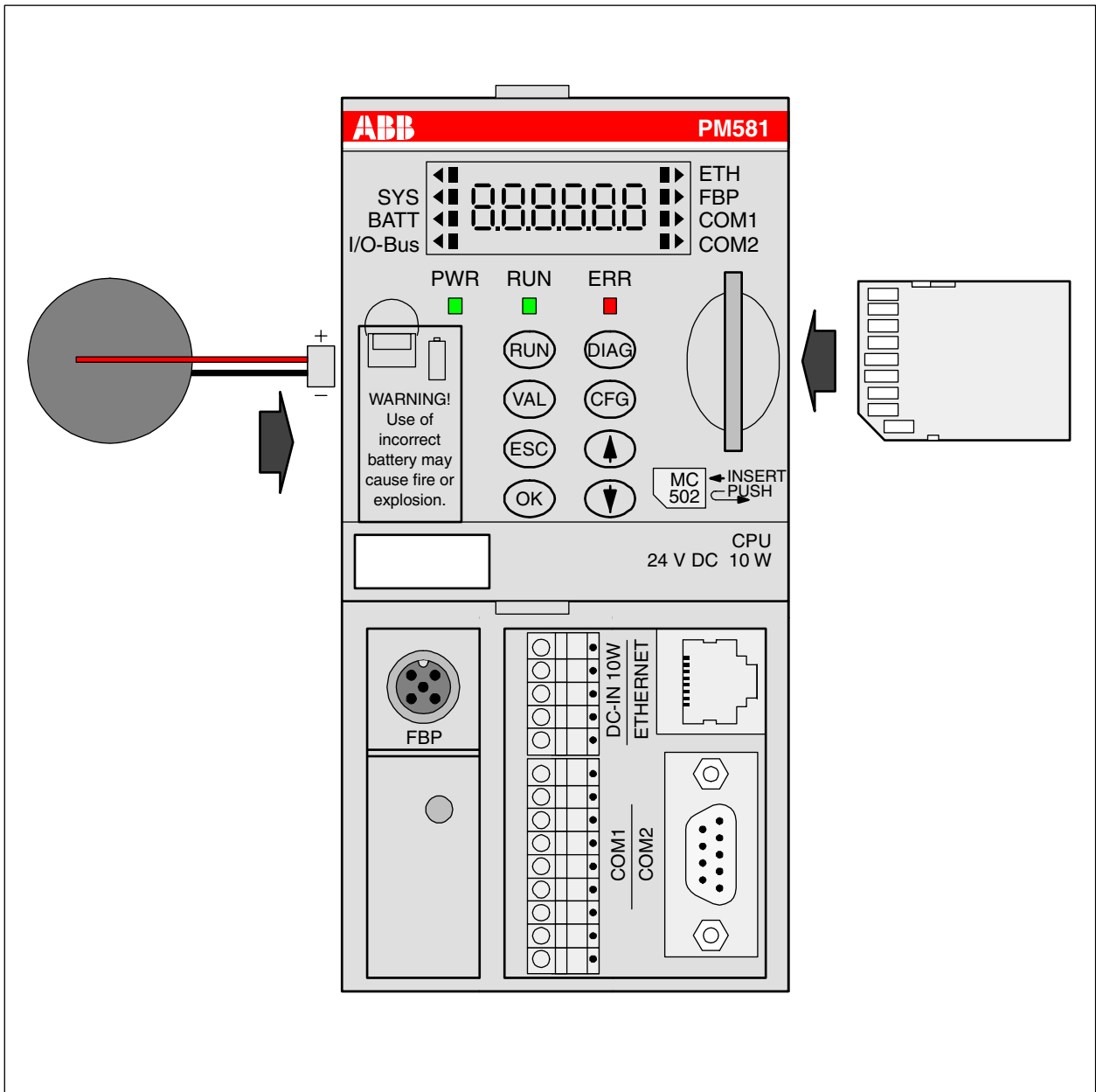


Scalable PLC  
for Individual Automation

System Technology  
of the CPUs





# Contents - System Technology of the AC500 CPUs

<b>1 Target Support Package</b> .....	7
<b>1.1 Introduction</b> .....	7
1.1.1 Control Builder PS501 versions.....	7
1.1.2 New functions in PS501 V1.2 .....	8
1.1.3 Compatibility of versions V1.0, V1.1 and V1.2 .....	9
File structure of the target system .....	9
Overview on target system files.....	9
Compatibility of CPU bootcode, CPU firmware, target system and CoDeSys .....	13
Conversion of a project created with version V1.0 or V1.1 to version V1.2 .....	14
<b>1.2 Selection of the target system - Target support settings</b> .....	15
<b>1.3 CPU parameters in the target support settings</b> .....	16
1.3.1 "Target Platform" settings .....	16
1.3.2 "Memory Layout" settings .....	17
1.3.3 "General" settings .....	19
1.3.4 "Network Functionality" settings .....	21
1.3.5 "Visualization" settings.....	21
<b>1.4 Overview on user program size and operands of AC500 CPUs</b> .....	22
<b>1.5 Installation of AC500 targets with the program installTarget.exe</b> .....	23
<b>2 AC500 inputs, outputs and flags</b> .....	26
<b>2.1 AC500 interfaces for inputs and outputs</b> .....	26
2.1.1 Address scheme for inputs and outputs .....	27
2.1.2 Example for addressing in BOOL / BYTE / WORD / DWORD .....	27
<b>2.2 Addressing of inputs and outputs</b> .....	28
<b>2.3 Processing of inputs and outputs in the multitasking system</b> .....	29
<b>2.4 Addressable flag area (%M area) in the AC500</b> .....	30
2.4.1 Allocation of the addressable flag area in the AC500 .....	30
2.4.2 Access to the %M area using the Modbus® Protocol .....	31
2.4.3 Access to operands in the addressable flag area .....	31
<b>2.5 Absolute addresses of operands</b> .....	32
2.5.1 Address operator ADR.....	32
2.5.2 Bit address operator BITADR .....	32

<b>2.6 Addressable PERSISTENT area (%R area) in the AC500 .....</b>	<b>34</b>
2.6.1 Special features of the addressable PERSISTENT area in the AC500 .....	34
2.6.2 Segmentation of the addressable PERSISTENT area in the AC500.....	35
2.6.3 Saving the buffered data of the AC500's %R area.....	35
2.6.4 Access to operands in the addressable PERSISTENT area (%R area) .....	37
<b>3 The AC500 PLC configuration .....</b>	<b>38</b>
<b>3.1 Overview on the PLC configuration .....</b>	<b>38</b>
3.1.1 PLC configuration functions.....	38
3.1.2 Export and import of configuration data.....	38
3.1.3 Default settings in the PLC configuration .....	39
3.1.4 Setting parameters in the PLC configuration.....	40
<b>3.2 Configuration of CPU parameters .....</b>	<b>40</b>
3.2.1 CPU parameters in PS501 versions V1.0 and V1.1 .....	40
Remark 1: Setting the parameters Auto run and MOD using the display/keypad.....	41
3.2.2 CPU parameters in version PS501 V1.2 .....	42
Remark 1: Setting the parameters Auto run and MOD using the display/keypad.....	44
Remark 2: Error LED .....	44
Remark 3: Behaviour of outputs in Stop .....	44
Remark 4: Reaction on floating point exceptions .....	44
Remark 5: Stop on error class .....	45
Remark 6: Warmstart.....	45
Remark 7: Start PERSISTENT %Rsegment.x and End PERSISTENT %Rsegment.x.....	46
<b>3.3 I/O bus configuration .....</b>	<b>47</b>
3.3.1 Setting the general I/O bus parameters.....	47
3.3.2 Inserting input and output modules .....	47
3.3.3 Configuring the input and output modules and channels .....	48
3.3.4 Module parameter "Ignore module" of S500 I/O devices .....	51
<b>3.4 Configuration of the serial interfaces (Interfaces / COM1 and COM2).....</b>	<b>52</b>
3.4.1 Setting the protocol of the serial interfaces .....	52
3.4.2 The setting 'COMx - Online access' .....	53
3.4.3 The setting 'COMx - ASCII'.....	53
Remark 1: Enable login .....	56
Remark 2: Usage of modems.....	56
Remark 3: Telegram ending identifier .....	57
Remark 4: Checksum .....	59
3.4.4 The setting 'COMx - Modbus' .....	60
3.4.5 The setting 'COM1 - CS31 Bus'.....	62
Connecting the DC551 and S500 I/O devices to the CS31 bus.....	64
Overview on input/output data of S500 I/O devices .....	68
Examples of impossible configurations .....	69

3.4.6 The setting 'COMx - SysLibCom' .....	71
Remark 1: Enable login .....	73
Remark 2: Usage of modems .....	73
Remark 3: Telegram ending identifier .....	73
Example for sending/receiving with "SysLibCom" .....	74
3.4.7 The setting 'COMx - Multi' .....	77
Functions of the block COM_SET_PROT .....	78
<b>3.5 FBP slave interface configuration (Interfaces / FBP slave) .....</b>	<b>79</b>
<b>3.6 Coupler configuration (Couplers) .....</b>	<b>81</b>
3.6.1 Configuring the internal coupler .....	82
3.6.1.1 The internal Ethernet coupler PM5x1-ETH .....	82
3.6.1.2 The internal ARCNET coupler PM5x1-ARCNET .....	84
Remark 1: Baudrate of the ARCNET coupler .....	85
Remark 2: Check of DIN identifier on receipt .....	85
3.6.2 Configuring the external couplers .....	87
<b>4 System start-up / program processing</b>	<b>88</b>
<b>4.1 Terms .....</b>	<b>89</b>
<i>Cold start</i> .....	89
<i>Warm start</i> .....	89
<i>RUN -&gt; STOP</i> .....	89
<i>START -&gt; STOP</i> .....	89
<i>Reset</i> .....	89
<i>Reset (cold)</i> .....	89
<i>Reset (original)</i> .....	89
<i>STOP -&gt; RUN</i> .....	89
<i>STOP -&gt; START</i> .....	90
<i>Download</i> .....	90
<i>Online Change</i> .....	90
<i>Data buffering</i> .....	90
<b>4.2 Start of the user program .....</b>	<b>91</b>
<b>4.3 Data backup and initialization .....</b>	<b>92</b>
4.3.1 Initialization of variables, overview .....	92
4.3.2 Notes regarding the declaration of retentive variables and constants .....	94
Declaration of retentive internal variables .....	94
Declaration of retentive variables in %M area .....	94
Declaration of constants .....	94
<b>4.4 Processing times .....</b>	<b>95</b>
4.4.1 Terms .....	95
4.4.2 Program processing time .....	95
4.4.3 Set cycle time .....	95
<b>4.5 Task configuration for the AC500 CPU .....</b>	<b>96</b>

<b>5 The diagnosis system in the AC500</b> .....	97
<b>5.1 Summary of diagnosis possibilities</b> .....	97
5.1.1 Structure of the diagnosis system .....	97
5.1.2 Diagnosis directly at the PLC by means of "ERR" LED, keypad and display .....	98
5.1.3 Plain-text display of error messages in the Control Builder status line during online mode .....	99
5.1.4 Diagnosis using the PLC browser commands of the Control Builder.....	99
5.1.5 Diagnosis with help of the user program .....	99
<b>5.2 Organization and structure of error numbers</b> .....	99
5.2.1 Error classes .....	100
5.2.2 Error identifiers .....	100
5.2.3 Possible error numbers.....	102
5.2.4 Error list.....	106
5.2.5 Coupler errors.....	113
<b>5.3 Diagnosis blocks for the AC500</b> .....	118
<b>5.4 AC500-specific PLC118 browser commands</b> .....	118
<b>6 The SD memory ca122rd in the AC500</b> .....	122
<b>6.1 SD card functions</b> .....	122
6.1.1 Summary of memory card functions.....	122
6.1.2 PLC browser commands for accessing the SD card.....	122
<b>6.2 SD card file system</b> .....	123
6.2.1 SD card file structure .....	123
File structure in versions V1.0 and V1.1 .....	123
File structure as of version V1.2 .....	124
6.2.2 The command file "SDCARD.INI" .....	126
File content in versions V1.0 and V1.1 .....	126
File content as of version V1.2 .....	127
6.2.3 Initializing an SD card .....	129
6.2.3.1 Initializing an SD card using the AC500 .....	129
6.2.3.2 Initializing the SD card using a PC .....	129
<b>6.3 Storing/loading the user program to/from an SD card</b> .....	130
6.3.1 Storing the user program to an SD card.....	130
6.3.2 Loading a user program from the SD card to the AC500 .....	130
<b>6.4 Storing/reading user data to/from an SD card</b> .....	131
6.4.1 Structure of data files stored on the SD card.....	131

6.4.2	Blocks for storing/reading user data to/from the SD card.....	132
6.4.3	Deleting a data file stored on the SD card.....	134
6.4.4	Storing user data to the SD card - data file without sectors .....	134
6.4.5	Storing user data to the SD card - data file with sectors .....	135
6.4.6	Loading user data from the SD card - data file without sectors.....	136
6.4.7	Loading user data from the SD card - data file with sectors.....	137
<b>6.5</b>	<b>Storing and loading retentive data to/from an SD card .....</b>	<b>138</b>
<b>6.6</b>	<b>Firmware update from the SD card .....</b>	<b>138</b>
6.6.1	Storing the firmware to the SD card .....	138
6.6.2	Updating the firmware of the AC500 CPU from the SD card .....	138
<b>6.7</b>	<b>Writing and reading the project sources to/from the SD card.....</b>	<b>139</b>
6.7.1	Writing the project sources from PC to SD card.....	140
6.7.2	Loading the project sources from the PLC's SD card into the PC.....	142
6.7.3	Loading the project sources from the SD card using the PC SD card reader.....	144
<b>6.8</b>	<b>SD card error messages .....</b>	<b>145</b>
<b>7</b>	<b>Data storage in Flash memory .....</b>	<b>146</b>
7.1	Blocks used for data storage.....	146
7.2	Example program for data storage .....	146
<b>8</b>	<b>Real-time clock and battery in the AC500.....</b>	<b>147</b>
8.1	General notes concerning the real-time clock in the AC500.....	147
8.2	Setting and displaying the real-time clock.....	147
8.2.1	Setting and displaying the real-time clock with the PLC browser.....	147
8.2.2	Setting and displaying the real-time clock with the user program .....	148
8.3	The AC500 battery .....	148
<b>9</b>	<b>The fast counters in the AC500 .....</b>	<b>149</b>
9.1	Activating the fast counters via the I/O bus.....	149
9.2	Counting modes of the fast counters.....	149
<b>10</b>	<b>Programming and testing.....</b>	<b>150</b>
10.1	Programming interfaces to the AC500 used by the Control Builder .....	150
10.2	Programming via the serial interfaces .....	151

10.2.1 Serial driver "Serial (RS232)" .....	152
10.2.2 Serial driver "ABB RS232 Route AC" .....	153
<b>10.3 Programming via ARCNET</b> .....	<b>156</b>
10.3.1 ARCNET driver "ABB Arcnet AC" .....	157
<b>10.4 Programming via Ethernet (TCP/IP)</b> .....	<b>159</b>
10.4.1 Ethernet driver "Tcp/Ip".....	160
10.4.2 Ethernet driver "ABB Tcp/Ip Level 2 AC".....	161
10.4.3 Ethernet ARCNET routing .....	164
<b>11 Communication with Modbus RTU</b> .....	<b>166</b>
<b>11.1 Protocol description</b> .....	<b>166</b>
<b>11.2 Modbus RTU with the serial interfaces COM1 and COM2</b> .....	<b>167</b>
11.2.1 Modbus operating modes of the serial interfaces.....	167
<b>11.3 Modbus on TCP/IP via Ethernet</b> .....	<b>167</b>
<b>11.4 Modbus addresses</b> .....	<b>168</b>
11.4.1 Modbus address table .....	168
11.4.2 Peculiarities for accessing Modbus addresses.....	170
11.4.3 Comparison between AC500 and AC31/90 Modbus addresses .....	171
<b>11.5 Modbus telegrams</b> .....	<b>173</b>
<b>11.6 Function block COM_MOD_MAST</b> .....	<b>180</b>
<b>12 Index - System Technology of the CPUs</b> .....	<b>181</b>



# 1 Target Support Package

## 1.1 Introduction

### 1.1.1 Control Builder PS501 versions

The AC500 basic units PM57x, PM58x and PM59x are programmed using the AC500 Control Builder (version V1.0 and later).

The Control Builder versions V1.0 and 1.1 are based on CoDeSys version V2.3 SP4 Patch 9 (V2.3.4.9+) and later. The Control Builder version V1.2 is based on CoDeSys version V2.3 SP8 Patch 0 (V2.3.8.0) and later.



**Note:** This documentation applies to all categories of the basic units PM57x, PM58x and PM59x. When the term "PM581" is given in the text, this text applies also to PM57x, PM58x and PM59x. Texts that are exclusively applicable for PM581 are expressly mentioned by a note.

To be able to program the AC500 controllers with the Control Builder, a so-called Target Support Package (TSP) must be installed. By default, the AC500 TSPs are automatically installed during installation of the Control Builder.

The default installation paths are as follows:

- **Control Builder:**  
..\%ProgramFiles%\3S Software\CoDeSys V2.3  
The environment variable %ProgramFiles% points to the directory "Program Files" on English operating systems and "Programme" on German operating systems.
- **TSP (Target Support Package):**  
..\%CommonProgramFiles%\CAA-Targets\ABB\_AC500  
The environment variable %CommonProgramFiles% points to the directory "Program Files\Common Files" on English operating systems and "Programme\Gemeinsame Dateien" on German operating systems.

The chapter "Installation of a Target Support Package" describes in detail how to install a TSP.

A Target Support Package (TSP) contains all configuration and expansion files necessary to operate a particular controller (target system) with an application. What has to be configured:

- code generator
- memory layout
- controller functions
- I/O modules
- interface assignment and parameterization

In addition, libraries, gateway drivers, target system specific help files, controller configuration files, error description file (Errors.ini) and the help file for the PLC browser (Browser.ini) are included.



**Note:** The basic units of the AC31 series 90 (07 KT 95, 07 KT 96, 07 KT 97 and 07 KT 98) are still programmed using the programming system 907 AC 1131.

## 1.1.2 New functions in PS501 V1.2

Version 1.2 implements the following new functions compared to versions V1.0 and V1.1:

- CD user interface in German, English, French and Spanish
- New CoDeSys version with display of the ABB Control Builder version. Available menu languages: German, English, French, Spanish, Italian and Russian
- Allows the installation of the CoDeSys HMI
- New AC500 targets PM5xy\_V12 for downward compatibility of projects created with versions V1.0/V1.1
- ABB EDS configurator for the generation of fix configurations from modular EDS files for DeviceNet
- Revised and expanded documentation
- Ethernet UDP: Selection of the UDP port in the PLC Configuration
- Ethernet/Online: Driver "ABB Tcp/Ip AC" (AA/55 protocol, same driver for AC500 and AC31 S90/07KT9x)
- COMx: Protocol "SysLibCom" - Support of the blocks contained in the 3S library SysLibCom.lib
- COMx: Protocol "Multi" - Switching between two protocols, for example ASCII/Modbus with the FB COM\_SET\_PROT
- COMx: Parameter 'Enable login' for CoDeSys login via COMx, if COMx is configured with the protocol "Modbus", "ASCII", "SysLibCom" or "Multi"
- Integration of the CS31 device DC551-CS31
- ARCNET online access using the driver "ABB ARCNET AC"
- ARCNET-\_5F\_ARC - 5F-ARCNET protocol (as in 07KT97/07KT98)
- ARCNET data exchange with the FBs ARC\_SEND, ARC\_REC, ARC\_STO
- ARCNET-FB: ARC\_INFO, ARC\_OWN\_NODE
- SD card: Firmware download for field bus couplers
- New PERSISTENT area "%R area", configurable as VAR RETAIN PERSISTENT in the PLC Configuration / CPU parameters
- PLC - Browser: Command "coupler settings" to display, for example, the IP address and the socket assignment of the Ethernet coupler
- Call stack in case of an application crash
- CPU parameters: 'Warmstart'->off/E2/voltage dip/e2 or voltage dip (automatic restart of the CPU after E2 errors and/or short voltage dips, as in 07KT98)
- STOP of the user program if a task has been suspended (all outputs set to FALSE/0)
- FPU-Exception: FB FPUEXINFO (for PM591, PM590 only)
- New CPU: PM590
- Blocks that enable the export/import of RETAIN data to/from the SD card (except %M area)
- Blocks that enable the export/import of the PERSISTENT area (%R area) to/from the SD card



**Note:** The new functions implemented in version V1.2 can only be used together with the targets PM5xy\_V12, CoDeSys as of version V2.3.8.0 and the AC500 firmware as of version V1.2.0. Further information about this can be found in the chapter "Compatibility of versions V1.0, V1.1 and V1.2".

### 1.1.3 Compatibility of versions V1.0, V1.1 and V1.2

The new functions in version 1.2 listed in the chapter above require extensive changes in the firmware of the AC500 CPUs, in the target files and also according changes in CoDeSys.

To ensure downward compatibility with versions V1.0 and V1.1, new target system files for all AC500 CPUs have been created for version V1.2. These files are installed in parallel to the target system files of version V1.1 when installing Control Builder V1.2. Only the newest online help files will be installed.

#### File structure of the target system

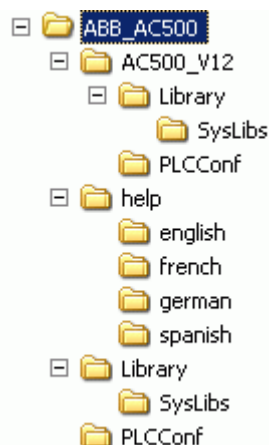
The target system files are structured as follows in version V1.2:

Installation path (as for V1.0 and V1.1):

..\%CommonProgramFiles%\CAA-Targets\ABB\_AC500

The environment variable %CommonProgramFiles% is points to the directory

- "Program Files\Common Files" on English operating systems and
- "Programme\Gemeinsame Dateien" on German operating systems.



#### Overview on target system files

The following files are part of the target:

No.	File / Directory	V1.0	V1.1	V1.2 see remark 1
<b>1. Target files</b>		<b>Root = ..\</b>		<b>Root = ..\</b>
1	AC500_PM571.trg	2005-07-28	2005-07-28	-
2	AC500_PM581.trg	2005-07-28	2005-07-28	-
3	AC500_PM591.trg	2005-07-28	2005-07-28	-
4	AC500_multi.tnf (install)	2005-06-01	2005-06-01	-
5	AC500_PM582.trg	-	2005-08-23	-
6	AC500_PM582.tnf (install)	-	2005-08-23	-
7	AC500_PM571_V12.trg	-	-	2007-05-16
8	AC500_PM581_V12.trg	-	-	2007-05-16
9	AC500_PM582_V12.trg	-	-	2007-05-16
10	AC500_PM590_V12.trg	-	-	2007-05-16
11	AC500_PM591_V12.trg	-	-	2007-05-16
12	AC500_V12.tnf (install)	-	-	2007-05-23

#### Remark 1:

The file date of the version V1.2 files is the date set on the CD PS501 V1.2 BETA 1. Thus, the release version may have other files.

No.	File / Directory	V1.0	V1.1	V1.2 see remark 1
<b>2. Target ancillary files</b>		<b>Root = ..\</b>		<b>Root = ..\AC500_V12</b>
1	AC500.hll	2004-05-27	2004-05-27	2007-04-26
2	Browser.ini	2005-09-07	2006-02-21	2006-06-02
3	Errors.ini	2005-05-23	2006-05-23	2006-05-23
4	Errors.xml (in CoDeSys directory)	2005-06-13	2006-05-29	2006-05-29
5	TaskConfig.xml	2004-04-21	2006-02-21	-
6	TaskConfig_PM57x.xml	-	-	2006-02-21
7	TaskConfig_PM58x.xml	-	-	2006-02-21
8	TaskConfig_PM59x.xml	-	-	2006-02-21

**Remark 1:**

The file date of the version V1.2 files is the date set on the CD PS501 V1.2 BETA 1. Thus, the release version may have other files.

No.	File / Directory	V1.0	V1.1	V1.2 see remark 1
<b>3. Configuration files see remark 2</b>		<b>Root = ..\PLCConf</b>		<b>Root = ..\AC500_V12\PLCConf</b>
1	AC500_COMx_V10.cfg	2005-07-26	2006-07-20	2007-04-19
2	AC500_CPU_V10.cfg	2005-07-26	2006-05-23	2007-04-19
3	AC500_FBPSlave_V10.cfg	2005-07-26	2006-05-23	2007-04-11
4	AC500_ModCS31_V10.cfg	2005-07-26	2006-07-24	2007-04-11
5	AC500_IOmodule_V10.cfg	2005-09-06	2006-07-28	2007-04-24
6	AC500_Coupler_V10.cfg	2005-09-07	2006-05-23	2007-04-13
7	AC500_CS31Base_V10.cfg	-	2006-05-23	2007-04-11
8	AC500_DC541_V11.cfg	-	2006-05-25	2007-04-13
9	AC500_IOmod2_V11.cfg	-	2006-07-04	2007-04-11
10	AC500_CAN_DevNet_V11.cfg	-	2006-07-13	2007-04-11
11	AC500_IOclass_V12.cfg	-	-	2007-04-11
12	AC500_ARCNET_V12.cfg	-	-	2007-04-11
13	AC500_COMNewProt_V12.cfg	-	-	2007-04-19

**Remark 1:**

The file date of the version V1.2 files is the date set on the CD PS501 V1.2 BETA 1. Thus, the release version may have other files.

**Remark 2:**

The directory ..\PLCConfig may only contain the \*.cfg files listed here because all \*.cfg files are loaded when loading a project. If the directory contains, for example, configuration files with partly the same content, both files are loaded. This can lead to unforeseeable effects!

No.	File / Directory	V1.0	V1.1	V1.2 see remark 1
<b>4. Libraries</b>		<b>Root = ..\Library</b>		<b>Root = ..\AC500_V12\Library</b>
1	BusDiag-lib	2004-08-27	2004-08-27	2004-08-27
2	CS31_AC500_V10.lib	2005-06-01	2005-06-01	2005-06-01
3	SysExt_AC500_V10.lib	2005-07-01	2005-07-01	2005-07-01
4	Serie90_AC500_V10.lib	2005-07-27	2005-07-27	2005-07-27
5	ASCII_AC500_V10.lib	2005-07-27	2005-07-27	2006-11-08
6	MODBUS_AC500_V10.lib	2005-07-28	2006-06-01	2006-06-01
7	Ethernet_AC500_V10.lib	2005-07-28	2005-07-28	2005-07-28
8	SysInt_AC500_V10.lib	2005-08-03	2006-06-02	2007-05-24
9	Diag_AC500_V10.lib	2005-08-10	2005-08-10	2005-08-10
10	PROFIBUS_AC500_V10.lib	2005-08-12	2005-08-12	2005-08-12
11	DeviceNet_AC500_V11.lib	-	2006-01-26	2006-01-26
12	CANopen_AC500_V11.lib	-	2006-03-02	2006-03-02
13	DC541_AC500_V11.lib	-	2006-06-01	2007-04-13
14	Counter_AC500_V11.lib	-	2006-06-02	2006-06-02
15	ARCNET_AC500_V12.lib	-	-	2007-03-01

**Remark 1:**

The file date of the version V1.2 files is the date set on the CD PS501 V1.2 BETA 1. Thus, the release version may have other files.

No.	File / Directory	V1.0	V1.1	V1.2 see remark 1
<b>5. System libraries (partly) supported by AC500</b>		<b>Root = ..\Library\SysLibs</b>		<b>Root = ..\AC500_V12\ Library\SysLibs</b>
1	SysLibCallback.lib	2005-07-18	2005-07-18	2005-07-18
2	SysLibCom.lib	2005-07-18	2005-07-18	2005-07-18
3	SysLibEvent.lib	2005-07-18	2005-07-18	2005-07-18
4	SysLiblecTasks.lib	2005-07-18	2005-07-18	2005-07-18
5	SysLibMem.lib	2005-07-18	2005-07-18	2005-07-18
6	SysLibPLCConfig.lib	2005-07-18	2005-07-18	2005-07-18
7	SysLibPlcCtrl.lib	2005-07-18	2005-07-18	2005-07-18
8	SysLibProjectInfo.lib	2005-07-18	2005-07-18	2005-07-18
9	SysLibRtc.lib	2005-07-18	2005-07-18	2005-07-18
10	SysLibSem.lib	2005-07-18	2005-07-18	2005-07-18
11	SysLibStr.lib	2005-07-18	2005-07-18	2005-07-18
12	SysLibTasks.lib	2005-07-18	2005-07-18	2005-07-18
13	SysLibTime.lib	2005-07-18	2005-07-18	2005-07-18
14	SysLibVisu.lib	-	-	2002-08-08
15	SysTaskInfo.lib	2005-07-18	2005-07-18	2005-07-18

**Remark 1:**

The file date of the version V1.2 files is the date set on the CD PS501 V1.2 BETA 1. Thus, the release version may have other files.

No.	File / Directory	V1.0	V1.1	V1.2 see remark 1
<b>6. Online help German version</b>		<b>Root = ..\help\german</b>		
1	CAA-Merger-1.chm	2005-09-30	2006-08-04	2006-08-04
2	CAA-Merger-2.chm	2006-04-07	2006-07-06	2007-05-22
3	CAA-Merger-3.chm	2005-08-11	2005-08-11	2005-08-11
4	CAA-Merger-6.chm	2006-03-27	2006-06-29	2007-05-14
5	CAA-Merger-7.chm	2006-03-16	2006-09-27	2006-09-27
6	CAA-Merger-8.chm	2005-09-14	2005-09-14	2005-09-14
<b>7. Online help English version</b>		<b>Root = ..\help\english</b>		
1	CAA-Merger-1.chm	2005-09-30	2006-08-04	2006-08-04
2	CAA-Merger-2.chm	2006-04-12	2006-07-06	2007-05-22
3	CAA-Merger-3.chm	2005-08-18	2005-08-18	2005-08-11
4	CAA-Merger-6.chm	2006-03-27	2006-06-29	2007-05-14
5	CAA-Merger-7.chm	2006-04-18	2006-10-04	2006-09-27
6	CAA-Merger-8.chm	2005-09-14	2005-09-14	2005-09-14

**Remark 1:**

The file date of the version V1.2 files is the date set on the CD PS501 V1.2 BETA 1. Thus, the release version may have other files.

No.	File / Directory	V1.0	V1.1	V1.2 see remark 1
<b>8. Online help French version</b>		<b>Root = ..\help\frrench (see remark 3)</b>		
1	CAA-Merger-1.chm	-	2006-08-04	2006-08-04
2	CAA-Merger-2.chm	-	2006-07-06	2007-05-22
3	CAA-Merger-3.chm	-	2005-08-18	2005-08-11
4	CAA-Merger-6.chm	-	2006-06-29	2007-05-14
5	CAA-Merger-7.chm	-	2006-10-04	2006-09-27
6	CAA-Merger-8.chm	-	2005-09-14	2005-09-14
<b>9. Online help Spanish version</b>		<b>Root = ..\help\spanish (see remark 3)</b>		
1	CAA-Merger-1.chm	-	-	2006-08-04
2	CAA-Merger-2.chm	-	-	2007-05-22
3	CAA-Merger-3.chm	-	-	2005-08-11
4	CAA-Merger-6.chm	-	-	2007-05-14
5	CAA-Merger-7.chm	-	-	2006-09-27
6	CAA-Merger-8.chm	-	-	2005-09-14

**Remark 1:**

The file date of the version V1.2 files is the date set on the CD PS501 V1.2 BETA 1. Thus, the release version may have other files.

**Remark 3:**

The online help is currently only available in German and English. As soon as the translation is completed, the according files will be replaced. The English help files are currently used in the Spanish and French version.

## Compatibility of CPU bootcode, CPU firmware, target system and CoDeSys

The following table shows the possible combinations of CPU bootcode, CPU firmware, target system files and CoDeSys.

		Bootcode			Firmware			Target			CoDeSys			
		V1.0.2	V1.1.3	V1.2.0	V1.0.2	V1.1.7	V1.2.0	V1.0	V1.1	V1.2	V2.3.4.9+	V2.3.5.x.. V2.3.7.5	V2.3.7.5+ Test	V2.3.8.0
Bootcode	V1.0.2				++	+	-	++	++	-	++	+	+	++
	V1.1.3				++	++	-	++	++	-	++	+	+	++
	V1.2.0				-	-	++	++	++	++	+	+	+	++
Firmware	V1.0.2	++	++	-				++	+	-	++	+	+	++
	V1.1.7	+	++	-				+	++	-	++	+	+	++
	V1.2.0	-	-	++				+	+	++	+	+	+	++
Target	V1.0	++	++	++	++	+	+				++	+	+	++
	V1.1	++	++	++	+	++	+				++	+	+	++
	V1.2	-	-	++	-	-	++				+	+	+	++
CoDeSys	V2.3.4.9+	++	++	+	++	++	+	++	++	+				
	V2.3.5.x .. V2.3.7.5	+	+	+	+	+	+	+	+	+				
	V2.3.7.5+ Test	+	+	+	+	+	+	+	+	+				
	V2.3.8.0	++	++	++	++	++	++	++	++	++				

++	=> all functions available
+	=> possible, but not all functions available
-	=> combination <b>NOT</b> possible

## Conversion of a project created with version V1.0 or V1.1 to version V1.2

Proceed as follows to convert a project created with version V1.0 or V1.1 to version V1.2:

1. Save the project (for example to the directory ..\save\_projects)
2. Open the project in CoDeSys
3. Project => Save as (for example under the name name\_V12.pro)
4. Resources => Target Settings => Select AC500 PM5xy V1.2
5. Execute Project => Clean all and Project => Rebuild all
6. File => Save
7. Online => Download
8. Online => Create boot project
9. Online => Run



**Caution:** A project created for the target PM5xy V1.2 can only be loaded into an AC500 PM5xy with bootcode as of version V1.2.0 and firmware as of version V1.2.0. If a controller with firmware version V1.1.7 is loaded, CoDeSys reports an according error after the download!

If a version V1.2 target is selected, the target system files, libraries and configuration files contained in the directory ..\ABB\_AC500\AC500\_V12 will be loaded. This ensures that the new functions will be available.

A **change back** to a target of version V1.0 or V1.1 is no longer possible as soon as a parameter, that is only available as of version V1.2, is set in the PLC configuration. For example, if the protocol "Multi" is set for the serial interface or the CPU parameter "Warmstart" is set to the value "On after E2 or short voltage dip". Since the according settings are not available in the configuration files of the targets of version V1.0 or V1.1, the PLC configuration cannot be loaded and has to be recreated, if necessary!

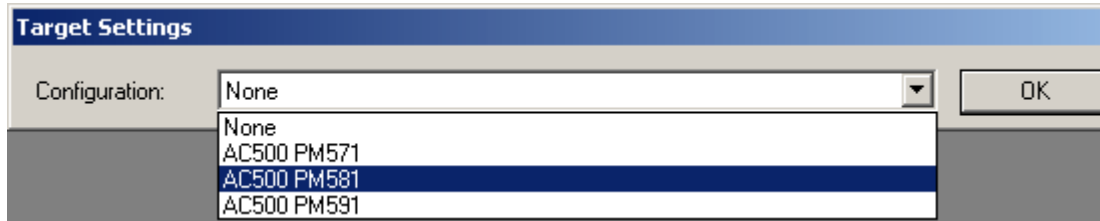
Thus, always save the projects before doing the conversion!



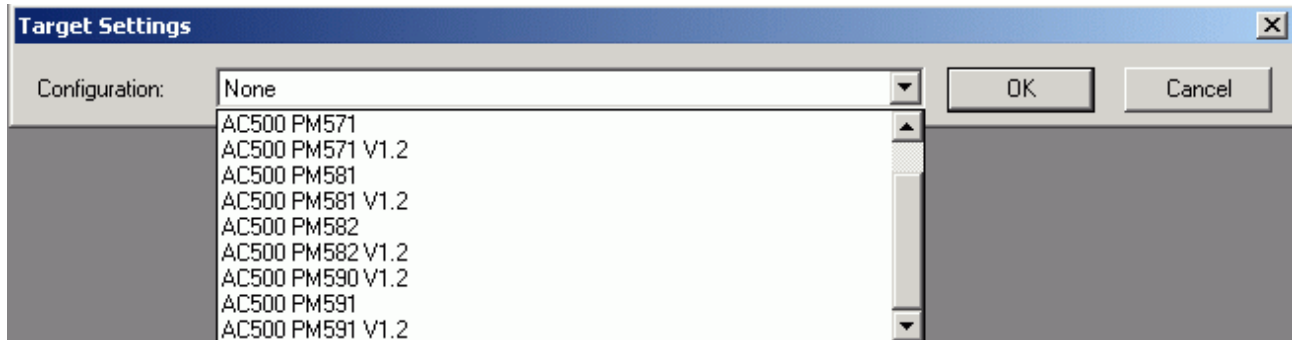
## 1.2 Selection of the target system - Target support settings

The assignment of a project to an AC500 CPU is done using the target support settings in the Control Builder.

When creating a new AC500 project with File / New, the target system must be selected first:



As of Control Builder version V1.2, the following AC500 targets can be selected.



Select the desired CPU and confirm your selection with OK.

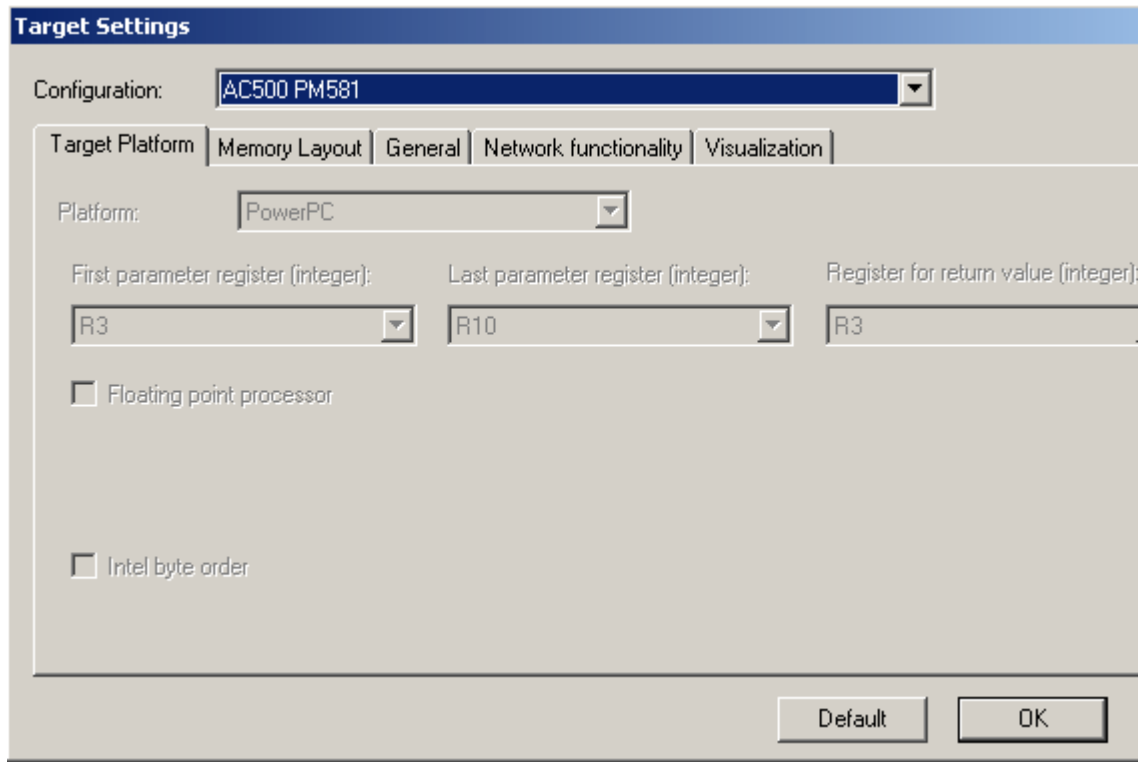


**Note:** If you want to download an existing project to another AC500 CPU, the new CPU must be set in the target system settings. To do this, select the object "Target Settings" in the "Resources" tab. In the appearing window, select the desired CPU from the "Configuration" list box.

## 1.3 CPU parameters in the target support settings

### 1.3.1 "Target Platform" settings

Once you have selected an AC500 CPU, the general CPU parameters are displayed:



The tab "Target Platform" displays the general settings for the CPU. These parameters cannot be changed.

The parameters have the following meaning:

Parameter	Meaning
Platform	Target system type - PowerPC
Floating point processor	PM57x/PM58x: Floating point operations are emulated PM59x: Floating point processor
First parameter register (integer)	Register where the first (integer) parameter of C function calls is passed (range depends on the operating system).
Last parameter register (integer)	Register where the last (integer) parameter of C function calls is passed (range depends on the operating system).
Register for return value (integer)	Register where the integer values of C function calls are returned (range depends on the operating system).
Intel byte order	If deactivated: Motorola byte address scheme is applied.

### 1.3.2 "Memory Layout" settings

The tab "Memory Layout" contains all information about the memory mapping within the CPU:

**Target Settings**

Configuration: AC500 PM581

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

	Base	Size	Area
Code :		16#40000	1
Global :		16#40000 per segment	4
Memory :		16#20000	5
Input :		16#6000	6
Output :		16#6000	2
Retain:		16#8000	

Retain in own segment

Maximum number of POU's: 1024

Total size of data memory: 16#0

Maximum number of global data segments: 1

Default OK Cancel

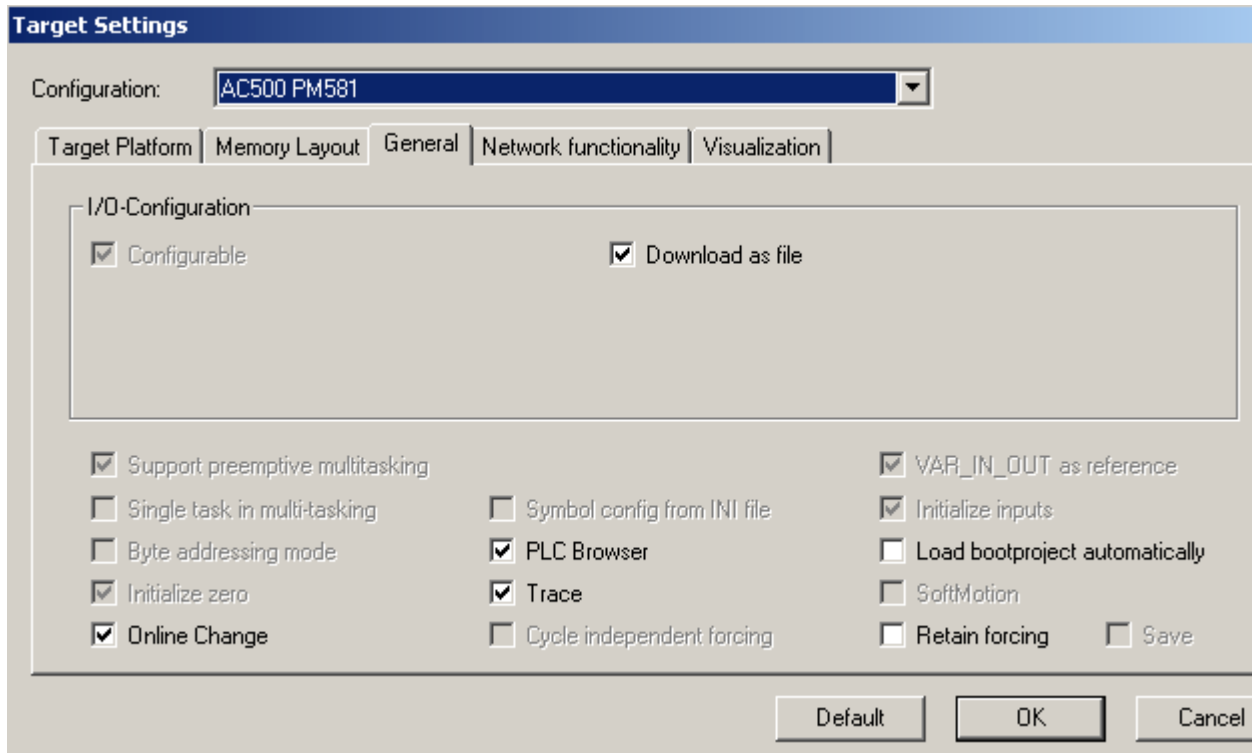
The parameters have the following meaning:

Parameter	Meaning										
Size (Code)	Code area size (user program) see chapter "User program size and operands of AC500 CPUs"										
Size per segment (Global)	Data area size (VAR ... END_VAR or VAR_GLOBAL & END_VAR) see chapter "User program size and operands of AC500 CPUs"										
Area (Memory)	Size of the addressable flag area (%M area) see chapter "User program size and operands of AC500 CPUs"										
Area (Input)	Size of the input process image (%I area) PM5x1: 6000 hex = 24 kB										
Area (Output)	Size of the output process image (%Q area) PM5x1: 6000 hex = 24 kB										
Size (Retain)	Size of the area for retentive data (VAR RETAIN) see chapter "User program size and operands of AC500 CPUs"										
Retain in own segment	If activated: Retentive data are administered in an own segment										
Total size of data memory	Can be ignored, set in firmware										
Maximum number of POUs	Maximum number of blocks allowed in the project see chapter "User program size and operands of AC500 CPUs"										
	<table border="1"> <tr> <td><b>The following applies:</b></td> <td></td> </tr> <tr> <td>- per function</td> <td>1 POU</td> </tr> <tr> <td>- per program</td> <td>1 POU</td> </tr> <tr> <td>- per function block</td> <td>2 POUs</td> </tr> <tr> <td>- per data type</td> <td>1 POU</td> </tr> </table>	<b>The following applies:</b>		- per function	1 POU	- per program	1 POU	- per function block	2 POUs	- per data type	1 POU
	<b>The following applies:</b>										
- per function	1 POU										
- per program	1 POU										
- per function block	2 POUs										
- per data type	1 POU										
For library POUs the same number of POUs for the relevant type is valid.											
Maximum number of global data segments	1 = all global data are administered in one segment.										

