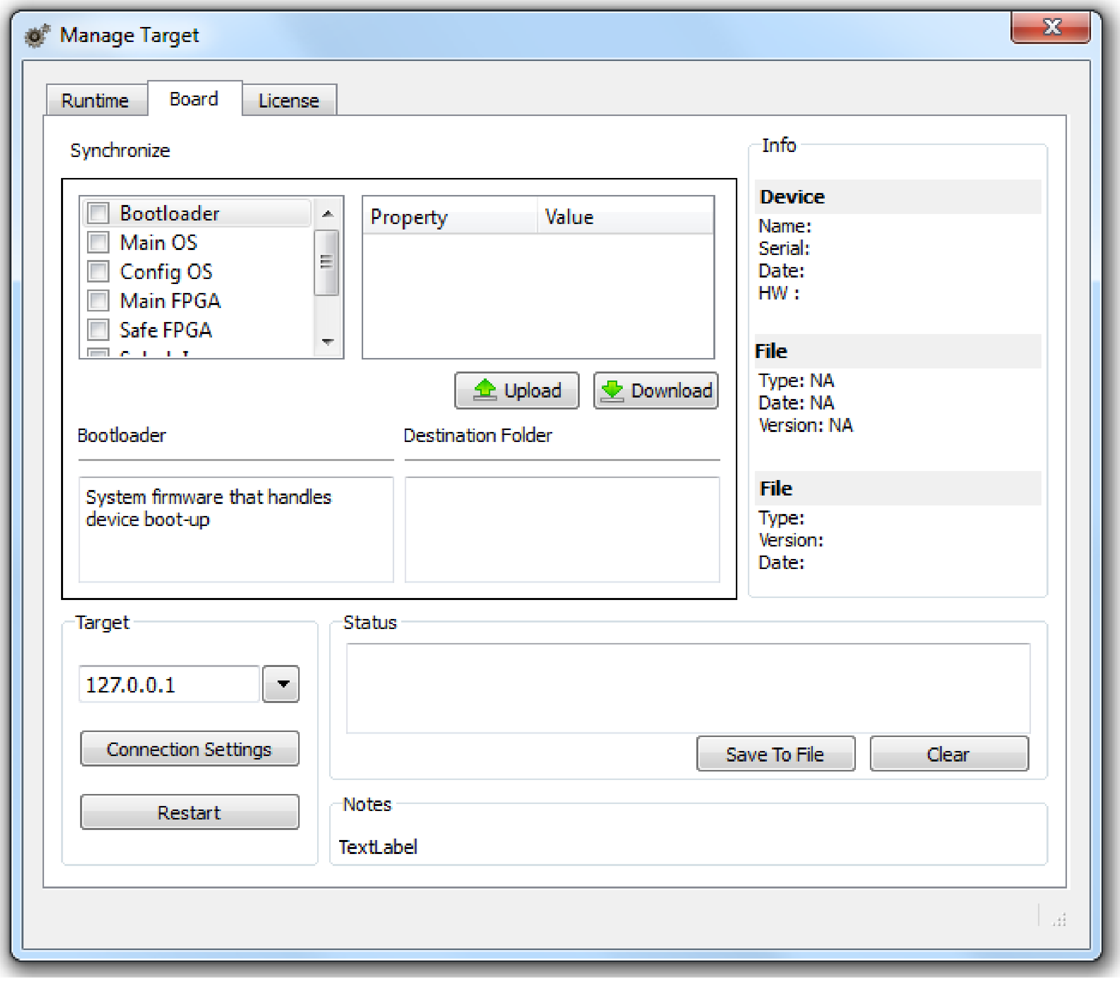
The HMI panels support the upgrade of the following components:

Component	**Explanation**
System Supervisor	Firmware of the system supervisor controller (sample file name: packaged_GekkoZigBee_v4.13.bin) Important: The System Supervisor Component can be upgraded only if the actual version on the panel is V4.13 or above. Version V4.08, V4.09, V4.10 and V4.11 MUST NOT be updated, they do not support automatic update from System Settings
Main FPGA	FPGA firmware (sample file name: h146xaf02r06.bin)
Safe FPGA	back-up copy of the Main FPGA that ensures unit booting in case of main FPGA corruption (may be after failed update) (Sample file name: h146xaf02r06.bin) Note: When updating FPGA firmware on the panel, the same file must be used for Main and Safe FPGA components
Bootloader	Loader to handle panel start-up (sample file name: redboot_UN20HS010025.bin)
Main OS	Main Operating System (sample file name: mainos_UN20HS0160M0237.bin)
Configuration OS	Back-up operating system that ensures units are recovering in case of main operating system corruption (may be after a failed update) (sample file name: configos_UN20HS0160C0237.bin)

Update of System Components from Panel Builder 600

Panel Builder 600 provides a dialog to update system components by downloading them to the target device using the Ethernet communication interface.