The numbers for the individual areas are required for administration in the firmware. The user cannot change any of these parameters.

### 1.3.3 "General" settings

The tab "General" contains general information about the target system and the relevant windows in the Control Builder.



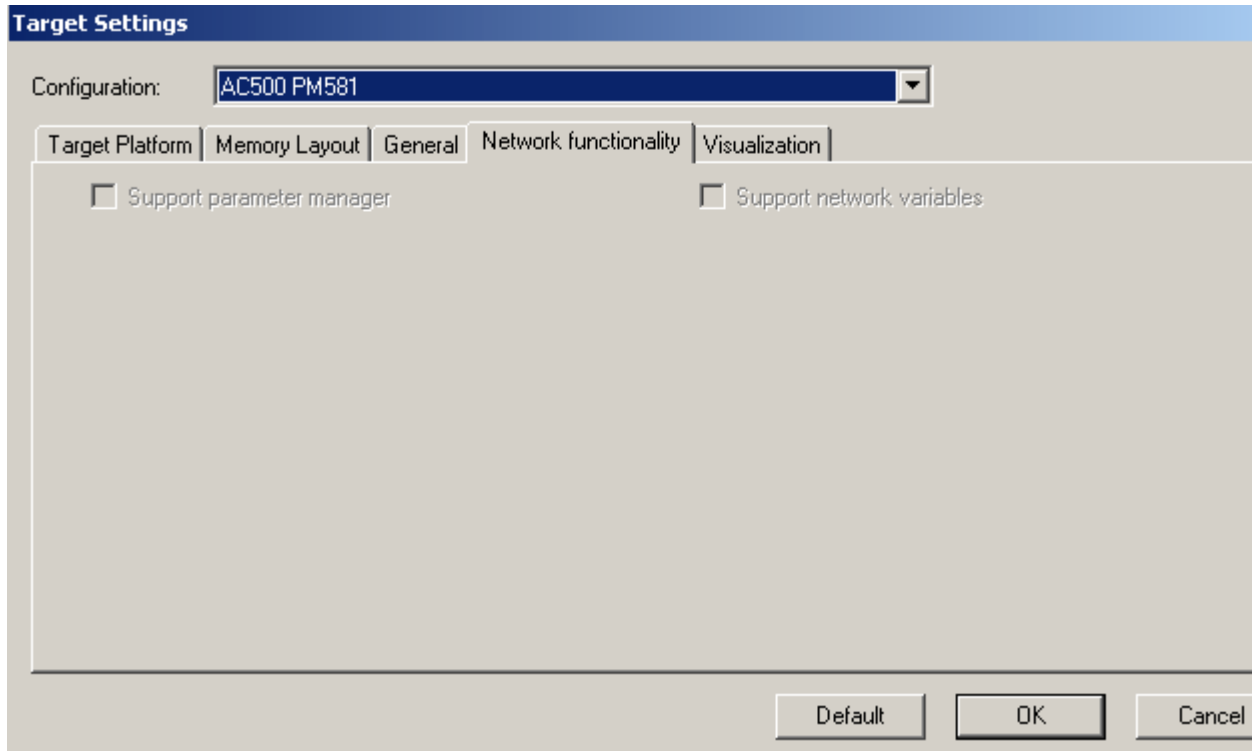
The parameters have the following meaning:

Parameter	Meaning
Configurable	<b>If activated:</b> Support configurable I/O configurations and download configuration descriptions to the controller
Download as file	<b>If activated:</b> When downloading, the I/O configuration file is downloaded to the controller together with the project. If the checkbox is <b>deactivated</b> , the I/O configuration will be downloaded to the controller in the download format <b>without displaying the download progress</b> .
Support preemptive multitasking	<b>If activated:</b> Support task configuration and download task description to controller
No address checking	<b>If deactivated:</b> When compiling the project, the IEC addresses are checked
Online Change	<b>If activated:</b> Online Change functionality
Singletask in multitasking	not yet implemented
Byte addressing mode	<b>If deactivated:</b> Byte addressing mode
Initialize zero	<b>If activated:</b> General initialization with zero
Download symbol file	<b>If deactivated:</b> The symbol file possibly created during the download will not be loaded into the controller.
Symbol config from INI file	<b>If deactivated:</b> The parameters for the symbol configuration are read from the project options window.
PLC Browser	<b>If activated:</b> PLC browser functionality supported.
Trace	<b>If activated:</b> Trace recording possible.
Cycle independent forcing	not yet implemented
VAR_IN_OUT as reference	<b>If activated:</b> At a function call, VAR_IN_OUT variables are called by reference (pointer); therefore no constants can be assigned and no read/write access is possible from outside the function block.
Initialize inputs	<b>If activated:</b> Init code is generated for the inputs declared with "AT %IX".
Load bootproject automatically	<b>If activated:</b> After the download, a bootproject is created automatically from the new program and sent to the PLC.
SoftMotion	<b>If deactivated:</b> No SoftMotion functionality supported
Retain forcing	<b>If activated:</b> Forced values are kept even at logout (provided the user has decided to retain the force list).
Save	If the options "Retain forcing" and "Save" are activated, the force list is stored to flash memory when "Creating the bootproject".

Parameters with gray background can be changed by the user.

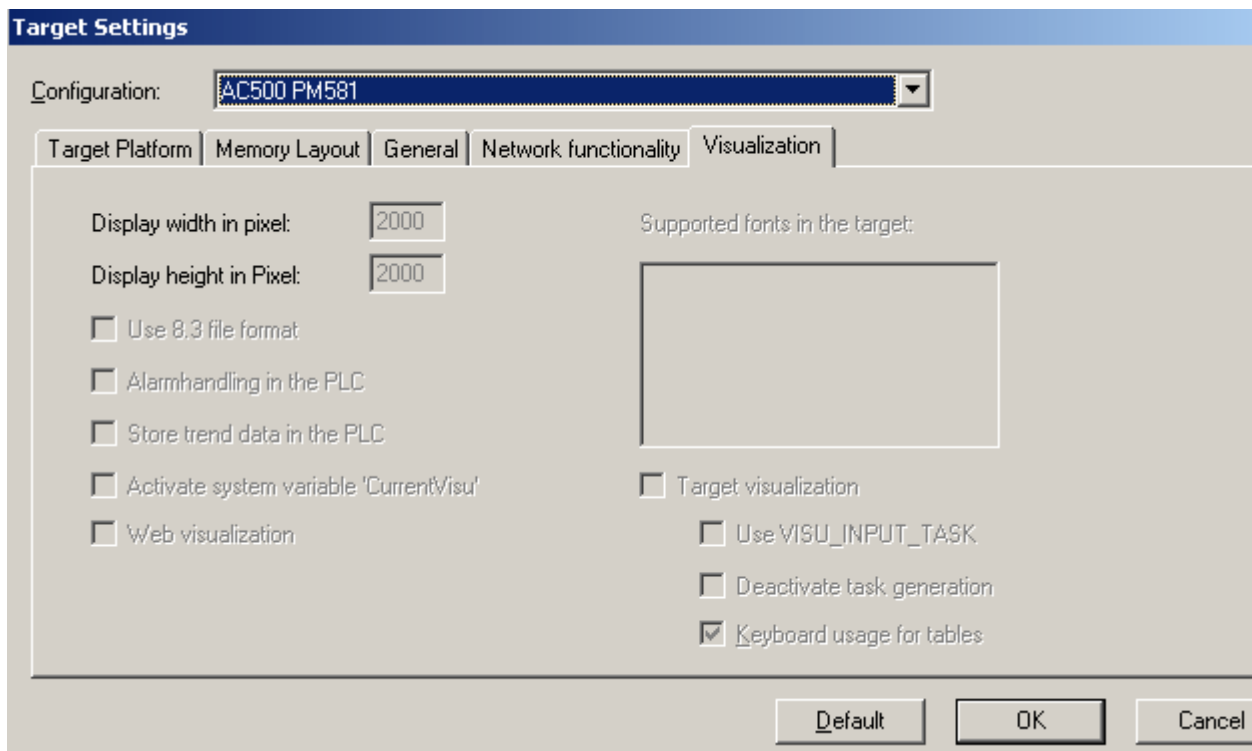
### 1.3.4 "Network Functionality" settings

The tab "Network functionality" contains the settings for the network variables. Network variables are not supported in the current version.



### 1.3.5 "Visualization" settings

The tab "Visualization" contains the settings for the visualization integrated in the PLC firmware. Target and WEB visualization are not supported in the current version.



## 1.4 Overview on user program size and operands of AC500 CPUs

The following table shows the values set for the program memory and the operands in AC500 targets:

Parameter / CPU	Unit Version	PM571	PM581	PM582	PM590	PM591
Available as of PS501 version		V1.0	V1.0	V1.1	V1.2	V1.1
Available as of firmware version		V1.0.2	V1.0.2	V1.1.7	V1.2.0	V1.1.7
User program (code) <b>see remark 1</b>	kB	64	256	512	2048 = 2 MB	4096 = 4 MB
Number of POU's		1024	1024	1024	4096	4096
Number of tasks		3	3	3	16	16
Floating point processor <b>see remark 2</b>		no	no	no	yes	yes
Global and local variables: VAR or VAR GLOBAL	kB	16	128	128	1024	2048
RETAIN area: VAR RETAIN or VAR RETAIN PERSISTENT <b>see remark 3</b>	kB up to V1.1.x	1	32	32	512	512
	kB V1.2.0 and later	4	32	32	512	512
Addressable flag area: VAR AT %Mx.y VAR RETAIN AT %Mx.y VAR RETAIN PERSISTENT AT %Mx.y	kB	4	128	128	512	512
PERSISTENT area: VAR AT %Rx.y	kB V1.2.0 and later	4	128	128	512	512
Inputs %I <b>see remark 4</b>	kB	24	24	24	24	24
Outputs %Q <b>see remark 4</b>	kB	24	24	24	24	24

### Remark 1:

The user program is composed of:

- the compiled code of all POU's called in the program,
- the initialization code for the variables and
- the code to restore the variables set as PERSISTENT (this does not include variables in the PERSISTENT area (%R area)!).

The configuration data are not considered in the user program size.

As of Control Builder version 1.2, the user program size (code size) is shown in the CoDeSys message box when compiling:

```
POU indexes: 262 (25%)
Size of data used: 1069 of 262144 bytes (0.41%)
Size of Retain data used: 0 of 32768 bytes (0.00%)
Code size: 6726 bytes
0 errors, 0 warnings
```

### Remark 2:

All AC500 CPUs can perform floating point operations. For CPUs without floating point processor (PM57x and PM58x), these operations are performed by an emulation library and are therefore slower. Emulation is faster for LREAL variables than for REAL variables. Thus, the use of LREAL variables is recommended.

### Remark 3:

The information shown in the message box does only contain the Retain data of the RETAIN area, and not the variables of the addressable flag area %Mx.y. that are declared as VAR RETAIN.



**Remark 4:**

The assignment of the inputs and outputs is described in detail in the chapter "AC500 inputs, outputs and flags".

## 1.5 Installation of AC500 targets with the program installTarget.exe

By default, all AC500 target files are installed when installing the Control Builder. When installing version V1.2, the target files of version V1.1 and the files of version V1.2 will be installed.

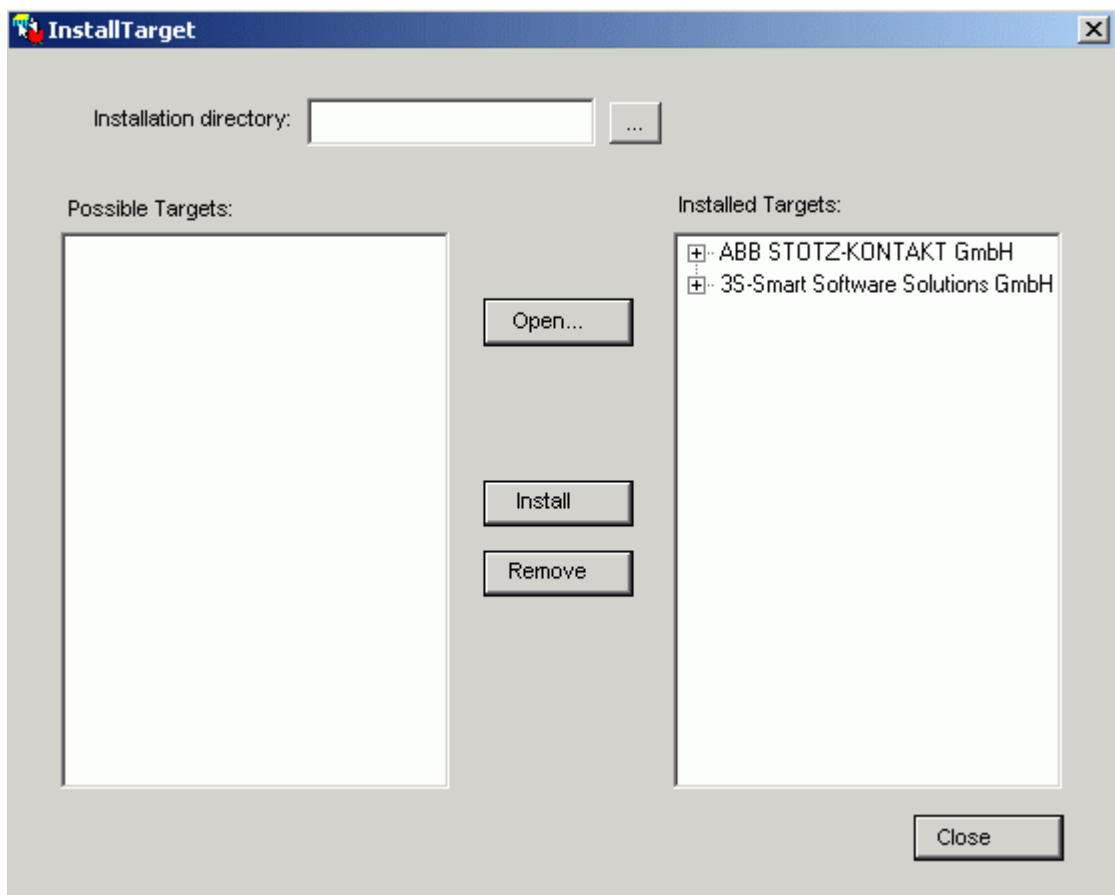
In certain cases the target files of several providers have to be installed on the PC at the same time. In this case, the target files of the AC500 CPUs can also be installed directly.



**Note:** The AC500 target of version V1.0 and V1.1 requires CoDeSys version V2.3.5.0 and later. Targets of version V1.2 require CoDeSys version V2.3.8.0 and later. AC500 targets cannot be used with CoDeSys V3.x.

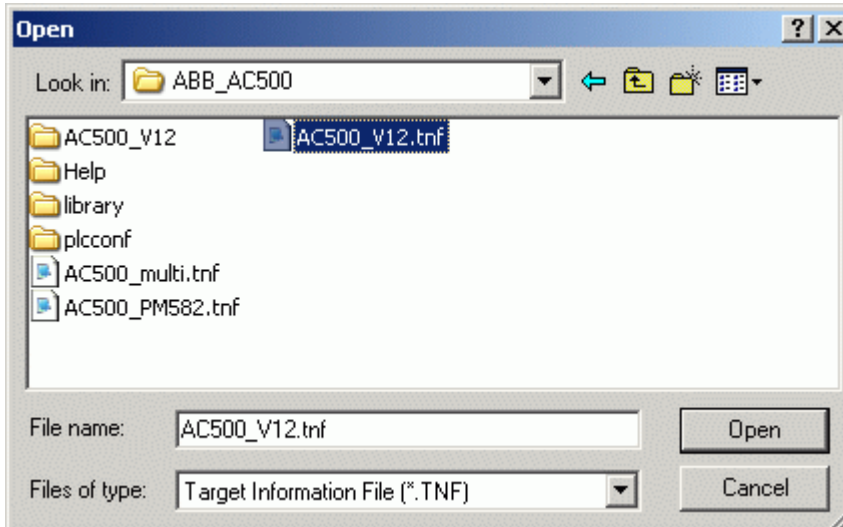
Proceed as follows to install the target:

1. Exit CoDeSys.
2. Exit the gateway, if necessary (right mouse click on the 3S icon in the Windows status bar => Exit).
3. Start the program InstallTarget.exe, for example by double clicking the file in the Windows Explorer. The program InstallTarget.exe is located in the installation directory of CoDeSys.exe, i.e., in case of a default installation in the directory ..\%ProgramFiles%\3S Software\CoDeSys V2.3.  
The environment variable %ProgramFiles% points to the directory  
- "Program Files" on English operating systems and  
- "Programme" on German operating systems.
4. The following user interface appears:



All targets currently installed (i.e., registered in the Windows registry) are displayed in the right-hand area "Installed targets".

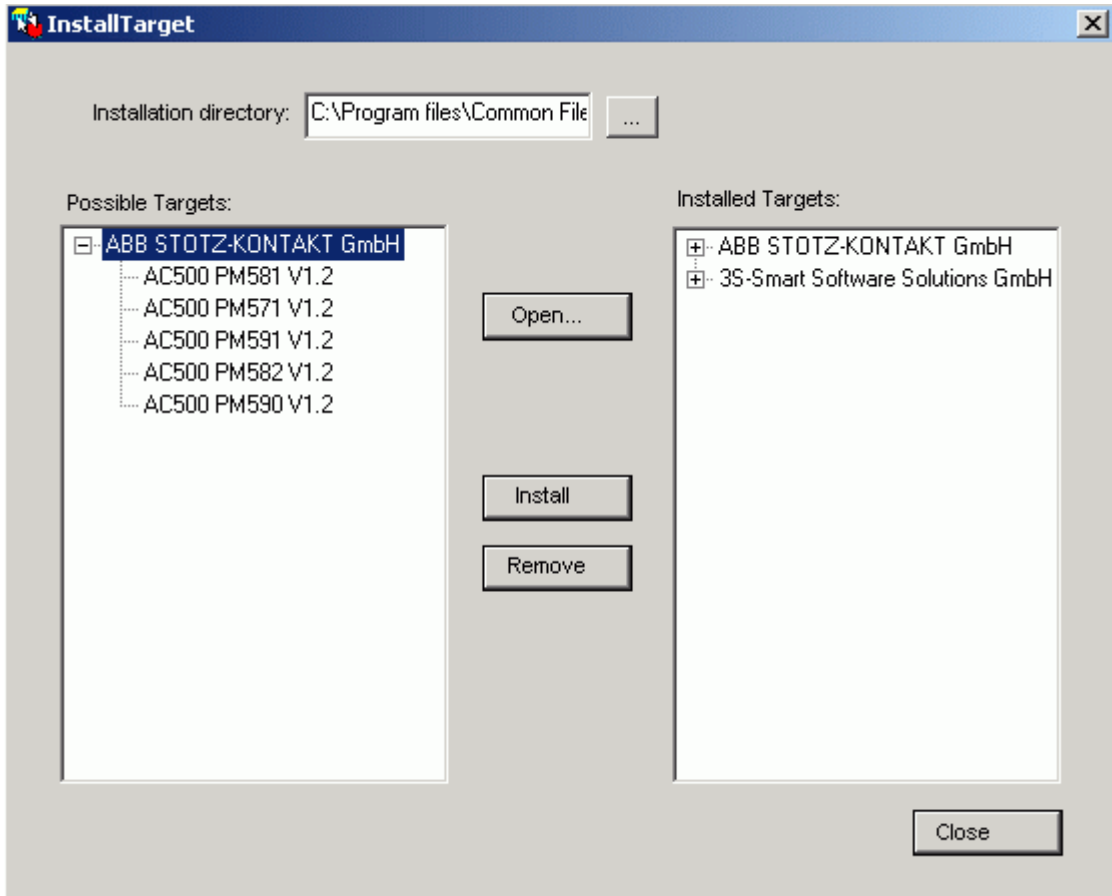
5. Click <Open> and select the directory that contains the installation file of the target(s) to be installed. The installation file of a target has the extension \*.tnf. The AC500 targets are located in the directory **CD-ROM drive:\CD\_AC500\CoDeSys\Targets\ABB\_AC500** on the CD PS501 Vx.y.
6. Select the desired installation file:



The following target installation files are available for the AC500 targets:  
AC500\_multi.tnf (PM571, PM581 and PM591 V1.1)  
AC500\_PM582.tnf (PM581 V1.1)  
AC500\_V12.tnf (PM571, PM581, PM582, PM590, PM591 V1.2)

Click <Open>.

7. In the left-hand area, all targets that are part of this installation file are displayed:



Select "ABB STOTZ-KONTAKT GmbH" or a specific target (for example AC500 PM581 V1.2) and click <Install>. All files belonging to the target will be installed. The "Installation directory" field shows the target directory.

8. To install further targets repeat steps 5 to 7.
9. Click <Close> to exit the program.
10. If you want to uninstall one or several targets, select the desired target(s) in the "Installed targets" area and click <Remove>.

**Remark:**

"Remove" does only delete the Windows registry entries. The files are still available in the installation directory. They have to be deleted manually, if necessary.

## 2 AC500 inputs, outputs and flags

All operands supported by CoDeSys are described in the Control Builder documentation. The documentation you are reading here describes in detail the "address" operands (%I for inputs and %Q for outputs) used in CoDeSys.

All addressable operands can be accessed bit-wise (X), byte-wise (B), word-wise (W) and double-word-wise (D) in the Control Builder. The Motorola byteorder is used for operand access.

### **Declaration of addressable operands:**

The declaration of the operands in the addressable flag area is done as follows:

**Symbol AT address : Type [:= initialization value]; (\* comment \*)**

[.] optional

The inputs and outputs are declared using the PLC configuration. Input and output devices that are directly coupled to the base unit are declared directly in the PLC configuration. Input and output devices connected to the coupler are configured using the field bus configurator SYCON.net which is part of the Control Builder (see topic Controller configuration with the Control Builder).

### **Caution:**

For multitasking, the digital inputs and outputs for every task are byte-wise cycle consistent, i.e., for instance inputs %IX0.0-%IX0.7 for task 1 and %IX1.0-%IX1.7 for task 2.

If, for example, task 1 has the higher priority and input %IX0.0 is used in task 1 and task 2, the value can change during the cycle of task 2 as it is updated every time task 1 is started.

This is not relevant for programs with only one task.

### 2.1 AC500 interfaces for inputs and outputs AC500

The following AC500 interfaces are available for inputs and outputs:

No.	Type	Designation	Number of inputs and outputs
<b>Configuration with CoDeSys PLC configuration (ConfConf)</b>			
1	I/O bus	Interface for I/O modules	Max 7 modules with a maximum of 32 channels (IX, QX, IW, QW) per module
2	COM1	CS31 bus master	Max 31 modules with a maximum of 32 channels per module, address 0-61
		Decentralized I/O expansion	RS-232 / RS-485 (version V2.0 and later)
3	COM2	Decentralized I/O expansion	RS-232 / RS-485 (version V2.0 and later)
4	FBP	FieldBusPlug - Slave	Max 8 modules with 16 IW + 16 QW + 16 IB + 16 QB with modular FBP, depending on fieldbus
5	Int. coupler	Internal coupler	ARCNET,.. (configuration without SYCON.net)
<b>Configuration with integrated SYCON.net</b>			
6	Line 0	Internal coupler	4 kB %I0.xx / %Q0.xx each
7	Line 1	Coupler 1	4 kB %I1.xx / %Q1.xx each
8	Line 2	Coupler 2	4 kB %I2.xx / %Q2.xx each
9	Line 3	Coupler 3	4 kB %I3.xx / %Q3.xx each
10	Line 4	Coupler 4	4 kB %I4.xx / %Q4.xx each

### 2.1.1 Address scheme for inputs and outputs

- The coupler I/Os are addressed as follows (two-stage process):

**%I(Q)BCouplerNumber.ByteCoupler**

The configuration is done using SYCON.net.

- No coupler numbers are assigned to I/Os that are connected to the CPU. These I/Os are configured with the PLC configuration (ConfConf) in the Control Builder.
- I/Os connected to the basic unit are assigned to the following address areas:

I/O bus:	%IB0 ..	%IB999	and %QB0 ..	%QB999
COM1:	%IB1000 ..	%IB1999	and %QB1000 ..	%QB1999
COM2 :	%IB2000 ..	%IB2999	and %QB2000 ..	%QB2999
FBP slave:	%IB3000 ..	%IB3999	and %QB3000 ..	%QB3999

- Addressing of the digital channels is done byte-oriented.
- Motorola byteorder is used to access the inputs and outputs.

### 2.1.2 Example for addressing in BOOL / BYTE / WORD / DWORD

The Motorola byteorder is used for addressing.

Address	Addr	Addr + 1	Addr + 2	Addr + 3
		16#xxxx x000	16#xxxx x001	16#xxxx x002
<b>BYTE</b>	<b>%IB0</b>	<b>%IB1</b>	<b>%IB2</b>	<b>%IB3</b>
<b>BOOL</b>	7 ... 0	7 ... 0	7 ... 0	7 ... 0
	%IX0.7 ... %IX0.0	%IX1.7 ... %IX1.0	%IX2.7 ... %IX2.0	%IX3.7 ... %IX3.0
<b>WORD</b>	<b>%IW0</b>		<b>%IW1</b>	
	15 ... 8	7 ... 0	15 ... 8	7 ... 0
<b>DWORD</b>	<b>%ID0</b>			
	31 ... 24	23 ... 16	15 ... 8	7 ... 0

Examples:

**%IX0.0** := TRUE  
 %IB0 := 1 := 16#01  
 %IW0 := 256 := 16#0100 (Bit 8 = TRUE)  
 %ID0 := 16777216 := 16#01000000 (Bit 24 = TRUE)

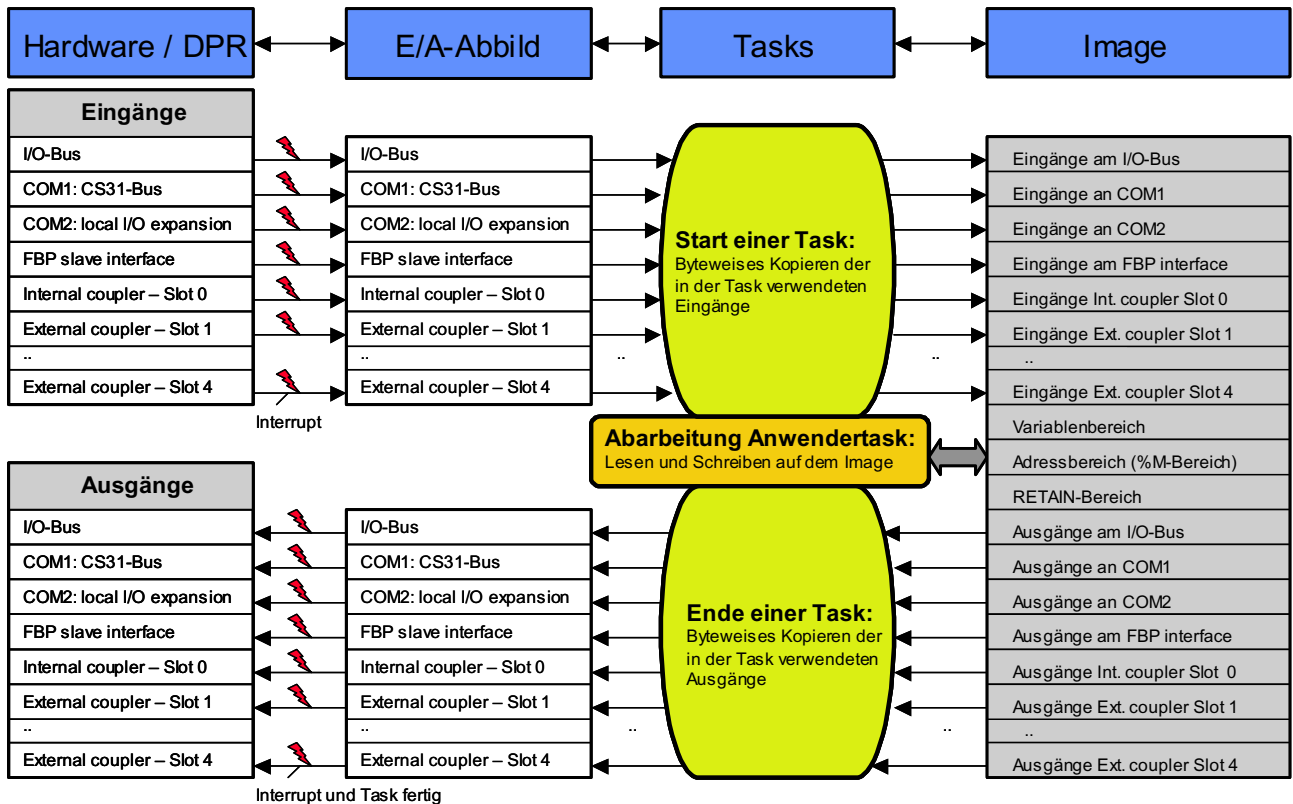
**%IX3.0** := TRUE  
 %IB3 := 1 := 16#01  
 %IW1 := 1 := 16#0001  
 %ID0 := 1 := 16#00000001

## 2.2 Addressing of inputs and outputs

No.	Device	Input/Output	Interface	Range	Addresses	
<b>Configuration with ConfConf (CPU I/Os) or SYCON.net (internal coupler)</b>						
0 ... 5	CPU I/Os and	Inputs (4kB)	C P U	I/O bus COM1 COM2 FBP	0000..0999 1000..1999 2000..2999 3000..4095	%IB0 ... %IB4095 %IW0 ... %IW2047 %ID0 ... %ID1023 %IX0.0 ... %IX4095.7
		Outputs (4kB)		I/O bus COM1 COM2 FBP	0000..0999 1000..1999 2000..2999 3000..4095	%QB0 ... %QB4095 %QW0 ... %QW2047 %QD0 ... %QD1023 %QX0.0 ... %QX4095.7
	Internal coupler	Inputs (4kB)	Line 0		0.0000 ... 0.4095	%IB0 ... %IB4095 %IW0 ... %IW2047 %ID0 ... %ID1023 %IX0.0 ... %IX4095.7
		Outputs (4kB)				%QB0 ... %QB4095 %QW0 ... %QW2047 %QD0 ... %QD1023 %QX0.0 ... %QX4095.7
<b>Configuration with SYCON.net</b>						
6	Coupler 1	Inputs (4kB)	Line 1		1.0000 ... 1.4095	%IB1.0 ... %IB1.4095 %IW1.0 ... %IW1.2047 %ID1.0 ... %ID1.1023 %IX1.0.0 ... %IX1.4095.7
		Outputs (4kB)				%QB1.0 ... %QB1.4095 %QW1.0 ... %QW1.2047 %QD1.0 ... %QD1.1023 %QX1.0.0 ... %QX1.4095.7
...						
9	Coupler 4	Inputs (4kB)	Line 4		4.0000 ... 4.4095	%IB4.0 ... %IB4.4095 %IW4.0 ... %IW4.2047 %ID4.0 ... %ID4.1023 %IX4.0.0 ... %IX4.4095.7
		Outputs (4kB)				%QB4.0 ... %QB4.4095 %QW4.0 ... %QW4.2047 %QD4.0 ... %QD4.1023 %QX4.0.0 ... %QX4.4095.7

## 2.3 Processing of inputs and outputs in the multitasking system

The following figure shows how the inputs and outputs are processed in the multitasking system.



### Generation of the input data image:

#### Inputs at the I/O bus:

After all I/O modules have been processed at the I/O bus, a corresponding interrupt is generated in the processor. The inputs are copied to the input data image during the Interrupt Service Routine (ISR). If the outputs were updated by a task, the outputs in the output data image are copied.

#### Inputs at the CS31 system bus:

After the CS31 driver has processed all I/O modules, a corresponding interrupt is generated in the processor. The inputs are copied to the input data image during the Interrupt Service Routine (ISR). If the outputs were updated by a task, the outputs in the output data image are copied.

#### Inputs of couplers line 0 to 4:

Once a coupler has received new data, a corresponding interrupt is generated in the processor. The inputs are copied from the DPR to the input data image of the processor during the Interrupt Service Routine (ISR). If the outputs were updated by a task, the outputs in the output data image are copied to the DPR.

Precondition for this is a valid coupler configuration.

### Starting a task:

When starting a task, **the inputs used in the task are copied byte-wise** from the input data image to the image. Byte-wise means that when using, for example, the input %IX0.0, the image of the inputs %IX0.0 ... IX0.7 will be copied to the image.

Because only those inputs are copied that are directly used in the task, it is not possible to **read the inputs indirectly**, if cycle consistency is required.

### Processing a task:

All tasks access the image, i.e., inputs are read from the image and outputs are written to the image. In ONLINE mode, the inputs/outputs of the image are displayed.

### Termination of processing the output data image by a task:

At the end of the task processing, the outputs used in the task are copied byte-wise from the image to the output data image. Byte-wise means that when using, for example, the output %QX0.0, the image of the outputs %QX0.0 ... QX0.7 will be copied from the image to the output data image. The internal variables "Output data image updated" for the CS31 processor and the couplers 0 .. 4 will be set.

### Writing the outputs:

#### Outputs at the I/O bus:

With the next interrupt of the I/O bus driver, the outputs of the output data image will be written and the variable "Output data image updated" will be reset.

#### Outputs at the CS31 system bus:

With the next interrupt of the CS31 processor, the outputs of the output data image will be written and the variable "Output data image updated" will be reset.

#### Outputs of the coupler line 0 to 4:

With the next interrupt of the coupler, the outputs of the output data image will be written to the DPR and the variable "Output data image updated" will be reset.

### I/O update task:

In order to update the inputs/outputs not used in the task, all inputs/outputs of the image are updated by a lower priority task (I/O update task). This task is only processed if no other user task runs.

## 2.4 Addressable flag area (%M area) in the AC500

### 2.4.1 Allocation of the addressable flag area in the AC500

The addressable flag area for the AC500 is divided into several segments with a size of 64 kbytes per segment. A maximum of 8 segments can be addressed. The availability of the segments or partial segments depends on the CPU. The size of the %M area can be found in the technical data of the CPUs (see Technical data of the CPU) and in the target system settings (see Target Support Package).

Segment	Operands	Size, cumulative [kB]	CPU PM57x	CPU PM58x	CPU PM59x
0	%MB0.0...%MB0.65535	64	4 kB	+	+
1	%MB1.0...%MB1.65535	128	-	+	+
2	%MB2.0...%MB2.65535	192	-	-	+
3	%MB3.0...%MB3.65535	256	-	-	+
4	%MB4.0...%MB4.65535	320	-	-	+
5	%MB5.0...%MB5.65535	284	-	-	+
6	%MB6.0...%MB6.65535	448	-	-	+
7	%MB7.0...%MB7.65535	512	-	-	+



## 2.4.2 Access to the %M area using the Modbus® Protocol

The Modbus® RTU protocol is implemented in the AC500. With the help of the Modbus® protocol, the segments 0 and 1 of the addressable flag area can be accessed.

The chapter Modbus in this documentation contains a detailed description of the Modbus® protocol and the corresponding addressing (see also Modbus protocol).



**Note:** For the AC500 CPU PM571, 4kB = %MB0.0 .. %MB0.4095 (i.e., not a complete segment) are available for the addressable flag area. Thus, not all Modbus addresses can be accessed.

## 2.4.3 Access to operands in the addressable flag area

The operands in the %M area can be accessed bit-wise, byte-wise, word-wise and double-word-wise.

Byte SINT / BYTE	Bit (byte-oriented) BOOL	Word INT / WORD	Double word DINT / DWORD
<b>Segment 0</b>			
%MB0.0	%MX0.0.0 ... %MX0.0.7	%MW0.0	%MD0.0
%MB0.1	%MX0.1.0 ... %MX0.1.7		
%MB0.2	%MX0.2.0 ... %MX0.2.7		
%MB0.3	%MX0.3.0 ... %MX0.3.7		
...	...	...	...
%MB0.65532	%MX0.65532.0 ... %MX0.65532.7	%MW0.32766	%MD0.16383
%MB0.65533	%MX0.65533.0 ... %MX0.65533.7		
%MB0.65534	%MX0.65534.0 ... %MX0.65534.7	%MW0.32767	
%MB0.65535	%MX0.65535.0 ... %MX0.65535.7		
<b>Segment 1</b>			
%MB1.0	%MX1.0.0 ... %MX1.0.7	%MW1.0	%MD1.0
%MB1.1	%MX1.1.0 ... %MX1.1.7		
%MB1.2	%MX1.2.0 ... %MX1.2.7	%MW1.1	
%MB1.3	%MX1.3.0 ... %MX1.3.7		
...	...	...	...
%MB1.65532	%MX1.65532.0 ... %MX1.65532.7	%MW1.32766	%MD1.16383
%MB1.65533	%MX1.65533.0 ... %MX1.65533.7		
%MB1.65534	%MX1.65534.0 ... %MX1.65534.7	%MW1.32767	
%MB1.65535	%MX1.65535.0 ... %MX1.65535.7		
<b>Segment 2</b>			
%MB2.0	%MX2.0.0 ... %MX2.0.7	%MW2.0	%MD2.0
%MB2.1	%MX2.1.0 ... %MX2.1.7		
%MB2.2	%MX2.2.0 ... %MX2.2.7	%MW2.1	
%MB2.3	%MX2.3.0 ... %MX2.3.7		
...	...	...	...
%MB2.65532	%MX2.65532.0 ... %MX2.65532.7	%MW2.32766	%MD2.16383
%MB2.65533	%MX2.65533.0 ... %MX2.65533.7		
%MB2.65534	%MX2.65534.0 ... %MX2.65534.7	%MW2.32767	
%MB2.65535	%MX2.65535.0 ... %MX2.65535.7		
...	...	...	...
<b>Segment 7</b>			
%MB7.0	%MX7.0.0 ... %MX7.0.7	%MW7.0	%MD7.0
%MB7.1	%MX7.1.0 ... %MX7.1.7		
%MB7.2	%MX7.2.0 ... %MX7.2.7	%MW7.1	
%MB7.3	%MX7.3.0 ... %MX7.3.7		
...	...	...	...
%MB7.65532	%MX7.65532.0 ... %MX7.65532.7	%MW7.32766	%MD7.16383
%MB7.65533	%MX7.65533.0 ... %MX7.65533.7		
%MB7.65534	%MX7.65534.0 ... %MX7.65534.7	%MW7.32767	
%MB7.65535	%MX7.65535.0 ... %MX7.65535.7		

## 2.5 Absolute addresses of operands

### 2.5.1 Address operator ADR

For particular blocks or in case of accessing operands via pointers, the absolute address of an operand must be determined. To do this, the Control Builder provides the address operator ADR.

The address operator ADR is described in the documentation for the Control Builder (see CoDeSys Documentation / ADR operator). The documentation you are reading here describes only the peculiarities of bit operands.

The addresses provided by the address operator can be used as inputs for blocks that require absolute addresses (such as xxx\_MOD\_MAST, COM\_SND). If these blocks shall be applied to internal variables, it must be guaranteed that the variables are set to successive addresses. This is achieved by declaring ARRAYS and STRINGS.

The address operator ADR provides the address of an operand in one double word DWORD (i.e., 32 bits). The address operator returns the address of the first byte of a variable (byte address). For the user-definable variables, variables of the type BOOL are stored as byte.

### 2.5.2 Bit address operator BITADR

For inputs, outputs and variables of the addressable flag area (%M area) or addressable PERSISTENT area (%R area), operands of the type BOOL occupy one bit. The address of this type of variables cannot be determined with the operator ADR.

When processing the statement:

```
dwAddress := ADR(%MX0.0.0);
```

the following error message appears:

**Error 4031:**  
**PLC\_PRG(xx): ADR is not allowed for bits! Use BITADR instead.**

BITADR returns the bit offset within the area %I, %Q or %M as DWORD.

The following table shows the position of the operands within the memory (considering %MD0.0 and %MD0.1 as example). Here you get information about which addresses the operator ADR returns and which offsets BITADR returns.



**Note:** The addresses shown are example addresses and thus can have other values.

Position of operands within memory and values of operators ADR and BITADR:

Byte SINT / BYTE	Word INT / WORD	Double word DINT / DWORD	Bit (byte-oriented) BOOL	ADR	BITADR
%MB0.0	%MW0.0	%MD0.0	%MX0.0.0	16#08000000	8
			%MX0.0.1		9
			%MX0.0.2		10
			%MX0.0.3		11
			%MX0.0.4		12
			%MX0.0.5		13
			%MX0.0.6		14
%MB0.1	%MW0.0		%MX0.0.7	16#08000001	15
			%MX0.1.0		0
			%MX0.1.1		1
			%MX0.1.2		2
			%MX0.1.3		3
			%MX0.1.4		4
			%MX0.1.5		5
%MB0.2	%MW0.1		%MX0.1.6	16#08000002	6
			%MX0.1.7		7
			%MX0.2.0		24
			%MX0.2.1		25
			%MX0.2.2		26
			%MX0.2.3		27
			%MX0.2.4		28
%MB0.3	%MW0.1		%MX0.2.5	16#08000003	29
			%MX0.2.6		30
			%MX0.2.7		31
			%MX0.3.0		16
			%MX0.3.1		17
			%MX0.3.2		18
			%MX0.3.3		19
%MB0.4	%MW0.2		%MX0.3.4	16#08000004	20
			%MX0.3.5		21
			%MX0.3.6		22
%MB0.5	%MW0.2		%MX0.3.7	16#08000005	23
			%MX0.4.0		40
%MB0.6	%MW0.3		..	16#08000006	..
			%MX0.4.7		47
			%MX0.5.0		32
%MB0.7	%MW0.3		..	16#08000007	..
			%MX0.5.7		39
			%MX0.6.0		56
..	16#08000007		..	..	
%MX0.6.7			63		
..	16#08000007		%MX0.7.0	48	
%MX0.7.7			55		

## 2.6 Addressable PERSISTENT area (%R area) in the AC500

### 2.6.1 Special features of the addressable PERSISTENT area in the AC500

As of Control Builder version V1.2 and CPU firmware V1.2.0, the new operand area "addressable PERSISTENT area" or %R area is available.



**Caution:** The %R area is only available in combination with Control Builder version V1.2 and later and AC500 firmware version V1.2.0 and later. For the %R area the following is relevant for Control Builder V1.2:

**CoDeSys.exe V2.3.8.0** or higher (shown under CoDeSys / Info)

**AC500 Target PM5xx\_V12** or higher (shown under InstallTarget / Installed targets)

The addressable PERSISTENT area or %R area has the following **peculiarities**:

1. Variables declared in the %R area are always located at the same position in the PLC's operand memory because they have addresses assigned (like the variables in the %R area).
2. Variables in the %R area are declared as follows:



**VAR** (**Caution: noRETAIN or PERSISTENT option**)

**Symbol AT %RTypeSegment.Offset : TYPE; (\* Comment \*)** or also

**aSymbol AT %RTypeSegment.Offset : ARRAY[start..end] OF TYPE; (\* Comment \*)**

**END\_VAR**

where:

Symbol	- symbolic name of the variable
Type	- X=BOOL (Bit), B=BYTE, W=WORD, D=DWORD
Segment	- 0..7 (availability depends on CPU type)
Offset	- 0..65535 (availability depends on CPU type)
TYPE	- BOOL, BYTE, WORD, DWORD or defined type (such as structure)
start	- Index of the first ARRAY element
end	- Index of the last ARRAY element

3. For each segment in the %R area, an area can be set in the PLC configuration which is buffered in case the **battery is installed and fully charged**. In this case, the variables behave like variables declared as VAR RETAIN PERSISTENT, i.e.,  
-> they keep their values even after
  - Online changes (like VAR RETAIN)
  - Voltage OFF/ON (like VAR RETAIN)
  - a download (like VAR PERSISTENT)
4. In contrast to the variables declared as PERSISTENT, these variables have the great advantage that no program code is required for dumping the variables during a download.
5. The buffered part of the %R area can be written to the SD card and read from the card (see chapter "Saving the buffered data of the %R area").

## 2.6.2 Segmentation of the addressable PERSISTENT area in the AC500

The addressable PERSISTENT area in the AC500 is divided into several segments with a size of 64 kbytes per segment. A maximum of 8 segments can be addressed. The availability of the segments or partial segments depends on the CPU:

Segment	Operands	Size, cumulative [kB]	CPU PM57x	CPU PM58x	CPU PM59x
0	%RB0.0...%RB0.65535	64	4 kB	+	+
1	%RB1.0...%RB1.65535	128	-	+	+
2	%RB2.0...%RB2.65535	192	-	-	+
3	%RB3.0...%RB3.65535	256	-	-	+
4	%RB4.0...%RB4.65535	320	-	-	+
5	%RB5.0...%RB5.65535	284	-	-	+
6	%RB6.0...%RB6.65535	448	-	-	+
7	%RB7.0...%RB7.65535	512	-	-	+

## 2.6.3 Saving the buffered data of the AC500's %R area

The buffered part of the %R area can be saved on the SD card and read from the card. This can be necessary, if, for example, the controller has to be replaced.

Saving data is done in two steps:

1. Copying the data from the %R area and writing it to the CPU's RAM disk as file
2. Saving the file to the SD card.

Reading data from the SD card is also done in two steps:

1. Loading the file from the SD card to the CPU's RAM disk.
2. Copying the data from the RAM disk to the %R area.

Saving and reading the data can be done using function blocks in the user program or with the PLC Browser contained in the Control Builder. The function blocks are contained in the library SysInt\_AC500\_V10.LIB.

Function	PLC Browser command	Function block
Copy from %R area to RAM disk	persistent save	PERSISTENT_SAVE
Save file to SD card	persistent export	PERSISTENT_EXPORT
Read file from SD card to RAM disk	persistent import	PERSISTENT_IMPORT
Copy data from RAM disk to %R area	persistent restore	PERSISTENT_RESTORE
Delete buffered data of the PERSISTENT area	persistent clear	PERSISTENT_CLEAR



**Caution:** If cycle consistency is required for the data, this has to be implemented in the user program. That means that the data may not be changed during copying to/from the %R area from/to the RAM disk.

If saving is done using the PLC Browser, this can be easily carried out by stopping the user program.



**Caution:** Copying the PERSISTENT area takes some milliseconds (see the following table). Thus, an according cycle time has to be set in the task configuration. Please note the remarks on the task configuration!

Action	Time in ms		
	CPU PM57x	CPU PM58x	CPU PM59x
<b>Restoring 1 kB (1024 bytes)</b>			
PERSISTENT_CLEAR	< 1	< 1	< 1
PERSISTENT_SAVE	2	2	2
PERSISTENT_EXPORT	1000	1000	500
PERSISTENT_IMPORT	500	1000	500
PERSISTENT_RESTORE	2	< 1	1
<b>Restoring 4 kB (4096 bytes)</b>			
PERSISTENT_CLEAR	< 1	< 1	< 1
PERSISTENT_SAVE	2	3	2
PERSISTENT_EXPORT	1000	1000	500
PERSISTENT_IMPORT	500	1000	500
PERSISTENT_RESTORE	3	3	2
<b>Restoring 64 kB (65536 bytes)</b>			
PERSISTENT_CLEAR	not possible	8	2
PERSISTENT_SAVE	not possible	11	6
PERSISTENT_EXPORT	not possible	2500	1000
PERSISTENT_IMPORT	not possible	2000	500
PERSISTENT_RESTORE	not possible	12	5
<b>Restoring max. PERSISTENT area</b>			
	<b>4 kB</b>	<b>128 kB</b>	<b>512 kB</b>
PERSISTENT_CLEAR	< 1	17	22
PERSISTENT_SAVE	2	22	35
PERSISTENT_EXPORT	1000	4000	8000
PERSISTENT_IMPORT	500	3000	4000
PERSISTENT_RESTORE	3	22	31

## 2.6.4 Access to operands in the addressable PERSISTENT area (%R area)

The operands in the %R area can be accessed bit-wise, byte-wise, word-wise and double-word-wise.

Byte SINT / BYTE	Bit (byte-oriented) BOOL	Word INT / WORD	Double word DINT / DWORD
<b>Segment 0</b>			
%RB0.0	%RX0.0.0...%RX0.0.7	%RW0.0	%RD0.0
%RB0.1	%RX0.1.0...%RX0.1.7		
%RB0.2	%RX0.2.0...%RX0.2.7	%RW0.1	
%RB0.3	%RX0.3.0...%RX0.3.7		
%RB0.65532	%RX0.65532.0...%RX0.65532.7	%RW0.32766	%RD0.16383
%RB0.65533	%RX0.65533.0...%RX0.65533.7		
%RB0.65534	%RX0.65534.0...%RX0.65534.7	%RW0.32767	
%RB0.65535	%RX0.65535.0...%RX0.65535.7		
<b>Segment 1</b>			
%RB1.0	%RX1.0.0...%RX1.0.7	%RW1.0	%RD1.0
%RB1.1	%RX1.1.0...%RX1.1.7		
%RB1.2	%RX1.2.0...%RX1.2.7	%RW1.1	
%RB1.3	%RX1.3.0...%RX1.3.7		
%RB1.65532	%RX1.65532.0...%RX1.65532.7	%RW1.32766	%RD1.16383
%RB1.65533	%RX1.65533.0...%RX1.65533.7		
%RB1.65534	%RX1.65534.0...%RX1.65534.7	%RW1.32767	
%RB1.65535	%RX1.65535.0...%RX1.65535.7		
<b>Segment 2</b>			
%RB2.0	%RX2.0.0...%RX2.0.7	%RW2.0	%RD2.0
%RB2.1	%RX2.1.0...%RX2.1.7		
%RB2.2	%RX2.2.0...%RX2.2.7	%RW2.1	
%RB2.3	%RX2.3.0...%RX2.3.7		
%RB2.65532	%RX2.65532.0...%RX2.65532.7	%RW2.32766	%RD2.16383
%RB2.65533	%RX2.65533.0...%RX2.65533.7		
%RB2.65534	%RX2.65534.0...%RX2.65534.7	%RW2.32767	
%RB2.65535	%RX2.65535.0...%RX2.65535.7		
<b>Segment 7</b>			
%RB7.0	%RX7.0.0...%RX7.0.7	%RW7.0	%RD7.0
%RB7.1	%RX7.1.0...%RX7.1.7		
%RB7.2	%RX7.2.0...%RX7.2.7	%RW7.1	
%RB7.3	%RX7.3.0...%RX7.3.7		
%RB7.65532	%RX7.65532.0...%RX7.65532.7	%RW7.32766	%RD7.16383
%RB7.65533	%RX7.65533.0...%RX7.65533.7		
%RB7.65534	%RX7.65534.0...%RX7.65534.7	%RW7.32767	
%RB7.65535	%RX7.65535.0...%RX7.65535.7		



**Note:** Only the first 4 kB in segment 0 are available for PM57x, i.e., %RB0.0..%RB0.4095 or %RW0.0..%RW0.2047 or %RD0.0..%RD0.1023.

## 3 The AC500 PLC configuration

### 3.1 Overview on the PLC configuration

#### 3.1.1 PLC configuration functions

The general operation of the PLC configuration is described in detail in the Control Builder documentation (see CoDeSys Documentation / PLC Configuration). This section describes the configuration of the AC500.

The PLC configuration describes the hardware of the project. This way, the following data can be made available in the project:

- General parameters of the AC500 CPU
- Inputs and outputs of all modules connected to the I/O bus
- Symbolic names and comments for the inputs and outputs
- Parameters of the input and output modules and the assigned I/O channels, if available
- Mode and parameter settings for the serial interfaces
- Inputs and outputs of all input/output modules connected to the serial interface COM1 in CS31 mode
- Coupler type, general parameters and logs of the installed couplers
- All required system libraries are automatically loaded according to the configuration when building the project (started by pressing <F11>)
- Creation of a new database for exporting and importing configuration data in XML data format

The PLC configuration for the AC500 allows to load a project into all AC500 CPUs (PM571, PM581, PM591, ..). In order to download a project to another CPU, the desired CPU has to be selected in the target settings. It is only necessary to change the PLC configuration, if the hardware structure of the PLC has changed, i.e., if, for example, other couplers are installed.

The PLC configuration allows to export and import the complete configuration or parts of it. Thus, it is for example possible to store previously edited symbolic names of the inputs/outputs of an I/O module on the PC and to import them into other projects. This is also possible for the interface settings (see also chapter "Export and import of configuration data").

#### 3.1.2 Export and import of configuration data

Exporting a module, all interface data or a complete configuration is done by selecting the desired element in the PLC configuration, opening the context menu by right-clicking the element and selecting the menu item "Export module". In the appearing window, the module can be saved to the corresponding configuration level as XML file under the selected module name.

Importing modules is done in the same way than exporting. That means, by right-clicking the desired module in the configuration tree and selecting the menu item "Import module" from the context menu.



**Note:** Modules can only be imported into the configuration level from which they were previously exported.

In case of I/O modules (such as I/O bus, CS31 bus), the position of the module can be ignored. Thus, a module installed in slot 1 can be exported and imported to slot 5. The input and output addresses are changed automatically.



Example of an export/import procedure:

### Example 1:

The input/output module DC532 at the I/O bus for machine part Axx shall be saved as DC532\_Axx and then be used in another project.

1. Append a DC532 module to the I/O bus by right-clicking the I/O bus element in the configuration tree and selecting the menu item "Append Subelement" / "DC532 - 16 digital input and 16 digital Inoutput" from the context menu.
2. Edit the symbolic names and comments of the inputs/outputs.
3. Export the DC532 module by right-clicking it in the configuration tree and selecting the menu item "Export module" from the context menu. In the appearing window, enter the file name "DC532\_Axx" and confirm the window.  
Now the file "DC532\_Axx.xml" is saved to the directory Compile which is a subdirectory of the Control Builder installation directory.
4. If necessary, save the project and open/create the project into which you want to insert the exported module.
5. Append a DC532 module to the I/O bus of this project.
6. Right-click the module DC532 in the configuration tree, select the context menu item "Import module" and then choose the file "DC532\_Axx" (contains the exported module) in the appearing window. The module will be inserted and the symbolic names and comments are available in the projects afterwards.

### Example 2:

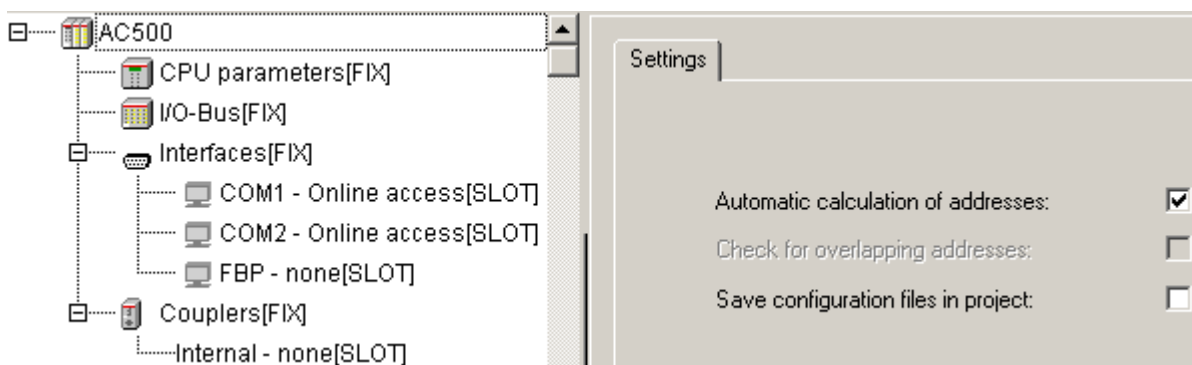
The Modbus RTU settings for serial interface COM1 shall be saved as "COM1\_MODBUS\_Slave1".

1. Change the COM1 parameters to "COM1 - MODBUS" by right-clicking the element in the configuration tree and selecting "Replace element" / "COM1 - MODBUS".
2. Configure the interface parameters in the "Module parameters" tab.
3. Export the new settings by right-clicking COM1 in the configuration tree and selecting the menu item "Export module" from the context menu. In the appearing window, enter the file name "COM1\_MODBUS\_Slave1" and confirm the window.  
Now the file "COM1\_MODBUS\_Slave1.xml" is saved to the directory Compile of the Control Builder.

### 3.1.3 Default settings in the PLC configuration

Once you have selected the AC500 CPU in the target settings, the CPU can be configured. To do this, select the object "PLC Configuration" in the "Resources" tab.

In case of a new AC500 project, the PLC configuration contains the following default settings:



**Note:** These default settings can be restored at any time by selecting "Extras" / "Standard configuration". **Do not change the default settings under "Settings"!** As of PS501 version V1.2, the parameter "Automatic calculation for addresses" is no longer editable.

The parameters represent the interfaces of the AC500 controllers. Each interface can be configured.

The individual parameters are used to configure the following elements:

Element	Configuration
CPU parameters	CPU parameters.
I/O bus	Input/output modules that are directly connected to the CPU.
Interfaces	Serial interfaces COM1 and COM2 and FBP slave interface.
Couplers	Parameters and protocols of the internal coupler and the external couplers. The real coupler configuration (ARCNET excluded) and the configuration of the connected input/output modules is done with the integrated fieldbus configurator SYCON.net (see also fieldbus configuration with SYCON.net).

The PLC configuration is based on the configuration files (\*.cfg) installed with the TSP.

### 3.1.4 Setting parameters in the PLC configuration

For all windows containing module parameters the following basic rules apply:

- All visible parameters of the configuration file are displayed. Only the values in the column Value can be edited.
- **Index:** The Index is a consecutive number (i) for the parameters within a module.
- **Name:** Name of the parameter.
- **Value :** Value of the parameter, editable. The default value is displayed initially. Values can be set directly or by means of symbolic names. If the entries in the configuration file are not set to 'Read Only', they can be edited. To edit a value, click on the edit field or select one of the entries from the scroll list.
- **Default:** Default value of the parameter.
- **Min.:** Minimum value of the parameter (applies only if no symbolic names are used).
- **Max.:** Maximum value of the parameter (applies only if no symbolic names are used).

A **tooltip** may give additional information on the currently selected parameter. This information is displayed according to the language setting for the Control Builder.

## 3.2 Configuration of CPU parameters

### 3.2.1 CPU parameters in PS501 versions V1.0 and V1.1

Selecting "CPU parameters" in the configuration tree opens the configuration window as shown as follows:

Index	Name	Value	Defa...
1	Auto run	On	On
2	Error LED	On	On
3	Check Battery	On	On
4	Behavior of outpu...	Off in hard...	Off in ...
5	Stop on error class	No effect	No eff...
6	Warmstart on E2	Off	Off

The following parameters can be set:

Parameter	Default value	Value	Meaning
Auto run / Start of user program when voltage ON <b>see remark 1</b>	On	On	If the Flash memory contains a valid project, the project will be loaded into the RAM memory and executed when switching on the controller.
		Off	If the Flash memory contains a valid project, this project will be loaded into the RAM memory but not executed when switching on the controller.
Error LED	On	On	The error LED lights up for errors of all classes.
		Off_by_E4	Warnings (E4) are not indicated by the error LED.
		Off_by_E3	Warnings (E4) and light errors (E3) are not indicated by the error LED.
Check Battery	On	On	The availability of the battery and the battery status are checked. If no battery is available or the battery is empty, a warning (E4) is generated and the LED ERR lights up.
		Off	The battery is not checked. No warning (E4) is generated. This also applies if a battery is installed but empty!
Behaviour of outputs in stop	Off in hardware and online	Off in hardware and online	In case of STOP, all outputs at the hardware and in the online display are set to FALSE or 0.
		Off in hardware and actual state online	In case of STOP, all outputs at the hardware are set to FALSE or 0. The online display indicates the status from the last cycle of the user program.
		Actual state in hardware and online	The status of the last cycle of the user program is kept for the outputs at the hardware and in the online display.
Stop on error class	No effect	No effect	In case of an error, the user program is not stopped.
		E1	In case of a fatal error (E1), the user program is stopped.
		E2	In case of a fatal or serious error (E1-E2), the user program is stopped.
		E3	In case of a fatal, serious or light error (E1-E3), the user program is stopped.
		E4	In case of a fatal, serious or light error (E1-E3) or a warning (E4), the user program is stopped.
Warmstart on E2	Off	Off	In case of a fatal error (E2), no warmstart is performed.
		On	In case of a fatal error (E2), a warmstart is performed automatically. (V1.2.0 and higher)

**Remark 1: Setting the parameters Auto run and MOD using the display/keypad**

Loading and running the user program also depends on the **setting for the parameter MOD using the display/keypad**. The display/keypad setting always has the higher priority.

The following applies:

MOD 00:	The user program will be loaded and run according to the setting for the CPU parameter "Auto run" (default setting).
MOD 01:	User program will not be loaded/run.
MOD 02:	The user program will be loaded and run independent of the setting for the CPU parameter "Auto run".

Keeping the "RUN" pushbutton pressed when booting the PLC automatically activates MOD 01, i.e., the user program is not loaded/run. Thus, it is possible to boot the PLC in Stop status. This may be required if, for example, both serial interfaces are set to Modbus and therefore no access with the Control Builder software is possible via the serial interface.

### 3.2.2 CPU parameters in version PS501 V1.2

The CPU parameters have been revised and expanded for PS501 version V1.2:

Index	Name	Value	Default	Min.	Max.
1	Auto run	On	On		
2	Error LED	On	On		
3	Check Battery	On	On		
4	Behavior of outputs in stop	Off in ha...	Off in ha...		
5	Stop on error class	E2	E2		
6	Warmstart	Off	Off		
7	Reaction on floatingpoint exception	E2 failure	E2 failure		
10	Start PERSISTENT %R0.x	0	0	0	65535
11	End PERSISTENT %R0.x	0	0	0	65535
12	Start PERSISTENT %R1.x	0	0	0	65535
13	End PERSISTENT %R1.x	0	0	0	65535
14	Start PERSISTENT %R2.x	0	0	0	65535
15	End PERSISTENT %R2.x	0	0	0	65535
16	Start PERSISTENT %R3.x	0	0	0	65535
17	End PERSISTENT %R3.x	0	0	0	65535
18	Start PERSISTENT %R4.x	0	0	0	65535
19	End PERSISTENT %R4.x	0	0	0	65535
20	Start PERSISTENT %R5.x	0	0	0	65535
21	End PERSISTENT %R5.x	0	0	0	65535
22	Start PERSISTENT %R6.x	0	0	0	65535
23	End PERSISTENT %R6.x	0	0	0	65535
24	Start PERSISTENT %R7.x	0	0	0	65535
25	End PERSISTENT %R7.x	0	0	0	65535

The following parameters can be set:

Parameter	Default value	Value	Meaning
Auto run / Start of user program when voltage ON <b>see remark 1</b>	On	On	If the Flash memory contains a valid project, the project will be loaded into the RAM memory and executed when switching on the controller.
		Off	If the Flash memory contains a valid project, this project will be loaded into the RAM memory but not executed when switching on the controller.
Error LED <b>see remark 2</b>	On	On	The error LED lights up for errors of all classes, no failsafe function activated.
		Off_by_E4	Warnings (E4) are not indicated by the error LED, no failsafe function activated.
		Off_by_E3	Warnings (E4) and light errors (E3) are not indicated by the error LED, no failsafe function activated.
		On+failsafe	The error LED lights up for errors of all classes and the failsafe function of the I/O bus is activated.
		Off_by_E4+failsafe	Warnings (E4) are not indicated by the error LED, the failsafe function of the I/O bus is activated.
		Off_by_E3+failsafe	Warnings (E4) and light errors (E3) are not indicated by the error LED, the failsafe function of the I/O bus is activated.
Check Battery	On	On	The availability of the battery and the battery status are checked. If no battery is available or the battery is empty, a warning (E4) is generated and the LED ERR lights up.
		Off	The battery is not checked. No warning (E4) is generated. This also applies if a battery is installed but empty!
Behaviour of outputs in stop <b>see remark 3</b>	Off in hardware and online	Off in hardware and online	In case of STOP, all outputs at the hardware and in the online display are set to FALSE or 0.
		Off in hardware and actual state online	In case of STOP, all outputs at the hardware are set to FALSE or 0. The online display indicates the status from the last cycle of the user program.
		Actual state in hardware and online	The status of the last cycle of the user program is kept for the outputs at the hardware and in the online display.
Reaction on floating point exceptions <b>see remark 4</b>	E2 failure	E2 failure	If a floating point exception occurs, an E2 error (Err=38) is triggered. The CPU goes to STOP. <b>Warning: PM59x only!</b>
		No failure	If a floating point exception occurs, no E2 error is triggered. Using the block FPU_EXINFO in the user program allows to react on a possibly occurred exception. <b>Warning: PM59x only!</b>

Parameter	Default value	Value	Meaning
Stop on error class <b>see remark 5</b>	E2	No effect	In case of a fatal or serious error (E1-E2), the user program is stopped.
		E1	In case of a fatal or serious error (E1-E2), the user program is stopped.
		E2	In case of a fatal or serious error (E1-E2), the user program is stopped.
		E3	In case of a fatal, serious or light error (E1-E3), the user program is stopped.
		E4	In case of a fatal, serious or light error (E1-E3) or a warning (E4), the user program is stopped.
Warmstart <b>see remark 6</b>	Off	Off	In case of a fatal error (E2), no warmstart is performed.
		On after E2 error	In case of a fatal error (E2), a warmstart is performed automatically.
		On after short voltage dip	A warmstart is performed after a short voltage dip.
		On after E2 or short voltage dip	In case of a fatal error (E2) or after a short voltage dip, a warmstart is performed automatically.
Start PERSISTENT %R0.x <b>see remark 7</b>	0	0..65535	Start offset for buffered area in PERSISTENT area %R0.x
End PERSISTENT %R0.x	0	0...65535	End offset for buffered area in PERSISTENT area %R0.x
...	..	..	..
Start PERSISTENT %R7.x	0	0...65535	Start offset for buffered area in PERSISTENT area %R7.x
End PERSISTENT %R7.x	0	0...65535	End offset for buffered area in PERSISTENT area %R7.x

### Remark 1: Setting the parameters Auto run and MOD using the display/keypad

See remark 1 under CPU parameters in PS501 versions V1.0 and V1.1

### Remark 2: Error LED

In addition to setting the behavior of the CPU's error LED ERR, this parameter is used to set the failsafe behavior of the I/O bus.

### Remark 3: Behaviour of outputs in Stop

The setting of the parameter "Behaviour of outputs in stop" directly influences the failsafe function of the outputs of the S500 I/O devices.

### Remark 4: Reaction on floating point exceptions

As of firmware version V1.2.0 of the AC500 CPUs and Control Builder version V1.2, the behavior of the CPUs PM59x regarding floating point exceptions can be set. In standard case, any floating point exception triggers an E2 error: class=E2, err=38, d1=9, d2=31, d3=31.

The CPU goes to STOP.

CPUs without floating point processor PM57x and PM59x do not trigger a floating point exception.

If the parameter "Reaction on floating point exceptions" is set to "No failure", no error is triggered in case of a floating point exception. The CPU remains in RUN mode.

By means of the function block FPU\_EXINFO (contained in SysInt\_AC500\_V10.LIB) it can be determined whether a floating point exception occurred during calculation. Depending on the result, either the calculation can be continued with default values or the machine can be shut down.

**Program example:**

```
PROGRAM PLC_PRG
```

```
VAR
```

```
    FPUEXINFO1    : FPU_EXINFO;
```

```
    rV1           : REAL := -1.0;
```

```
    rV2           : REAL;
```

```
    bError        : BOOL;
```

```
    bWarning      : BOOL;
```

```
END_VAR
```

```
bWarning := bError := FALSE;
```

```
rV2 := SQRT(rV1);
```

```
(* floating point calculation *)
```

```
FPUEXINFO1();
```

```
(* check for exception occurred *)
```

```
IF FPUEXINFO1.ERR THEN
```

```
    (* evaluation of exception *)
```

```
    (* for example, shut down system, continue calculation with default values or corrected values *)
```

```
    rV1 := 1.0;
```

```
    bWarning := TRUE;
```

```
    (* same calculation with corrected values *)
```

```
    rV2 := SQRT(rV1);
```

```
    FPUEXINFO1();
```

```
(* recheck.. *)
```

```
    IF FPUEXINFO1.ERR = TRUE THEN
```

```
        bError := TRUE;
```

```
    END_IF
```

```
END_IF
```

```
(* here, for example, evaluation of bWarning, bError.. *)
```

**Remark 5: Stop on error class**

As of firmware version V1.2.0 of the AC500 CPUs, the user program is stopped with any serious error (class E2) independent of the setting for the parameter "Stop on error class". The settings "No effect" and "E1" have the same behavior as the setting "E2".

The texts could not be changed due to downward compatibility to PS501 V1.0 and V1.1 projects.

**Remark 6: Warm start**

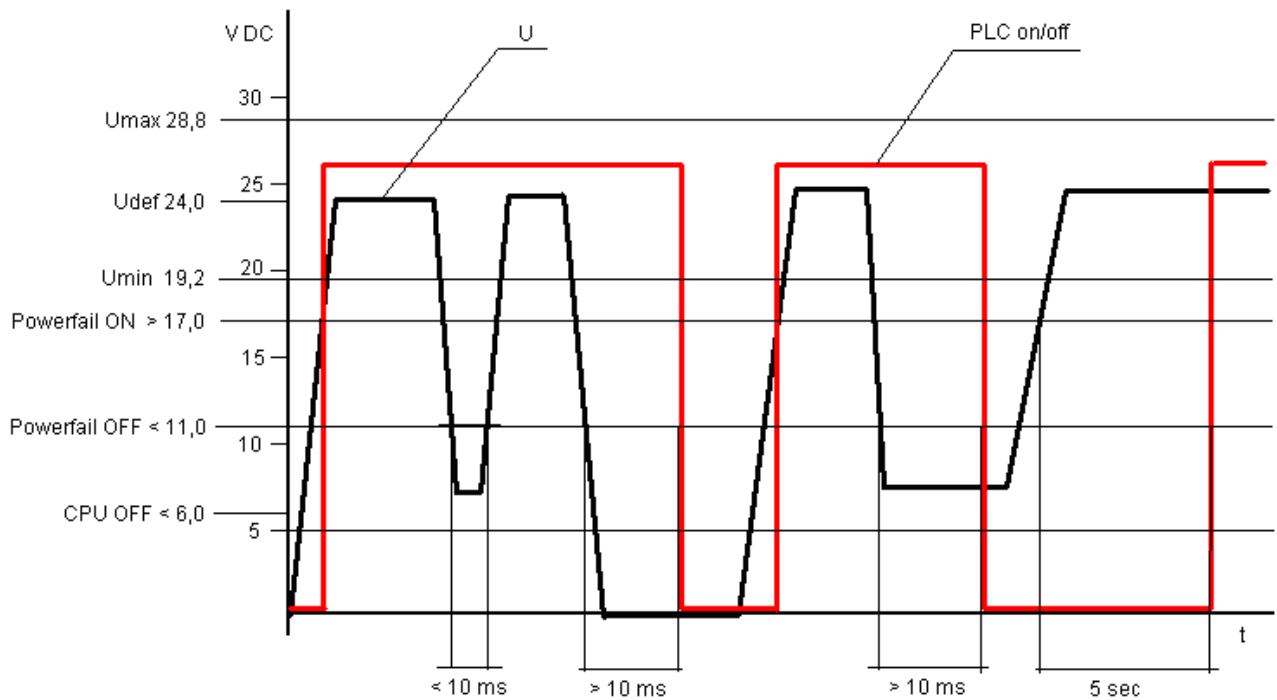
The parameter "Warmstart" allows to set the behavior of the CPU in case of

- serious errors (class E2) and
- short voltage dips.

If the default setting is used, the CPU changes to STOP mode if a serious error occurs. The CPU is switched off for voltage dips >10ms. The display shows "AC500".

The new settings allow to perform a warmstart of the CPU after a serious error or after short voltage dips or in case of both events.

The following figure shows the behavior of the CPU for different control voltage signals.



Short voltage dips, i.e., the control voltage falls below a value lower than "Powerfail OFF" (<11 VDC) for less than 10 ms, are bridged by the PLC, i.e., the CPU remains on.

If the control voltage is switched off, the CPU remains on for > 10 ms.

If the control voltage is lower than 11 V DC (but > 6 V DC) for longer than 10 ms and then goes back to the normal value, the behavior of the CPU depends on the setting for the parameter "Warmstart". If the parameter is set to "Off", the CPU remains in power fail mode, i.e., it does not restart. A restart of the CPU can only be done by switching the control voltage OFF/ON. If the parameter is set to "On after short voltage dip" or "On after E2 or short voltage dip", the CPU is restarted when the control voltage is greater than 17 V DC for 5 seconds. However, if the control voltage falls once more below 11 V DC within these 5 seconds, the time is restarted. Thus, the control voltage must have a value > 17 V DC for 5 seconds.

#### Remark 7: Start PERSISTENT %Rsegment.x and End PERSISTENT %Rsegment.x

With version V1.2 of PS501 and firmware V1.2.0 the new addressable variables area %Rx.x is available. The parameters "Start PERSISTENT %Rsegment.x" and "End PERSISTENT %Rsegment.x" are used to buffer this area. In the particular segment, "Start PERSISTENT %Rsegment.x" specifies the start byte and "End PERSISTENT %Rsegment.x" the end byte of the area to be buffered.

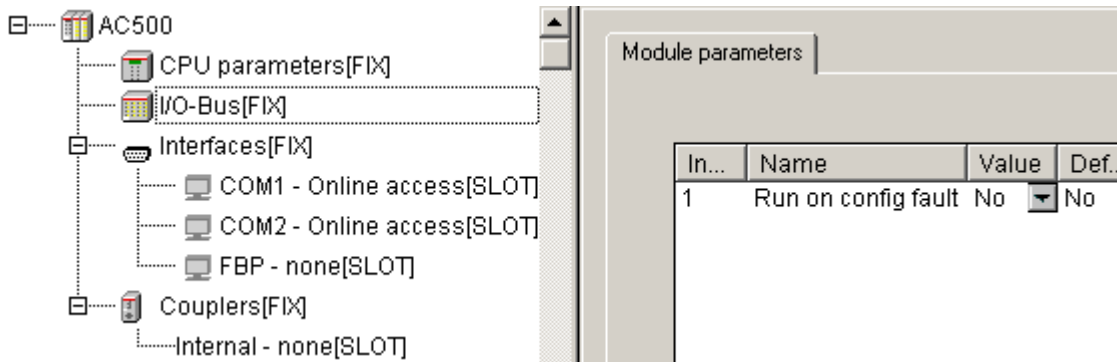
The new operand area is described in detail in chapter "The addressable PERSISTENT area %Rsegment.x".



### 3.3 I/O bus configuration

#### 3.3.1 Setting the general I/O bus parameters

Selecting "I/O-Bus" in the configuration tree opens the following configuration window:



In the same way as described for the CPU parameters, the general parameters for the CPU's I/O bus can be set in this window. The following parameter can be set:

Parameter	Default value	Value	Meaning
Run on config fault	No	No	In case of a configuration error, the user program is not started.
		Yes	The user program is run independent of a faulty I/O bus configuration.

#### 3.3.2 Inserting input and output modules

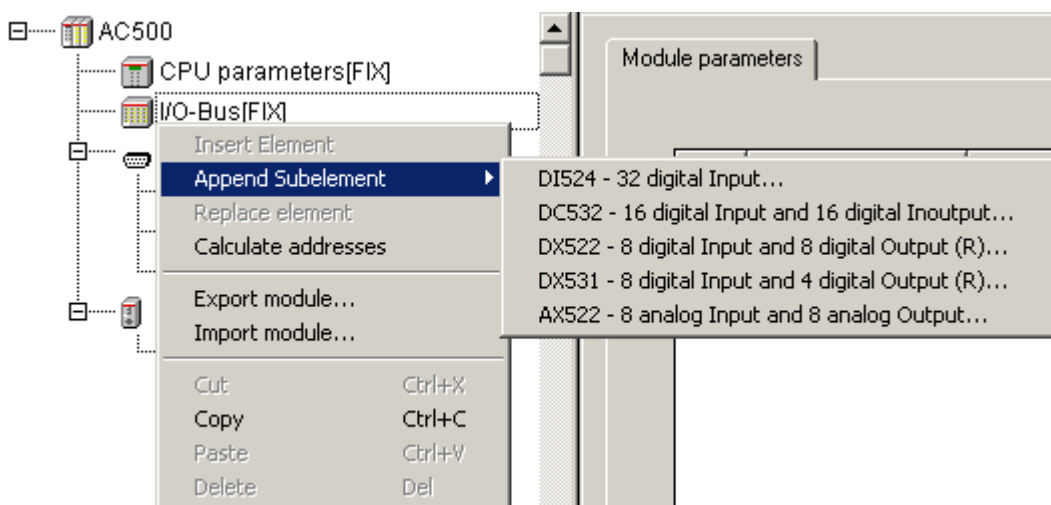
To make the inputs and/or outputs of the input and output modules connected to the I/O bus available in the project, the hardware must be reproduced in the PLC configuration.

Input and output modules connected to the I/O bus of the CPU occupy the I/O following area:

%IB0 .. %IB999 or %QB0 .. %QB999.

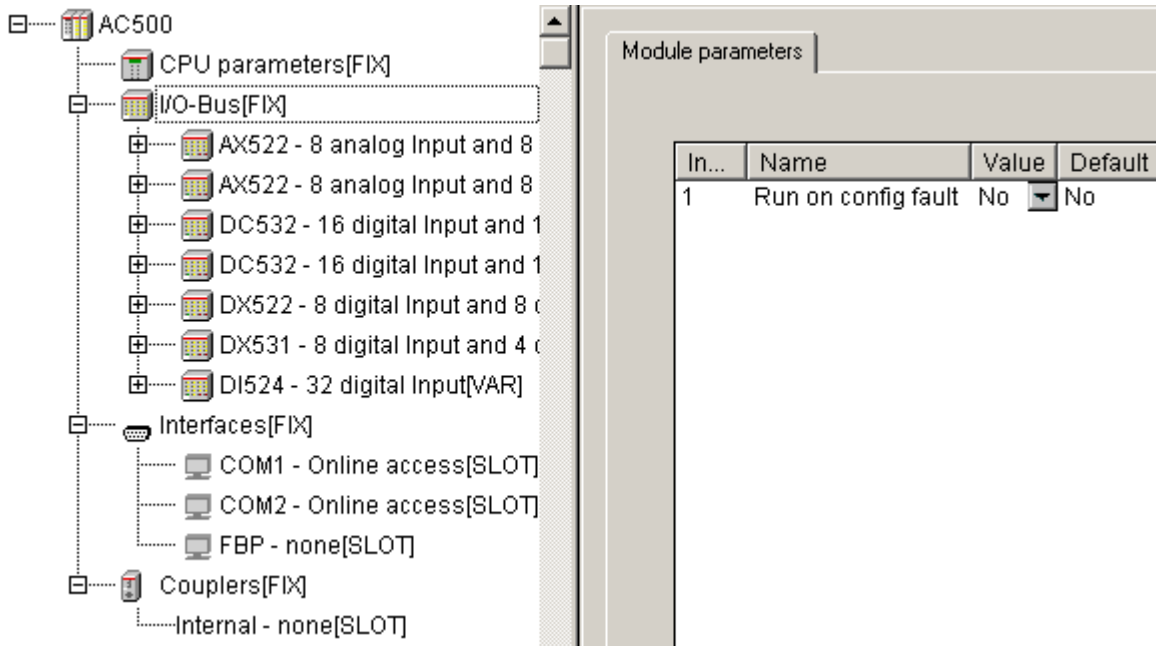
There is no fix assignment between module number and the input/output addresses of the channels.

Right-clicking the "I/O-Bus" element in the configuration tree opens the context menu where you can change the "I/O\_Bus" mode. Select "Append Subelement". The sub menu displays all available input and output modules:



Select the desired module depending on its hardware configuration. Repeat this step for all modules. A maximum of 7 input/output modules (10 modules as of V1.2.0) can be appended to the I/O bus.

The following figure shows an example for a configuration with the maximum number of modules:



If the maximum number of modules (7 modules or 10 modules as of V1.2.0) are appended, the context menu item "Append Subelement" can no longer be selected.

Changing the configuration is possible by deleting modules and inserting or appending new modules.

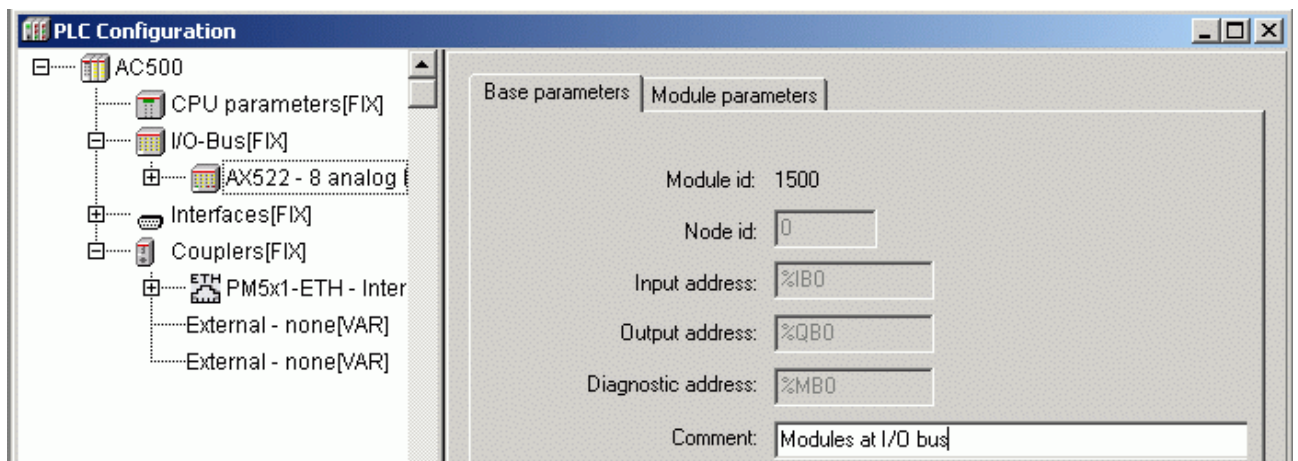


**Note:** As of firmware version V1.2.0 and PS501 version V1.2, 10 input/output modules can be appended to the CPU's I/O bus.

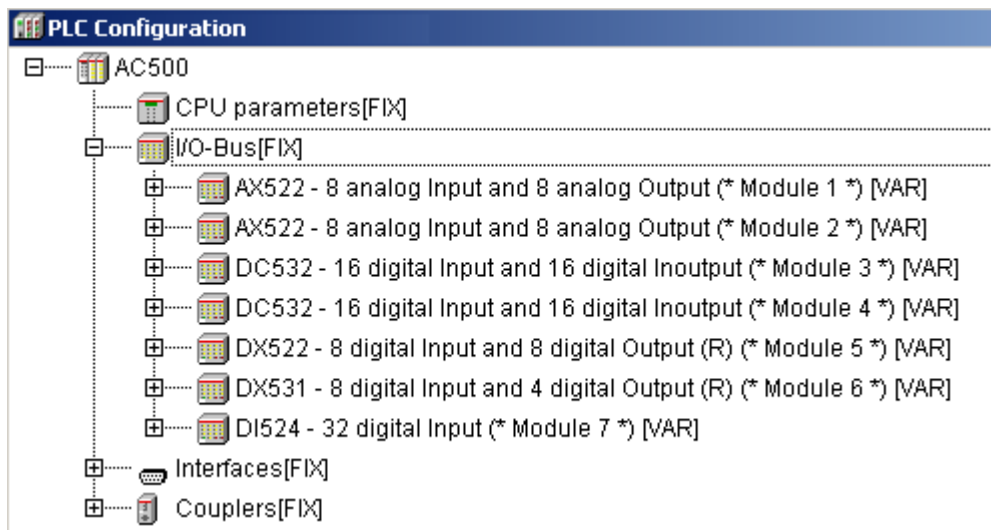
### 3.3.3 Configuring the input and output modules and channels

All inputs and/or outputs of the module are created when inserting the input/output module. In case of digital modules, the channels are provided as WORD, BYTE and BOOL.

If version V1.0 or V1.1 of the Control Builder is used, the module parameters are directly shown when clicking on an input/output module. As of Control Builder version V1.2, the Base parameters tab is opened.



A module name can be entered into the Comment field. This name also appears in the tree structure.



If you expand, for instance, the analog module AX522 and select the module parameters, the following is displayed:

Ind...	Name	Val...	De...
2	Ignore modu...	No	No
4	Check supply	On	On
5	Analog data ...	D...	Def.
6	Behaviour o...	Off	Off

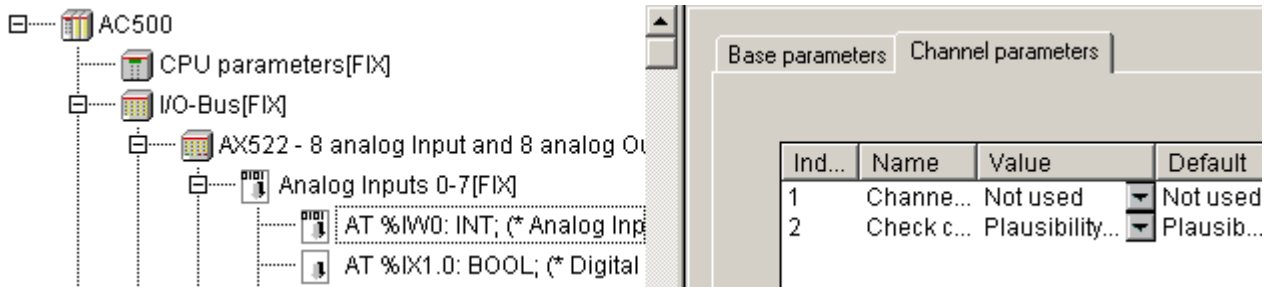
PLCconf\_IO4\_E.gif

Because the analog inputs can also be configured as digital inputs, bit 0 of each channel is also available as BOOL.

The following settings are possible:

1. The window with the module-specific parameters is displayed by selecting the module in the configuration tree. The parameters differ for the individual modules. For a description of the module parameters refer to the documentation for the input/output modules (parameterization) (see also I/O Device Description / Modules).

If an input/output module contains channel-related parameters, the following window appears when selecting the corresponding channel in the configuration tree:



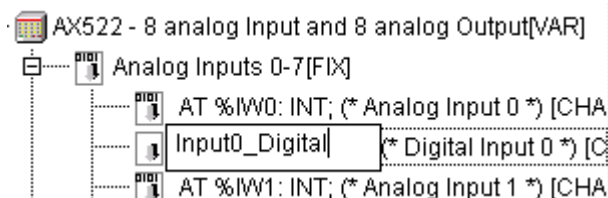
The parameters differ for the individual modules. For a description of the module parameters refer to the documentation for the input/output modules.

2. The symbolic name of a channel can be entered in front of the string "AT" in the channel declaration.

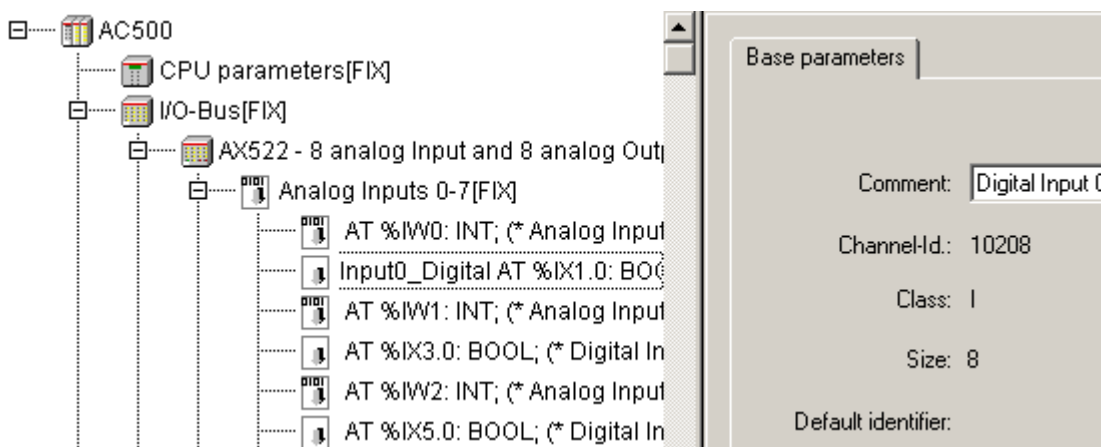


**Note:** All channels should have a symbolic name and only symbolic names should be used in the program code. If the hardware configuration has changed or if you want to download the project to a PLC with another hardware configuration and thus the PLC configuration has to be changed, the addresses of the inputs and outputs can change. In case of symbolic programming (i.e., symbolic names are used), the program code does not have to be changed.

Example how to enter a symbolic name:



3. For each channel, a comment can be entered into the field "Comment" in the "Base parameters" tab.



### 3.3.4 Module parameter "Ignore module" of S500 I/O devices

All S500 I/O devices have the module parameter "Ignore module". This parameter allows to set whether the I/O device specified in the PLC configuration is considered or not when checking the configuration data.

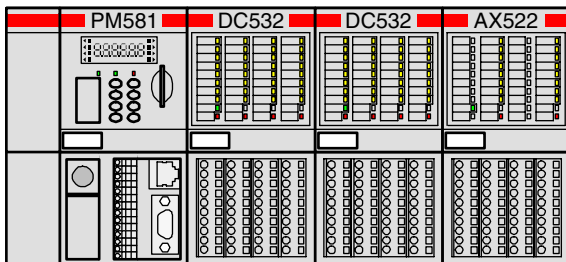
The parameter setting No (default setting) requires that the device is physically available.

If the parameter is set to Yes, the device must not be connected!

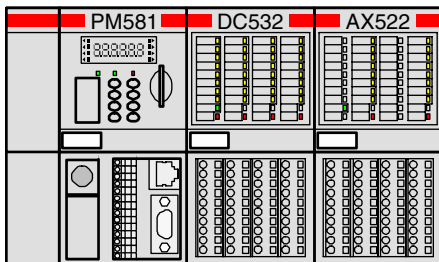
Thus, it is for example possible to create a project for machines with different hardware configuration and to exclude unnecessary input/output devices from checking by setting the parameter Ignore module to TRUE.