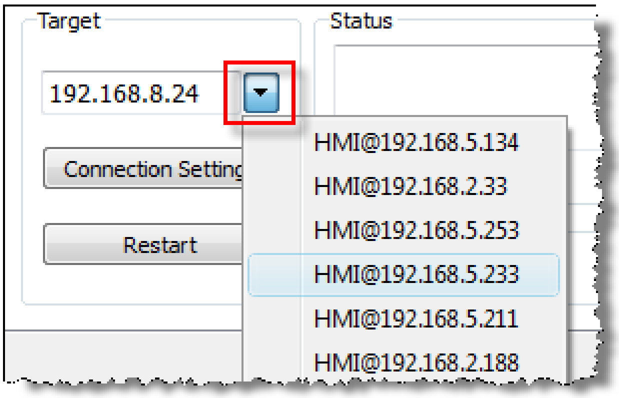
The dialog is available in Run -> Manage Target -> Board.



System firmware that handles device boot-up

127.0.0.1

The first step is to use the Target discovery function to locate the panel IP from the local network. Click on the little arrow symbol and identify the HMI panel from the list of units recognized in the network. In case the panel is not listed, you can try a second time or type the IP directly in the box. Then click out of the box to accept the inserted IP.



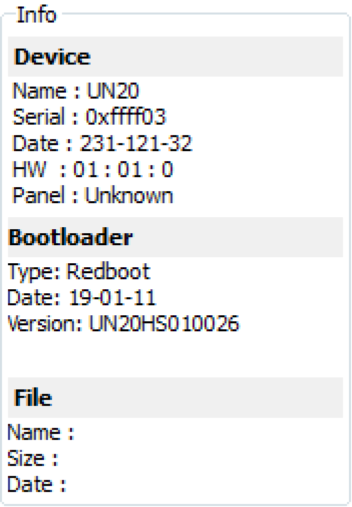
192.168.8.24

HMI@192.168.5.134
HMI@192.168.2.33
HMI@192.168.5.253
HMI@192.168.5.233
HMI@192.168.5.211
HMI@192.168.2.188

NOTE

Discovery service is a broadcast service. When a remote connection is done via VPN or from external networks discovery is not working, so, type directly IP address of target to connect to it.

When the device is recognized the Info box shows the target details as shown as an example in the figure below.



Info

Device

Name : UN20
Serial : 0xffff03
Date : 231-121-32
HW : 01 : 01 : 0
Panel : Unknown

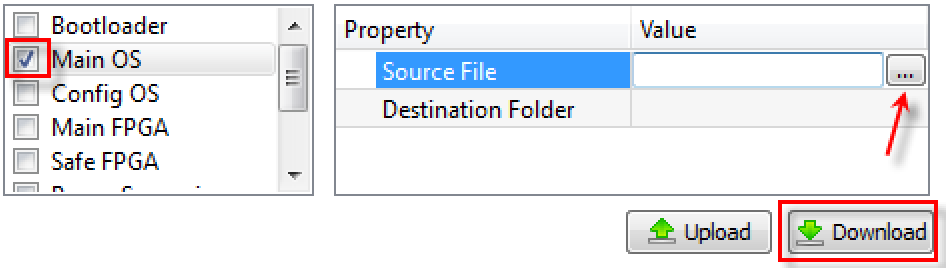
Bootloader

Type: Redboot
Date: 19-01-11
Version: UN20HS010026

File

Name :
Size :
Date :

In the component list locate the one you need to update, check the box and browse for the file from the "Source file" box as shown in the figure below.



□ Bootloader
☑ Main OS
□ Config OS
□ Main FPGA
□ Safe FPGA

Property	Value
Source File	
Destination Folder	

Then click download and check the progress from the Status box below.

NOTE In the component selection you can mark more than one check box and provide the related file to be downloaded. The system will then execute the transfer of the all the elements, one after the other, and at the end you will need to cycle the power of the system.

Manage target also allows you to replace the default splash screen image shown by the devices during the power up phase. Image for the splash screen must be provided in bitmap format saved in RGB 565 format.