#### Example:

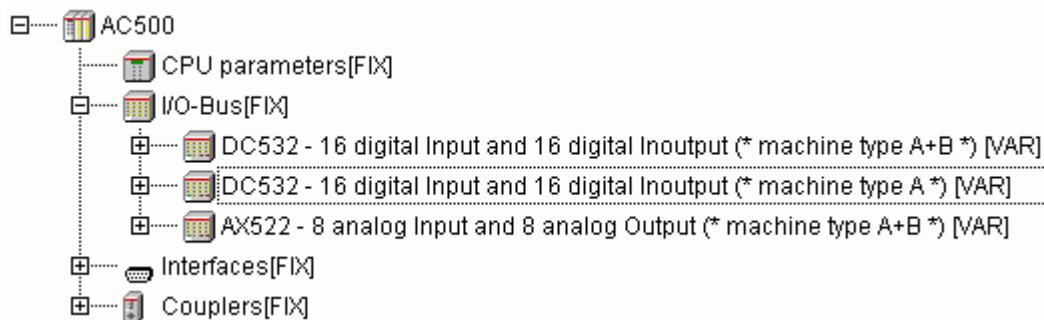
In full installation (type A), a machine shall be controlled with an AC500 with the following hardware configuration: CPU PM581 + 2xDC532 + 1xAX522



For a variant (type B) of the machine, the second DC532 is not required. This results in the following PLC hardware configuration: CPU PM581 + 1xDC532 + 1xAX522



The PLC configuration is identical for both machines:



In the project for machine type B, the module parameter Ignore module is set to TRUE for the second DC532. Thus, all inputs and outputs have the same addresses.

A further advantage of this parameter is that, for example, not all devices must be available for test purposes.

### 3.4 Configuration of the serial interfaces (Interfaces / COM1 and COM2)

The AC500 CPU is equipped with the two interfaces COM1 and COM2 which can be operated as RS 232 and RS 485.



**Note:** RS 485 operation of an interface is only possible, if the parameter "RTS control" is set to "telegram".

#### 3.4.1 Setting the protocol of the serial interfaces

By default, the serial interfaces are set to 'Online access', i.e., the access is done with help of the Control Builder.

Index	Name	Value	Default
11	Baudrate	19200	19200
12	Parity	none	none
13	Data bits	8	8
14	Stop bits	1	1

The protocol of the serial interfaces can be changed by right-clicking the interface 'COM1' or 'COM2' in the configuration tree and selecting the context menu item 'Replace element'.

Index	Name
11	Baudrat
12	Parity
13	Data bit

That means, the interface protocol is directly set in the PLC configuration. No block (such as MODINIT, COMINIT) is required.

The serial interface settings can be read in online mode using the PLC browser commands "com settings" and "com protocols". Chapter "AC500-specific PLC browser commands" contains a description of these commands.

### 3.4.2 The setting 'COMx - Online access'

If 'COMx - Online access' is selected, the interface parameters are set to the following fixed values:

Baudrate=19200 Baud, Stop bit=1, Parity=none, Data bits=8



**Note:** As of firmware version V1.1.7 and Control Builder version V1.2, the parameter "RTS control" is set to "telegram". This also allows programming via RS 485 (for example using an according converter).

Index	Name	Value	Default	Min.	Max.
2	RTS control	telegram	telegram		
11	Baudrate	19200	19200		
12	Parity	none	none		
13	Data bits	8	8		
14	Stop bits	1	1		

The parameters are read-only (not editable).

The serial interface settings must match the settings for the serial gateway driver in the Control Builder (see also Programming and Testing / Serial Driver).

### 3.4.3 The setting 'COMx - ASCII'

With the selection "ASCII", the initialization of the serial interface is done for the "free protocol", i.e., all interface parameters can be set and any protocol can be realized.

Sending and receiving data is done by means of the blocks COM\_SEND and COM\_REC (contained in library ASCII\_AC500\_V10.LIB). A detailed description of these blocks can be found in the ASCII\_AC500\_V10.LIB documentation.



**Caution:** To be able to receive data using the block COM\_REC, a buffer of the size **272 bytes** must be available (for example abyRecData : ARRAY[0..271] OF BYTE). This is also required if only short telegrams are received.

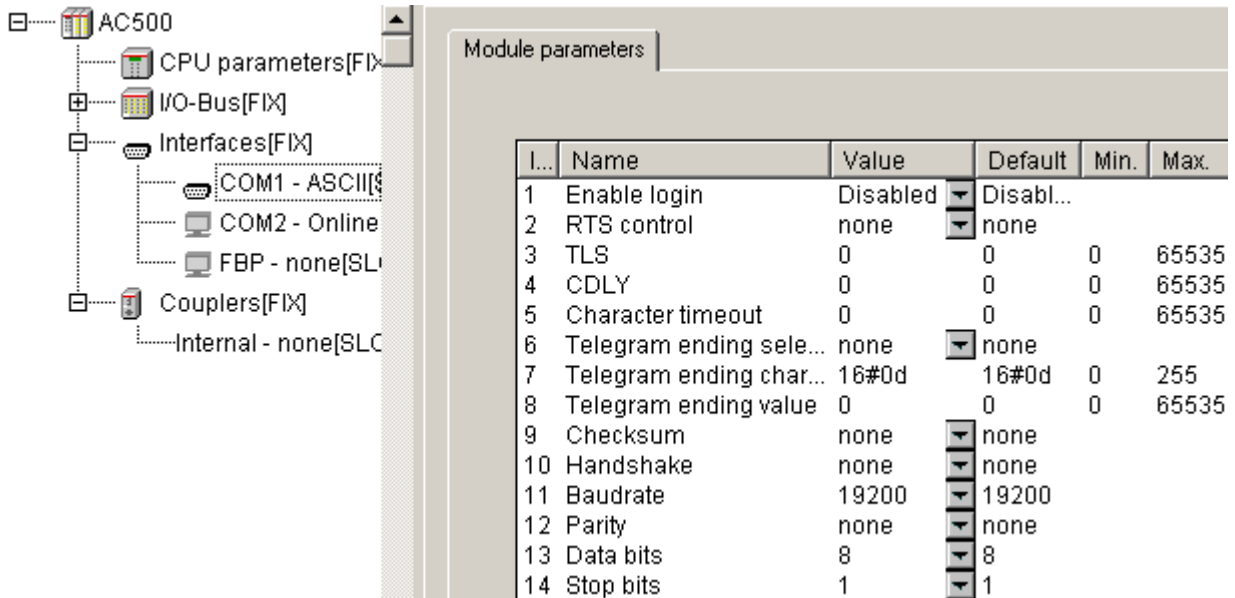
The operating system provides a total of 32 buffers with 272 bytes each for the transmission and reception of data. If the PLC is in STOP mode (= pause) or the input EN at the block COM\_REC is set to FALSE or the block is not called, these buffers run full.

If the block COM\_REC is called again (with EN:=TRUE) before all buffers are used, the data received meanwhile are made available.

If all buffers were full, the error Invalid handle with ERR=TRUE and ERNO=16#2001=8193 is reported for one cycle. After this the reception is reset.

The reception is always reset after a download or the command Online/Reset.

Selecting "ASCII" displays the following window:



The parameters define how the serial interface will be initialized. The parameters can be grouped. They are used to initialize the following functions:

- Monitoring the programming login:  
Enable login
- Modem control and RS485:  
RTS control, TLS, CDLY
- Recognition of telegram ending for reception:  
Character timeout, Telegram ending selection, Telegram ending value, Telegram ending character
- Checksum
- Transmission parameters:  
Baudrate, Parity, Data bits, Stop bits

The following settings are possible:

Parameter	Default value	Value	Meaning
Enable login <b>see remark 1</b>	Disabled	Disabled	There is no check with regard to the Control Builder login telegram.
		Enabled	Telegrams received are checked with regard to the Control Builder login sequence. If the sequence is detected, the protocol setting is changed to 'Online access'. -> available as of firmware 1.2.0 and PS501 V1.2
RTS control <b>see remark 2</b>	none	None	No RTS control (direction control)
		telegram	RTS control activated (absolutely necessary for RS 485!)
TLS <b>see remark 2</b>	0	0...65535	Carrier lead time in [ms] (TLS > CDLY)
CDLY <b>see remark 2</b>	0	0...65535	Carrier delay time in [ms] (CDLY <= TLS)
Character timeout	0	0...65535	Character timeout in characters (must be 0 if Telegram ending selection = Character timeout)




<b>see remark 3</b>			
Telegram ending selection <b>see remarks 3 and 4</b>	none	none	No telegram ending identifier
		String (check receive)	2 characters, e.g. <CR><LF> (16#0d, 16#0a -> 16#0d0a) in parameter "Telegram ending value"
		Telegram length	Telegram ending identifier set by telegram length
		Duration	Telegram ending identifier set by time
		Character timeout	Telegram ending identifier set by character timeout
Telegram ending character <b>see remark 3</b>	16#0d	0...255	Up to version V1.1.x: Telegram ending character
	0	0...1	As of version V1.2.0: Number of end characters in case of telegram ending selection "String"
Telegram ending value <b>see remark 3</b>	0	0...65535	Up to version V1.1.x: Telegram ending identifier value for settings "Duration" and "Character timeout"
	0	0...65535	As of version V1.2.0: Telegram ending identifier value for settings "Duration", "Character timeout" and "String"
Checksum <b>see remark 4</b>	none	None	No checksum
		CRC8	CRC8 checksum -> available as of firmware V1.2.0
		CRC16	CRC16 checksum (Motorola format) -> available as of firmware V1.2.0
		LRC	Add all values to byte (ignore overflow), result multiplied by -1 -> available as of firmware V1.2.0
		ADD	Add all values to byte (ignore overflow) -> available as of firmware V1.2.0 and PS501 V1.2
		CS31	CS31 bus checksum -> available as of firmware V1.2.0 and PS501 V1.2
		CRC8-FBP	CRC8 FBP field bus neutral protocol -> available as of firmware V1.2.0 and PS501 V1.2
		XOR	XOR all values to byte (ignore overflow) -> available as of firmware V1.2.0 and PS501 V1.2
		CRC16 (Intel)	Like CRC16, result swapped -> available as of firmware V1.2.0 and PS501 V1.2
Handshake	none	None	No handshake
		RTS/CTS	Hardware handshake
		XON/XOFF	Not yet implemented
		3964R master	Not yet implemented
		3964R slave	Not yet implemented
Baudrate	19200	300 1200 4800 9600 14400 19200 38400 57600 115200 125000 187500	Character length in bits/s

Parity	none	None	No parity check
		Odd	Odd parity
		Even	Even parity
		Mark	Parity bit := TRUE
		Space	Parity bit := FALSE
Data bits	8	5, 6, 7, 8	Character length in bits/character
Stop bits	1	1, 2	Number of stop bits

**Remark 1: Enable login**

This parameter is available as of firmware version V1.2.0 and PS501 V1.2!

If "Enable login" is set to Yes, all received telegrams are checked with regard to the CoDeSys login service.

 **Caution:** It is recommended to activate the automatic login detection only for those projects for which this function is absolutely required because it slows down communication via the serial interface and also influences the PLC performance.

If the connection is directly made via RS 232, a login telegram will only be detected if the same parameters as used by CoDeSys (Baudrate=19200 Baud, Stop bits=1, Parity=None, Data bits=8 Bit) are set when initializing the interface.

The same applies if the connection is made via RS 232/RS 485 interface converters. The login telegram can only be detected, if the initialization parameters have the same values as the parameters set in CoDeSys. Because for such an application usually more than one device are connected to the RS 485 transmission line, the following has to be observed additionally:

The CoDeSys login telegram does not contain a device address. Thus, the service is first identified by all devices connected to the RS 485 transmission line that can be programmed using CoDeSys and the interface of which is able to read the login telegram (interface with Enable login=Yes). Due to this, telegram collisions can occur during the subsequent acknowledgement of the login request by these devices, resulting in an interruption of the communication.

If the connection between CoDeSys and the PLC is established via modem, the communication is not influenced by the interface parameters set in the PLC configuration. The parameter values required for the modem used have to be set. Once the initialization is completed, the mode processes the received telegrams according to the parameter settings. Also the assignment between login request and an individual PLC is guaranteed because the connection is established using the modem's phone number or MSN.

The login with CoDeSys first causes a reinitialization of the interface. All blocks accessing this interface are locked during the online session, i.e., they do not perform any function. During this period the block outputs have the following values:

```
DONE = FALSE
ERR = TRUE
ERNO = PROTOCOL_PROTECTED = 16#301F = 12319
```

The blocks will be re-activated after the logout by CoDeSys.

The login monitoring for an interface is only done if CoDeSys is not already logged in via another interface (Ethernet, ARCNET or other COM).

**Remark 2: Usage of modems**

The ASCII protocol considers the special properties of modems, interface converters and repeaters. If these devices are used at a serial interface operated in 'free mode', the compression mechanism possibly supported by these devices has to be deactivated. For detailed information, please refer to the operation manual of the used device.

Some repeaters, modems or interface converters require a control signal in order to set the transfer direction. The direction control can be enabled or disabled via the input RTSCTRL.

Various devices additionally require a lead time to stabilize their carrier signal. These devices can only transfer data in send direction after this time has elapsed. This carrier lead time can be set via the input TLS.

Additionally, for some devices it is necessary to sustain the carrier signal in send direction for some time after data transfer is completed. Only if this time has elapsed, the complete transfer of a telegram is ensured and the devices are ready for data transfer in opposite direction. This carrier delay time can be set via the input CDLY.



**Note:** Carrier lead time (TLS) and carrier delay time (CDLY) must be adjusted for all communication devices connected to the same transmission line. The times are only considered for RTS control = telegram.

### Remark 3: Telegram ending identifier

The telegram ending identifier is set using the parameters Character timeout, Telegram ending selection, Telegram ending character and Telegram ending value.

#### Character silent time monitoring:

Monitoring of the character timeout can be set for all possible telegram ending settings (except Character timeout).

If the parameter "Character timeout" = 0, no character timeout monitoring is done.

With "Character timeout" > 0 the character timeout monitoring is activated.

The character silent time is defined in number of characters. The number of characters and the interface parameters (Baudrate, Parity, Data bits and Stop bits) are used to calculate the silent time.

**Example:** Baudrate=9600 Baud, Parity=none, Data bits=8, Stop bits=1, Character timeout=3

This results in a frame of 10 bits/character:  
1 start bit + 8 data bits + 0 parity bit + 1 stop bit

Character silent time =  $1000 \times \text{Character timeout} \times \text{Frame} / \text{Baudrate} \text{ [ms]}$   
Character timeout =  $1000 \times 3 \times 10 / 9600 = 3.125 \text{ ms} \sim 4 \text{ ms}$ .

If the time between the reception of two characters exceeds the character silent time, the reception is aborted with an error and the characters received up to this moment are made available.

The following parameter combinations are possible:

Character timeout	Telegram ending selection	Telegram ending character	Telegram ending value	Description
Character timeout see remark on character silent time monitoring	Type of telegram ending identifier	Telegram ending character - = ignored	Telegram ending value - = ignored	
Number of characters 0 or >0	None	-	-	No telegram ending identifier, i.e., the characters received since last call are provided. The maximum number of characters is limited to 256.
Number of characters 0 or >0	String (check receive)	Number of telegram ending characters 1 or 2	2 characters (for example 16#0d0a)	According to value set for "Telegram ending character", it is checked for 1 or 2 ending characters. The ending character(s) is (are) not passed, i.e., they are not contained in DATA area.
	1	1	16#0d = 13dec = <CR>	After reception of 16#0d, telegram received is reported.
	2	2	16#0d0a = 3338dec = <CR><LF>	After reception of 16#0d and subsequently 16#0a, telegram received is reported.
Number of characters 0 or >0	Telegram length	-	Number of characters >0 and <=256	Telegram received is reported once the number of characters defined in "Telegram ending value" is received.
Number of characters 0 or >0	Duration	-	Time in [ms]	Telegram received is reported once the time set for "Telegram ending value" (in [ms]) is elapsed. The time starts with the first FALSE -> TRUE edge at input EN of the receive block COM_REC.
0	Character timeout	-	Number of characters >0 and <=256	The number of characters set for "Telegram ending value" and the interface parameters (Baudrate, Parity, Data bits and Stop bits) are used to calculate the silent time. Telegram received is reported if the silent time between two characters is >= the calculated silent time.



**Caution:** The setting for the telegram ending selection "String" has been changed for firmware version V1.2.x and Control Builder version V1.2. This setting is not compatible with the setting in the firmware versions V1.0.x and V1.1.x and Control Builder versions V1.0 and V1.1.

Up to version V1.2.x, the telegram ending character was set with the parameter "Telegram ending character", the parameter "Telegram ending value" was ignored. Only one telegram ending character could be set.

**Thus, the user program has to be changed accordingly when updating to V1.2.x!**

#### Remark 4: Checksum:

The parameter "Checksum" takes effect as of CPU firmware version V1.2.0.

#### Sending with block COM\_SEND:

With "Checksum" <> none, the selected checksum is appended when sending. If the parameter "Telegram ending selection" is set to "String (check receive)", the checksum of the ending character(s) is entered. The character(s) is (are) appended according to the inputs END\_LEN and END\_CH of the block COM\_SEND.

#### Receiving with block COM\_REC:

With "Checksum" <> none, the selected checksum is checked during reception. If the parameter "Telegram ending selection" is set to "String (check receive)", the checksum of the ending character(s) is expected.

The ending character(s) and the checksum are not output, i.e., they are not contained in the DATA area.

A telegram should look as follows:

Data 0	Data 1	Data 2	..	Data n	Check 1	[Check 2]	End 1	[End 2]
--------	--------	--------	----	--------	---------	-----------	-------	---------

The values enclosed in [] are only relevant for 16 bit checksum or 2 ending characters.

At the blocks COM\_SEND and COM\_REC, the area addressed via the input DATA contains the following values:

Data 0	Data 1	Data 2	..	Data n
--------	--------	--------	----	--------

#### Example:

#### Setting in PLC configuration:

"Telegram ending selection" = String

"Telegram ending character" = 2

"Telegram ending value" = 16#0d0a

"Checksum" = CRC16 (i.e., Motorola format)

#### Send with COM\_SEND:

LEN = n+1

END\_LEN = 2

END\_CH = 16#0d0a

The area addressed via input **DATA** contains the following data:

Data 0	Data 1	Data 2	..	Data n
--------	--------	--------	----	--------

The following data are sent via the interface:

Data 0	Data 1	Data 2	..	Data n	CRC16 high	CRC16 low	16#0d	16#0a
--------	--------	--------	----	--------	------------	-----------	-------	-------

#### Reception with COM\_REC:

The interface receives the following telegram:

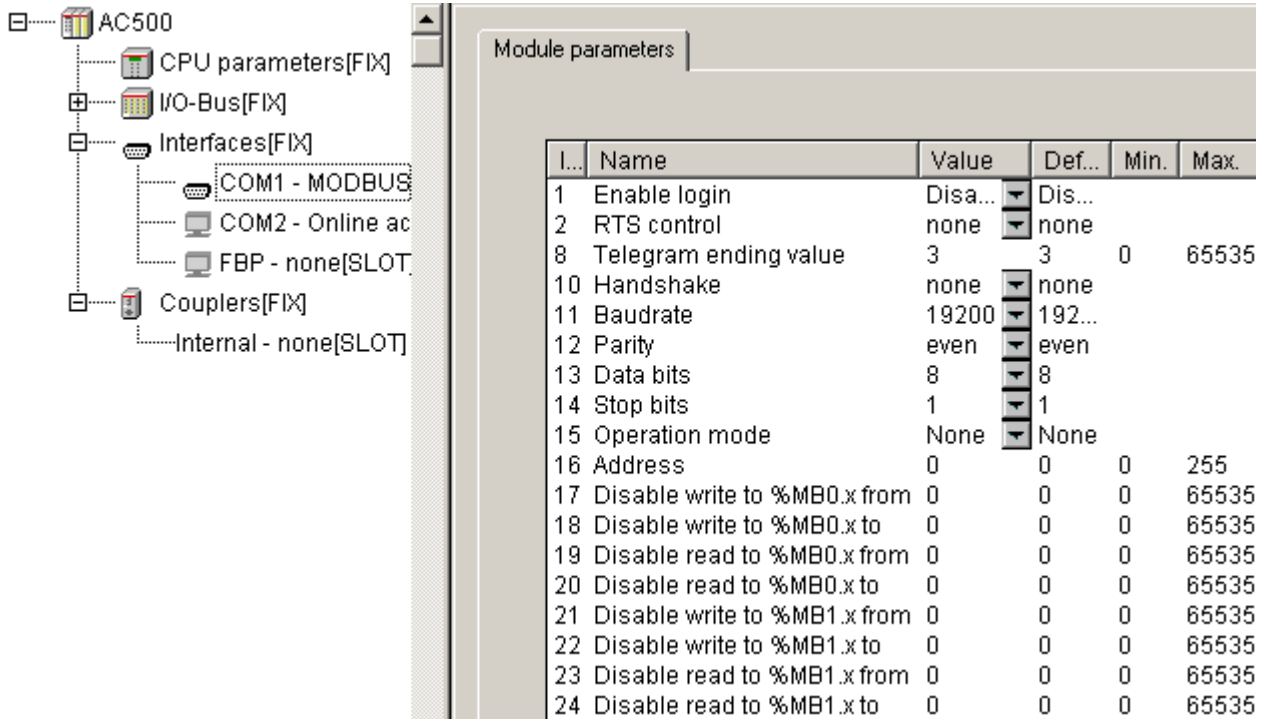
Data 0	Data 1	Data 2	..	Data n	CRC16 high	CRC16 low	16#0d	16#0a
--------	--------	--------	----	--------	------------	-----------	-------	-------

The following data are written to the area addressed via DATA:

Data 0	Data 1	Data 2	..	Data n
--------	--------	--------	----	--------

### 3.4.4 The setting 'COMx - Modbus'

For the protocol setting 'MODBUS', the following window is displayed:



The following settings are possible:

Parameter	Default value	Value	Meaning
Enable login	Disabled	Disabled	There is no check with regard to the Control Builder login telegram.
		Enabled	Telegrams received are checked with regard to the Control Builder login sequence. If the sequence is detected, the protocol setting is changed to 'Online access'. -> available as of firmware V1.2.0
RTS control	None	None	No RTS control
		Telegram	RTS control for telegram activated -> available as of firmware V1.2.0
TLS	0	0...65535	Carrier lead time in [ms] or characters (TLS > CDLY) -> available as of firmware V1.2.0
CDLY	0	0...65535	Carrier delay time in [ms] or characters (CDLY <= TLS) -> available as of firmware V1.2.0
Telegram ending value	3	0...65535	Number of characters for character timeout
Handshake	None	None	No flow control
		RTS/CTS	Hardware handshake -> available as of firmware V1.2.0
		XON/XOFF	Software handshake -> Not yet implemented
Baudrate	19200	300 1200 4800 9600 14400	Character length in bits/s

		19200 38400 57600 115200 125000 187500	
Parity	Even	None	No parity
		Odd	Odd parity
		Even	Even parity
		Mark	Parity bit := TRUE
		Space	Parity bit := FALSE
Data bits	8	5, 6, 7, 8	Number of data bits, 5 to 8
Stop bits	1	1, 2	Number of stop bits, 1 or 2
Operation mode	None	None	None
		Master	Master
		Slave	Slave
Address	0	0...255	Address for Modbus slave
Disable write to %MB0.x from	0	0...65535	Disable write access for segment 0 starting at %MB0.x
Disable write to %MB0.x to	0	0...65535	Disable write access for segment 0 up to %MB0.x
Disable read to %MB0.x from	0	0...65535	Disable read access for segment 0 starting at %MB0.x
Disable read to %MB0.x to	0	0...65535	Disable read access for segment 0 up to %MB0.x
Disable write to %MB1.x from	0	0...65535	Disable write access for segment 1 starting at %MB1.x
Disable write to %MB1.x to	0	0...65535	Disable write access for segment 1 up to %MB1.x
Disable read to %MB1.x from	0	0...65535	Disable read access for segment 1 starting at %MB1.x
Disable read to %MB1.x to	0	0...65535	Disable read access for segment 1 up to %MB1.x

The selection "COMx - MODBUS" sets the serial interface x to the Modbus RTU protocol (see also Modbus protocol).

For **Modbus slave operation**, an area without read and/or write access can be set in the segments %M0.x and %M1.x. Reading/writing is disabled beginning at the set address and is valid up to the set end address (inclusive).

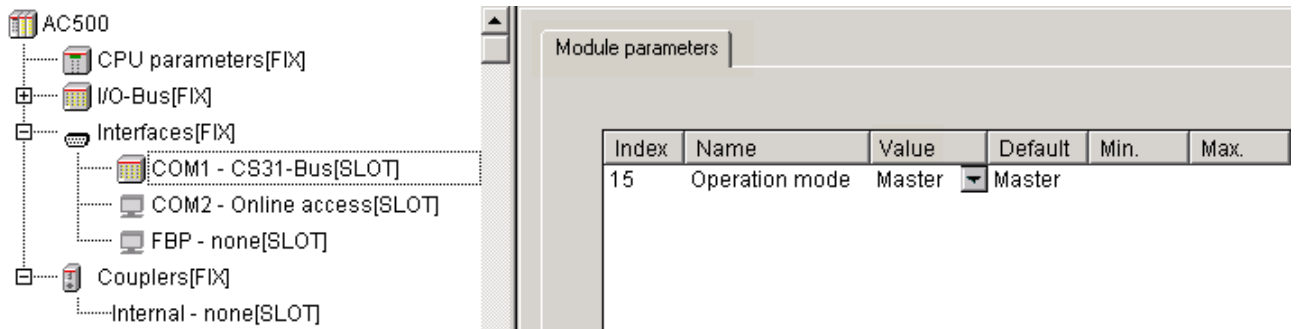


**Note:** The parameter "Data bits" always has to be set to 8 for Modbus.

### 3.4.5 The setting 'COM1 - CS31 Bus'

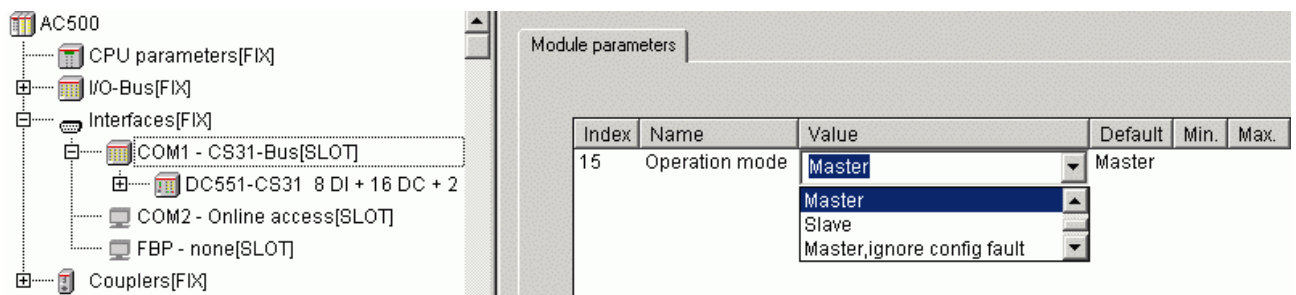
If the protocol 'CS31-Bus' is selected for the interface COM1, the interface is definitely set as CS31 bus master.

COM2 cannot be used as CS31 bus interface.



As of AC500 firmware version V1.2.0 and Control Builder version V1.2, the parameter "Operation mode" can be set to "Master" (default value) and "Master, ignore config fault" for the CS31.

**Note:** The settings "Slave" or "Slave, ignore config fault" are not allowed for the "CS31 bus" protocol. The default value "Master" is applied if these values are set.



The parameter "Operation mode" influences the beginning of the inputs/outputs update and the start of the user program.

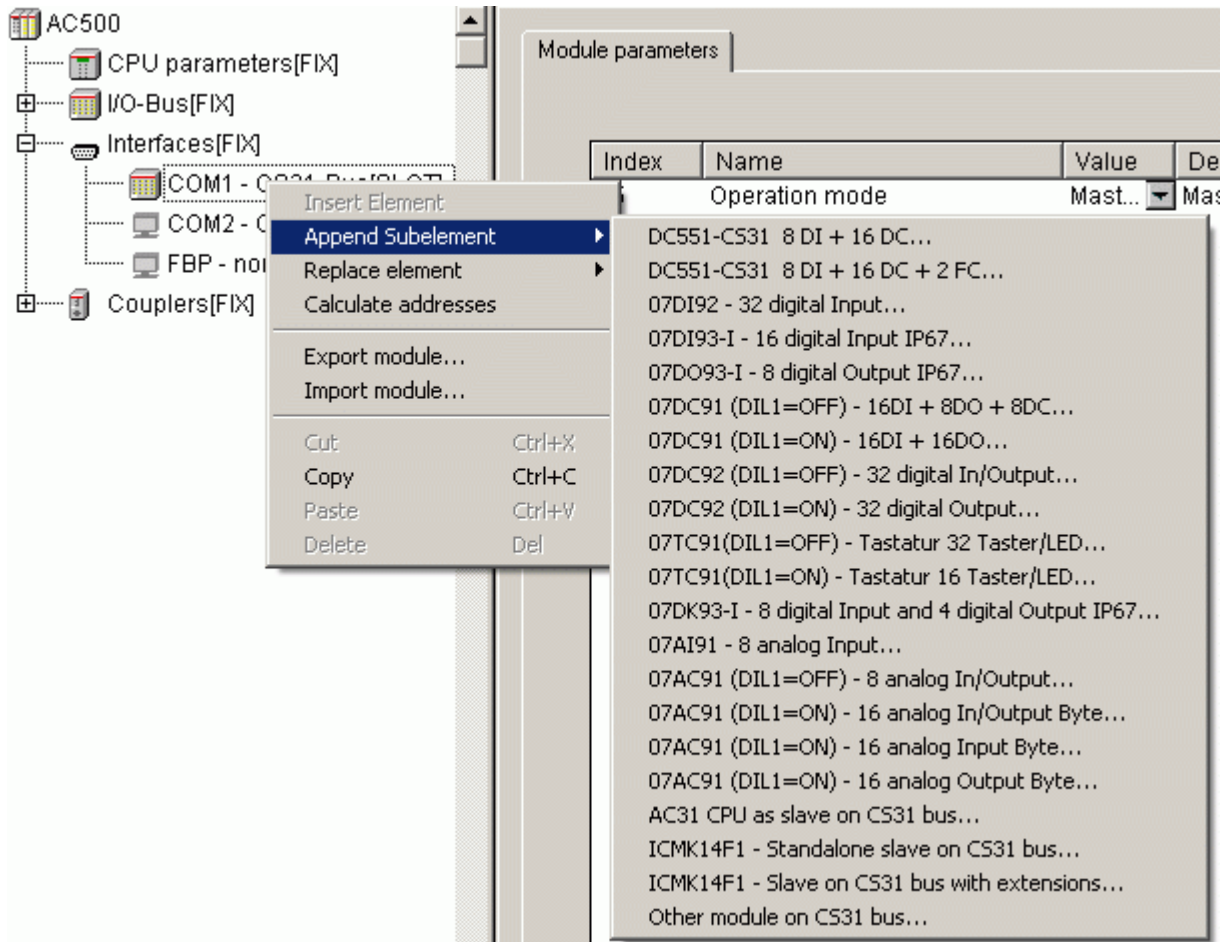
Setting "Master" (default):

Setting "Master, ignore config fault":

To make the inputs and/or outputs of the input and output modules connected to the CPU available in the project, the hardware **must** be reproduced in the PLC configuration.




Right-clicking the "COM1 - CS31-Bus" element in the configuration tree opens the context menu where you can change the module "COM1". Select "Append Subelement". The sub menu displays all input and output modules available for the CS31 bus:

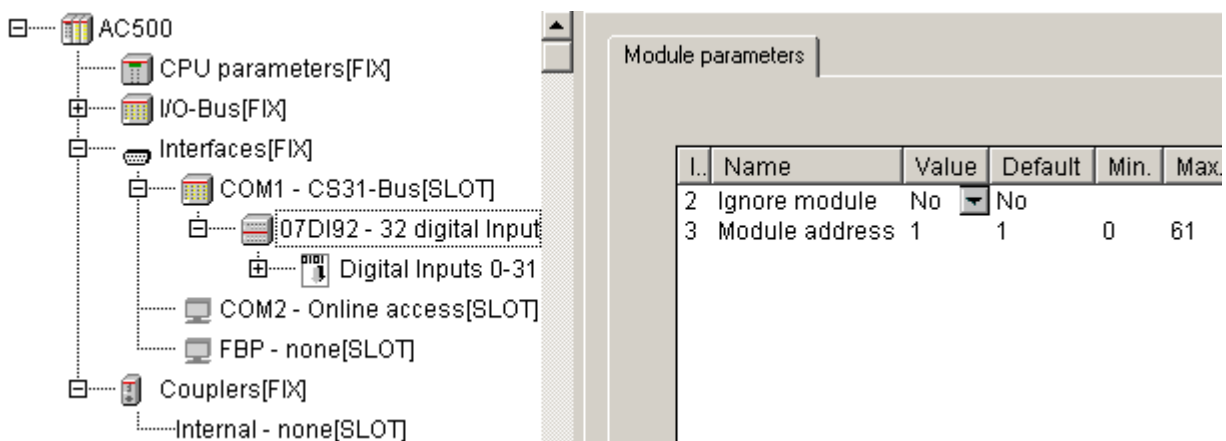


Select the desired input/output module.

A maximum of 31 modules (slaves) can be connected to the CS31 bus. Please note that for a module containing digital and analog expansions two modules are registered on the CS31 bus.

The configuration is described using the digital input module 07DI92 as an example. Once the module is inserted, the inputs and outputs of the module are available. In the 'Module parameters' window, set the 'Module address' to the module's hardware address (this is the address defined at the module with DIL switches).

 **Note:** For AC31 CPUs used as CS31 slave, the address is set via software.



The module address has to be set for all modules connected to the CS31 bus.

**! Caution:** There is no fix connection between module address and the input/output addresses of the channels. The input/output addresses are assigned automatically and change when inserting new modules.

When setting the module address, observe the following rules:

1. It is recommended to set a unique address for each module.
2. It is allowed to specify the same module address for a digital module and an analog module, but this is not recommended.
3. If the same module address is set for a digital input module and a digital output module, only the first module in the PLC configuration is detected. This also applies to analog modules!
4. DIL switch 8 (used to allocate the channels 8..15 for series 90) is ignored as all inputs/outputs are byte-oriented.
5. In case of expandable CS31 modules, the maximum configurations have to be observed.

Input/output modules connected to COM1 occupy the following I/O area:

**COM1: %IB1000 .. %IB1999 or %QB1000 .. %QB1999**

Further parameters and settings for the CS31 modules are optional and can be found in the descriptions for the individual modules.

The S500 modules are described here: S500 module description. AC31 modules are described in the 907 AC 1131 documentation.

### Connecting the DC551 and S500 I/O devices to the CS31 bus

The base module "DC551-CS31" is available in two versions in the PLC configuration:

1. DC551-CS31 8 DI + 16 DC / without fast counter

The addresses 00...69 can be set at the module and in the PLC configuration.

2. DC551-CS31 8 DI + 16 DC + 2FC / with 2 fast counters

The hardware addresses 70...99 can be set at the module. This corresponds to the module addresses 00...29 with activated counter. In the PLC configuration and at the block CNT\_DC551, the module address (00..29) is set.

(See also library Counter\_AC500\_V11.LIB / CNT\_DC551).

The following parameters can be set in the PLC configuration for the DC551:

Index	Name	Value	Default
2	Ignore module	No	No
3	Module address	8	1
13	Error LED	On	On
16	Check supply	On	On
17	Input delay	8 ms	8 ms
18	Fast counter	1-1 Up counter	0-No counter
19	Detection short circuit at outputs	On	On
20	Behaviour outputs at communication fault	Off	Off
21	Substitute Value	0	0

Parameter	Default value	Value	Meaning
Ignore module <b>see remark 1</b>	No	No	It is checked whether the module exists on the CS31 bus.
		Yes	Module is not checked. -> available as of CPU firmware V1.2.0 and PS501 V1.2
Module address	0	0...69	Module address of the DC551 without fast counter
		0...29	Module address of the DC551 with fast counter
Error-LED <b>see remark 2</b>	On	On	The error LED lights up for errors of all classes, no failsafe function activated.
		Off_by_E4	Warnings (E4) are not indicated by the error LED, no failsafe function activated.
		Off_by_E3	Warnings (E4) and light errors (E3) are not indicated by the error LED. No failsafe function activated.
		On+failsafe	The error LED lights up for errors of all classes and the failsafe function of the CS31 bus is activated. -> available as of CPU firmware V1.2.0, DC551 firmware V1.9 and PS501 V1.2
		Off_by_E4+failsafe	Warnings (E4) are not indicated by the error LED, the failsafe function of the CS31 bus is activated. -> available as of CPU firmware V1.2.0, DC551 firmware V1.9 and PS501 V1.2
		Off_by_E3+failsafe	Warnings (E4) and light errors (E3) are not indicated by the error LED, the failsafe function of the CS31 bus is activated. -> available as of CPU firmware V1.2.0, DC551 firmware V1.9 and PS501 V1.2
Check supply	On	On	Control voltage monitoring ON
		Off	Control voltage monitoring OFF
Input delay	8 ms	0.1 / 1 / 8 / 32 ms	Input delay 0.1 / 1 / 8 / 32 ms
Fast counter	0-No counter	0-No counter	Operation mode of the fast counter (see also hardware / description of fast counters)
Detection short-circuits at outputs	On	On	Output short-circuit detection ON
		Off	Output short-circuit detection OFF
Behaviour of outputs at communication fault <b>see remark 2</b>	Off	Off	Behavior of outputs at communication faults on the CS31 bus OFF
		Last value	Last value -> available as of CPU firmware V1.2.0, DC551 firmware V1.9 and PS501 V1.2
		Substitute value	Substitute value -> available as of CPU firmware V1.2.0, DC551 firmware V1.9 and PS501 V1.2
		Last value 5 sec.	Last value for 5 seconds -> available as of CPU firmware V1.2.0, DC551 firmware V1.9 and PS501 V1.2
		Substitute value 5 sec.	Substitute value for 5 seconds -> available as of CPU firmware V1.2.0, DC551 firmware V1.9 and PS501 V1.2

		Last value 10 sec.	Last value for 10 seconds -> available as of CPU firmware V1.2.0, DC551 firmware V1.9 and PS501 V1.2
		Substitute value 10 sec.	Substitute value for 10 seconds -> available as of CPU firmware V1.2.0, DC551 firmware V1.9 and PS501 V1.2
Substitute value <b>see remark 2</b>	0	0...65535 0000 <sub>hex</sub> ...FFFF <sub>hex</sub>	Substitute value for the outputs, one bit per output, bit 0=C8 .. bit 15=C23 -> available as of CPU firmware V1.2.0, DC551 firmware V1.9 and PS501 V1.2

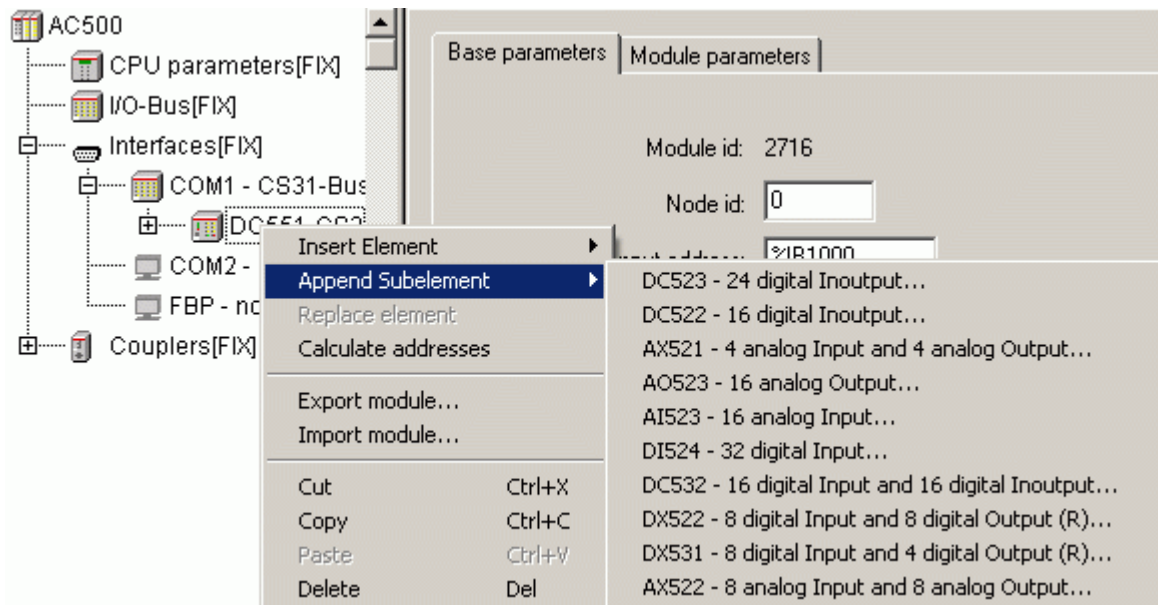
### Remark 1: Ignore module

A detailed description of the parameter "Ignore module" can be found in the chapter "The module parameter 'Ignore module' of S500 I/O devices".

### Remark 2: Failsafe function of CS31 bus

Further information on the failsafe function of the CS31 bus are contained in the chapter "The failsafe function of S500 I/O devices".

Further S500 modules can be coupled to the base module "DC551-CS31" via the I/O bus. Right-clicking the element "DC551-CS31" in the configuration tree and selecting the menu item "Append Subelement" displays all available input/output modules that can be added to the module "DC551-CS31".



A maximum of 7 expansions with a total of 240DI/240DO and 32AI/32AO can be appended to the module.

The following **peculiarities concerning the CS31 bus in the AC500** must be observed when addressing S500 I/O devices at the CS31 bus:

1. A CS31 software module can occupy a maximum of  
-> 15 bytes of inputs and 15 bytes of outputs in the digital area.  
This corresponds to  $15 \times 8 = 120$  digital inputs and 120 outputs.
2. A CS31 software module can allocate a maximum of  
-> 8 words of inputs and 8 words of outputs in the analog area.
3. A maximum of 31 of these CS software modules are allowed for connection to the CS31 bus.
4. If a device has more than 15 bytes or 8 words of inputs or outputs, it occupies 2 or more of the 31 CS31 software modules.

5. The DC551 can internally manage **2 CS31 software modules in the digital area and 4 CS31 software modules in the analog area**. This corresponds to a maximum of:
  - 240 digital inputs (2 x 15 bytes) and
  - 240 digital outputs (2 x 15 bytes) and
  - 32 analog inputs (4 x 8 words) and
  - 32 analog outputs (4 x 8 words).
6. Address setting is done at the DC551 using two rotary switches at the module's front plate.
7. To enable the fast counter of the DC551, the hardware address (HW\_ADR) has to be set to the module address + 70. With activated fast counter, the module addresses 0..28 (hardware address setting 70..98) are allowed.  
Then, the DC551 registers as 2 CS31 software modules using the module address (hardware address 70), once in the digital area and once in the analog area.
8. CS31 software module 1 in digital area:  
-> registers using the module address.  
CS31 software module 2 in digital area:  
-> registers using module address+7 and bit "Channel >= 7" set.  
CS31 software module 1 in analog area:  
-> registers using the module address.  
CS31 software module 2 in analog area:  
-> registers using module address and bit "Channel >= 7" set.  
CS31 software module 3 in analog area:  
-> registers using the module address+1.  
CS31 software module 4 in analog area:  
-> registers using module address+1 and bit "Channel >= 7" set.
9. The DC551 can manage a maximum of 255 parameters.  
This does not cause any restrictions in all configurations with the currently available S500 I/O devices.
10. The next free address for a DC551 is derived from the highest address occupied in the digital area or the analog area of the previous DC551.
11. When connecting several S500 expansion modules to a DC551 via the I/O bus, their inputs and outputs follow the DC551's inputs and outputs without gap. Such a cluster can occupy up to 5 CS31 software modules.
12. A maximum of 7 S500 expansion modules (extensions) can be connected to a DC551.

A configuration consisting of two combined input/output modules could look as follows:

I.	Name	Value	Default	Min.	Max.
2	Module address	1	1	0	61
6	Check supply	On	On		
7	Input delay	8 ms	8 ms		



**Note:** The fast counters of the input/output modules (e.g., "DC532") are only available if the modules are connected to the CPU's I/O bus.

### Summary of input/output data of S500 I/O devices

The following input/output data and parameters are available with S500 I/O devices:

Device	ID	I/O range	Digital area		Analog area		Parameter
			Inputs	Outputs	Inputs	Outputs	
			Byte	Byte	Words	Words	Byte
DC551	2716	8 DI + 16 DC	3	2	0	0	15
DC551+FC	2715	8 DI + 16 DC + FC	5	4	4	8	16
AI523	1515	16 AI	0	0	16	0	36
AO523	1510	16 AO	0	0	0	16	41
AX521	1505	4 AI + 4 AO	0	0	4	4	23
AX522	1500	8 AI + 8 AO	0	0	8	8	39
DC522	1220	16 DC	2	2	0	0	8
DC523	1215	24 DC	3	3	0	0	10
DC532	1200	16 DI + 16 DC	4	2	0	0	8
DI524	1000	32 DI	4	0	0	0	4
DX522	1210	8 DI + 8 DX	1	1	0	0	6
DX531	1205	8 DI + 4 DX	1	1	0	0	6

## Examples of impossible configurations

Due to the peculiarities concerning the CS31 bus and the DC551 described at the beginning of this chapter, some configurations cannot be realized. Here are some examples:

### Example 1: DC551 + 6 x DC532

Device	I/O range	Digital area		Analog area		Parameter
		Inputs	Outputs	Inputs	Outputs	
		Byte	Byte	Words	Words	
DC551	8 DI + 16 DC	3	2	0	0	15
DC532	16 DI + 16 DC	4	2	0	0	9
DC532	16 DI + 16 DC	4	2	0	0	9
DC532	16 DI + 16 DC	4	2	0	0	9
DC532	16 DI + 16 DC	4	2	0	0	9
DC532	16 DI + 16 DC	4	2	0	0	9
DC532	16 DI + 16 DC	4	2	0	0	9
DC532	16 DI + 16 DC	4	2	0	0	9
<b>Total</b>	<b>120 DI + 128 DC</b>	<b>31</b>	<b>16</b>	<b>0</b>	<b>0</b>	<b>78</b>

This configuration is not possible because the DC551 can manage a maximum of 30 bytes in the digital area (= 120 inputs/outputs).