NOTE Splash screen images must NOT be bigger than 500 KB and they must have a black background to ensure the best optical results.

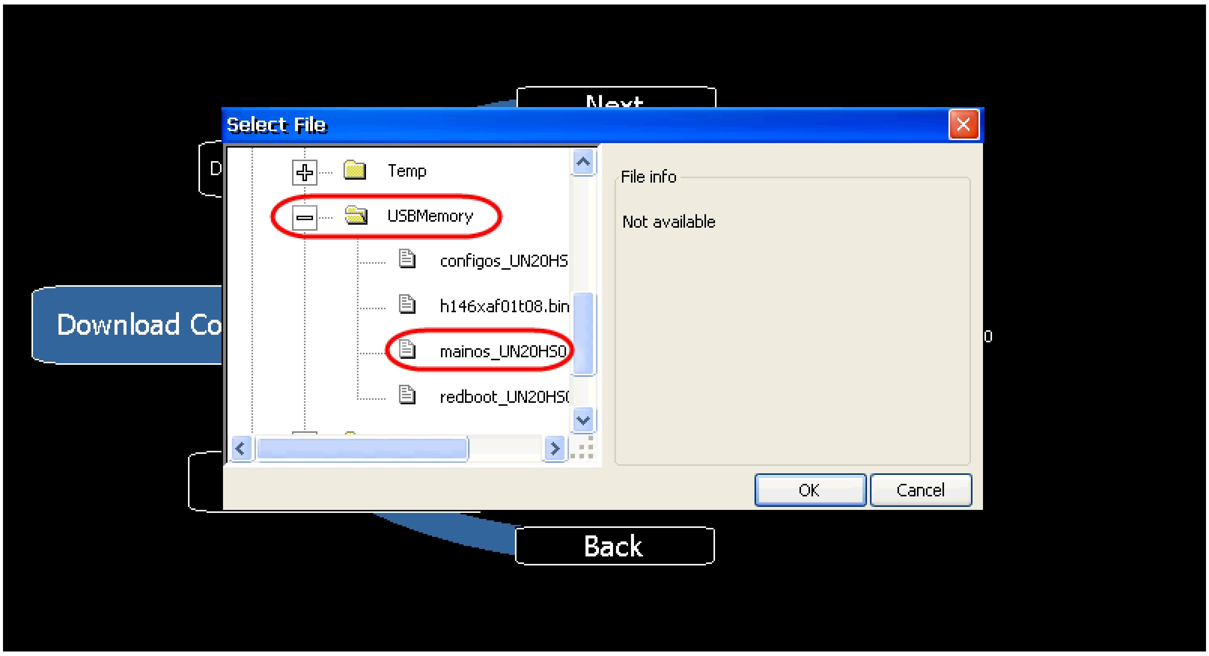
Update of the System Components via USB Flash Drive

System components can be updated via USB flash drives. For each component a specific file is provided. Checksum file with an .md5 extension is required to be present in the same location as the system file to be upgraded.

To update a system component proceed as follows:

Copy all the files you need to upgrade to a USB Memory and plug this into the USB port of the panel.

Start the System Settings tool with the special procedure for getting this in **System Mode**, and then locate the desired item in the rotating menu. Click directly on the item (the blue button with white label) and browse to locate the proper file stored on the pen drive (USBMemory). The figure below shows an example of the Main OS components.



NOTE

Select the "Download" command to transfer files to the panel. Select the "Upload" command to get files from the panel.

Then follow the instructions on the screen to proceed with the update. A progress bar on the screen will inform you about the status of the operation. Please make sure to NOT turn off power to the panel while a system component is being upgraded. Some of the components will require some time for the upgrade to complete.

NOTE

Upgrade procedure may change depending on the hardware revision or operating system version from which you start; please contact technical support offices for any detail about the exact sequence.

Access Protection to HMI Devices

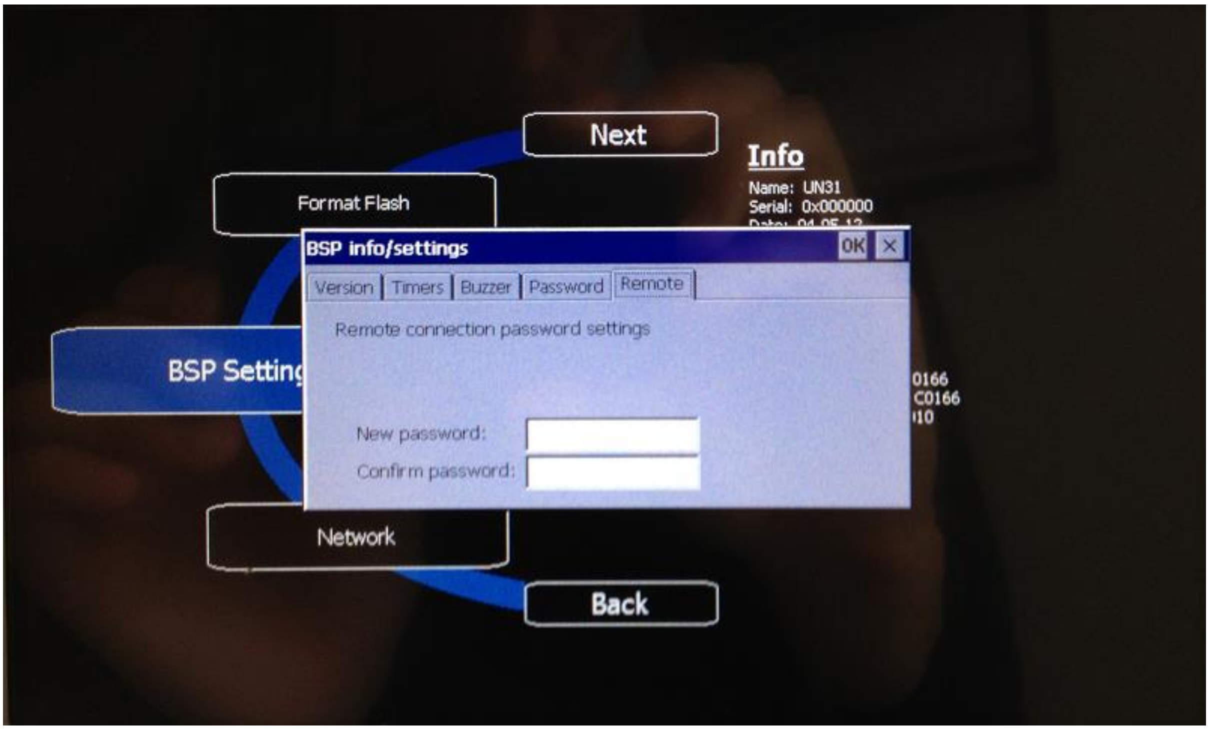
The following operations can be protected with a password:

- Runtime management: Install runtime and update runtime
- Board management: replace main BSP components such as MainOS, ConfigOS, Bootloader, etc.
- Download and upload of project files

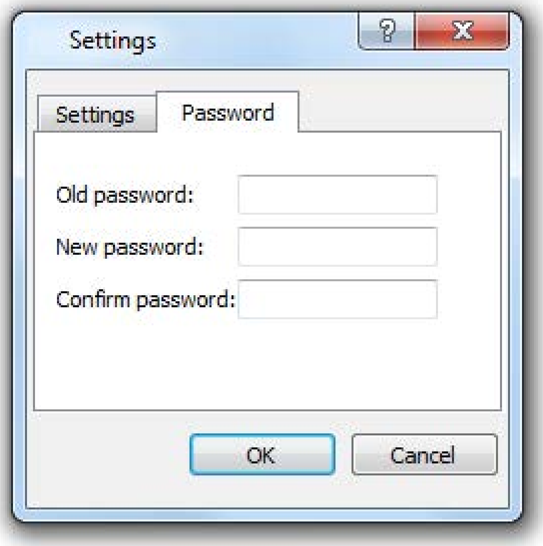
A default value for this password is used by the HMI runtime and Panel Builder 600.

There are three ways to change this password in the HMI runtime:

1. Using the tab **Remote** in the BSP Settings (in system mode) dialog box in System menu includes (starting from BSP versions V1.64 UN30/31 and V2.73 UN20).



2. Using the tab **Password** in Settings from the runtime Context Menu.



Panel Builder 600 shows a dialog asking for the password to match the password defined in the HMI device. The new password will be stored in the computer to be used for further connections.

- Using **Set Target Password** in update package. Password is updated by runtime just after update process is completed. If update failed (for example because **Old password** not match hmi password) a popup inform user about it.

You can enter also the same password in Panel Builder 600 using Manage Target -> Board -> **Connection setting** (to allow in Panel Builder 600 to access runtime).

Default port used for this remote service is 2100.

NOTE

A format of hmi panel reset password.

NOTE

For Win32 Runtime, password is saved into Users\[username]\AppData\Roaming\ABB\buildNumber\server\config\RemoteUpdateConfig.xml.