### Example 2: DC551 + 5 (or more) AX522

Device	I/O range	Digital area		Analog area		Parameter
		Inputs	Outputs	Inputs	Outputs	
		Byte	Byte	Words	Words	
DC551	8 DI + 16 DC	3	2	0	0	15
AX522	8 AI + 8 AO	0	0	8	8	40
AX522	8 AI + 8 AO	0	0	8	8	40
AX522	8 AI + 8 AO	0	0	8	8	40
AX522	8 AI + 8 AO	0	0	8	8	40
AX522	8 AI + 8 AO	0	0	8	8	40
<b>Total</b>	<b>8 DI + 16 DC + 32 AI + 32 AO</b>	<b>3</b>	<b>2</b>	<b>40</b>	<b>40</b>	<b>215</b>

This configuration is not possible because the DC551 can manage a maximum of 32 words in the analog area. For 6 or 7 AX522, the number of analog channels increases accordingly.

### Example 3: DC551 + 3 (or more) AO523

Device	I/O range	Digital area		Analog area		Parameter
		Inputs	Outputs	Inputs	Outputs	
		Byte	Byte	Words	Words	
DC551	8 DI + 16 DC	3	2	0	0	15
AO523	16 AO	0	0	0	16	42
AO523	16 AO	0	0	0	16	42
AO523	16 AO	0	0	0	16	42
<b>Total</b>	<b>8 DI + 16 DC + 48 AO</b>	<b>3</b>	<b>2</b>	<b>0</b>	<b>48</b>	<b>141</b>

This configuration is not possible because the DC551 can manage a maximum of 32 words in the analog area. For each further AO523, the number of analog channels increases accordingly.

**Example 4: DC551 + 3 (or more) AI523**

Device	I/O range	Digital area		Analog area		Parameter
		Inputs	Outputs	Inputs	Outputs	
		Byte	Byte	Words	Words	
DC551	8 DI + 16 DC	3	2	0	0	15
AI523	16 AI	0	0	16	0	37
AI523	16 AI	0	0	16	0	37
AI523	16 AI	0	0	16	0	37
<b>Total</b>	<b>8 DI + 16 DC + 48 AI</b>	<b>3</b>	<b>2</b>	<b>48</b>	<b>0</b>	<b>126</b>

This configuration is not possible because the DC551 can manage a maximum of 32 words in the analog area. For each further AI523, the number of analog channels increases accordingly.

**Example 5: DC551 + 5 (or more) PD501**

Device	I/O range	Digital area		Analog area		Parameter
		Inputs	Outputs	Inputs	Outputs	
		Byte	Byte	Words	Words	
DC551	8 DI + 16 DC	3	2	0	0	15
PD501	8 DO + 8 AI	0	1	8	0	21
PD501	8 DO + 8 AI	0	1	8	0	21
PD501	8 DO + 8 AI	0	1	8	0	21
PD501	8 DO + 8 AI	0	1	8	0	21
PD501	8 DO + 8 AI	0	1	8	0	21
<b>Total</b>	<b>8 DI + 16 DC + 48 DO + 40 AI</b>	<b>3</b>	<b>7</b>	<b>40</b>	<b>0</b>	<b>120</b>

This configuration is not possible because the DC551 can manage a maximum of 32 words in the analog area. For 6 or 7 PD501, the number of analog channels increases accordingly.

**Example 6: DC551 with FC + 2 (or more) AO523**

Device	I/O range	Digital area		Analog area		Parameter
		Inputs	Outputs	Inputs	Outputs	
		Byte	Byte	Words	Words	
DC551	8 DI + 16 DC + FC	5	4	4	8	16
AO523	16 AO	0	0	0	16	42
AO523	16 AO	0	0	0	16	42
<b>Total</b>	<b>8 DI + 16 DC + FC + 32 AO</b>	<b>5</b>	<b>4</b>	<b>4</b>	<b>40</b>	<b>100</b>

This configuration is not possible because the DC551 can manage a maximum of 32 words in the analog area. For each further AO523, the number of analog channels increases accordingly.



### Example 7: DC551 with FC + 2 (or more) AI523

Device	I/O range	Digital area		Analog area		Parameter
		Inputs	Outputs	Inputs	Outputs	
		Byte	Byte	Words	Words	
DC551	8 DI + 16 DC + FC	5	4	4	8	16
AI523	16 AI	0	0	16	0	37
AI523	16 AI	0	0	16	0	37
<b>Total</b>	<b>8 DI + 16 DC + FC + 32 AI</b>	<b>5</b>	<b>4</b>	<b>36</b>	<b>8</b>	<b>90</b>

This configuration is not possible because the DC551 can manage a maximum of 32 words in the analog area. For each further AI523, the number of analog channels increases accordingly.

#### 3.4.6 The setting 'COMx - SysLibCom'

As of Control Builder version V1.2 and firmware version V1.2.0 the protocol

- 'COMx - SysLibCom'

is available.

If the protocol 'COMx - SysLibCom' is selected for the serial interface COMx, the interface is prepared for operation with the blocks contained in the library "SysLibCom.lib" and the according protocols.

The library SysLibCom.lib contains the following functions:

Function	Meaning	Note
SysComClose	Closes an interface	
SysComGetVersion2300	Internal version synchronization	Only internal
SysComOpen	Opens an interface	
SysComRead	Reads data from an interface	
SysComSetSettings	Parameterization of an interface	
SysComSetSettingsEx	Extended parameterization of an interface	Currently not supported
SysComWrite	Write data to an interface	



**Note:** All blocks in the "SysLibCom.lib" library are functions. The blocks are processed until the according action is completed or processing is aborted due to a possibly set timeout. During sending, the block waits, for example, until the characters have been actually output. This can cause the suspension of the task in case of too small task cycle times!

The interface is initialized in the PLC configuration according to the defined settings when starting the PLC, after a download or after a reset. The following settings are possible:

Parameter	Default value	Value	Meaning
Enable login see remark 1	Disabled	Disabled	There is no check with regard to the Control Builder login telegram.
		Enabled	Telegrams received are checked with regard to the Control Builder login sequence. If the sequence is detected, the protocol setting is changed to 'Online access'.
RTS control see remark 2	None	None	No RTS control (direction control)
		telegram	RTS control activated (absolutely necessary for RS 485!)

TLS see remark 2	0	0...65535	Carrier lead time in [ms] (TLS > CDLY)
CDLY see remark 2	0	0...65535	Carrier delay time in [ms] (CDLY <= TLS)
Character timeout see remark 3	0	0...65535	Character timeout in characters (must be 0 if Telegram ending selection = Character timeout)
Telegram ending selection see remark 3	None	None	No telegram ending identifier
		String (check receive)	2 characters, e.g. <CR><LF> (16#0d, 16#0a -> 16#0d0a) in parameter "Telegram ending value" <b>Setting not recommended!</b>
		Telegram length	Telegram ending identifier set by telegram length <b>Setting not recommended!</b>
		Duration	Telegram ending identifier set by time <b>Setting not recommended!</b>
		<b>Character timeout</b>	Telegram ending identifier set by character timeout
Telegram ending character see remark 3	0	0...1	Number of end characters in case of telegram ending selection "String"
Telegram ending value see remark 3	0	0...65535	Telegram ending identifier value for settings "Duration", "Character timeout" and "String"
Handshake	None	None	No handshake
		RTS/CTS	Hardware handshake
		XON/XOFF	Not yet implemented
		3964R master	Not yet implemented
		3964R slave	Not yet implemented
Baudrate	19200	300 1200 4800 9600 19200 38400 57600 115200 125000 187500	Character length in bits/s
Parity	None	None	No parity check
		Odd	Odd parity
		Even	Even parity
		Mark	Parity bit := TRUE
		Space	Parity bit := FALSE
Data bits	8	5, 6, 7, 8	Character length in bits/character
Stop bits	1	1, 2	Number of stop bits

### Remark 1: Enable login

See remark 1 under "ASCII" protocol settings

## Remark 2: Usage of modems

See remark 2 under "ASCII" protocol settings

## Remark 3: Telegram ending identifier

See remark 3 under "ASCII" protocol settings

Because the receive function SysComRead() interrupts the processing of the user program until the ending criteria (telegram ending selection or timeout) is detected, it is recommended to set the telegram ending selection only to the following values:

- None
- Character timeout

During processing of the user program, the following parameters can be changed using the function SysComSetSettings():

- Baudrate
- Number of stop bits
- Parity

This is done by adding the parameters to the structure COMSETTINGS. The structure is as follows:

Parameter	Type	Default value	Meaning / valid values
dwBaudRate	DWORD	none	Baudrate 300, 1200, 4800, 9600, 14400, 19200, 38400 57600, 115200, 125000, 187500 <b>Warning:</b> The structure must have a valid value assigned to it. The function reports an error for any invalid values (also 0).
byStopBits	BYTE	0	Number of stop bits 0=1, 1=1,5, 2=2 stop bits
byParity	BYTE	0	Parity 0=No, 1=odd, 2=even
dwTimeout	DWORD	0	Currently not supported! Specified values are ignored.
dwBufferSize	DWORD	0	
dwScan	DWORD	0	

## Example for sending/receiving with "SysLibCom"

The following example shows how data are sent/received with the protocol "SysLibCom".  
 -> Telegrams of 32 bytes length are to be received and sent.

### 1. Setting in PLC configuration:

Index	Value	Wert	Default	Min.	Max.
1	Enable login	Dis...	Disabl...		
2	RTS control	none	none		
3	TLS	0	0	0	65535
4	CDLY	0	0	0	65535
5	Character timeout	0	0	0	65535
6	Telegram ending selection	none	none		
7	Telegram ending character	0	0	0	255
8	Telegram ending value	0	0	0	65535
10	Handshake	none	none		
11	Baudrate	192...	19200		
12	Parity	none	none		
13	Data bits	8	8		
14	Stop bits	1	1		

### 2. Declaration part of the program PROGRAM proSysLibCom\_Test

VAR

```

strComSettings      : COMSETTINGS;          (* Structure of COM settings *)
dwHandle            : DWORD;
byStep              : BYTE;                 (* Step chain *)
dwRead              : DWORD;               (* Number of characters received *)
dwWritten           : DWORD;               (* Number of characters sent *)
bEnSend            : BOOL;                 (* Enable sending *)
byCom               : BYTE := COM2;        (* COM number *)
dwBaudrate          : DWORD := 19200;      (* Baudrate *)
wLenRec             : WORD := 32;          (* Number of characters to be received *)
wLenTele           : WORD := 32;          (* Telegram length, here 32 characters for
                                           example *)
wLenSend           : WORD := 32;          (* Number of characters to be sent, for example
                                           32 characters *)
dwTimeoutSend      : DWORD := 0;          (* Timeout in [ms] for sending *)
dwTimeoutRec       : DWORD := 0;          (* Timeout in [ms] for receiving *)
abyRecBuffer       : ARRAY[0..271] OF     (* Receive buffer, 272 bytes min.! *)
  BYTE;
abyTeleBuffer      : ARRAY[0..543] OF     (* Telegram buffer, 2 x receive buffer min. *)
  BYTE;
abySendBuffer      : ARRAY[0..271] OF     (* Send buffer, telegram length max.! *)
  BYTE;
strDataRec         : StrucReceiveData;    (* Structure of receive telegram *)
strDataSend        : StrucSendData        (* Structure of send telegram *)

```

END\_VAR

### 3. Code part of the program

-> Processing of a step chain containing the following steps

CASE byStep OF

```
0:      (* Step 0: Open the interface COMx -> SysComOpen -> get handle *)
        strComSettings.Port := byCom;                (* COM_Number *)
        dwHandle := SysComOpen(strComSettings.Port); (* Open COM interface -> get
                                                    handle *)

        byStep := SEL( dwHandle <> INVALID_HANDLE, 250, (* handle ok -> Step 1,
                                                    otherwise error step 250 *)
1):      (* Step 1: Transfer of COMx interface parameters *)
        strComSettings.dwBaudRate := dwBaudrate;    (* Set baudrate *)
        (* Enter at this point the number of stop bits and parity, if necessary *)

        (* set COM settings -> if OK, run step 2, in case of an error step 250 *)
        byStep := SEL( SysComSetSettings(dwHandle, ADR(strComSettings)), 250, 2);

2:      (* Step 2: Initialization completed successfully -> now receiving and/or sending *)

        (* Receive data:
        read all data received since last run, but wLenRec max.! *)
        dwRead := SysComRead( dwHandle,
                               ADR(abyRead),
                               WORD_TO_DWORD(wLenRec),
                               dwTimeoutRec);        (* READ DATA *)

        IF (dwRead > 0) THEN (* Number of characters received; in bytes *)
            (* here, ignore for example all characters until valid telegram start detected *)

            (* Number of receiving cycles for the telegram *)
            dwNumReadPerTele[byCom] := dwNumReadPerTele[byCom] + 1;
            (* Copy data to buffer *)
            SysMemCpy( dwDest := ADR(abySumDataRead[dwSum]DataRead),
                      dwSrc := ADR(abyRead[0]),
                      dwCount := dwRead );

            (* Sum of read data of a telegram *)
            dwSumDataRead := dwSumDataRead + dwRead;

            IF dwSumDataRead >= wLenTele THEN        (* Telegram complete ? *)
                dwRecCount := dwRecCount + 1;        (* Number of telegrams received *)
                (* Copy received telegram to structure strDataRec *)
                SysMemCpy( dwDest := ADR(strDataRec,
                                         dw Src := ADR(abySumDataRead[0]),
                                         dwCount := wLenTele );

                dwNumReadPerTele := 0;                (* init for following telegram *)

                dwSumDataRead := 0;

                (* here the evaluation of data starts !!! *)

            END_IF;                                  (* Telegram complete *)
```

```

END_IF; (* Data received *)

(* Send data *)
(* Enabling the sending of data can be done, for example, cyclical or by program control *)
IF bEnSend THEN (* Enable sending *)
    (* Copy data to be sent to send buffer *)
    SysMemCpy( dwDest := ADR(abyDataSend[0]),
               dwSrc := ADR(strDataSend),
               dwCount := wLenSend);

    (* Send data *)
    dwWritten := SysComWrite( dwHandle,
                              ADR(abyDataSend[0]),
                              WORD_TO_DWORD(wLenSend),
                              dwTimeoutSend); SEND DATA !!! *)

    IF (dwWritten <> wLenSend THEN
        byStep := 250; (* Error when sending *)
    END_IF; (* dwWritten *)
    bEnSend := FALSE; (* Deactivate enable sending *)
END_IF; (* bEnSend *)

250: (* Step 250: Error step -> Close COMx and start with step 0 *)
    bResult := SysComClose(dwHandle); (* Close COM interface *)
    dwHandle := 0;
    byStep := 0;

END_CASE; (* End of step chain *)

```

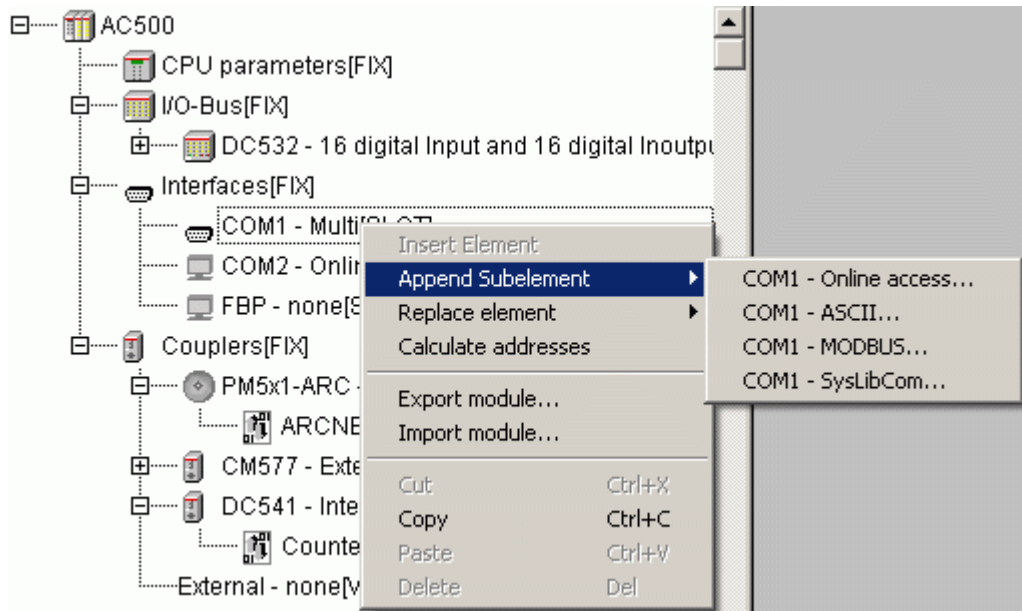
### 3.4.7 The setting 'COMx - Multi'

As of Control Builder version V1.2 and firmware version V1.2.0 the protocol 'COMx- Multi' can be used.

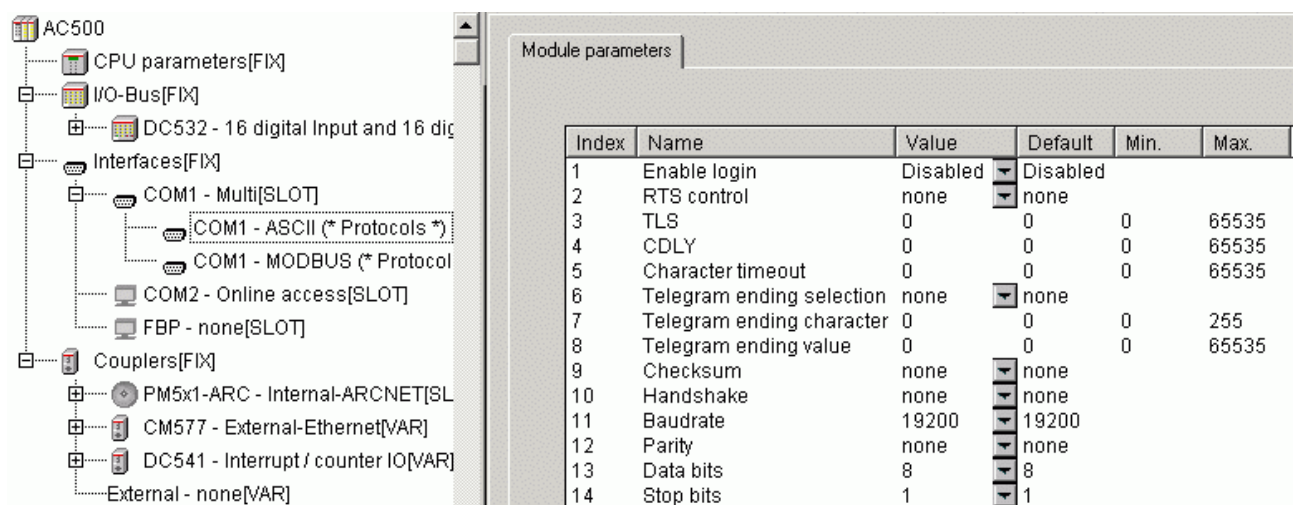
If the protocol 'COMx - Multi' is set for the serial interface COMx, the interface is prepared for operation with two selectable protocols.

Both protocols are appended to the protocol 'COMx - Multi' as modules. This is done in the same way than appending input/output modules to the I/O bus:

i.e., by right-clicking the interface, selecting Append Subelement and selecting the desired protocol.



Once both protocols have been appended, the parameters have to be set for each protocol:



The protocol parameters are identical to the parameters described for the individual protocols.

When restarting the program, i.e., after switching power ON, a download or a reset, the protocol appended first is always active.

Switching between the protocols is done using the block "COM\_SET\_PROT" (contained in the library SysInt\_AC500\_V10.lib). At the block input COM, the number of the serial interface is applied and at the input IDX the protocol index is set. The protocol appended first in the PLC configuration has the index 0, the second protocol the index 1.

## Functions of the block COM\_SET\_PROT

The block COM\_SET\_PROT can be used for different functions:

- Switching between two different protocols, for example ASCII / Modbus
- Switching the interface parameters of a protocol, for example changing the baudrate
- Re-initialization of an interface protocol (for example, if an interface "hangs up")
- Switching between "Online access" and ASCII/Modbus/SysLibCom depending on the current PLC mode, for example STOP=Online access, RUN=Modbus (or ASCII, SysLibCom). In this case, the parameter "Enable login" does not have to be activated and the interface can use other interface parameters than required for "Online access" (see the following program example).

Example for switching COM2 between "Online access" and Modbus (master)

### 1. Setting in the PLC configuration:

-> Protocol with index 0: Online access

Index	Name	Value	Default	Min.	Max.
2	RTS control	telegram	telegram		
11	Baudrate	19200	19200		
12	Parity	none	none		
13	Data bits	8	8		
14	Stop bits	1	1		

-> Protocol with index 1: Modbus (master)

Index	Name	Value	Default	Min.	Max.
1	Enable login	Disabled	Disabled		
2	RTS control	none	none		
8	Telegram ending value	3	3	0	65535
10	Handshake	none	none		
11	Baudrate	115200	19200		
12	Parity	even	even		
13	Data bits	8	8		
14	Stop bits	1	1		
15	Operation mode	Master	None		
16	Address	0	0	0	255
17	Disable write to %MB0.x from	0	0	0	65535
18	Disable write to %MB0.x to	0	0	0	65535
19	Disable read to %MB0.x from	0	0	0	65535
20	Disable read to %MB0.x to	0	0	0	65535
21	Disable write to %MB1.x from	0	0	0	65535
22	Disable write to %MB1.x to	0	0	0	65535
23	Disable read to %MB1.x from	0	0	0	65535
24	Disable read to %MB1.x to	0	0	0	65535

### 2. Setting the system events START and STOP in the task configuration:

Name	Description	called POU
<input checked="" type="checkbox"/> start	Called when program starts	callback_Start
<input checked="" type="checkbox"/> stop	Called when program stops	callback_Stop
<input type="checkbox"/> before_reset	Called before reset takes place	



### 3. Call of block COM\_SET\_PROT in system events

FUNCTION callback\_Start: DWORD

VAR\_INPUT

dwEvent : DINT;

dwFilter: DINT;

dwOwner : DINT;

END\_VAR

COM\_SET\_PROT(EN := FALSE); (\* for edge creation \*)

COM\_SET\_PROT(EN := TRUE, COM := 2, IDX := 1); (\* switch to Modbus \*)

FUNCTION callback\_Stop : BOOL

VAR\_INPUT

dwEvent : DINT;

dwFilter: DINT;

dwOwner : DINT;

END\_VAR

COM\_SET\_PROT(EN := FALSE); (\* for edge creation \*)

COM\_SET\_PROT(EN := TRUE, COM := 2, IDX := 0); (\* switch to Online access \*)

## 3.5 FBP slave interface configuration (Interfaces / FBP slave)

The FBP slave interface is used to connect the AC500 controllers as fieldbus slave via FieldBusPlug (FBP).

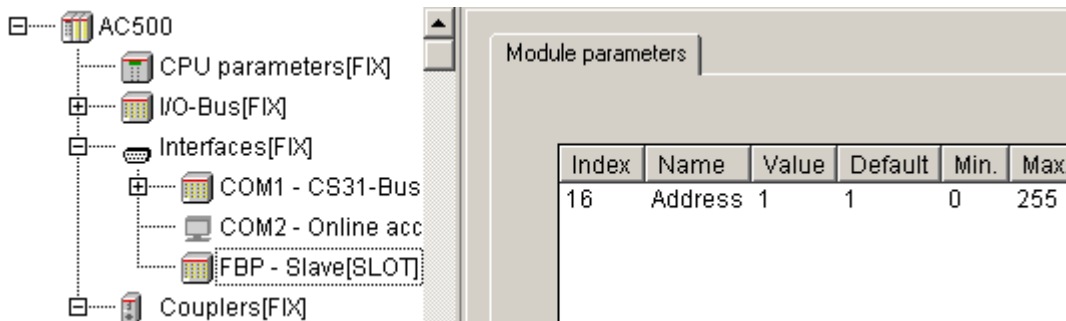
No protocol is set for the FBP interface in the standard configuration (setting "FBP - none").



**Note:** The FBP interface is configured as "Online access" for PM57x and PM58x (as of firmware version V1.1.7) and PM59x (as of version V1.2.0).

This allows programming via the FBP slave interface using the device UTF21-FBP adapter USB.

Setting the protocol "FBP - Slave" is done using the command "Replace element".



The following setting is possible:

Parameter	Default value	Value	Meaning
Address	0	0...255	Address as FBP slave



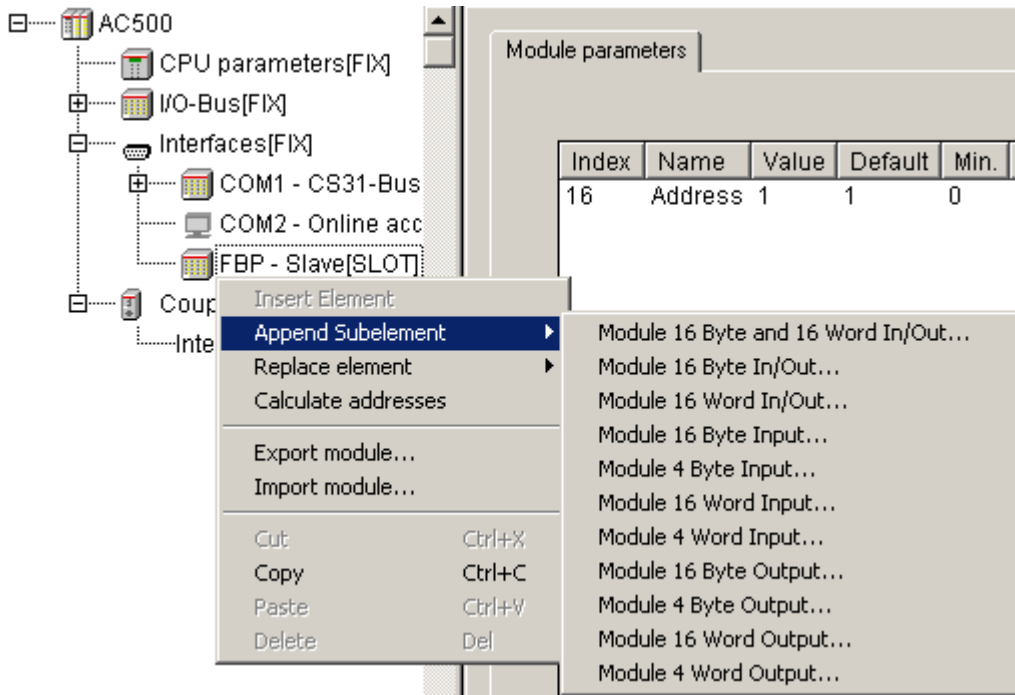
**Note:** If the FBP slave interface address (ADR > 0) is set using the display/keypad, this address has priority over the PLC configuration setting.

The FBP slave interface occupies the I/O area:

**%IB3000 .. %IB3999 or %QB3000 .. %QB3999.**

Depending on the fieldbus master, the AC500 CPU can exchange a different amount of input/output data with the master.

Right-clicking the "FBP-Slave" element in the configuration tree opens the context menu where you can change the "FBP" module. Select "Append Subelement". The sub menu displays all available input and output modules for the FBP slave interface:



A maximum of 8 modules can be appended to the FBP slave interface. A module can have a maximum of 16 byte and 16 word inputs and outputs. The number and size of possible modules depends on the used FBP, fieldbus and fieldbus master coupler.

FBP	Fieldbus	I/O range
PDP21	PROFIBUS DP V0	1 module with a maximum of 16 bytes and 16 words (inputs or outputs), for example 1 x "Module 16 Byte and 16 Word In/Out"
PDP22 DP V1 modular	PROFIBUS DP V0/V1	8 modules, but a total of 32 bytes and 128 words in/out, 244 bytes max. per direction, a total of 368 bytes per slave
DNP21	DeviceNet	1 module with a maximum of 16 bytes and 16 words (inputs or outputs), for example 1 x "Module 16 Byte and 16 Word In/Out"
DNP21 modular	DeviceNet	8 modules, but a total of 16 bytes and 16 words (input or output)

The Byte inputs and outputs are provided as BYTE and BOOL and the Word inputs and outputs as WORD, BYTE and BOOL.

The I/O modules saved in the PLC configuration and their addresses must match the entries in the configuration of the respective fieldbus master.

If you want to exchange less data than the maximum allowed amount of I/O data with the fieldbus master, you can setup a configuration consisting of different modules.

In the following example, the AC500 CPU operating as fieldbus slave no. 10 shall exchange 8 Byte inputs, 4 Byte outputs, 4 Word inputs and 16 Word outputs with a fieldbus master.

The following modules are appended using the context menu items:

- 2 x "4 Byte Input"
- 1 x "4 Byte Output"
- 1 x "4 Word Input" and
- 1 x "16 Word Output".

The final configuration looks as follows:

The screenshot shows a configuration tree for an AC500 system. The tree includes: CPU parameters[FIX], I/O-Bus[FIX], Interfaces[FIX], COM1 - CS31-Bus[SLOT], COM2 - Online access[SLOT], FBP - Slave[SLOT], Module 4 Byte Input[VAF], Module 4 Byte Input[VAF], Module 4 Byte Output[VAF], Module 4 Word Input[VAF], Module 16 Word Output, and Couplers[FIX]. The 'Module parameters' dialog box is open, showing a table with the following data:

Index	Name	Value	Default	Min.	Max.
16	Address	1	1	0	255

In this configuration, the Byte inputs and outputs are provided as BYTE and BOOL and the Word inputs and outputs as WORD, BYTE and BOOL.

### 3.6 Coupler configuration (Couplers)

In the standard configuration selected with "File" / "New" or "Extras" / "Standard configuration", the PLC configuration contains no couplers:

The screenshot shows a configuration tree for an AC500 system. The tree includes: CPU parameters[FIX], I/O-Bus[FIX], Interfaces[FIX], COM1 - Online access[SLOT], COM2 - Online access[SLOT], FBP - none[SLOT], Couplers[FIX], and Internal - none[SLOT].

PLCconf\_Coupler1.gif

The following assignment between coupler line and slot applies to the couplers:

- Line 0 corresponds to the internal coupler (installed in the CPU's housing)
- Line 1 is the coupler installed in the first slot on the left of the CPU
- Line 2 is the coupler installed in the second slot on the left of the CPU
- Line 3 and 4 are installed in the third and fourth slot on the left of the CPU

The allocation of inputs/outputs for the couplers is done slot-oriented and independent of the coupler type.



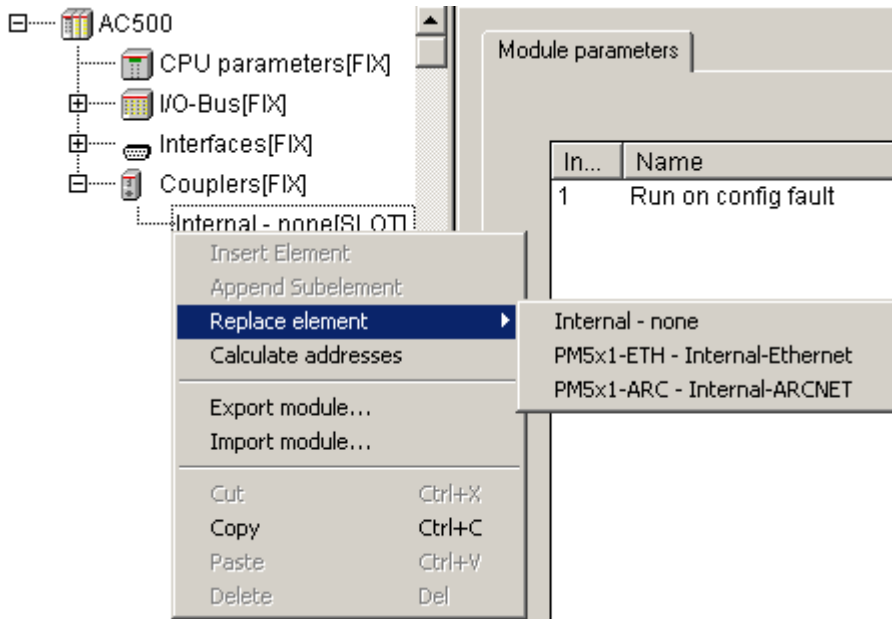
**Note:** If a coupler slot is empty, the entry "External none" has to be set for this coupler slot. This is not necessary, if **no** external coupler is inserted.

**Example:** Internal Ethernet coupler and PROFIBUS coupler in slot 2

1. Replace element: "Internal none" by "PM5x1 Internal Ethernet"
2. Append subelement: "External none" to empty slot 1
3. Append subelement: "CM572 External PROFIBUS DP Master" in slot 2

### 3.6.1 Configuring the internal coupler

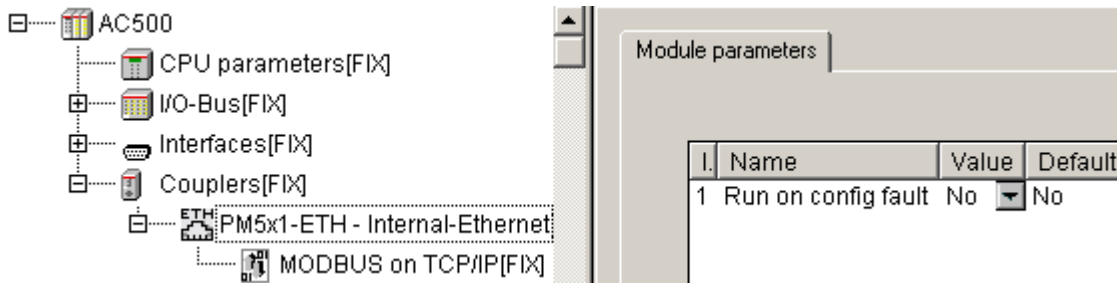
For an AC500 controller with internal coupler, the coupler must be specified in the PLC configuration. This is done by right-clicking the element "Internal - none" in the configuration tree and selecting the sub menu "Replace element". All available internal couplers are shown:



Select the coupler required for the used hardware.

#### 3.6.1.1 The internal Ethernet coupler PM5x1-ETH

The following parameters can be set for the internal Ethernet coupler "PM5x1-ETH - Internal Ethernet":



Parameter	Default value	Value	Meaning
Run on config fault	No	No	In case of a configuration error, the user program is not started.
		Yes	The user program is started even if the internal Ethernet coupler is configured incorrectly.

Only the behavior of the CPU in case of a configuration error of the coupler and the required protocols are set in the PLC configuration. The actual configuration of the Ethernet coupler such as the setting of the IP address is done with the integrated fieldbus configurator SYCON.net (see SYCON.net).

The protocol "MODBUS on TCP/IP" is available by default. Here, the following parameters can be set:

L...	Name	Value	Def...	Min.	Max.
3	Disable write to %MB0.x from	0	0	0	65535
4	Disable write to %MB0.x to	0	0	0	65535
5	Disable read to %MB0.x from	0	0	0	65535
6	Disable read to %MB0.x to	0	0	0	65535
7	Disable write to %MB1.x from	0	0	0	65535
8	Disable write to %MB1.x to	0	0	0	65535
9	Disable read to %MB1.x from	0	0	0	65535
10	Disable read to %MB1.x to	0	0	0	65535

Like for Modbus RTU on the serial interfaces COMx, it is also possible to disable read and/or write access to individual segments for slave operation. Reading/writing is disabled beginning at the set address and is valid up to the set end address (inclusive).

Parameter	Default value	Value	Meaning
Disable write to %MB0.x from	0	0...65535	Disable write access for segment 0 starting at %MB0.x
Disable write to %MB0.x to	0	0...65535	Disable write access for segment 0 up to %MB0.x
Disable read to %MB0.x from	0	0...65535	Disable read access for segment 0 starting at %MB0.x
Disable read to %MB0.x to	0	0...65535	Disable read access for segment 0 up to %MB0.x
Disable write to %MB1.x from	0	0...65535	Disable write access for segment 1 starting at %MB1.x
Disable write to %MB1.x to	0	0...65535	Disable write access for segment 1 up to %MB1.x
Disable read to %MB1.x from	0	0...65535	Disable read access for segment 1 starting at %MB1.x
Disable read to %MB1.x to	0	0...65535	Disable read access for segment 1 up to %MB1.x

The protocol "UDP data exchange" can be appended to the coupler by right-clicking the internal Ethernet coupler "PM5x1-ETH - Internal Ethernet" in the configuration tree and selecting the context menu item "Append UDP data exchange".

In...	Name	Value	Default
1	Run on config fault	No	No

The following parameters can be set for the "UDP data exchange" protocol:

Name	Value	De...	Min.	Max.
1 Size of receive buffer	8192	8192	1464	65535
2 Size of transmit buffer high prio	4096	4096	0	65535
3 Size of transmit buffer low prio	4096	4096	0	65535
4 Size of timeout buffer	2048	2048	0	65535
5 Number of header data	10	10	0	1464
6 Receive broadcast	Dis...	Dis...		
7 Behavior on receive buffer ov...	Ov...	Ove...		

Parameter	Default value	Value	Meaning
Size of receive buffer	8192	1464..65535	Size of receive buffer in bytes The minimum size is equal to the maximum size of an UDP telegram
Size of transmit buffer high prio	4096	0..65535	Size of transmit buffer (in bytes) for telegrams with high priority
Size of transmit buffer low prio	4096	0..65535	Size of transmit buffer (in bytes) for telegrams with low priority
Size of timeout buffer	2048	0..65535	Size of buffer (in bytes) for timeout data packages
Number of header data	10	0..1464	Number of header data to be copied to the timeout buffer for timeout packages (in bytes)
Receive broadcast	Disable	Disable	Reception of broadcast telegrams disabled (data packages to all stations)
		Enable	Reception of broadcast telegrams enabled (data packages to all stations)
Behavior on receive buffer overflow	Overwrite	Overwrite	Behavior on overflow of the receive buffer: The oldest data packages stored in the receive buffer are overwritten with the new incoming data packages.
		Reject	Behavior on overflow of the receive buffer: New incoming data are dismissed.

### 3.6.1.2 The internal ARCNET coupler PM5x1-ARCNET



**Note:** The controller with internal ARCNET coupler will be available as of version V1.2.

In the PLC configuration, the following parameters can be set for the internal ARCNET coupler "PM5x1-ARCNET - Internal ARCNET":

Index	Name	Value	Default	Min.	Max.
1	Run on config fault	No	No		
2	Address	0	0	0	255
3	Baudrate	2,5Mbs	2,5Mbs		
4	Extended timeout ET1/ET2	Very small net	Very small net		
5	Long packets	Enable	Enable		
6	Evaluate DIN on receipt	Enable	Enable		

Parameter	Default value	Value	Meaning
Run config fault	No	No	In case of a configuration error, the user program is not started.
		Yes	The user program is started independent of a faulty configuration of the internal ARCNET coupler.
Address	0	0...255	Address (node ID) of the ARCNET coupler
Baudrate see remark 1	2.5 MB/s	2.5 MB/s	Baudrate set for the ARCNET coupler
		1.25 MB/s	
		625 kB/s	
		312.5 kB/s	
Extended timeout ET1/ET2	Very small net	Very small net (=0)	ARCNET timeout setting. The following applies: Bit 0 configures ET2 of the coupler Bit 1 configures ET1 of the coupler  Value ET1 ET2 Meaning ----- 0 1 1 Max. network expansion 2 km 1 1 0 2 0 1 3 1 1 for large networks
		Small net (=1)	
		Big net (=2)	
		Very big net (=3)	
Long packets	Enable	Enable	Enable long data packages (512 bytes)
		Disable	Incoming long data packages are received and dismissed. The SEND block indicates an error in case of long data packages.
Evaluate DIN on receipt see remark 2	Enable	Enable	Enable check of DIN identifier on receipt
		Disable	Disable check of DIN identifier on receipt

### Remark 1: Baudrate of the ARCNET coupler



**Note:** If the baudrate set for the ARCNET coupler differs from the default value (2.5 MB/s), programming via ARCNET using the SoHard-ARCNET PC boards is no longer possible. The same baudrate has to be set for all subscribers of the ARCNET network. The ARCNET PC boards are firmly set to 2.5 MB/s.

### Remark 2: Check of DIN identifier on receipt

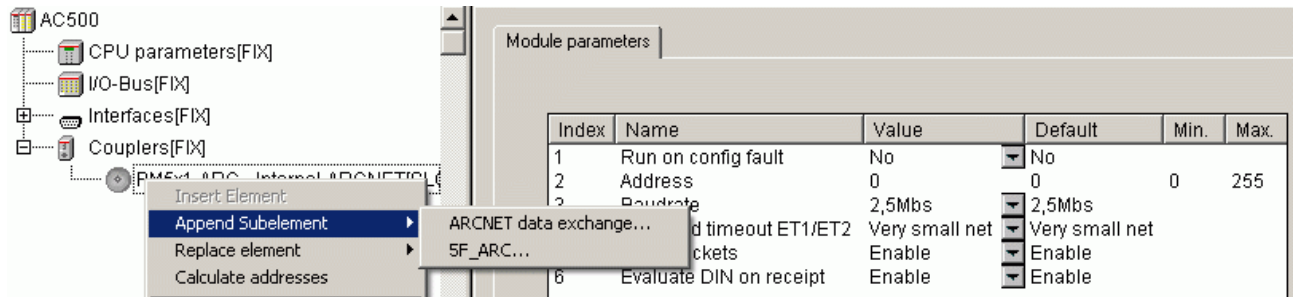
If the parameter "Evaluate DIN on receipt" is enabled (default setting), the following DIN identifiers are reserved:

DIN identifier		Protocol
Hex	Dec	
4F	79	"Online access" - Programming/OPC with CoDeSys / AC1131
5F	95	5F_ARCNET (MODBUS functions for ARCNET)
6F	111	PC331 programming (not used for AC1131/CoDeSys)
7F	127	Default DIN identifier for data exchange with function blocks: ARC_REC, ARC_SEND, ARC_STO, ARC_INFO All DIN identifiers except the reserved identifiers can be used for data exchange.

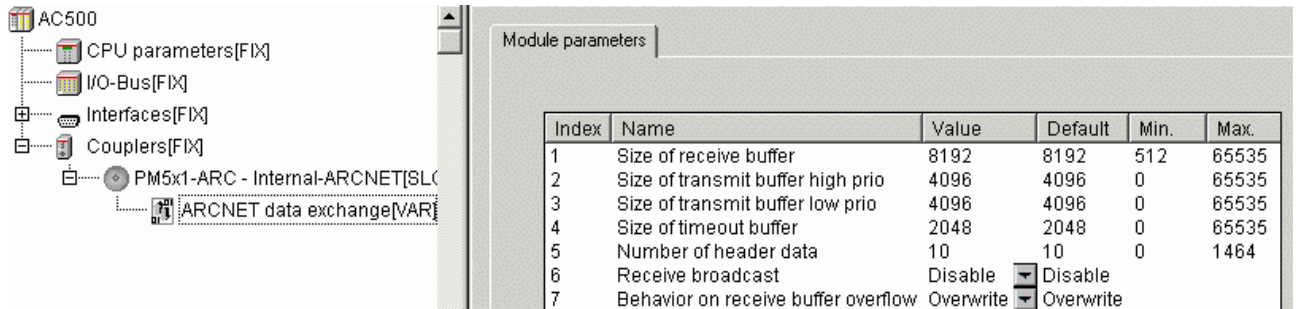


**Caution:** If the parameter "Evaluate DIN on receipt" is disabled, programming and/or OPC via ARCNET is not possible!

The protocols "ARCNET data exchange" and/or "5F\_ARC" can be appended to the coupler by right-clicking the internal ARCNET coupler "PM5x1-ARC - Internal ARCNET" in the configuration tree and selecting the corresponding context menu item.



The following parameters can be set for the "ARCNET data exchange" protocol:



Parameter	Default value	Value	Meaning
Size of receive buffer	8192	512...65535	Receive buffer size in bytes. The minimum size is equal to the maximum size of an UDP telegram.
Size of transmit buffer high prio	4096	0...65535	Size of transmit buffer (in bytes) for telegrams with high priority.
Size of transmit buffer low prio	4096	0...65535	Size of transmit buffer (in bytes) for telegrams with low priority.
Size of timeout buffer	2048	0...65535	Size of buffer (in bytes) for timeout data packages.
Number of header data	10	0...1464	Number of header data to be copied to the timeout buffer for timeout packages (in bytes).
Receive broadcast	Disable	Disable	Reception of broadcast telegrams disabled (data packages to all stations).
		Enable	Reception of broadcast telegrams enabled (data packages to all stations).
Behavior on receive buffer overflow	Overwrite	Overwrite	Behavior on overflow of the receive buffer. The oldest data packages stored in the receive buffer are overwritten with the new incoming data packages.
		Reject	Behavior on overflow of the receive buffer. New incoming data are dismissed.



The following parameters can be set for the "5F\_ARC" protocol:

Index	Name	Value	Default	Min.	Max.
3	Disable write to %MB0.x from	0	0	0	65535
4	Disable write to %MB0.x to	0	0	0	65535
5	Disable read to %MB0.x from	0	0	0	65535
6	Disable read to %MB0.x to	0	0	0	65535
7	Disable write to %MB1.x from	0	0	0	65535
8	Disable write to %MB1.x to	0	0	0	65535
9	Disable read to %MB1.x from	0	0	0	65535
10	Disable read to %MB1.x to	0	0	0	65535

Parameter	Default value	Value	Meaning
Disable write to %MB0.x from	0	0...65535	Disable write access for segment 0 starting at %MB0.x
Disable write to %MB0.x to	0	0...65535	Disable write access for segment 0 up to %MB0.x
Disable read %MB0.x from	0	0...65535	Disable read access for segment 0 starting at %MB0.x
Disable read %MB0.x to	0	0...65535	Disable read access for segment 0 up to %MB0.x
Disable write to %MB1.x from	0	0...65535	Disable write access for segment 1 starting at %MB1.x
Disable write to %MB1.x to	0	0...65535	Disable write access for segment 1 up to %MB1.x
Disable read %MB1.x from	0	0...65535	Disable read access for segment 1 starting at %MB1.x
Disable read %MB1.x to	0	0...65535	Disable read access for segment 1 up to %MB1.x



**Note:** For the AC500 CPU PM571, 4kB = %MB0.0 .. %MB0.4095 (i.e., not a complete segment) are available for the addressable flag area. Thus, not all Modbus addresses can be accessed.

### 3.6.2 Configuring the external couplers

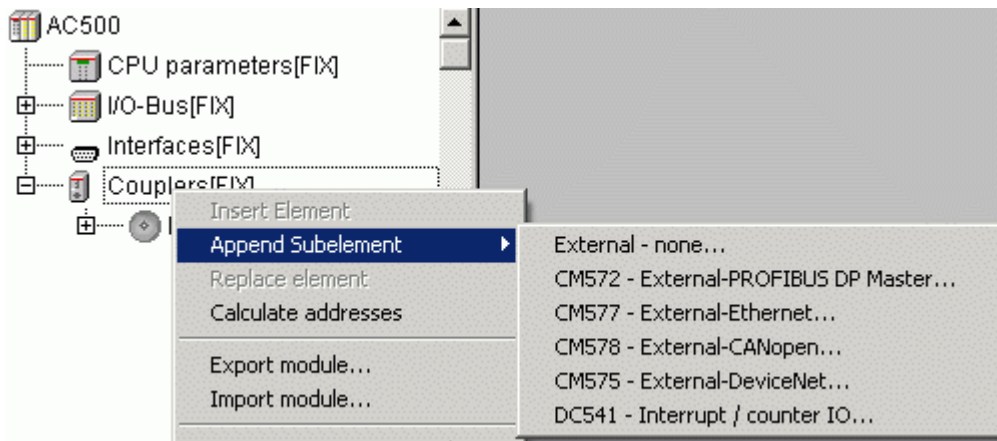
For an AC500 controller with external couplers, the couplers must be specified in the PLC configuration in the same order as they are installed in the hardware.

The following assignment between coupler line and slot applies to the couplers:

- Line 0 corresponds to the internal coupler (installed in the CPU's housing)
- Line 1 is the coupler installed in the first slot on the left of the CPU
- Line 2 is the coupler installed in the second slot on the left of the CPU
- Line 3 and 4 are installed in the third and fourth slot on the left of the CPU

The allocation of inputs/outputs for the couplers is done slot-oriented and independent of the coupler type.

To append the external coupler, right-click the element "Couplers" in the configuration tree and select the sub menu "Append Subelement". All available external couplers are shown:



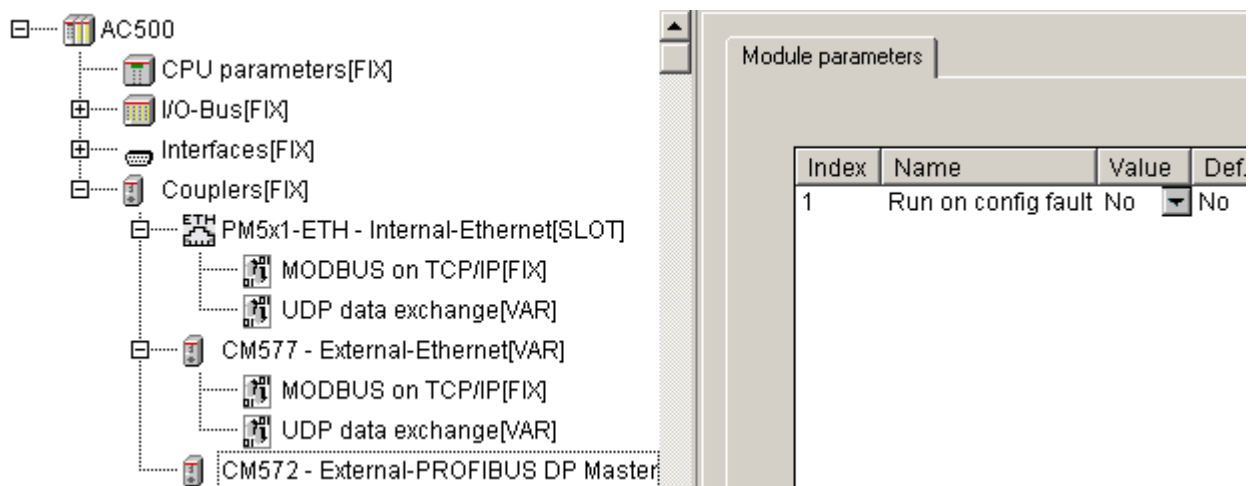
**Note:** The external couplers "CM578 - External CANopen", "CM575 - External DeviceNet" and the module "DC541 - Interrupt / counter IO" are available as of version V1.1.

For external couplers, the same applies as for internal couplers: only the behavior of the CPU in case of a coupler configuration error and the required protocols are set in the PLC configuration. The actual configuration of the external couplers is done with the integrated fieldbus configurator SYCON.net. For a detailed description of this procedure refer to the chapter "System Technology of the Couplers" (see also System Technology of the Couplers).

The following settings are possible:

Parameter	Default value	Value	Meaning
Run config fault	No	No	In case of a configuration error, the user program is not started.
		Yes	The user program is started independent of a faulty configuration of the particular external coupler.

The following figure shows an example of a PLC configuration consisting of an internal Ethernet coupler "PM5x1-ETH - Internal-Ethernet", an external Ethernet coupler "CM577 - External-Ethernet" and a PROFIBUS DP master coupler "CM572 - External-PROFIBUS DP Master":



PLCconf\_Coupler6\_E.gif

For the external Ethernet coupler "CM577 - External-Ethernet", the same parameters and protocol settings are possible as for the internal Ethernet coupler "PM5x1 - Internal-Ethernet" (see also Internal Ethernet coupler).

## 4 System start-up / program processing

### 4.1 Terms

#### **Cold start:**

- A cold start is performed by switching power OFF/ON if no battery is connected.
- All RAM memory modules are checked and erased.
- If no user program is stored in the Flash EPROM, the default values (as set on delivery) are applied to the interfaces.
- If there is a user program stored in the Flash EPROM, it is loaded into RAM.
- The default operating modes set by the PLC configuration are applied.

#### **Warm start:**

- A warm start is performed by switching power OFF/ON with a battery connected.
- All RAM memory modules are checked and erased except of the buffered operand areas and the RETAIN variables.
- If there is a user program stored in the Flash EPROM, it is loaded into RAM.
- The default operating modes set by the PLC configuration are applied.

#### **RUN -> STOP:**

- RUN -> STOP means pressing the RUN key on the PLC while the PLC is in RUN mode (PLC display "run").
- If a user program is loaded into RAM, execution is stopped.
- All outputs are set to FALSE or 0.
- Variables keep their current values, i.e., they are not initialized.
- The PLC display changes from "run" to "StoP".

#### **START -> STOP:**

- START -> STOP means stopping the execution of the user program in the PLC's RAM using the menu item "Online/Stop" in the programming system.
- All outputs are set to FALSE or 0.
- Variables keep their current values, i.e., they are not initialized.
- The PLC display changes from "run" to "StoP".

#### **Reset:**

- Performs a START -> STOP process.
- Preparation for program restart, i.e., the variables (VAR) are set to their initialization values.
- Reset is performed using the menu item "Online/Reset" in the programming system.

#### **Reset (cold):**

- Performs a START -> STOP process.
- Preparation for program restart, i.e., the variables (VAR) are set to their initialization values.
- Reset (cold) is performed using the menu item "Online/Reset (cold)" in the programming system.

#### **Reset (original):**

- Resets the controller to its original state (deletion of Flash, SRAM (%M, RETAIN), coupler configurations and user program!).
- Reset (original) is performed using the menu item "Online/Reset (original)" in the programming system.

#### **STOP -> RUN:**

- STOP -> RUN means pressing the RUN key on the PLC while the PLC is in STOP mode (PLC display "StoP").

- If a user program is loaded into RAM, execution is continued, i.e., variables will not be set to their initialization values.
- The PLC display changes from "StoP" to "run".

### **STOP -> START:**

- STOP -> START means continuing the execution of the user program in the PLC's RAM using the menu item "Online/Start" in the programming system.
- If a user program is loaded into RAM, execution is continued, i.e., variables will not be set to their initialization values.
- The PLC display changes from "StoP" to "run".

### **Download:**

- Download means loading the complete user program into the PLC's RAM. This process is started by selecting the menu item "Online/Download" in the programming system or after confirming a corresponding system message when switching to online mode (menu item "Online/Login").
- Execution of the user program is stopped.
- In order to store the user program to the Flash memory, the menu item "Online/Create boot project" must be called after downloading the program.
- Variables are set to their initialization values according to the initialization table.
- RETAIN variables can have wrong values as they can be allocated to other memory addresses in the new project!
- A download is forced by the following:
  - changed PLC configuration
  - changed task configuration
  - changed library management
  - changed compile-specific settings (segment sizes)
  - execution of the commands "Project/Clean all" and "Project/Rebuild All".

### **Online Change:**

- After a project has changed, only these changes are compiled when pressing the key <F11> or calling the menu item "Project/Build". The changed program parts are marked with a blue arrow in the block list.
- The term Online Change means loading the changes made in the user program into the PLC's RAM using the programming system (after confirming a corresponding system message when switching to online mode, menu item "Online/Login").
- Execution of the user program is not stopped. After downloading the program changes, the program is re-organized. During re-organization, no further online change command is allowed. The storage of the user program to the Flash memory using the command "Online/Create boot project" cannot be initiated until re-organization is completed.
- Online Change is not possible after:
  - changes in the PLC configuration
  - changes in the task configuration
  - changes in the library management
  - changed compile-specific settings (segment sizes)
  - performing the commands "Project/Clean all" and "Project/Rebuild All".

### **Data buffering:**

- Data buffering, i.e., maintaining data after power ON/OFF, is only possible, if a battery is connected. The following data can be buffered completely or in parts:
  - Data in the addressable flag area (%M area)
  - RETAIN variable
  - PERSISTENT variable (number is limited, no structured variables)
  - PERSISTENT area (%R area, as of V1.2)
- In order to buffer particular data, the data must be excluded from the initialization process.

## 4.2 Start of the user program

The user program (UP) is started according to the following table. Here it is assumed that a valid user program is stored to the Flash memory.

Action	No SD card with UP 1) installed, Auto run = <b>ON</b>	No SD card with UP 1) installed, Auto run = <b>OFF</b>	SD card with UP 1) installed, Auto run = <b>ON</b>	SD card with UP 1) installed, Auto run = <b>OFF</b>
<b>Voltage ON</b> or <b>Warm start</b>  or <b>Cold start</b>	UP is loaded from Flash into RAM and started from Flash.	No UP is loaded from Flash. When logging in, the message "No program available in the controller ..." is displayed.	UP is loaded from the SD card into Flash memory and RAM and then started from RAM.	UP is loaded from the SD card to the Flash memory. RAM remains empty. When logging in, the message "No program available in the controller ..." is displayed.
<b>STOP -&gt; RUN</b>	UP in RAM is started.	UP in RAM is started.	UP in RAM is started.	UP in RAM is started.
<b>STOP -&gt; START</b>	UP in RAM is started.	UP in RAM is started.	UP in RAM is started.	UP in RAM is started.
<b>Download 2)</b>	The UP currently stored in the CPU's RAM is stopped. The built UP is loaded from the PC into the PLC's RAM.	The built UP is loaded from the PC into the PLC's RAM.	The UP currently stored in the CPU's RAM is stopped. The built UP is loaded from the PC into the PLC's RAM.	The built UP is loaded from the PC into the PLC's RAM.
<b>Online Change 3)</b>	Processing of the UP currently stored in the CPU's RAM is continued. The changes made to the UP are loaded from the PC into the PLC's RAM. The UP is re-organized and processed.	The changes made to the UP are loaded from the PC into the PLC's RAM. The UP is re-organized.	Processing of the UP currently stored in the CPU's RAM is continued. The changes made to the UP are loaded from the PC into the PLC's RAM. The UP is re-organized and processed.	The changes made to the UP are loaded from the PC into the PLC's RAM. The UP is re-organized.

### Remarks:

- 1) The version of the PLC operating system used to generate the SD card and the version of the PLC operating system to which the UP is to be loaded from the SD card must be the same. If the versions differ, the SD card is not loaded.
- 2) After the download is completed, the program is not automatically stored to the Flash memory. To perform this, select the menu item "Online/Create boot project". If the UP is not stored to the Flash memory, the UP is reloaded from the Flash memory after voltage OFF/ON.

The program is started either by pressing the RUN/STOP key or using the menu item "Online/Start" in the programming system.

- 3) After the Online Change process is completed, the program is not automatically stored to the Flash memory. To perform this, select the menu item "Online/Create boot project" after re-organization is completed. During re-organization and flashing, no further online change command is allowed.

If the UP is not stored to the Flash memory, the UP is reloaded from the Flash memory after voltage OFF/ON.

## 4.3 Data backup and initialization

### 4.3.1 Initialization of variables, overview

The initialization of variables to 0 or to the initialization value is performed by switching voltage ON, by a reset or after downloading the user program.

If internal variables shall be buffered, these variables have to be marked as "VAR\_RETAIN" or "VAR\_RETAIN PERSISTENT". This applies to both the internal variables and the variables of the addressable flag area (%M area).



**Note:** The order of the internal RETAIN variables is only kept when using the online change command. If the program is re-built, the order can change and, due to this, the buffered variables do not match.

See also CoDeSys / Retentive variables, chapter "Behavior of RETAIN variables on download".

The following table shows an overview of the initialization values of the individual variables:

Variable	Action		
VAR		0	Voltage ON
VAR := Value		unch.	STOP → RUN (pushbutton)
VAR %MDx.x		unch.	STOP → START (PC)
VAR %MDx.x := Value		0	Download
VAR_RETAIN		unch.	Online Change
VAR_RETAIN := Value		0	Reset
VAR_RETAIN %MDx.x		0	Reset (cold)
VAR_RETAIN %MDx.x := Value		0	Voltage OFF/ON after Reset (cold)
VAR_PERSISTENT		???	Reset (origin), factory setting
VAR_PERSISTENT := Value		0	Download after Reset (origin)
VAR_PERSISTENT %MDx.x		0	
VAR_PERSISTENT %MDx.x := Value		0	
VAR_RETAIN PERSISTENT		0	
VAR_RETAIN PERSISTENT := Value		0	
VAR_RETAIN PERSISTENT %MDx.x		0	
VAR_RETAIN PERSISTENT %MDx.x := Value		0	
without battery      unch. = unchanged			
with battery      unch. = unchanged			
		unch.	Voltage ON
		unch.	STOP → RUN (pushbutton)
		unch.	STOP → START (PC)
		0	Download
		unch.	Online Change
		0	Reset
		0	Reset (cold)
		0	Voltage OFF/ON after Reset (cold)
		???	Reset (origin), factory setting
		0	Download after Reset (origin)
		0	
		unch.	Voltage ON
		unch.	STOP → RUN (pushbutton)
		unch.	STOP → START (PC)
		0	Download
		unch.	Online Change
		0	Reset
		0	Reset (cold)
		0	Voltage OFF/ON after Reset (cold)
		???	Reset (origin), factory setting
		0	Download after Reset (origin)

### 4.3.2 Notes regarding the declaration of retentive variables and constants

To guarantee the correct initialization or backing up of variables according to the table shown above, the following rules have to be observed when declaring variables.

#### **Declaration of retentive internal variables:**

The variables have to be declared as **VAR\_RETAIN** or **VAR\_GLOBAL RETAIN**.

#### **Example:**

(\* Declaration in the global variable lists \*)

```
VAR_GLOBAL RETAIN
    byVar : BYTE;
    wVar : WORD;
    rVar : REAL;
END_VAR
```

(\* Declaration in the program \*)

```
VAR RETAIN
    byVar1 : BYTE;
END_VAR
```

#### **Declaration of retentive variables in %M area:**

The variables have to be declared as **VAR\_RETAIN** or **VAR\_GLOBAL RETAIN**.



**Note:** As of Control Builder version V1.2 and firmware version V1.2.0, a new persistent area is available which can be buffered by a specific setting in the PLC configuration (see also chapter "The addressable PERSISTENT area").

#### **Declaration of constants:**

Constants are declared as **VAR\_GLOBAL CONSTANT** or **VAR\_CONSTANT**.

Example:

(\* Declaration as global constants \*)

```
VAR_GLOBAL CONSTANT
    byConst_1 : BYTE := 1;
END_VAR
```

(\* Declaration in the program \*)

```
VAR CONSTANT
    byConst_2 : BYTE := 2;
END_VAR
```



**Note:** Using the option "Replace constants" available under "Project" => "Options" => "Build", it is possible to specify whether constants are treated as variables (i.e., writing the variable is possible) or the value is directly entered into the code when building the project.



## 4.4 Processing times

### 4.4.1 Terms

The most important times for the use of the AC500 basic unit with or without connected remote modules are:

- The **reaction time** is the time between a signal transition at the input terminal and the signal response at the output terminal.  
For binary signals, the reaction time is composed of the input delay, the cycle time of the program execution and the bus transmission time if the system is expanded by remote modules.
- The **cycle time** determines the time intervals after which the processor restarts the execution of the user program.  
The cycle time has to be specified by the user. It should be greater than the program processing time of the user program plus the data transfer times and the related waiting times.  
The cycle time is also the time base for some time-controlled functions, such as for the INTK.
- The **program processing time** is the net time for processing the user program.

### 4.4.2 Program processing time

Statements	PM57x	PM58x	PM59x
<b>- Binary statements of the type:</b>			
!M /M &M =M !NM /NM &NM =NM !M /M &M =SM !NM /NM &NM =RM Processing time for 1000 statements:	0.3 ms	0.15 ms	0.02 ms
<b>- Word statements of the type:</b>			
!MW +MW -MW =MW !-MW -MW +MW =-MW !MW *MW :MW =MW !-MW *-MW :-MW =-MW Processing time for 1000 statements:	0.3 ms	0.15 ms	0.01 ms
<b>- Floating point:</b>			
Processing time for 1000 statements:	6 ms	3 ms	0.02 ms

### 4.4.3 Set cycle time

It is assumed that the processor always gets access in a moment with a worst-case condition.

The cycle time is stored in the task configuration and can be selected in steps of 1 ms. If the selected cycle time is too short, the processor will not be able to completely process the tasks assigned to it every cycle. This will result in a time delay.

If this lack of time becomes too large over several cycles, the processor aborts the program execution and outputs an error (E2).

For some function blocks, such as the PID controller, the error-free execution depends on an exact timing sequence. Make sure that there is a larger time reserve.

To check the correct cycle time, perform the following steps:

- Load the user program into the basic unit.
- Check the capacity utilization with "Online/PLC Browser/cpload".
- Change the cycle time until the capacity utilization is below 80 %.

**When setting the cycle time, consider the following values:**

- Time for reading and copying the input signals from the I/O driver to the I/O image.
- Time for copying the input signals of the user task from the I/O image to the image memory.
- Program processing time
- Time for copying the output signals of the user task from the image memory to the I/O image.
- Time for copying the output signals from the I/O image to the I/O driver and applying the I/Os to the I/O module.
- Receiving/sending interrupts from coupler telegrams within the cycle time.
- Receiving/sending interrupts from the serial interface within the cycle time.
- Task changes.
- Runtime of the watchdog task.

## 4.5 Task configuration for the AC500 CPU

How to use the task configuration for the Control Builder is described in detail in the chapter "Resources / Task configuration" (see also 3S: CoDeSys Programming System / Resources / Task configuration).

This section describes the specialities for the task configuration for the AC500.

The possible number of tasks depends on the CPU type. For PM571 and PM581, a maximum of 3 user tasks can be created, for PM591 a maximum of 16 user tasks is allowed.

If **no task configuration** is specified in the project, a task with the following properties is created automatically.

Type = cyclic  
Priority = 10  
Cycle time = t#10ms  
Program call= PLC\_PRG.

In version 1.0, the following task types are possible for the AC500 CPU: "**cyclic**" and "**free running**". The types "event triggered" and "external event triggered" are not possible.

As of Control Builder version V1.1 and firmware version V1.1.7, also the task type "**external event triggered**" can be selected for the interrupt and counting device DC541 (see also System Technology DC541 - Interrupt-Mode).

All 32 priorities can be selected for the user tasks, where 0 is the highest priority and 31 the lowest. The default priority is 10.

Priorities **lower than 10** are reserved for high-priority processes with a **very short program execution time**. The priorities 10 to 31 are intended for "normal" user tasks or tasks with a long program execution time.



**Caution:** Using, for example, a priority lower than 10 for a task with a long program execution time can cause, for example, the CS31 bus and/or the FBP interface to fail.

## 5 The diagnosis system in the AC500

### 5.1 Summary of diagnosis possibilities

#### 5.1.1 Structure of the diagnosis system

The AC500 contains a diagnosis system that allows to manage up to 100 error messages in a circular buffer. For each of these events, a time stamp with date and time based on the controller's real-time clock (RTC) is generated in the runtime system. The time stamp consists of three entries:

- Error occurrence (come)
- Error disappearance (gone)
- Error acknowledgement

If no battery is inserted in the PLC, the PLC clock is set to the following value when switching on the control voltage:

01. January 1970, 00:00

Each error message has a unique error number. This number provides the following information:

- State (come, gone, acknowledged)
- Error class
- Faulty component
- Faulty device
- Faulty module
- Faulty channel
- Error identifier

For a description of the error numbers, please refer to section Organization and structure of error numbers later in this chapter.

The error numbers are divided into the following error classes:

Class	Type	Description	Example
E1	Fatal error	Safe operation of the operating system is no longer ensured.	Checksum error in system Flash, RAM error
E2	Serious error	The operating system works correctly, but the error-free execution of the user program is not ensured.	Checksum error in user Flash, task cycle times exceeded
E3	Light error	It depends on the application whether the user program has to be stopped by the operating system or not. The user decides which reaction is to be done.	Flash memory cannot be programmed, I/O module failed
E4	Warning	Errors that occur on peripheral devices or that will have an effect only in the future. The user decides which reactions are to be done.	Short circuit in an I/O module, battery empty/not installed

There are different possibilities to access the error messages:

- Diagnosis directly at the PLC by means of "ERR" LED, keypad and display
- Plain-text display of the error messages in the status line of the Control Builder in online mode
- Diagnosis with the PLC browser commands of the Control Builder
- Diagnosis with help of the user program using the diagnosis blocks of the library SysInt\_AC500\_Vxx.LIB

### 5.1.2 Diagnosis directly at the PLC by means of "ERR" LED, keypad and display

If the PLC contains a non-acknowledged error, the red error LED "ERR" lights up.



**Note:** The CPU parameter "Error LED" in the PLC configuration allows to set for which error class the LED indicates an error. The default setting is "On", i.e., errors of all error classes are indicated. If the parameter is set for example to "Off\_by\_E3", the LED "ERR" does not light up in case of errors of the classes E3 and E4. However, if an E2 error occurs, the LED always lights up. See also chapter "Configuration of the CPU parameters".

If one or several non-acknowledged errors exist, the errors can be displayed and acknowledged according to their occurrence order using the <DIAG> key. Pressing the <DIAG> key the first time displays the error class and error identifier. After this, pressing the <DIAG> key several times browses through the detail information d1=component, d2=device, d3=module and d4=channel. If the "d4" information is displayed and the <DIAG> key is pressed once more, the error class/error identifier is re-displayed.

If you quit the diagnostic display by pressing <ESC>, the error is not acknowledged and displayed again when pressing the <DIAG> key.

If you quit the diagnostic display with the <QUIT> key, the error is acknowledged.

The LED "ERR" goes off when all errors are acknowledged.

#### Example:

The error "Battery empty or not installed" appears in the PLC display as follows:

Pushbutton	Display	Meaning
<DIAG>	E4 008	E4=Warning / Identifier = Empty/Not available
<DIAG>	d1 009	Detail information d1 = 009 -> Component=CPU
<DIAG>	d2 022	Detail information d2 = 022 -> Device=Battery
<DIAG>	d3 031	Detail information d3 = 031 -> Module=no specification
<DIAG>	d4 031	Detail information d4 = 031 -> Channel=no specification
<DIAG>	E4 008	E4=FK4 / Identifier = Empty/Not available
<ESC>	run/StoP	Diagnostic display is quit without error acknowledgement.
<DIAG>	E4 008	E4=FK4 / Identifier = Empty/Not available
<QUIT>	run/StoP	Diagnostic display is quit with error acknowledgement. If no further non-acknowledged errors exist, the LED "ERR" goes off.

For a description of the error numbers refer to the chapter Organization of the error numbers.

### 5.1.3 Plain-text display of error messages in the Control Builder status line during online mode

When the Control Builder is switched to online mode, incoming error messages or status changes of an error message (come, gone, acknowledged) are displayed as plain-text in the status line. For example, the acknowledgment of the error message "Battery empty or not installed" described in the previous section is displayed in online mode as follows:

```
#152502216: x 1970-01-01 06:33:53 E4 :' No battery or battery empty
```

**With:**

#152502216	Online error number
x	Error acknowledged (+ = Error come, - = Error gone)
1970-01-01 06:33:53	Time stamp (time of acknowledgement)
E4 :	Error class 4 = Warning
No battery or battery empty	Error text (according to language setting for the Control Builder)



**Note:** The error text is read from the file Errors.ini and displayed according to the online error number. The language of the error text depends on the language setting for the Control Builder. Errors which do not have an entry in the file Errors.ini are displayed without error text. The file Errors.ini is part of the Target Support Package (TSP) and located in the directory ..\Targets\ABB\_AC500 or ..\Targets\ABB\_AC500\AC500\_V12.

### 5.1.4 Diagnosis using the PLC browser commands of the Control Builder

All errors or errors of a certain error class can be displayed and/or acknowledged using the PLC browser of the Control Builder. Also the complete diagnosis system can be deleted.

The PLC browser commands are described in detail in the chapter AC500-specific PLC browser commands.

### 5.1.5 Diagnosis with help of the user program

The entries in the diagnosis system can also be accessed from the user program with the help of specific diagnosis blocks. These blocks are described in the chapter The diagnosis blocks of the AC500.

## 5.2 Organization and structure of error numbers

For each error, an error number is stored in the firmware. This error number is coded as follows:

Status	Error class	Faulty component	Faulty device	Faulty module	Faulty channel	Error identifier
	28...29	24...27	16...23	11...15	6...10	0...5
4 bits	2 bits	4 bits	8 bits	5 bits	5 bits	6 bits
	0...3	0...15	0...255	0...31	0...31	0...63

Status value	Meaning
Bit 0	not used
Bit 1	Error occurrence (come)
Bit 2	Error removed (gone)
Bit 3	Error acknowledgement

In addition to the error information, the diagnosis message also contains status information (1 bit per status). Each status is set by a specific event:

- Error occurrence (come)
- Error acknowledgement
- Error removal (gone)

The diagnosis message is generated when an error occurs. In this case, the status bit 1 is set. If this error is acknowledged or removed afterwards, the corresponding status bits are set additionally.

In the Control Builder, the **online error number** is displayed. This error number is sent to the Control Builder when working in online mode and decoded **language-dependent** using the controller description file **Errors.ini**. The error number is coded as follows:

Faulty component	Faulty device	Faulty module	Faulty channel	Error identifier
24...27	16...23	11...15	6...10	0...5
4 bits	8 bits	5 bits	5 bits	6 bits
0...15	0...255	0...31	0...31	0...63

The error status (come, gone, acknowledged) and the error class are hidden in the online error number and displayed as plain-text.

### 5.2.1 Error classes

The error classes are coded as follows:

Value error class			Meaning
Bit 29	Bit 28	Class	
0	0	1	E1 fatal errors
0	1	2	E2 serious errors
1	0	3	E3 light errors
1	1	4	E4 warnings

### 5.2.2 Error identifiers

The following error identifiers are defined. The identifiers are kept generally in order to reach a maximum systematic. The exact meaning of each error depends on the further information provided by the error messages. For example, the error identifiers 'Highest level' and 'Lowest level' for analog channels correspond to the 'Out of Range' message.

Error Identifier		
Err	Meaning	Meaning
0	Fehler allgemein	General
1	Fehler falscher Wert	Wrong value
2	Fehler ungültiger Wert	Invalid value
3	Fehler Timeout	Timeout
4	Fehler oberster Grenzwert	Highest level
5	Fehler oberer Grenzwert	High level
6	Fehler unterer Grenzwert	Low level
7	Fehler unterster Grenzwert	Lowest level
8	Fehler leer/fehlt	Empty or missing
9	Fehler voll	Full
10	Fehler zu groß	Too big
11	Fehler zu klein	Too small
12	Fehler beim Lesen	Read
13	Fehler beim Schreiben	Write
14	Fehler beim Löschen	Delete
15	Fehler beim Reservieren von Speicher	Alloc memory
16	Fehler beim Freigeben von Speicher	Free memory
17	Fehler beim Zugriff	Access
18	Fehler beim Testen	Test
19	Fehler Checksumme	Checksum
20	Fehler Message	Message
21	Fehler bei PutMessage	Put message
22	Fehler bei GetMessage	Get message
23	Fehler Warten auf freie Message	Wait message
24	Fehler Message gelöscht	Message deleted
25	Fehler Warten auf Antwort	Wait answer
26	Fehler Konfiguration	Config data
27	Fehler keine Konfiguration	No config
28	Fehler Unterschied Soll-/Ist-Konfiguration	Different config
29	Fehler beim Schreiben der Konfiguration	Write config
30	Fehler beim Lesen der Konfiguration	Read config
31	Fehler anderer Typ / anderes Modell	Wrong type or model
32	Fehler unbekannter Typ / unbekanntes Modell	Unknown type or model
33	Fehler WaitReset	Wait reset
34	Fehler WaitReady	Wait ready
35	Fehler WaitRun	Wait run
36	Fehler WaitCom	Wait com
37	Fehler Zykluszeit	Cycle time
38	Fehler Exception	Exception
39	Fehler unbekannter Baustein	Unknown POU
40	Fehler Version	Version
41	Fehler Übertragung	Transmit
42	Fehler Empfang	Receive
43	Fehler intern	Internal
44	Fehler keine Abgleichwerte	No adjustment values
45	Drahtbruch	Cut wire
46	Überlast	Overload
47	Kurzschluss	Short circuit
48	Überlast / Drahtbruch	Overload / Cut wire
49	Kurzschluss / Drahtbruch	Short-circuit / Cut wire
50	Überlast / Kurzschluss	Overload / Short-circuit
51	Überlast / Kurzschluss / Drahtbruch	Overload / Short-circuit / Cut wire
63 (max.)	weitere	others

### 5.2.3 Possible error numbers

The following tables contain the possible combinations of error numbers.

Component		Device		Module or type		Channel		Remark				
No.	Comp	No.	Dev	No.	Mod	No.	Ch	<- PLC browser				
	d1		d2		d3		d4	<- Display				
9	CPU	0	CPU	1	Operating system	1	Initialization error					
							2	Runtime error				
							3	Configuration error				
							31	Operating system				
							2	Runtime system	1	Initialization error		
										2	Runtime error	
				3	Configuration error							
				31	Operating system							
				4	IEC task online display %s	1	Initialization error					
							2	Runtime error				
				1	External coupler 1...6 or internal			1	Initialization	0	not used	
								2	Runtime error			
		3	Configuration					26,				
		4	Protocol									
		31	Coupler									
		2	External coupler 2			1	Initialization	0	not used			
						2	Runtime error					
						3	Configuration					
						4	Protocol					
						31	Coupler					
		3	External coupler 3			1	Initialization	0	not used			
						2	Runtime error					
						3	Configuration			26,		
						4	Protocol					
						31	Coupler					
		4	External coupler 4			1	Initialization	0	not used			
						2	Runtime error					
						3	Configuration			26,		
						4	Protocol					
						31	Coupler					
		10	Internal coupler			1	Initialization	0	not used			
						2	Runtime error					
						3	Configuration			26,		
						4	Protocol					
						31	Coupler					
		11	COM1			1	Initialization	0	not used			
						2	Runtime error					
						3	Configuration			26,		
						4	Protocol					
						31	COM					
		12	COM2			1	Initialization	0	not used			
						2	Runtime error					
						3	Configuration			26,		
						4	Protocol					
						31	COM					
		13	FBP			1	Initialization	0	not used			
						2	Runtime error					
						3	Configuration			26,		
4	Protocol											
31	FBP											



		14	I/O bus	1	Initialization	0	not used	18, 15
				2	Runtime error			
				3	Configuration			
				4	Protocol			
				31	I/O bus			
		16	System EPROM	0...31	Sector, block no. or similar	0...31	Sector, block no. or similar (%s)	
		17	RAM	0...31		0...31		
		18	Flash EPROM	0...31		0...31		
		19	HW watchdog	31	Watchdog	31	Watchdog	
		20	SD Memory Card	1	Initialization	0...31	Sector, block no. or similar (%s)	
				2	Runtime error	0...31		
				3	Configuration	0...31		
				4	Protocol	0...31		
				31	SD card	0...31		
		21	Display	1	Initialization	0	not used	
				2	Runtime error			
				4	Protocol			
				31	Display			
		22	Battery	31	Battery	31	Battery	8,
		23	Clock	1	Initialization	0	not used	
				2	Runtime error			
				3	Configuration			
4	Protocol							
31	Clock							

Component		Device		Module or type		Channel		Remark				
No.	Comp	No.	Dev	No.	Mod	No.	Ch	<- PLC browser				
	d1		d2		d3		d4	<- Display				
1..4 10	External coupler 1..4 or internal	0..254	Address/ Socket: Fieldbus: Slave ARCNET: ID partner - 1 Modbus: Comm partner	0..29	Module number	0..31	Channel number					
				30	Module number > 29	0..31	Channel number					
				31	Slave device	31	Slave device					
		255	Coupler	1	Initialization	0	not used	0...15 Bit 4...7 of the error number reported by the coupler  see chapter "Coupler errors"	0...15 Bit 0...3 of the error number reported by the coupler  see chapter "Coupler errors"			
				2	Runtime error							
				3	Configuration							
				4	Protocol							
				5	Operating system coupler							
				6	Task 1 coupler							
				7	Task 2 coupler							
				8	Task 3 coupler							
				9	Task 4 coupler							
				10	Task 5 coupler							
				11	Task 5 coupler							
12	Task 7 coupler											
13	Watchdog coupler											
31	Coupler											
11	COM1	0..254	Address: CS31: Slave Dec. expansion: Slave Modbus: Comm partner	0..29	Module number CS31: Module type: 00 - Digital input 01 - Analog input 02 - Digital output 03 - Analog output 04 - Digital in/output 05 - Analog in/output	0..31	Channel number	8, 48				
				30	Module number > 29	0..31	Channel number					
				31	Slave device	31	Slave device					
		255	COM	1	Initialization	0	not used	0...15 Bit 4...7 of the error number reported by the coupler  see chapter "Coupler errors"	0...15 Bit 0...3 of the error number reported by the coupler  see chapter "Coupler errors"			
				2	Runtime error							
				3	Configuration							
				4	Protocol							
				31	COM							
		12	COM2	0..254	Address: CS31: Slave Dec. expansion: Slave Modbus: Comm partner	0..29	Module number			0..31	Channel number	
						30	Module number > 29			0..31	Channel number	
31	Slave device					31	Slave device					
255	COM			1	Initialization	0	not used			0...15 Bit 4...7 of the error number reported by the coupler  see chapter "Coupler errors"	0...15 Bit 0...3 of the error number reported by the coupler  see chapter "Coupler errors"	
				2	Runtime error							
				3	Configuration							
				4	Protocol							
31	COM											

Component		Device		Module or type		Channel		Remark					
No.	Comp	No.	Dev	No.	Mod	No.	Ch	<- PLC browser					
	d1		d2		d3		d4	<- Display					
13	FBP	0...254	Module number Parameter number	0..30	Slot number	0..31	Channel number						
									255	FBP	1	Initialization	0
		2	Runtime error										
		3	Configuration										
		4	Protocol										
31	FBP												
14	I/O bus	0...254	I/O bus module	0..6	Module type: 00 - Digital input 01 - Analog input 02 - Digital output 03 - Analog output 04 - Digital in/output 05 - Analog in/output 06 - others (e.g., fast counter)	0..31	Channel number	%s					
									31	Module	1	Initialization error	
											2	Runtime error	
											3	Configuration	26,
											4	Protocol	
		31	Module										
		255	I/O bus	1	Initialization	0	not used						
								2	Runtime error				
								3	Configuration				
								4	Protocol				
31	I/O bus												
15	User	0...255	any	0..31	any	0..31	any, meaning is project-specific						

## 5.2.4 List of all errors

E1..E4	d1	d2	d3	d4	Identifier 000...063	AC500 display	<- displayed in 5)
Class	Comp	Dev	Mod	Ch	Err	PS501 PLC browser	
Byte 6 Bit 6..7	-	Byte 3	Byte 4	Byte 5	Byte 6 Bit 0..5	FBP- diagnosis block	
Class	Inter- face	De- vice	Mod- ule	Chan- nel	Error identifier	Error message	Remedy
	1)	2)	3)	4)			
<b>AC500 CPU errors</b>							
<b>Errors directly reported by the CPU</b>							
<b>Serious errors</b>							
2	9	0	2	0	15	Not enough memory to generate the external reference list	
2	9	0	2	2	37	Cycle time is greater than the set watchdog time	Change task configuration
2	9	0	2	2	38	Access violation by an IEC task (for example zero pointer) (detail in call hierarchy as of V1.2)	Correct program
2	9	1..4/10	1	0	15	Watchdog task could not be installed	Check coupler
2	9	1..4/10	1	0	15	Installation of the coupler bus driver failed	Check coupler
2	9	1..4/10	1	0	15	Initialization error, not enough memory	- Check coupler - Check CPU FW version
2	9	1..4/10	1	0	17	Accessing test to the coupler failed	- Check coupler - Check CPU FW version
2	9	1..4/10	1	0	18	Watchdog test for the coupler failed	Check coupler
2	9	1..4/10	1	0	34	Timeout when setting the warm start parameters of the coupler	Check coupler
2	9	1..4/10	1	0	38	Installation of the coupler driver failed	Check coupler and FW version
2	9	1..4/10	2	0	15	Error occurred when creating the I/O description list of the coupler	Check coupler and FW version
2	9	1..4/10	4	0	15	Installation of a driver for the coupler failed	Check coupler and FW version
2	9	1..4/10	31	0..7	3	Watchdog error coupler/channel	Check coupler and FW version
2	9	11..13	1	0	15	Installation of a protocol driver for the serial interface failed, not enough memory	Check CPU FW version
2	9	11..13	1	0	17	Initialization of the protocol driver for the serial interface failed	Check CPU FW version
2	9	11..13	1	0	38	Installation of the hardware for the serial interface failed	Check CPU FW version
2	9	14	1	0	15	Not enough resources for the I/O-Bus	Check CPU FW version
2	9	14	1	0	17	Installation of the I/O-Bus driver failed	Check CPU FW version
2	9	14	1	0	43	Incorrect data format of the hardware driver of the I/O-Bus	Check CPU FW version
2	9	24	2	1	38	FPU division by zero	Clear up program
2	9	24	2	2	38	FPU overflow	Clear up program
2	9	24	2	3	38	FPU underflow	Clear up program
2	9	24	2	4	38	Forbidden FPU operation (e.g. 0/0)	Clear up program
2	9	24	2	6	38	FPU library function generated	Clear up program
<b>Light errors</b>							
3	9	0	2	1	17	Registering the handler for persistent data areas, areas do not work correctly	Check CPU FW version
3	9	0	2	2	37	User program contains an endless loop, a stop by hand is necessary	Correct user program
3	9	0	2	3	26	Configuration error	Adapt PLC configuration

3	9	0	4..31	3	26	Event-controlled task, unknown external event	Check task configuration
3	9	1..4/10	2	0	12	Error occurred when reading the I/O description	Check coupler configuration
3	9	1..4/10	2	0	26	Program was not started because of invalid configuration data of the coupler	Configure coupler
3	9	1..4/10	2	0	29	Error occurred when deleting the configuration	Check coupler
3	9	1..4/10	3	0	26	In the PLC configuration, the coupler was not configured correctly or not at all	Adapt PLC configuration
3	9	11..13	3	0	26	Configuration error of the serial interface	Check configuration
3	9	14	3	0	26	Parameter "Error LED"=Failsafe is only allowed, if parameter "Behaviour of outputs in stop=actual state in hardware and online"	Change configuration
3	9	16	1	0	13	Deleting of the boot project failed (possibly no valid boot project available)	Reload project
3	9	21	31	0	17	Display could not be installed	Check FW
<b>Warnings</b>							
4	9	0	2	2	20	Program not started because of an existing error (see PLC configuration CPU parameters, stop on error class)	Eliminate error and acknowledge
4	9	0	2	2	37	Cycle time exceeded, but shorter than watchdog time	Adapt task configuration
4	9	0	2	2	43	Control system was restarted by FK2 or power dip according to PLC configuration	
4	9	0	3	1	40	Boot code versions V1.1.3. (or older versions) support smaller RAM disk	Update boot code to 1.2.0
4	9	1..4/10	1	0	32	Installation of the coupler driver failed, unknown coupler type	Check configuration
4	9	1..4/10	2	0	3	Within the specified time, connection could not be established to all of the slaves, I/O data may be (partly) invalid	Check slave for existence or set times according to the slave behaviour
4	9	1..4/10	2	0	4	No socket available	Check coupler settings with PLC browser
4	9	18	1	0	17	PLC Config file could not be read	Reload project
4	9	18	2	0	8	Error firmware update of SD card, file could not be opened	Check SD card, e.g. removed without "ejected"
4	9	20	1	2	2	Invalid value for FunctionOfCard in SDCARD.INI, value will be ignored	Check file SDCARD.INI on the SD card
4	9	20	1	10..14	8	Error Firmware update of the SD card, file could not be opened	Check SD card
4	9	20	1	10..14	12	Error Firmware update of the SD card, error while reading the file	Check SD card
4	9	20	1	10..14	13	Error Firmware update of the SD card, error while writing the file	Check CPU + coupler
4	9	20	1	10..14	17	Error Firmware update of the SD card, error while accessing the coupler	Check coupler
4	9	20	1	10..14	31	Error Firmware update of the SD card, file does not match the coupler type	Check SD card
4	9	20	3	20..24	8	Error while reading/writing the configuration data from/to the SD card, file could not be opened	Check SD card
4	9	20	3	20..24	12	Error while reading/writing the configuration data from/to the SD card, error while reading the file	Check SD card
4	9	20	3	20..24	13	Error while reading/writing the configuration data from/to the SD card, error while writing the file	Check SD card
4	9	20	3	20..24	17	Error while reading/writing the	Check SD card and

						configuration data from/to the SD card, error while accessing the coupler	hardware configuration
4	9	20	3	20..24	31	Error while reading/writing the configuration data from/to the SD card, file does not match the coupler type	Check SD card and hardware configuration
4	9	20	31	0..15	0..15	Coupler update failed, error message of the coupler: Channel = Bit 4..7 Error = Bit 0..3	see table "Coupler errors"
4	9	20	31	1	2	File does not exist	Check SD card
4	9	20	31	1	8	Error: Invalid file	Check SD card
4	9	20	31	1	40	Version is not supported, e.g. obsolete software	Check FW version, update to latest version
4	9	20	31	1	43	Other error, e.g. not enough memory or file system error	Update CPU FW
4	9	20	31	2	12	No SD card inserted or SDCARD.INI file not found	Check SD card
4	9	20	31	3	13	SD card errors: - SDCARD.INI on SD card is missing, default was generated - Copying the boot project from the SD card failed - Copying the boot project from the SD card failed (may be that there is no valid boot project) - Creating of the boot project failed (may be that there is no valid boot project)	Check SD card
4	9	20	31	5	13	Loading the source code failed	Check SD card, reload
4	9	20	31	31	8	Missing or exhausted battery	Insert battery or set parameter "Check Battery" to "Off"
<b>Error messages of the I/O-Bus</b>							
<b>Serious errors</b>							
2	14	1..10	31	1	34	Timeout while initializing an I/O module	Replace module
2	14	1..10	31	4	42	Failure of the module, more than the max. permissible communication errors have occurred in sequence	Check module, FW version (PLC-browser: IO-bus desc)
<b>Light errors</b>							
3	14	1..10	31	1	32	Master and module could not agree on any protocol variant, no variant found which is supported by both the master and the module	Check FW version CPU / I/O module
3	14	1..10	31	3	26	Configuration error PLC configuration master	Check configuration
3	14	255	2	0	3	Timeout while updating the I/O data at the program start	Check FW version CPU / I/O modules
3	14	255	2	0	26	Program was not started because of configuration error of the I/O-Bus	Check configuration
2	14	255	3	0	26	Configuration error PLC configuration master	Check configuration
<b>Warnings</b>							
4	14	1..10	31	1	34	Timeout during parameterization	Check FW version CPU / IO modules
4	14	1..10	31	31	44	Module has not passed factory test	Replace module
<b>Error messages of the coupler interface</b>							
<b>Serious errors</b>							
2	1..4/10	255	3	0	2	Same Node ID twice in the net	Use Node IDs only once
<b>Light errors</b>							
3	1..4/10	255	3	0	26	Incorrect or missing coupler configuration	Configure coupler
3	1..4/10	255	5	0..15	0..15	Error message of the operating system of the coupler:	see table "Coupler errors"

						Channel = Bit 4..7 Error = Bit 0..3	
3	1..4/10	255	6..12	0..15	0..15	Error message of the task x of the coupler: Task x = 'Module' - 5 Channel = Bit 4..7 Error = Bit 0..3	see table "Coupler errors"
3	1..4/10	255	31	0	33	Timeout while waiting for reset of the coupler	Check coupler
3	1..4/10	255	31	0	34	Timeout while waiting for readiness of the coupler	Check coupler
<b>Warnings</b>							
4	1..4/10	0..7	31	31	47	Short-circuit coupler/channel	Fix short-circuit
4	1..4/10	0..254	31	0..15	0..15	Communication error to the slave, Channel = Bit 4..7 Error = Bit 0..3	see table "Coupler errors"
4	1..4/10	255	2	0..15	0..15	Communication error of the coupler, Channel = Bit 4..7 Error = Bit 0..3	see table "Coupler errors"
<b>Error messages of the serial interfaces</b>							
<b>Serious errors</b>							
2	11..13	255	31	0	17	Access errors: - Interface could not be closed - Interface could not be opened - Timeslotmode could not be activated	Check FW version, replace CPU if necessary
<b>Warnings</b>							
4	13	255	4	0	42	Receiving error or timeout of the FBP slave interface	
<b>Error messages of the CS31 bus (COM1 = CS31 master)</b>							
Class	Interface	Address	Module type	Channel	Error identifier	Error message	Remedy
<b>Light errors</b>							
3	11	0..61	0..5	0	8	No module found on the CS31 bus	Adapt configuration
3	11	0..61	1..8	0..31	0..63	S500 class 3 diagnostic sent by DC551	see table "S500 errors"
3	11	255	1	0	8	No module found on the CS31 bus	Check configuration
<b>Warnings</b>							
4	11	0..61	0..5	0	8	ICMK 14 with extensions configured, the extensions was not found on the bus	Check configuration
4	11	0..61	0..5	0	28	Module discarded and registered again; is only reported at the start	
4	11	0..61	0..5	0	32	Not configured module found on the bus, again discarded	Check CS31 bus, insert CS31_DIAG function block into the project
4	11	0..61	0..5	0..15	1	Internal error (error 1), reported by an AC31 I/O module	Check module
4	11	0..61	0..5	0..15	28	Configured module does not match the module registered to the bus	Check PLC configuration, insert CS31_DIAG function block into the project
4	11	0..61	0..5	0..15	32	Not configured module registered to the bus	Check PLC configuration, insert CS31_DIAG function block into the project
4	11	0..61	0..5	0..15	47	Short-circuit on CS31 module/channel	Fix short-circuit
4	11	0..61	0, 2, 4	0..15	2	Cut wire (Error 2) of an AC31 I/O module	Remove error at the module or the channel
4	11	0..61	0, 2, 4	0..15	4	Overload (Error 4) of an AC31 I/O module	
4	11	0..61	0, 2, 4	0..15	6	Overload + cut wire (Error 6) of an AC31 I/O module	
4	11	0..61	0, 2, 4	0..15	10	Short-circuit + cut wire or "out of range" at analog modules (Error 10) of an AC31 I/O module	
1	11	0..61	0, 2, 4	0..15	12	Overload + short-circuit (Error 12) of	

						an AC31 I/O module	
4	11	0..61	0, 2, 4	0..15	14	Short-circuit + overload + cut wire (Error 14) of an AC31 I/O module	
4	11	0..61	1, 3, 5	0..15	3	Analog value exceeded (Error 3) of an AC31 I/O module	
4	11	0..61	1, 3, 5	0..15	9	Cut wire (Error 9) of an AC31 analog module	
4	11	0..61	1..8	0..31	0..63	E4 error messages of DC551 and S500 I/O modules, see table below	see tablee "S500 errors"
4	11	0..61	31	31	9	Impossible configuration DC551 and S500 I/O modules (too many I/Os in one cluster)	Change configuration
4	11	0..61	31	31	31	Impossible configuration DC551 and S500 I/O modules (too many parameters in one cluster)	Change configuration
4	11	0..61	31	31	34	Outputs are written before the configuration of the modules DC551 + S500 I/O have been finished	

### Notes

- 1) AC500 uses the following interface identification:  
14 = I/O bus, 11 = COM1 (e.g., CS31 bus), 12 = COM2  
FBP diagnosis blocks do not contain this identifier.
- 2) The assignment for "Device" is as follows:  
31 = Module, 1..7 = Expansion 1..7
- 3) The assignment for "Module" is as follows:  
31 = Module or module type (2=DO)
- 4) In case of module errors, "31 = Module" is reported for the channel.
- 5) In the current AC500 CPU firmware version, errors reported to the AC500 CPU by the DC505 FBP cannot be directly made visible on the display or in the PLC browser of the Control Builder PS501.