NOTE

Leave “Old password” empty as default if target password is not set.

Firewalling

Following ports today are used by Panel Builder 600, Windows Client and ActiveX to access to runtime and should be opened in office firewalls and forwarded to runtime if remote access is required:

- 80/tcp (or 443/tcp)
- 21/tcp (FTP in passive mode)
- 2100/tcp
- 16384/tcp
- 5900/tcp (VNC)
- 5100/tcp (JavaScript remote debugger)

For systems using Codesys 2.3 integrated into HMI device, port 1200/tcp is used.

Following ports are used by the studio in HMI discovery and usually are not necessary for remote access but just in local area networks:

- 990/UDP broadcast
- 991/UDP broadcast
- 998/UDP broadcast
- 999/UDP broadcast

When broadcast service is not available (ex. in VPN networks), user have to type exact IP Address directly to reach device from Panel Builder 600.

Factory Restore

If you´re having problems with HMI device, you can try to restore factory settings from system mode.

To restore factory settings proceed as follows:

1. Enter in System Mode
2. Use one of the following operations available in rotating menu:
   - Format Flash for cleanup entire flash disk and registry configuration
   - Restore Factory Settings allows user to select components to cleanup.

Both operations do not manage firmware factory restore (Mains, Congo’s, Boot loader, FPGA images etc.

For more information related to Format Flash and Restore factory Settings please ref. to System Mode.

PB610 Panel Builder 600 Functional Specifications and Compatibility

The scope of this chapter is to provide a clear overview of the supported functions and related limitations for both programming software and HMI Runtime system.

What is listed below in this document is a safe limitation, above which a proper operation and state-of-the-art performance of the system is not guaranteed.

Table of Functions and Limits

Function \ Feature	**Max allowed**
Number of pages	1000 (up to 10k x 10k as resolution based on HMI model)
Number of basic widgets	2000 per page
Number of tags	10000
Number of dialog pages	50 (max 5 can be opened in the same time)
Number of objects of any type in one page	2000
Number of recipes	32
Number of parameter sets for a recipe	1000
Number of elements per recipe	1000
Number of user groups	50
Number of users	50
Number of concurrent clients	4
Number of schedulers	30
Number of alarms	500/2000 (depends on the control panel model)
Number of templates pages	50
Number of actions programmable per button state	32
Number of trend buffers	30
Number of curves per trend widget	5
Number of curves per page	10
Number of samples per trend buffer	200000
Number of trend buffer samples for a project	1200000
Number of messages in a message field	1024
Number of languages	12
Number of events per buffer	2048
Number of event buffers	4
JavaScript file size per page	16 kB
Size of project on disk	30 MB

Compatibility

Starting from the first official release of PB610 Panel Builder 600 V1.58 (00) we have applied the following policy for compatibility:

- PB610 Panel Builder 600 version MUST always be aligned with PB610 Panel Builder 600 Runtime on the Control Panel. The user has the responsibility to update runtime components on the target device together with any PB610 Panel Builder 600 update. A runtime update can be done directly from PB610 Panel Builder 600 using the “Update Target” command available in the “Run\Manage Target” dialog.
- Any version of PB610 Panel Builder 600 newer than V1.58 (00) is able to open and properly handle projects created on an older version, but not older than V1.58 (00).
- Projects created with older versions of PB610 Panel Builder 600, but not older than V1.58 (00), opened with later versions and deployed to compatible runtime, are ensured to maintain the performance and functionality just as before.
- Compatibility between newer versions of runtime and those projects created and deployed with older versions of PB610 Panel Builder 600 is not ensured.
- Do not edit projects with a version of Panel Builder 600 older than the one used to create them. It can result in a damage of the project and to runtime unstability.

Contact us

ABB Automation Products GmbH
Wallstadter Str. 59
68526 Ladenburg, Germany
Phone: +49 62 21 701 1444
Fax: +49 62 21 701 1382
E-Mail: plc.sales@de.abb.com

www.abb.com/plc

Note:
We reserve the right to make technical changes or modify the contents of this document without prior notice. With regard to purchase orders, the agreed particulars shall prevail. ABB AG does not accept any responsibility whatsoever for potential errors or possible lack of information in this document.

We reserve all rights in this document and in the subject matter and illustrations contained therein. Any reproduction, disclosure to third parties or utilization of its contents – in whole or in parts – is forbidden without prior written consent of ABB AG.

Power and productivity
for a better world™