E1..E4	d1	d2	d3	d4	Identifier 000...063	AC500 display	<- displayed in 5)
Class	Comp	Dev	Mod	Ch	Err	PS501 PLC browser	
Byte 6 Bit 6..7	-	Byte 3	Byte 4	Byte 5	Byte 6 Bit 0..5	FBP diagnosis block	
Class	Inter- face	De- vice	Mod- ule	Chan- nel	Error identifier	Error message	Remedy
	1)	2)	3)	4)			
<b>Errors of the S500 I/O modules</b>							
<b>S500 I/O module errors</b>							
<b>Light errors</b>							
3	14	1..7	31	31	3	Timeout in the I/O module	Replace I/O module
	11 / 12	ADR	1..7				
3	14	1..7	31	31	9	Overflow diagnosis buffer	Restart
	11 / 12	ADR	1..7				
3	14	1..7	31	31	11	Process voltage too low	Check process voltage
	11 / 12	ADR	1..7				
3	14	1..7	31	31	19	Checksum error in the I/O module	Replace I/O module
	11 / 12	ADR	1..7				
3	14	1..7	31	31	26	Parameter error	Check master
	11 / 12	ADR	1..7				
3	14	1..7	31	31	36	Internal data interchange disturbed	Replace I/O module
	11 / 12	ADR	1..7				
3	14	1..7	31	31	40	Different hardware and firmware versions in the module	Replace I/O module
	11 / 12	ADR	1..7				
3	14	1..7	31	31	43	Interner Fehler im Gerät	Replace I/O module
	11 / 12	ADR	1..7				
3	14	1..7	31	31	47	Sensor voltage too low	Check sensor voltage
	11 / 12	ADR	1..7				
<b>Warnings</b>							
4	14	1..7	31	31	45	Process voltage switched off (ON->OFF)	Process voltage ON
	11 / 12	ADR	1..7				
<b>Channel errors of the S500 I/O modules</b>							
<b>Warnings</b>							
4	14	1..7	1	0..7	7	Measurement underflow at the analog input	Check input value
	11 / 12	ADR	1..7				
4	14	1..7	1	0..7	47	Short-circuit at the analog input	Check terminal
	11 / 12	ADR	1..7				
4	14	1..7	1	0..7	48	Measurement overflow or cut wire at the analog input	Check input value and terminal
	11 / 12	ADR	1..7				
4	14	1..7	2	0..23	47	Short-circuit at the digital output	Check terminal
	11 / 12	ADR	1..7				
4	14	1..7	3	0..7	48	Measurement overflow at the analog output	Check output value
	11 / 12	ADR	1..7				
4	14	1..7	3	0..7	7	Measurement underflow at the analog output	Check output value
	11 / 12	ADR	1..7				
<b>Module errors DC551-CS31</b>							
<b>Light errors</b>							
3	11	ADR	31	31	3	Timeout in the I/O module	Replace I/O module
3	11	ADR	31	31	9	Overflow of diagnosis buffer	Restart
3	11	ADR	31	31	11	Process voltage too low	Check process voltage
3	11	ADR	1..7	31	17	No communication with the I/O module	Replace I/O module
3	11	ADR	31	31	19	Checksum error in the I/O module	Replace I/O module
3	11	ADR	31	31	26	Parameter error	Check master
3	11	ADR	31	31	36	Internal data interchange disturbed	

3	11	ADR	31	31	40	Different hardware and firmware versions in the module	Replace I/O module
3	11	ADR	31	31	43	Internal error in the module	Replace I/O module
<b>Warnings</b>							
4	11	ADR	31	31	45	Process voltage ON/OFF	Process voltage ON
4	11	ADR	31/1..7	31	34	No response during initialization of the I/O module	Replace I/O module
4	11	ADR	31/1..7	31	32	Wrong I/O module on the slot	Replace I/O module and check configuration
<b>Channel errors DC551-CS31</b>							
<b>Warnings</b>							
4	11	ADR	31/1..7	8..23	47	Short-circuit at the digital output	Check terminal
<b>Module errors DC505-FBP</b>							
<b>Light errors</b>							
3	-	1..7	31	31	11	Process voltage too low	Check process voltage
3	-	1..7	31	31	17	No communication with the I/O module	Replace I/O module
3	-	31	31	31	3	Timeout in the I/O module	
3	-	31	31	31	19	Checksum error in the I/O module	
3	-	31	31	31	36	Internal data interchange disturbed	
3	-	31	31	31	40	Different hardware and firmware versions in the module	
3	-	31	31	31	43	Internal error in the module	
3	-	31	31	31	9	Overflow of diagnosis buffer	Restart
3	-	31	31	31	26	Parameter error	Check master
<b>Warnings</b>							
4	-	31	31	31	45	Process voltage ON/OFF	Process voltage ON
4	-	31/1..7	31	31	32	Wrong I/O module on the slot	Replace I/O module and check configuration
4	-	31/1..7	31	31	34	No response during initialization of the I/O module	Replace I/O module
<b>Channel errors DC505-FBP</b>							
<b>Warnings</b>							
4	-	31	2	8..15	47	Short-circuit at the digital output	Check terminal

## Notes

- 1) AC500 uses the following interface identification:  
14 = I/O bus, 11 = COM1 (e.g., CS31 bus), 12 = COM2  
FBP diagnosis blocks do not contain this identifier.
- 2) The assignment for "Device" is as follows:  
31 = Module, 1..7 = Expansion 1..7
- 3) The assignment for "Module" is as follows:  
31 = Module or module type (2=DO)
- 4) In case of module errors, "31 = Module" is reported for the channel.
- 5) In the current AC500 CPU firmware version, errors reported to the AC500 CPU by the DC505 FBP cannot be directly made visible on the display or in the PLC browser of the Control Builder PS501.

## 5.2.5 Coupler errors

E1..E4	d1	d2	d3	d4	Identifier 000...063	AC500 display	<- displayed in 5)
Class	Comp	Dev	Mod	Ch	Err	PS501 PLC browser	
Byte 6 Bit 6..7	-	Byte 3	Byte 4	Byte 5	Byte 6 Bit 0..5	FBP diagnosis block	
Class	Interface	Device	Module	Channel	Error identifier	Error message	Remedy
	1)	2)	3)	4)			
<b>Error of the coupler's operating system</b>							
<b>General operating system errors</b>							
3	1..4/10	255	5	0	1	01 <sub>hex</sub> = 1 <sub>dec</sub> Error priority MAX	
3	1..4/10	255	5	0	2	02 <sub>hex</sub> = 2 <sub>dec</sub> Error priority NULL	
3	1..4/10	255	5	0	3	03 <sub>hex</sub> = 3 <sub>dec</sub> Error priority DOUBLE	
3	1..4/10	255	5	0	4	04 <sub>hex</sub> = 4 <sub>dec</sub> Stack size error	
3	1..4/10	255	5	0	5	05 <sub>hex</sub> = 5 <sub>dec</sub> EPROM size error	
3	1..4/10	255	5	0	6	06 <sub>hex</sub> = 6 <sub>dec</sub> RAM size error	
3	1..4/10	255	5	0	7	07 <sub>hex</sub> = 7 <sub>dec</sub> Segment counter error	
3	1..4/10	255	5	0	8	08 <sub>hex</sub> = 8 <sub>dec</sub> Segment size error	
3	1..4/10	255	5	0	9	09 <sub>hex</sub> = 9 <sub>dec</sub> Cycle time error	
3	1..4/10	255	5	0	10	0A <sub>hex</sub> = 10 <sub>dec</sub> Frequency error	
3	1..4/10	255	5	0	11	0B <sub>hex</sub> = 11 <sub>dec</sub> Trace buffer size error	
3	1..4/10	255	5	0	12	0C <sub>hex</sub> = 12 <sub>dec</sub> Error min. RAM	
3	1..4/10	255	5	0	13	0D <sub>hex</sub> = 13 <sub>dec</sub> Device address error	
3	1..4/10	255	5	0	14	0E <sub>hex</sub> = 14 <sub>dec</sub> MCL token error	
3	1..4/10	255	5	0	15	0F <sub>hex</sub> = 15 <sub>dec</sub> Driver type error	
3	1..4/10	255	5	1	0	10 <sub>hex</sub> = 16 <sub>dec</sub> SCC error	
3	1..4/10	255	5	1	1	11 <sub>hex</sub> = 17 <sub>dec</sub> Flash type OPT error	
3	1..4/10	255	5	1	2	12 <sub>hex</sub> = 18 <sub>dec</sub> Flash type BSL error	
3	1..4/10	255	5	1	3	13 <sub>hex</sub> = 19 <sub>dec</sub> Flash DIR name error	
3	1..4/10	255	5	1	4	14 <sub>hex</sub> = 20 <sub>dec</sub> Function table error	
3	1..4/10	255	5	1	5	15 <sub>hex</sub> = 21 <sub>dec</sub> RAM type error	
3	1..4/10	255	5	1	6	16 <sub>hex</sub> = 22 <sub>dec</sub> Flash DIR type error	
3	1..4/10	255	5	3	2	32 <sub>hex</sub> = 50 <sub>dec</sub> RAM test error	
3	1..4/10	255	5	3	3	33 <sub>hex</sub> = 51 <sub>dec</sub> Data segment error	
3	1..4/10	255	5	3	4	34 <sub>hex</sub> = 52 <sub>dec</sub> RAM error	
3	1..4/10	255	5	3	5	35 <sub>hex</sub> = 53 <sub>dec</sub> EPROM error	
3	1..4/10	255	5	3	6	36 <sub>hex</sub> = 54 <sub>dec</sub> DONGLE error	
3	1..4/10	255	5	3	7	37 <sub>hex</sub> = 55 <sub>dec</sub> Wrong RCS identifier error	
3	1..4/10	255	5	3	8	38 <sub>hex</sub> = 56 <sub>dec</sub> Error allocating memory	
3	1..4/10	255	5	6	4	64 <sub>hex</sub> = 100 <sub>dec</sub> RCS task not ready	
3	1..4/10	255	5	6	5	65 <sub>hex</sub> = 101 <sub>dec</sub> Task 1 not ready	
3	1..4/10	255	5	6	6	66 <sub>hex</sub> = 102 <sub>dec</sub> Task 2 not ready	
3	1..4/10	255	5	6	7	67 <sub>hex</sub> = 103 <sub>dec</sub> Task 3 not ready	
3	1..4/10	255	5	6	8	68 <sub>hex</sub> = 104 <sub>dec</sub> Task 4 not ready	
3	1..4/10	255	5	6	9	69 <sub>hex</sub> = 105 <sub>dec</sub> Task 5 not ready	
3	1..4/10	255	5	6	10	6A <sub>hex</sub> = 106 <sub>dec</sub> Task 6 not ready	
3	1..4/10	255	5	6	11	6B <sub>hex</sub> = 107 <sub>dec</sub> Task 7 not ready	
3	1..4/10	255	5	6	12	6C <sub>hex</sub> = 108 <sub>dec</sub> Task 8 not ready	
3	1..4/10	255	5	6	13	6D <sub>hex</sub> = 109 <sub>dec</sub> Task 9 not ready	
3	1..4/10	255	5	6	14	6E <sub>hex</sub> = 110 <sub>dec</sub> Task 10 not ready	
3	1..4/10	255	5	6	15	6F <sub>hex</sub> = 111 <sub>dec</sub> Task 11 not ready	
3	1..4/10	255	5	7	0	70 <sub>hex</sub> = 112 <sub>dec</sub> Task 12 not ready	
3	1..4/10	255	5	7	1	71 <sub>hex</sub> = 113 <sub>dec</sub> Task 13 not ready	

3	1..4/10	255	5	7	2	72 <sub>hex</sub> = 114 <sub>dec</sub> Task 14 not ready	
3	1..4/10	255	5	7	3	73 <sub>hex</sub> = 115 <sub>dec</sub> Task 15 not ready	
3	1..4/10	255	5	7	8	78 <sub>hex</sub> = 120 <sub>dec</sub> MCL 0 missing	
3	1..4/10	255	5	7	9	79 <sub>hex</sub> = 121 <sub>dec</sub> MCL 1 missing	
3	1..4/10	255	5	7	10	7A <sub>hex</sub> = 122 <sub>dec</sub> MCL 2 missing	
3	1..4/10	255	5	8	0	80 <sub>hex</sub> = 128 <sub>dec</sub> MCL double	
3	1..4/10	255	5	8	1	81 <sub>hex</sub> = 129 <sub>dec</sub> MCL start address	
3	1..4/10	255	5	8	2	82 <sub>hex</sub> = 130 <sub>dec</sub> MCL 0 error	
3	1..4/10	255	5	8	3	83 <sub>hex</sub> = 131 <sub>dec</sub> MCL 1 error	
3	1..4/10	255	5	8	4	84 <sub>hex</sub> = 132 <sub>dec</sub> MCL 2 error	
3	1..4/10	255	5	8	10	8A <sub>hex</sub> = 138 <sub>dec</sub> MCL mode	
3	1..4/10	255	5	8	12	8C <sub>hex</sub> = 140 <sub>dec</sub> RCS 0 missing	
3	1..4/10	255	5	8	13	8D <sub>hex</sub> = 141 <sub>dec</sub> RCS 1 missing	
3	1..4/10	255	5	8	14	8E <sub>hex</sub> = 142 <sub>dec</sub> RCS 2 missing	
3	1..4/10	255	5	8	15	8F <sub>hex</sub> = 143 <sub>dec</sub> RCS 3 missing	
3	1..4/10	255	5	9	0	90 <sub>hex</sub> = 144 <sub>dec</sub> RCS 4 missing	
3	1..4/10	255	5	9	1	91 <sub>hex</sub> = 145 <sub>dec</sub> RCS 5 missing	
3	1..4/10	255	5	9	2	92 <sub>hex</sub> = 146 <sub>dec</sub> RCS 6 missing	
3	1..4/10	255	5	9	3	93 <sub>hex</sub> = 147 <sub>dec</sub> RCS 7 missing	
3	1..4/10	255	5	9	4	94 <sub>hex</sub> = 148 <sub>dec</sub> RCS double	
3	1..4/10	255	5	9	5	95 <sub>hex</sub> = 149 <sub>dec</sub> RCS start address	
3	1..4/10	255	5	9	6	96 <sub>hex</sub> = 150 <sub>dec</sub> RCS 0 error	
3	1..4/10	255	5	9	7	97 <sub>hex</sub> = 151 <sub>dec</sub> RCS 1 error	
3	1..4/10	255	5	9	8	98 <sub>hex</sub> = 152 <sub>dec</sub> RCS 2 error	
3	1..4/10	255	5	9	9	99 <sub>hex</sub> = 153 <sub>dec</sub> RCS 3 error	
3	1..4/10	255	5	9	10	9A <sub>hex</sub> = 154 <sub>dec</sub> RCS 4 error	
3	1..4/10	255	5	9	11	9B <sub>hex</sub> = 155 <sub>dec</sub> RCS 5 error	
3	1..4/10	255	5	9	12	9C <sub>hex</sub> = 156 <sub>dec</sub> RCS 6 error	
3	1..4/10	255	5	9	13	9D <sub>hex</sub> = 157 <sub>dec</sub> RCS 7 error	
3	1..4/10	255	5	10	0	A0 <sub>hex</sub> = 160 <sub>dec</sub> LIB 0 missing	
3	1..4/10	255	5	10	1	A1 <sub>hex</sub> = 161 <sub>dec</sub> LIB 1 missing	
3	1..4/10	255	5	10	2	A2 <sub>hex</sub> = 162 <sub>dec</sub> LIB 2 missing	
3	1..4/10	255	5	10	3	A3 <sub>hex</sub> = 163 <sub>dec</sub> LIB 3 missing	
3	1..4/10	255	5	10	4	A4 <sub>hex</sub> = 164 <sub>dec</sub> LIB 4 missing	
3	1..4/10	255	5	10	5	A5 <sub>hex</sub> = 165 <sub>dec</sub> LIB 5 missing	
3	1..4/10	255	5	10	6	A6 <sub>hex</sub> = 166 <sub>dec</sub> LIB 6 missing	
3	1..4/10	255	5	10	7	A7 <sub>hex</sub> = 167 <sub>dec</sub> LIB 7 missing	
3	1..4/10	255	5	10	8	A8 <sub>hex</sub> = 168 <sub>dec</sub> LIB double	
3	1..4/10	255	5	10	9	A9 <sub>hex</sub> = 169 <sub>dec</sub> LIB start address	
3	1..4/10	255	5	10	10	AA <sub>hex</sub> = 170 <sub>dec</sub> LIB 0 error	
3	1..4/10	255	5	10	11	AB <sub>hex</sub> = 171 <sub>dec</sub> LIB 1 error	
3	1..4/10	255	5	10	12	AC <sub>hex</sub> = 172 <sub>dec</sub> LIB 2 error	
3	1..4/10	255	5	10	13	AD <sub>hex</sub> = 173 <sub>dec</sub> LIB 3 error	
3	1..4/10	255	5	10	14	AE <sub>hex</sub> = 174 <sub>dec</sub> LIB 4 error	
3	1..4/10	255	5	10	15	AF <sub>hex</sub> = 175 <sub>dec</sub> LIB 5 error	
3	1..4/10	255	5	11	0	B0 <sub>hex</sub> = 176 <sub>dec</sub> LIB 6 error	
3	1..4/10	255	5	11	1	B1 <sub>hex</sub> = 177 <sub>dec</sub> LIB 7 error	
3	1..4/10	255	5	12	8	C8 <sub>hex</sub> = 200 <sub>dec</sub> unknown IRQ	
3	1..4/10	255	5	12	9	C9 <sub>hex</sub> = 201 <sub>dec</sub> Watchdog	
3	1..4/10	255	5	12	10	CA <sub>hex</sub> = 202 <sub>dec</sub> SCC TX IRQ	
3	1..4/10	255	5	12	11	CB <sub>hex</sub> = 203 <sub>dec</sub> SCC RX IRQ	
3	1..4/10	255	5	12	12	CC <sub>hex</sub> = 204 <sub>dec</sub> Task state	
3	1..4/10	255	5	14	6	E6 <sub>hex</sub> = 230 <sub>dec</sub> Task 0	
3	1..4/10	255	5	14	7	E7 <sub>hex</sub> = 231 <sub>dec</sub> Task 1	
3	1..4/10	255	5	14	8	E8 <sub>hex</sub> = 232 <sub>dec</sub> Task 2	
3	1..4/10	255	5	14	9	E9 <sub>hex</sub> = 233 <sub>dec</sub> Task 3	
3	1..4/10	255	5	14	10	EA <sub>hex</sub> = 234 <sub>dec</sub> Task 4	
3	1..4/10	255	5	14	11	EB <sub>hex</sub> = 235 <sub>dec</sub> Task 5	

3	1..4/10	255	5	14	12	EChex = 236dec Task 6	
3	1..4/10	255	5	14	13	EDhex = 237dec Task 7	
3	1..4/10	255	5	15	0	F0hex = 240dec DBG task 0 segment	
3	1..4/10	255	5	15	1	F1hex = 241dec DBG task 1 segment	
3	1..4/10	255	5	15	2	F2hex = 242dec DBG task 2 segment	
3	1..4/10	255	5	15	3	F3hex = 243dec DBG task 3 segment	
3	1..4/10	255	5	15	4	F4hex = 244dec DBG task 4 segment	
3	1..4/10	255	5	15	5	F5hex = 245dec DBG task 5 segment	
3	1..4/10	255	5	15	6	F6hex = 246dec DBG task 6 segment	
3	1..4/10	255	5	15	7	F7hex = 247dec DBG task 7 segment	
<b>General task errors</b>							
3	1..4/10	255	6..12	0	1	01hex = 1dec No communication	
3	1..4/10	255	6..12	0	2	02hex = 2dec Idle	
3	1..4/10	255	6..12	3	2	32hex = 50dec Base initialization	
3	1..4/10	255	6..12	6	4	64hex = 100dec Parity error	
3	1..4/10	255	6..12	6	5	65hex = 101dec Frame error	
3	1..4/10	255	6..12	6	6	66hex = 102dec Overrun	
3	1..4/10	255	6..12	6	7	67hex = 103dec Data count	
3	1..4/10	255	6..12	6	8	68hex = 104dec Checksum error	
3	1..4/10	255	6..12	6	9	69hex = 105dec Timeout	
3	1..4/10	255	6..12	6	10	6Ahex = 106dec Protocol error	
3	1..4/10	255	6..12	6	11	6Bhex = 107dec Data error	
3	1..4/10	255	6..12	6	12	6Chex = 108dec NACK	
3	1..4/10	255	6..12	6	14	6Ehex = 110dec Protocol base	
3	1..4/10	255	6..12	9	6	96hex = 150dec Invalid message header	
3	1..4/10	255	6..12	9	7	97hex = 151dec Invalid message length	
3	1..4/10	255	6..12	9	8	98hex = 152dec Invalid message command	
3	1..4/10	255	6..12	9	9	99hex = 153dec Invalid message structure	
3	1..4/10	255	6..12	9	10	9Ahex = 154dec Message error	
3	1..4/10	255	6..12	9	11	9Bhex = 155dec Message timeout	
3	1..4/10	255	6..12	9	12	9Chex = 156dec Invalid message sequence	
3	1..4/10	255	6..12	9	13	9Dhex = 157dec Invalid message number	
3	1..4/10	255	6..12	9	14	9Ehex = 158dec Unable to run the command, since execution of the previous command is not yet finished	
3	1..4/10	255	6..12	10	0	A0hex = 160dec Error in telegram header	
3	1..4/10	255	6..12	10	1	A1hex = 161dec Invalid device address	
3	1..4/10	255	6..12	10	2	A2hex = 162dec Wrong address data area	
3	1..4/10	255	6..12	10	3	A3hex = 163dec Data address and data count cause a buffer overflow	
3	1..4/10	255	6..12	10	4	A4hex = 164dec Invalid data index	
3	1..4/10	255	6..12	10	5	A5hex = 165dec Invalid data count	
3	1..4/10	255	6..12	10	6	A6hex = 166dec Unknown data type	
3	1..4/10	255	6..12	10	7	A7hex = 167dec Unknown function	
3	1..4/10	255	6..12	10	10	AAhex = 170dec Message base	
3	1..4/10	255	6..12	12	8	C8hex = 200dec Task not initialized, coupler not configured	
3	1..4/10	255	6..12	12	9	C9hex = 201dec Busy	
3	1..4/10	255	6..12	12	10	CAhex = 202dec No segment of RCS received	
3	1..4/10	255	6..12	12	11	CBhex = 203dec Unknown or wrong sender of a command message	
3	1..4/10	255	6..12	13	2	D2hex = 210dec No database	

3	1..4/10	255	6..12	13	3	D3 <sub>hex</sub> = 211 <sub>dec</sub> Error writing the database	
3	1..4/10	255	6..12	13	4	D4 <sub>hex</sub> = 212 <sub>dec</sub> Error reading the database	
3	1..4/10	255	6..12	13	5	D5 <sub>hex</sub> = 213 <sub>dec</sub> Error registering the diagnosis structure	
3	1..4/10	255	6..12	13	6	D6 <sub>hex</sub> = 214 <sub>dec</sub> Parameter error	
3	1..4/10	255	6..12	13	7	D7 <sub>hex</sub> = 215 <sub>dec</sub> Configuration	
3	1..4/10	255	6..12	13	8	D8 <sub>hex</sub> = 216 <sub>dec</sub> Function list	
3	1..4/10	255	6..12	13	9	D9 <sub>hex</sub> = 217 <sub>dec</sub> System	
3	1..4/10	255	6..12	13	10	DA <sub>hex</sub> = 218 <sub>dec</sub> Not enough internal memory available	
3	1..4/10	255	6..12	13	11	DB <sub>hex</sub> = 219 <sub>dec</sub> No DPR	
3	1..4/10	255	6..12	13	12	DC <sub>hex</sub> = 220 <sub>dec</sub> System base	

E1..E4	d1	d2	d3	d4	Identifier 000...063	AC500 display	<- displayed in 5)
Class	Comp	Dev	Mod	Ch	Err	PS501 PLC browser	
Byte 6 Bit 6..7	-	Byte 3	Byte 4	Byte 5	Byte 6 Bit 0..5	FBP diagnosis block	
Class	Interface	Device	Module	Channel	Error identifier	Error message	Remedy
	1)	2)	3)	4)			

**Ethernet coupler errors**

**IP task (task 7) errors, see remark 1**

3	1..4/10	255	12	3	2	32 <sub>hex</sub> = 50 <sub>dec</sub> No TCP task initialized	
3	1..4/10	255	12	3	3	33 <sub>hex</sub> = 51 <sub>dec</sub> Error when initializing the task configuration	
3	1..4/10	255	12	3	4	34 <sub>hex</sub> = 52 <sub>dec</sub> No Ethernet address	
3	1..4/10	255	12	3	5	35 <sub>hex</sub> = 53 <sub>dec</sub> Wait for warm start	
3	1..4/10	255	12	3	6	36 <sub>hex</sub> = 54 <sub>dec</sub> Invalid flags	
3	1..4/10	255	12	3	7	37 <sub>hex</sub> = 55 <sub>dec</sub> Invalid IP address	
3	1..4/10	255	12	3	8	38 <sub>hex</sub> = 56 <sub>dec</sub> Invalid net mask	
3	1..4/10	255	12	3	9	39 <sub>hex</sub> = 57 <sub>dec</sub> Invalid gateway	
3	1..4/10	255	12	3	11	3B <sub>hex</sub> = 59 <sub>dec</sub> Unknown hardware	
3	1..4/10	255	12	3	12	3C <sub>hex</sub> = 60 <sub>dec</sub> No IP address	
3	1..4/10	255	12	3	13	3D <sub>hex</sub> = 61 <sub>dec</sub> Error initializing the driver	
3	1..4/10	255	12	3	14	3E <sub>hex</sub> = 62 <sub>dec</sub> No IP address configuration	
3	1..4/10	255	12	3	15	3F <sub>hex</sub> = 63 <sub>dec</sub> Invalid serial number	
3	1..4/10	255	12	4	0	40 <sub>hex</sub> = 64 <sub>dec</sub> No memory on chip	
3	1..4/10	255	12	6	14	6E <sub>hex</sub> = 110 <sub>dec</sub> Timeout	
3	1..4/10	255	12	6	15	6F <sub>hex</sub> = 111 <sub>dec</sub> Timeout invalid	
3	1..4/10	255	12	7	3	73 <sub>hex</sub> = 115 <sub>dec</sub> Target not reachable	
3	1..4/10	255	12	7	6	76 <sub>hex</sub> = 118 <sub>dec</sub> IP address invalid	
3	1..4/10	255	12	7	12	7C <sub>hex</sub> = 124 <sub>dec</sub> Ethernet address invalid	
3	1..4/10	255	12	8	2	82 <sub>hex</sub> = 130 <sub>dec</sub> Unknown mode	
3	1..4/10	255	12	8	3	83 <sub>hex</sub> = 131 <sub>dec</sub> ARP cache full	
3	1..4/10	255	12	8	6	86 <sub>hex</sub> = 134 <sub>dec</sub> No ARP entry found	
3	1..4/10	255	12	9	5	95 <sub>hex</sub> = 149 <sub>dec</sub> Unexpected response	

**TCP/UDP task (task 6) errors, see remark 1**

3	1..4/10	255	11	3	2	32 <sub>hex</sub> = 50 <sub>dec</sub> Init of IP task not completed	
3	1..4/10	255	11	3	3	33 <sub>hex</sub> = 51 <sub>dec</sub> Error when initializing the task configuration	

3	1..4/10	255	11	3	4	34 <sub>hex</sub> = 52 <sub>dec</sub> Init of IP task failed	
3	1..4/10	255	11	3	7	37 <sub>hex</sub> = 55 <sub>dec</sub> No memory available for init	
3	1..4/10	255	11	6	14	6E <sub>hex</sub> = 110 <sub>dec</sub> Timeout	
3	1..4/10	255	11	6	15	6F <sub>hex</sub> = 111 <sub>dec</sub> Invalid timeout	
3	1..4/10	255	11	7	0	70 <sub>hex</sub> = 112 <sub>dec</sub> Invalid socket	
3	1..4/10	255	11	7	1	71 <sub>hex</sub> = 113 <sub>dec</sub> Socket status	
3	1..4/10	255	11	7	3	73 <sub>hex</sub> = 115 <sub>dec</sub> Target not reachable	
3	1..4/10	255	11	7	4	74 <sub>hex</sub> = 116 <sub>dec</sub> Option not supported	
3	1..4/10	255	11	7	5	75 <sub>hex</sub> = 117 <sub>dec</sub> Invalid parameter	
3	1..4/10	255	11	7	6	76 <sub>hex</sub> = 118 <sub>dec</sub> Invalid IP address	
3	1..4/10	255	11	7	7	77 <sub>hex</sub> = 119 <sub>dec</sub> Invalid port	
3	1..4/10	255	11	7	8	78 <sub>hex</sub> = 120 <sub>dec</sub> CONN closed	
3	1..4/10	255	11	7	9	79 <sub>hex</sub> = 121 <sub>dec</sub> CONN reset	
3	1..4/10	255	11	7	10	7A <sub>hex</sub> = 122 <sub>dec</sub> Unknown protocol	
3	1..4/10	255	11	7	11	7B <sub>hex</sub> = 123 <sub>dec</sub> No sockets	
3	1..4/10	255	11	8	2	82 <sub>hex</sub> = 130 <sub>dec</sub> Unknown mode	
3	1..4/10	255	11	8	3	83 <sub>hex</sub> = 131 <sub>dec</sub> Max. data length exceeded	
3	1..4/10	255	11	8	4	84 <sub>hex</sub> = 132 <sub>dec</sub> Message count exceeded	
3	1..4/10	255	11	8	5	85 <sub>hex</sub> = 133 <sub>dec</sub> Max. group exceeded	
3	1..4/10	255	11	9	5	95 <sub>hex</sub> = 149 <sub>dec</sub> Unexpected response message	
<b>OMB task (Modbus TCP) errors (task 3), see remark 1</b>							
3	1..4/10	255	8	3	4	34 <sub>hex</sub> = 52 <sub>dec</sub> Invalid configuration data, server connections	
3	1..4/10	255	8	3	5	35 <sub>hex</sub> = 53 <sub>dec</sub> Invalid configuration data, task timeout	
3	1..4/10	255	8	3	6	36 <sub>hex</sub> = 54 <sub>dec</sub> Invalid configuration data, OMB timeout	
3	1..4/10	255	8	3	7	37 <sub>hex</sub> = 55 <sub>dec</sub> Invalid configuration data, mode	
3	1..4/10	255	8	3	8	38 <sub>hex</sub> = 56 <sub>dec</sub> Invalid configuration data, send timeout	
3	1..4/10	255	8	3	9	39 <sub>hex</sub> = 57 <sub>dec</sub> Invalid configuration data, connect timeout	
3	1..4/10	255	8	3	10	3A <sub>hex</sub> = 58 <sub>dec</sub> Invalid configuration data, close timeout	
3	1..4/10	255	8	3	11	3B <sub>hex</sub> = 59 <sub>dec</sub> Invalid configuration data, swap	
3	1..4/10	255	8	3	12	3C <sub>hex</sub> = 60 <sub>dec</sub> TCP_UDP task not found, TCP task not ready	
3	1..4/10	255	8	3	13	3D <sub>hex</sub> = 61 <sub>dec</sub> PLC task not found, PLC task not ready	
3	1..4/10	255	8	3	14	3E <sub>hex</sub> = 62 <sub>dec</sub> Error initializing OMB task	
3	1..4/10	255	8	3	15	3F <sub>hex</sub> = 63 <sub>dec</sub> Error initializing PLC task mode	
3	1..4/10	255	8	6	15	6F <sub>hex</sub> = 111 <sub>dec</sub> Unknown sender of a response	
3	1..4/10	255	8	7	0	70 <sub>hex</sub> = 112 <sub>dec</sub> Error code in response	
3	1..4/10	255	8	7	1	71 <sub>hex</sub> = 113 <sub>dec</sub> No socket found in searched status	
3	1..4/10	255	8	7	2	72 <sub>hex</sub> = 114 <sub>dec</sub> Invalid value in request	
3	1..4/10	255	8	7	3	73 <sub>hex</sub> = 115 <sub>dec</sub> Error message of TCP task	
3	1..4/10	255	8	7	4	74 <sub>hex</sub> = 116 <sub>dec</sub> Modbus error	
3	1..4/10	255	8	7	5	75 <sub>hex</sub> = 117 <sub>dec</sub> No socket available	
3	1..4/10	255	8	7	6	76 <sub>hex</sub> = 118 <sub>dec</sub> Invalid socket handle	
3	1..4/10	255	8	7	7	77 <sub>hex</sub> = 119 <sub>dec</sub> Timeout in client socket	

3	1..4/10	255	8	7	8	78 <sub>hex</sub> = 120 <sub>dec</sub> Socket closed, without response to command	
3	1..4/10	255	8	7	9	79 <sub>hex</sub> = 121 <sub>dec</sub> Not ready flag set	
3	1..4/10	255	8	7	10	7A <sub>hex</sub> = 122 <sub>dec</sub> TCP task no longer ready	
3	1..4/10	255	8	7	11	7B <sub>hex</sub> = 123 <sub>dec</sub> Watchdog event	
3	1..4/10	255	8	7	12	7C <sub>hex</sub> = 124 <sub>dec</sub> Device in reconfiguration	
3	1..4/10	255	8	7	13	7D <sub>hex</sub> = 125 <sub>dec</sub> PLC task not initialized	
3	1..4/10	255	8	7	14	7E <sub>hex</sub> = 126 <sub>dec</sub> OMB server socket closed	

**Remark 1:**

The error information is also available at the output ERNO of the blocks used for the coupler. The following applies: ERNO := 6000<sub>hex</sub> OR error.

### 5.3 Diagnosis blocks for the AC500

The folder "**Diagnosis**" in the AC500 library **SysInt\_AC500\_Vxx.LIB** contains the following diagnosis blocks:

Block	Function
DIAG_EVENT	Generates an error entry in the diagnosis system
DIAG_GET	Provides detailed information and the error code for the next error of the selected error class
DIAG_INFO	Indicates that an error of the class 1..4 exists
DIAG_ACK	Acknowledges an error with error code
DIAG_ACK_ALL	Acknowledges all errors of an error class (except errors that have to be acknowledged exclusively)

The diagnosis blocks are described in detail in the documentation for the SysInt\_AC500\_Vxx.LIB library.

### 5.4 AC500-specific PLC browser commands

The PLC browser interface of the Control Builder provides CoDeSys standard commands as well as AC500-specific commands. The general operation of the PLC browser is described in the according user manual.

This section only describes AC500-specific commands and commands that provide special data for the AC500.

For all commands online help information is available. The help information is displayed language-dependent by entering "?command" when operating in online mode. The command "?" lists all available firmware commands.

The commands listed in online mode can differ from the commands shown when pressing the button [...] as the Control Builder version and firmware version can differ. The commands listed when clicking the button [...] are defined in the file "Browser.ini" that belongs to the selected target system package (TSP).



The PLC browser provides the following AC500-specific commands:

Command	Meaning	Implementation
?	Displays all implemented commands	Standard
mem	Memory dump from up to	Standard
memc	Memory dump relative to code area	Standard
memd	Memory dump relative to data area	Standard
reflect	Reflect actual command line (for test purposes)	Standard
dpt	Displays the data pointer table	Standard
ppt	Displays the block pointer table	Standard
pid	Displays the project ID	Standard
pinf	Displays project information in the format: pinf Address of Structure: 16#0013CF74 Date: 4213949F Project Name: MODBUS_Test_BB.pro Project Title: Test MODBUS Project Version: V1.0 Project Author: Brigitte Blei Project Description: Test of serial interfaces End of Project-info.	Standard
tsk	Displays the IEC task list with task information in the format: tsk Number of Tasks: 1 Task 0: Main program, ID: 1519472 Cycle count: 45402 Cycle time: 1 ms Cycle time (min): 1 ms Cycle time (max): 1 ms Cycle time (avg): 1 ms Status: RUN Mode: CONTINUE ----- Priority: 10 Interval: 5 ms Event: NONE ----- Function pointer: 16#00601584 Function index: 131	Standard
startprg	Starts the user program	Standard
stopprg	Stops the user program	Standard
resetprg	Resets the user program	Standard
resetprgcold	Resets the user program (cold)	Standard
resetprgorg	Resets the user program (origin)	Standard
reload	Reloads the boot project from Flash	Standard
getprgprop	Displays program properties in the format: getprgprop Name: MODBUS_FBP_Test_BB.pro Title: Test MODBUS Version: V1.0 Author: Brigitte Blei Date: 4213949F	Standard
getprgstat	Displays the program status in the format: getprgstat Status: Run Last error: Id 00000000 TimeStamp 000055F3 Parameter 00000000 Text Flags:	Standard

filecopy	File command copy	No
filerename	File command rename	No
filedelete	File command delete	No
filedir	File command dir	No
saveretain	In V1.0 and V1.1: Saves the RETAIN variables to the SD card. As of V1.2: Writes the RETAIN variables to RAM (same as retain save)	Specific
restoreretain	In V1.0 and V1.1: Restores the RETAIN variables from the SD card. As of V1.2: Restores the RETAIN variables from RAM (same as retain restore)	Specific
setpwd	Sets the PLC password (required at logon!)	Standard
delpwd	Deletes the PLC password	Standard
plcload	Displays the PLC utilization (system+IEC+tasks+communication)	Standard
rtsinfo	Displays the firmware information (version, driver) in the format: rtsinfo rts version: 2.4.5.2 OS version: SMX smxPPC 3.5.2 uses IO driver interface rts api version: 2.407 4 driver(s) loaded driver 1: AC500 CPU driver, device interface version: 2.403 driver 2: AC500 I/O-BUS driver, device interface version: 2.403 driver 3: AC500 COM driver, device interface version: 2.403 driver 4: AC500 Coupler driver, device interface version: 2.403 AC500 PM581 (DISP) : V1.0 AC500 PM581 (BOOT) : V1.2.0, (Build:May 3 2007,12:33:32,Release) AC500 PM581 (FW) : V1.2.0, (Build:May 10 2007,16:32:40,Release)	Specific
traceschedon	Enables task tracing	No
traceschedoff	Disables task tracing	No
traceschedstore	Stores task trace to RAM	No
ipaddr	Sets the IP address of the CPU	No
basetick	Sets the basetick to $\mu$ s	No
diagreset	Resets the diagnosis system	Specific
diagack all	Acknowledges all errors (except errors that have to be quit exclusively)	Specific
diagack x	Acknowledges all errors of the class X (with X= 1...4)	Specific
diagshow all	Shows all errors in the format: diagshow all  --- All errors --- State                    Class Comp Dev Mod Ch Err  0152502216 active and quitted 4      9      22 31 31 8 1970-01-01 00:00:08 occurred disappeared 1970-01-01 00:00:15 quitted	Specific

	<pre> 0152369165 active not quitted 4 9 20 31 0 13 1970-01-01 01:19:12 occurred - disappeared - quitted  --- end --- </pre>	
time	Displays and sets the time of the realtime clock	Specific
date	Displays and sets the date of the realtime clock	Specific
batt	Polls the battery status	Specific
sdappl	Saves the boot project to the SD card	Specific
sdfunc	Displays and changes the SD card function	Specific
sdboot	Updates the bootcode from the SD card	Specific
sdfirm	Updates the firmware from the SD card	Specific
sdcoupler x	Updates the firmware of coupler x from the SD card	Specific
cpuload	Displays the CPU load (current, min., max., average)	Specific
delappl	Deletes the user program in the Flash memory	Specific
retain	<p>Saving and restoring the RETAIN variables:</p> <p>retain clear -&gt; Clears all RETAIN variables</p> <p>retain save -&gt; Saves the RETAIN variables to the RAM disk</p> <p>retain restore -&gt; Restores the RETAIN variables from the RAM disk</p> <p>retain export -&gt; Exports the RETAIN variables from the RAM disk to the SD card</p> <p>retain import -&gt; Imports the RETAIN variables from the SD card to the RAM disk</p>	Specific as of V1.2
persistent	<p>Saving and restoring the persistent area %R area:</p> <p>persistent clear -&gt; Clears the %R area</p> <p>persistent save -&gt; Saves the buffered %R area to the RAM disk</p> <p>persistent restore -&gt; Restores the buffered %R area from the RAM disk</p> <p>persistent export -&gt; Exports the buffered %R area from the RAM disk to the SD card</p> <p>persistent import -&gt; Imports the buffered %R area from the SD card to the RAM disk</p>	Specific as of V1.2
io-bus stat	Displays the I/O bus statistic	Specific
io-bus desc	Displays the I/O bus configuration	Specific
com protocols	Displays the protocols available for the serial interfaces	Specific
com settings	Displays the serial interface settings	Specific
coupler desc	Displays information on the coupler interfaces (type, firmware, serial number, date)	Specific
coupler settings	Displays the current coupler settings, for example, IP address and socket assignment	Specific as of V1.2
reboot	Reboots the PLC (CoDeSys performs a logout when restarting or logout possible up to 3 seconds after command input)	Specific

## 6 The SD memory card in the AC500

### 6.1 SD card functions

#### 6.1.1 Summary of memory card functions

The AC500 controller contains a FLASH memory card of the type "SD Memory Card" (in short SD card) as external storage medium which is accessed by the PLC like a floppy disk drive. The SD card is used to transfer data between a commercially available PC with SD card interface and the AC500.

In the AC500, the SD card can be used to:

- update the AC500-CPU processor firmware
- update the CPU boot code
- update the display controller firmware (as of version V2.0)
- update the coupler firmware (as of version V1.2.0)
- update the firmware of the I/O modules connected to the I/O bus (as of version V2.0)
- load and save user programs (boot project)
- load and save the source code of the user program
- load and save retentive variables (RETAIN, %R area)
- load and save user data (with blocks)

The SD card can be operated by:

- writing/reading files using a standard PC card reader with SD card interface
- specific PLC browser commands
- reading and writing data from the user program using specific blocks

#### 6.1.2 PLC browser commands for accessing the SD card

Command	Function
sdfunc	Displays and sets the SD card function "FunctionOfCard": 0 None 1 Load user program 2 Load firmware 3 Load user program and firmware
sdappl	Saves the user program (boot project) stored in Flash memory to the SD card (files Default.prg and Default.chk) and sets the SD card function to "Load user program" Set FunctionOfCard=+1 (bit 0 = 1)
saveretain	Up to V1.1: Saves the RETAIN variables "RETAIN.BIN" to the SD card (not from %M area)
restoreretain	Up to V1.1: Restores the RETAIN variables "RETAIN.BIN" from SD card to SRAM (not from %M area)
retain	As of V1.2: Saving and restoring the RETAIN variables: retain export -> Exports the RETAIN variables from the RAM disk to the SD card retain import -> Imports the RETAIN variables from the SD card to the RAM disk
persistent	As of V1.2: Saving and restoring the persistent area %R area: persistent export -> Exports the buffered %R area from the RAM disk to the SD card persistent import -> Imports the buffered %R area from the SD card to the RAM disk

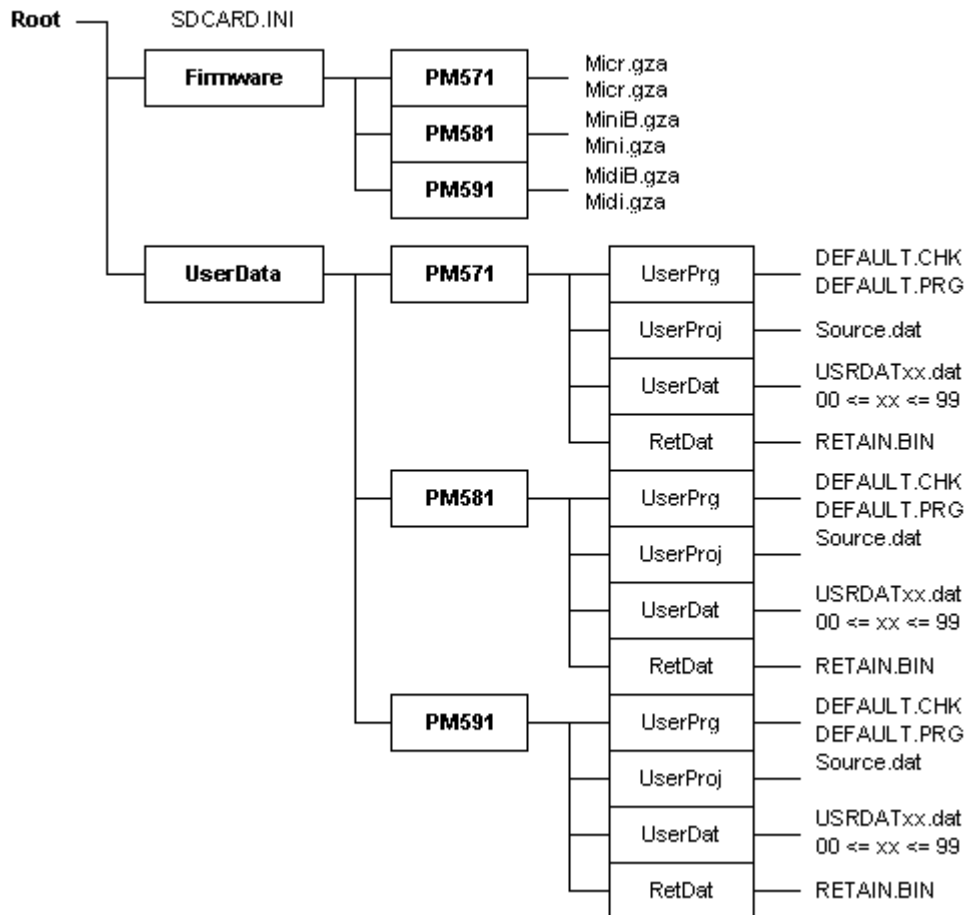
## 6.2 SD card file system

### 6.2.1 SD card file structure

In the PLC, the SD card is accessed like a PC floppy disk drive. The type of the file system is FAT (Microsoft DOS format). The file names are stored in 8.3 format (no "long" names) on the SD card.

#### File structure in versions V1.0 and V1.1

In versions V1.0 and V1.1, the file structure on the SD card looks as follows:



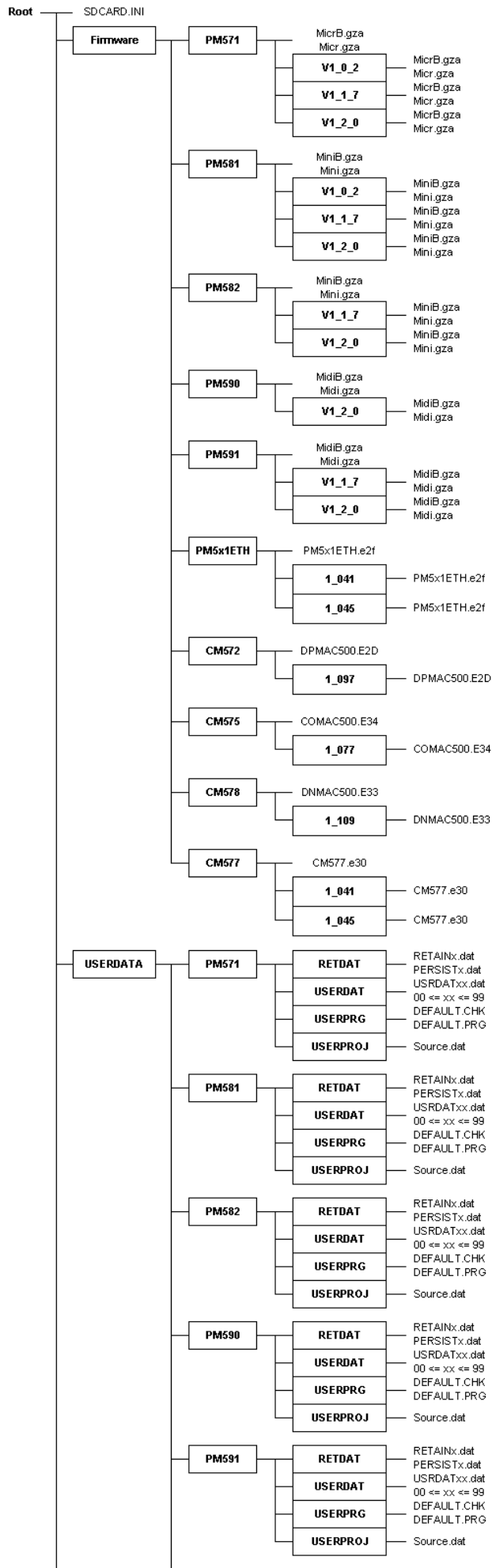
- The root directory on the SD card contains the command file SDCARD.INI (for a detailed description see the following section).
- The subdirectory "Firmware" contains further subdirectories for the:
  - CPUs PM571, PM581 and PM591.
 Each of these subdirectories contains the boot code MixxB.gza and the CPU firmware Mixx.gza. Boot code and firmware for the CPU PM582 are contained in the directory PM581.
- The subdirectory "UserData" contains the subdirectories:
  - "UserPrg", user program
  - "UserProj", source code of the user program
  - "UserDat", user data and
  - "RetDat", retentive data
 These directories contain the user data.
- xx in the file name represents the data file number.

## File structure as of version V1.2

The SD card file structure has been revised and expanded for version V1.2. The following new functions are implemented:

1. Management of several boot code and firmware versions on one SD card.
2. Each CPU has its own directory.  
Up to version V1.1, there were only directories for controller classes PM57x, PM58x and PM59x. Due to this, the PM581 and PM582 data were stored to the same directory, for example ..\PM581.
3. Loading the field bus coupler firmware from the SD card by specific settings in the file SDCARD.INI or using the PLC browser command.
4. Management of several field bus coupler firmware versions.
5. Saving/restoring RETAIN data (%M area excluded) to/from the SD card via the user program and/or the PLC browser.
6. Saving/restoring the PERSISTENT area (%R area) to/from the SD card via the user program and/or the PLC browser.

As of version V1.2, the file structure looks as follows:



## 6.2.2 The command file "SDCARD.INI"

The command file "SDCARD.INI" located in the root directory on the SD card determines the behavior when starting the AC500 with installed SD card and when installing the card (firmware update, loading the user program, etc.).

The file SDCARD.INI is created in Windows INI format.

### File content in versions V1.0 and V1.1

```
[Status]
FunctionOfCard=0

[FirmwareUpdate]
CPUPM5x1=0
CPUEC500=0
Display=0
Coupler_0=0
Coupler_1=0
Coupler_2=0
Coupler_3=0
Coupler_4=0

[UserProg]
UserProgram=0
RetainData=0
AddressData=0
```

The entries have the following meaning:

- **[Status]**  
**FunctionOfCard=0**

The parameter FunctionOfCard determines which function is performed when inserting the SD card.

With:

FunctionOfCard=0 Perform no function when inserting the card or voltage ON

FunctionOfCard=1 Load user program according to entry in group [UserProg]

FunctionOfCard=2 Start firmware update according to entry in group [FirmwareUpdate]

FunctionOfCard=4 Reserved for factory test

If you want to load the firmware and user program, set FunctionOfCard = 3.

- **[FirmwareUpdate]**  
**CPUPM5x1=0** (0 = no update, 1 = Update CPU firmware)  
**Display=0** (0 = no update, 1 = Update display controller)  
**Coupler\_0=0** (0 = no update, 1 = Update internal coupler)  
**Coupler\_1=0** (0 = no update, 1 = Update coupler in slot 1)  
**Coupler\_2=0** (0 = no update, 1 = Update coupler in slot 2)  
**Coupler\_3=0** (0 = no update, 1 = Update coupler in slot 3)  
**Coupler\_4=0** (0 = no update, 1 = Update coupler in slot 4)

The parameters in the group [FirmwareUpdate] specify which firmware is loaded. In version V1.0, only the CPU processor firmware can be loaded.

- **[UserProg]**  
**UserProgram=0** (0 = no update, 1 = Update user program)  
**RetainData=0** (0 = no update, 1 = Update RETAIN variables)

The parameters in the group [UserProg] specify which user data are loaded. Up to version V1.1, only the user program can be loaded.



## File content as of version V1.2

### [Status]

FunctionOfCard=0

### [FirmwareUpdate]

CPUPM5x1=0

CPUEC500=0

Display=0

Coupler\_0=0

Coupler\_1=0

Coupler\_2=0

Coupler\_3=0

Coupler\_4=0

### [UserProg]

UserProgram=0

RetainData=0

AddressData=0

### [PM571]

TYPE=1

VERSION=1\_2\_0

PLCBOOT=1\_2\_0

### [PM581]

TYPE=1

VERSION=1\_2\_0

PLCBOOT=1\_2\_0

### [PM582]

TYPE=1

VERSION=1\_2\_0

PLCBOOT=1\_2\_0

### [PM590]

TYPE=1

VERSION=1\_2\_0

PLCBOOT=1\_2\_0

### [PM591]

TYPE=1

VERSION=1\_2\_0

PLCBOOT=1\_2\_0

### [PM5x1]

TYPE=4

VERSION=1\_045

### [CM572]

TYPE=4

VERSION=1\_097

### [CM575]

TYPE=4

VERSION=1\_077

### [CM577]

TYPE=4

VERSION=1\_045

### [CM578]

TYPE=4

VERSION=1\_109

The entries have the following meaning:

- **[Status]**

- **FunctionOfCard=0**

The parameter FunctionOfCard determines which function is performed when inserting the SD card.

With:

FunctionOfCard=0 Perform no function when inserting the card or voltage ON

FunctionOfCard=1 Load user program according to entry in group [UserProg]

FunctionOfCard=2 Update firmware according to entry in group [FirmwareUpdate]

FunctionOfCard=3 Update firmware according to entry in group [FirmwareUpdate] and load user program according to entry in [UserProg]

FunctionOfCard=4 Reserved for factory test

FunctionOfCard=8 Functions 0..4 + save debug data in case of possible failures

Entry: 8 + function, Example: Function=1 ? FunctionOfCard=8+1=9

- **[FirmwareUpdate]**

- **CPUPM5x1=0**

**Display=0** (0 = no update, currently no further mode available)

**Coupler\_0=0** (0 = no update, 2/3 = Update internal coupler)

**Coupler\_1=0** (0 = no update, 2/3 = Update coupler in slot 1)

**Coupler\_2=0** (0 = no update, 2/3 = Update coupler in slot 2)

**Coupler\_3=0** (0 = no update, 2/3 = Update coupler in slot 3)

**Coupler\_4=0** (0 = no update, 2/3 = Update coupler in slot 4)

0 = no update,

1 = Update CPU firmware from base directory on the CPU

This mode is fully compatible to the firmware update of versions V1.0 and V1.1.

2 = Update with specific version

In mode 2, the update is only performed if the key 'version' in a product section (for example [PM581]) returns a different result than the version on the PLC. If the key 'version' is missing, no update is performed.

3 = Update firmware only if the SD card firmware is newer than the PLC firmware.

For mode 3 the same applies as for mode 2. However, an update is only performed if the firmware is newer than the PLC's firmware version.

For this mode it must be ensured that the firmware versions can be read safely.

The parameters of the group [FirmwareUpdate] specify which firmware is loaded.

- **[UserProg]**

UserProgram=0 (0 = no update, 1 = Update user program)

RetainData=0 (0 = no update, 1 = Update RETAIN variables)

The parameters of the group [UserProg] specify which user data are loaded. In version V1.0, only the user program can be loaded.

- **[PM571] or [PM581], [PM582], [PM590], [PM591]**

TYPE=1

VERSION=1\_2\_0

PLCBOOT=1\_2\_0

Which firmware or boot code version is to be loaded is entered to the key for the according CPU PM5xy. The key is only evaluated for mode 2 and 3 in the key

**[FirmwareUpdate] / CPUPM5x1=2 or 3.**

The following applies for TYPE:

TYPE=1 CPU

TYPE=2 DISPLAY

TYPE=3 I/O device at the CPU's I/O bus

TYPE=4 Coupler (0-internal, 1..4-external)

The CPU firmware version is set with VERSION. For example, VERSION has to be set to VERSION=1\_2\_0 for version V1.2.0 and VERSION=1\_1\_7 for version V1.1.7.

The entry PLCBOOT specifies the boot code version of the CPU. For example, VERSION has to be set to VERSION=1\_2\_0 for version V1.2.0 and VERSION=1\_1\_3 for version V1.1.3.

- **[PM5x1] or [CM572], [CM575], [CM577], [CM578]**  
TYPE=4  
VERSION=1\_045

For the couplers always TYPE=4 has to be entered.

VERSION specifies the firmware version of the coupler. For the Ethernet coupler, VERSION has to be set to, for example, VERSION=1\_045 for version V01.045.

## 6.2.3 Initializing an SD card

### 6.2.3.1 Initializing an SD card using the AC500

The file structure described above is created on the SD card when a formatted SD card is inserted into the AC500. The file SDCARD.INI contains the following entries:

```
[Status]
FunctionOfCard=0
[FirmwareUpdate]
CPUPM5x1=0
Display=0
Coupler_0=0
Coupler_1=0
Coupler_2=0
Coupler_3=0
Coupler_4=0
[UserProg]
UserProgram=0
RetainData=0
AddressData=0
```

The following error message is displayed if you insert a non-formatted SD card:

152369164 FK4 Warning Unable to read the SD card
--

Other files or subdirectories in the root directory on the SD card are kept unchanged.

### 6.2.3.2 Initializing an SD card using a PC

It is also possible to create the above described file structure on the hard disk of a PC with SD card interface.

To do this, create the required directories and an ASCII file named SDCARD.INI with Notepad, for example.

It is also possible to copy the file structure from an initialized SD card to the hard disk and then from the hard disk to SD cards that are not initialized.

## 6.3 Storing/loading the user program to/from an SD card

### 6.3.1 Storing the user program to an SD card

To store the user program to the SD card, proceed as follows:

1. Build the complete project using the menu items "Project" / "Clean all" and "Project" / "Rebuild all".
2. Load the project into the AC500.
3. Create the boot project on the controller using "Online" / "Create boot project".  
The boot project files (DEFAULT.PRG and DEFAULT.CHK) are loaded into the AC500 and flashed.  
The RUN LED on the AC500 flashes while data flashing is in progress.
4. Insert the SD card. If the SD card does not already contain the required file structure, the structure will be created  
(see also: "Initializing an SD card using the AC500").



**Caution:** If a user program is already stored on the SD card, i.e., the directory UserData\PM5x1\UserPrg already contains the files DEFAULT.PRG and DEFAULT.CHK, these files will be overwritten without any warning.

If you want to store several user programs to the SD card, you have to copy them into other directories using the PC.

5. Open the PLC Browser from the Resources tab and enter the command "**sdappl**" <ENTER>.  
The files DEFAULT.PRG and DEFAULT.CHK are loaded from the Flash memory and stored to the directory UserData\PM5x1\UserPrg on the SD card.  
In the file SDCARD.INI, the parameter "FunctionOfCard" is set to 1 (bit 0 = 1) and the parameter "UserProgram" is set to 1, i.e., the function "Load the user program" is activated.  
The RUN LED on the AC500 flashes while writing to the SD card is in progress.

If you insert the SD card containing the user program into the AC500, the SD card is loaded into the Flash memory of the AC500 (see next section).

### 6.3.2 Loading a user program from the SD card to the AC500

If an SD card is inserted into the AC500 when the **PLC is in STOP mode** or if the SD card is already inserted when switching on the control voltage, the file structure on the SD card is checked. If the file structure exists, the file SDCARD.INI is read. If the parameter "FunctionOfCard" is set to 1 (bit 0 = 1) and the parameter "UserProgram" = 1, the files DEFAULT.PRG and DEFAULT.CHK in the directory UserData\PM5x1\UserPrg on the SD card are loaded into the Flash memory of the AC500.



**Caution:** In versions V1.0 and V1.1, the user program can only be loaded with control voltage ON if FunctionOfCard=3 (i.e., firmware update and user program are loaded). To disable the firmware update, the according firmware file has to be deleted!

The RUN LED on the AC500 flashes while loading and flashing the user program is in progress.

The loaded program is activated after a PLC restart.

If the user program cannot be loaded (for example, due to missing files, wrong directory structure or mismatching project for the controller), a corresponding error message appears.

A summary of the SD card errors can be found in the section "SD card error messages".

If you insert the SD card into the AC500 when the **PLC is in RUN mode**, the user program is not loaded independent of the settings for the parameters "FunctionOfCard" (bit 0=1) and "UserProgram" (=1). Thus, the function "Load user program" can be deactivated with the PLC browser command "sdfunc 0" even if no PC card reader is available.

## 6.4 Storing/reading user data to/from an SD card

### 6.4.1 Structure of data files stored on the SD card

Depending on the AC500 CPU type, the data are stored in the following SD card directory:

AC500 CPU	Directory	File
PM571	..\UserData\PM571\UserDat	USRDATxx.DAT
PM581	..\UserData\PM581\UserDat	USRDATxx.DAT
PM591	..\UserData\PM591\UserDat	USRDATxx.DAT

A maximum of 100 files (USRDAT00.DAT...USRDAT99.DAT) can be stored in one directory.

Each data file USRDATxx.DAT can be divided into individual sectors, if necessary. The "sector label" enclosed in square brackets (such as [Sector\_01]<CR><LF>) indicates the start of the sector. Within a sector, data are saved as data sets in ASCII format. The individual values of a data set are separated by semicolon. Each data set is closed with <CR><LF> (0Dhex, 0Ahex).

This enables the direct import/export of the data from/to EXCEL. The data files can be viewed and edited using a standard ASCII editor (such as Notepad).

When saving/loading data files, observe the following rules:

- Data sets within a sector must always have the same number of values.
- Data sets in different sectors can have a different number of values.
- Values of integer data types can be stored. REAL or LREAL variables cannot be stored.
- The values of a data set must have the same data format (BYTE, WORD, INT,..).
- A sector can have data sets with different data format.  
(Warning: The user has to know the structure of the data for reading them.)
- The data sets are always appended to the end of the file when writing.
- Searching for a "sector label" within a file is possible when reading it.
- Data sets can be read starting from a particular "sector label".
- A particular data set of a sector cannot be read or written.
- If you want to read each data set individually, a "sector label" must be inserted before each data set.
- Reading and writing the data with help of the user program is done with the blocks SD\_READ and SD\_WRITE.
- The values of a data set must be available in variables successively stored in the PLC (e.g., ARRAY, STRING, %M area).
- A data file can be deleted with help of the PLC program.
- Individual data sets and/or sectors cannot be deleted with the user program.  
This has to be done on the PC using an ASCII editor such as Notepad.

## Data file examples:

### Example 1:

Data file USRDAT5.DAT without sectors:  
-> 5 data sets, each with 10 DINT values:

```
600462;430;506;469;409;465;466;474;476;-1327203
600477;446;521;484;425;480;482;490;491;-1327187
600493;461;537;499;440;496;497;505;507;-1327172
600508;477;552;515;456;511;513;521;522;-1327156
600524;492;568;530;471;527;528;536;538;-1327141
```

### Example 2:

Data file USRDAT7.DAT with sectors:  
-> 3 sectors, each with 3 data sets and 10 DINT values per data set:

```
[Sector_01]
610439;10408;10483;10446;10387;10442;10444;10452;10453;-1317225
610455;10423;10499;10462;10402;10458;10460;10467;10469;-1317209
610476;10445;10520;10483;10424;10479;10481;10489;10490;-1317188
```

```
[Sector_02]
610570;10539;10614;10577;10518;10573;10575;10583;10584;-1317094
610585;10554;10630;10592;10533;10589;10591;10598;10600;-1317078
610602;10571;10646;10609;10550;10605;10607;10615;10616;-1317062
```

```
[Sector_03]
610701;10670;10746;10708;10649;10704;10706;10714;10715;-1316963
610717;10686;10761;10724;10665;10720;10722;10730;10731;-1316947
610739;10708;10783;10746;10686;10742;10744;10751;10753;-1316926
```

## 6.4.2 Blocks for storing/reading user data to/from the SD card

The following blocks are used to write and read user data from the PLC program to/from the SD card:

**Library:** SysInt\_AC500\_Vxx.lib

**Folder:** ..\Data Storage\ SD card

The library SysExt\_AC500\_Vxx.lib is also required. Both libraries are loaded automatically when creating a project for an AC500 CPU.

**Blocks:** SD\_WRITE - Writes user data  
SD\_READ - Reads user data

The blocks SD\_WRITE and SD\_READ are described in the documentation for the block library SysInt\_AC500\_Vxx.lib.

The inputs and outputs of the block SD\_WRITE and their functions are as follows:

Name	Type	Assignment
<b>Inputs</b>		
EN	BOOL	The FALSE->TRUE edge starts the write process
ATTRIB	BYTE	Write attribute of the block: 1 - Delete file 2 - Write append 3 - Write sector label
FILENO	BYTE	Consecutive file number 0 <= xx <= 99 (USRDATxx.DAT)
SEG	POINTER TO STRING	Pointer to sector label string (via ADR operator)
FORMAT	BYTE	Data format: 00 hex - 0 - BYTE 01 hex - 1 - CHAR 10 hex - 16 - WORD 11 hex - 17 - INT 20 hex - 32 - DWORD 21 hex - 33 - DINT
NVAR	WORD	Number of variables to be written
ADRVAR	DWORD	Address of the first variable, starting from which the data are available in the PLC (via ADR operator)
<b>Outputs</b>		
DONE	BOOL	Function completed
ERR	BOOL	Error: FALSE=no error, TRUE=error
ERNO	INT	Error number

The inputs and outputs of the block SD\_READ and their functions are as follows:

Name	Type	Assignment
<b>Inputs</b>		
EN	BOOL	The FALSE->TRUE edge starts the read process
ATTRIB	BYTE	Read attribute of the block: 1 - Open file, seek sector, read data set 2 - Open file, read data set 3 - Open and read next data set 4 - Read data set, close file 5 - Close file
FILENO	BYTE	Consecutive file number 0 <= xx <= 99 (USRDATxx.DAT)
SEG	POINTER TO STRING	Pointer to sector label string (via ADR operator)
FORMAT	BYTE	Data format: 00 hex - 0 - BYTE 01 hex - 1 - CHAR 10 hex - 16 - WORD 11 hex - 17 - INT 20 hex - 32 - DWORD 21 hex - 33 - DINT
NVAR	WORD	Number of variables to be read
ADRVAR	DWORD	Address of the first variable starting from which the data are stored to the PLC (via ADR operator)
<b>Outputs</b>		
DONE	BOOL	Function completed
ERR	BOOL	Error: FALSE=no error, TRUE=error
ERNO	INT	Error number

The error messages of the blocks SD\_READ and SD\_WRITE are described in the chapter Function block error messages.

### 6.4.3 Deleting a data file stored on the SD card

To delete a data file from the SD card, proceed as follows:

1. Insert the SD card.
2. Call the block SD\_WRITE with the following settings:  
EN := TRUE  
ATTRIB := 1 (\* delete \*)  
FILENO := 0...99 (\* number of the file to be deleted \*)  
SEG, FORMAT, NVAR, ADRVAR - any

### 6.4.4 Storing user data to the SD card - data file without sectors

Proceed as follows to store user data to the SD card in a data file without sectors:

1. Insert the SD card.
2. Write a data set by calling the block SD\_WRITE with the following settings:  
EN := TRUE (\* FALSE/TRUE edge starts writing \*)  
ATTRIB := 2 (\* write append \*)  
FILENO := 0...99 (\* number of the file to which the data set is to be appended \*)  
SEG := Address of the variable of the sector label (\* any \*)  
FORMAT := Data format  
NVAR := Number of values in data set  
ADRVAR := Address of the first variable to be written

If no corresponding file exists, then it is created.

The write process is successfully completed when output DONE:=TRUE and output ERR:=FALSE. A write error is indicated by ERR:=TRUE and ERNO<>0.

3. Further data sets can be written with the same block settings after the completion message is indicated (output DONE=TRUE). This process is started with a FALSE/TRUE edge at input EN.



**Note:** The file USRDATxx.DAT is saved as USRDATxx.BAK for each write process and an "Open file / Write file / Close file" procedure is performed.



## 6.4.5 Storing user data to the SD card - data file with sectors

Proceed as follows to store user data to the SD card in a data file with sectors:

1. Insert the SD card.
2. Write the sector label by calling the block SD\_WRITE with the following settings:  
EN := TRUE  
ATTRIB := 3 (\* write sector \*)  
FILENO := 0...99 (\* number of the file to which the data set is to be appended \*)  
SEG := Address of the variable of the sector label  
FORMAT := Data format  
NVAR := Number of values in data set  
ADRVAR := Address of the first variable to be written  
If no corresponding file exists, then it is created.  
The sector is successfully written when output DONE:=TRUE and output ERR:=FALSE. A write error is indicated by ERR:=TRUE and ERNO<>0.
3. Write a data set by calling the block SD\_WRITE with the following settings:  
EN := TRUE (\* FALSE/TRUE edge starts writing \*)  
ATTRIB := 2 (\* write append \*)  
FILENO := 0...99 (\* number of the file to which the data set is to be appended \*)  
SEG := Address of the variable of the sector label  
FORMAT := Data format  
NVAR := Number of values in data set  
ADRVAR := Address of the first variable to be written  
The write process is successfully completed when output DONE:=TRUE and output ERR:=FALSE. A write error is indicated by ERR:=TRUE and ERNO<>0.
4. Further data sets can be written with the same block settings after the completion message is indicated (output DONE=TRUE). This process is started with a FALSE/TRUE edge at input EN.
5. If you want to write further sectors and data sets, repeat steps 2..4.



**Note:** The file USRDATxx.DAT is saved as USRDATxx.BAK for each write process and an "Open file / Write file / Close file" procedure is performed.

## 6.4.6 Loading user data from the SD card - data file without sectors

Proceed as follows to read user data from a data file without sectors on the SD card and write them to the PLC:

1. Insert the SD card.
2. Read a data set by calling the block SD\_READ with the following settings:  
EN := TRUE (\* FALSE/TRUE edge starts reading \*)  
ATTRIB := 2 (\* open / read \*)  
FILENO := 0...99 (\* number of the file which is to be read \*)  
SEG := Address of the variable of the sector label (\* any \*)  
FORMAT := Data format  
NVAR := Number of values in data set  
ADRVAR := Address of the first variable into which data are to be written  
The read process is successfully completed when output DONE:=TRUE and output ERR:=FALSE. A read error is indicated by ERR:=TRUE and ERNO<>0.
3. Further data sets can be read with the following settings after the completion message is displayed (output DONE=TRUE). This process is started with a FALSE/TRUE edge at input EN:  
EN := TRUE (\* FALSE/TRUE edge starts reading \*)  
ATTRIB := 3 (\* continue read \*)  
FILENO := 0...99 (\* number of the file which is to be read \*)  
SEG := Address of the variable of the sector label (\* any \*)  
FORMAT := Data format  
NVAR := Number of values in data set  
ADRVAR := Address of the first variable into which data are to be written  
If an unexpected sector label or the end of file (EOF) is detected when reading, a corresponding error message is generated.
4. To read a further data set and to close the file afterwards, call the block SD\_READ with the following settings after the completion message (output DONE=TRUE) and start the process with a FALSE/TRUE edge at input EN:  
EN := TRUE (\* FALSE/TRUE edge starts reading \*)  
ATTRIB := 4 (\* read / close \*)  
FILENO := 0...99 (\* number of the file which is to be read \*)  
SEG := Address of the variable of the sector label (\* any \*)  
FORMAT := Data format  
NVAR := Number of values in data set  
ADRVAR := Address of the first variable into which data are to be written  
If an unexpected sector label or the end of file (EOF) is detected when reading, a corresponding error message is generated.
5. To close the file, call the block SD\_READ with the following settings after the completion message (output DONE=TRUE) and start the process with a FALSE/TRUE edge at input EN:  
EN := TRUE (\* FALSE/TRUE edge starts close process \*)  
ATTRIB := 4 (\* close \*)  
FILENO := 0...99 (\* number of the file which is to be read \*)  
SEG := Address of the variable of the sector label (\* any \*)  
FORMAT := Data format  
NVAR := Number of values in data set (\* any \*)  
ADRVAR := Address of first variable (\* any \*)

## 6.4.7 Loading user data from the SD card - data file with sectors

Proceed as follows to read user data from a data file with sectors on the SD card and write them to the PLC:

1. Insert the SD card.
2. Seek a sector label and read a data set by calling the block SD\_READ with the following settings:  
EN := TRUE (\* FALSE/TRUE edge starts reading \*)  
ATTRIB := 1 (\* open / seek / read \*)  
FILENO := 0...99 (\* number of the file which is to be read \*)  
SEG := Address of the variable of the sector label  
FORMAT := Data format  
NVAR := Number of values in data set  
ADRVAR := Address of the first variable into which data are to be written  
The read process is successfully completed when output DONE:=TRUE and output ERR:=FALSE. A seek error is indicated by ERR:=TRUE and ERNO<>0.
3. Further data sets can be read with the following settings after the completion message is indicated (output DONE=TRUE). This process is started with a FALSE/TRUE edge at input EN:  
EN := TRUE (\* FALSE/TRUE edge starts reading \*)  
ATTRIB := 3 (\* continue read \*)  
FILENO := 0...99 (\* number of the file which is to be read \*)  
SEG := Address of the variable of the sector label (\* any \*)  
FORMAT := Data format  
NVAR := Number of values in data set  
ADRVAR := Address of the first variable into which data are to be written  
If an unexpected sector label or the end of file (EOF) is detected when reading, a corresponding error message is generated.
4. If you want to read further sectors / data sets, close the file and then repeat steps 2 and 3.
5. To read a further data set and to close the file afterwards, call the block SD\_READ with the following settings after the completion message (output DONE=TRUE) and start the process with a FALSE/TRUE edge at input EN:  
EN := TRUE (\* FALSE/TRUE edge starts reading \*)  
ATTRIB := 4 (\* read / close \*)  
FILENO := 0...99 (\* number of the file which is to be read \*)  
SEG := Address of the variable of the sector label  
FORMAT := Data format  
NVAR := Number of values in data set  
ADRVAR := Address of the first variable into which data are to be written  
If an unexpected sector label or the end of file (EOF) is detected when reading, a corresponding error message is generated.
6. To close the file, call the block SD\_READ with the following settings after the completion message (output DONE=TRUE) and start the process with a FALSE/TRUE edge at input EN:  
EN := TRUE (\* FALSE/TRUE edge starts close process \*)  
ATTRIB := 4 (\* close \*)  
FILENO := 0...99 (\* number of the file to be read \*)  
SEG := Address of the variable of the sector label  
FORMAT := Data format  
NVAR := Number of values in data set (\* any \*)  
ADRVAR := Address of first variable (\* any \*)

## 6.5 Storing and loading retentive data to/from an SD card

Retentive variables (RETAIN variables) declared with VAR\_RETAIN .. END\_VAR or VAR\_GLOBAL RETAIN .. END\_VAR are set to their initialization values during download. If the RETAIN variables shall keep their values also after a download, they have to be saved before starting the download and reloaded after the download is completed.

This is possible using the PLC browser commands "saveretain" and "restoreretain". The command "saveretain" saves the RETAIN variables to the file RETAIN.BIN on the SD card.

Depending on the CPU type, the files are stored to the following directories on the SD card:

AC500 CPU	Directory	Files
PM571	..\UserData\PM571\RetDat	RETAIN.BIN
PM581	..\UserData\PM581\RetDat	RETAIN.BIN
PM591	..\UserData\PM591\RetDat	RETAIN.BIN

## 6.6 Firmware update from the SD card

### 6.6.1 Storing the firmware to the SD card

Storing the firmware to the SD card is done using a standard PC card reader with SD card interface.

To do this, proceed as follows:

1. Initialize the SD card, i.e., create the file structure the PLC requires by, for example, inserting a new SD card into the AC500 (see also: "Initializing an SD card using the AC500").
2. Copy the firmware files into the corresponding directory:

AC500 CPU	Directory	File
PM571	..\Firmware\PM571	MICR.GZA
PM581	..\Firmware\PM581	MINI.GZA
PM591	..\Firmware\PM591	MIDI.GZA

3. Change the command file SDCARD.INI located in the root directory on the SD card as follows:  
Parameter "FunctionOfCard=2" (or =3 for firmware and user program update)  
Parameter CPUPM5x1=1

A specific firmware version can be loaded as of version V1.2. This is done by setting the parameter CPUPM5x1=2 or 3 and creating an according key for the CPU. The firmware has to be copied to the according directory. See the chapter The command file "SDCARD.INI".

### 6.6.2 Updating the firmware of the AC500 CPU from the SD card

To update the firmware of the AC500 CPU via SD card, proceed as follows:

1. Prepare the SD card as described in the section "Storing the firmware to the SD card".
2. Switch off the PLC control voltage.
3. Insert the SD card.
4. Switch on the PLC control voltage.  
The file structure on the SD card is checked when booting the PLC. If the file structure exists, the file SDCARD.INI is read. If the parameter "FunctionOfCard" is set to 2 (bit 1 = 1) and the parameter "CPUPM5x1" = 1, the file MIxx.gzaS19 is searched in the directory Firmware\PM5x1 (depends on the used CPU type) and then loaded into the PLC, checked and flashed.  
The individual steps are indicated as follows:

Process	Indication	Remark
Reading the firmware	RUN LED flashes fast	If you remove the SD card during reading, the previously stored firmware version is kept.
Flashing the firmware	RUN LED and ERR LED flash fast	<b>Warning:</b> If the control voltage is switched off during flashing, the firmware will be corrupted!
Firmware update completed successfully	RUN LED flashes slow (app. 1Hz)	
Incorrect firmware update	RUN LED and ERR LED flash slow (app. 1Hz)	

A specific firmware version can be loaded as of version V1.2. This is done by setting the parameter CPU5x1=2 or 3 and creating an according key for the CPU. The firmware has to be copied to the according directory. See the chapter The command file "SDCARD.INI".



**Note:** If the file SDCARD.INI contains the parameter setting "FunctionOfCard=3" (firmware update / load user program), first the firmware and then the user program are read from the SD card and then stored in the according Flash memory.

## 6.7 Writing and reading the project sources to/from the SD card

Usually it is sufficient to load the boot project (compiled user program) to the PLC. Sometimes, however, the user may wish to transfer the entire project sources to the PLC. This is why two different commands are available for this: The command "Online" => "Create boot project" writes the boot project to the flash memory of the PLC whereas the command "Online" => "Sourcecode download" stores the project sources to the SD card.

The project sources and all parts of a project are packed in the compressed file "SOURCE.DAT":

- All blocks in all IEC languages with all comments and symbols
- All visualizations
- Task configuration
- PLC configuration
- Online change information

The following is also contained if set accordingly in the project options:

- All loaded libraries
- All required configuration files

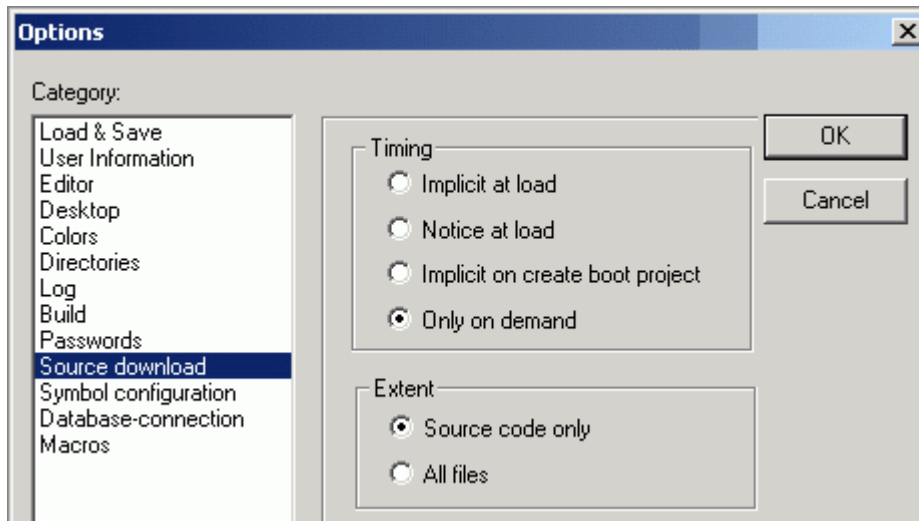
Reading the project sources can be done in two ways:

- Reading the compressed file SOURCE.DAT from the PLC (with an SD card inserted)
- Opening the compressed project directly on the PC using an SD card reader or opening a copy of the file SOURCE.DAT stored on the hard disk of the PC.

## 6.7.1 Writing the project sources from PC to SD card

Writing the project sources to the SD card is done as follows:

- Select the project sources to be written by selecting: "Project" => "Options" => "Source download"

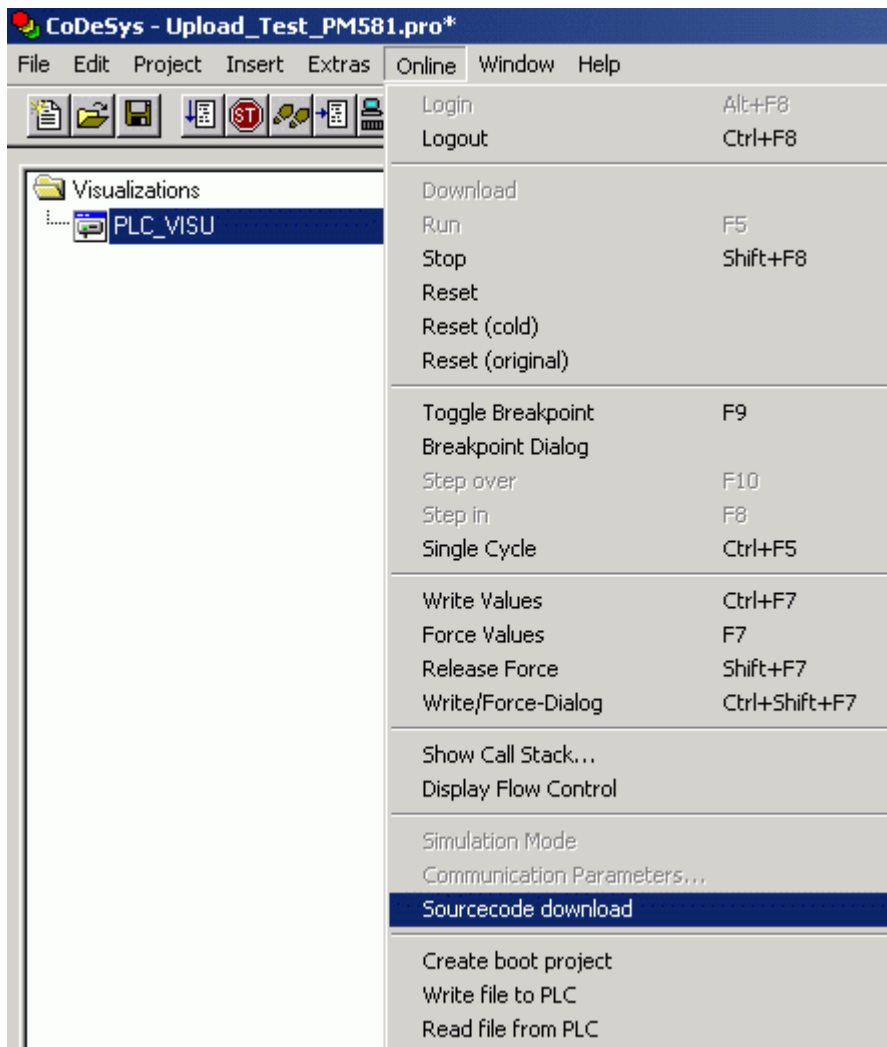


The "Timing" options specify the time when the project sources are to be written. The default setting is "On demand".

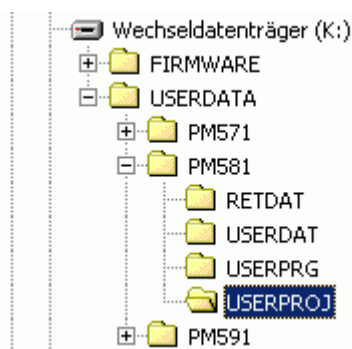
The "Extent" options define which files are to be written. Using the option "Source code only" writes all files that are part of the project. The option "All files" also includes the libraries and configuration files.

Especially with the option "All files" the compressed file SOURCE.DAT can have a size of several megabytes (in particular if visualizations with bitmaps are included!).

- Insert the SD card into the CPU. In case a new SD card is used, the directory structure required for the AC500 is created. This is indicated by the flashing LED RUN.
- Login to the PLC (via serial interface, Ethernet or ARCNET, depending on the interfaces available for your PLC).
- If possible, stop the PLC with the command "Online" => "Stop". (In Stop mode of the PLC, loading the project sources is much faster than in Run mode.)
- Select "Online" => "Sourcecode download".



- The compressed file SOURCE.DAT is created, downloaded to the PLC and written to the SD card. The file is located in the following directory on the SD card:  
[device]:\USERDATA\USERPROJ\SOURCE.DAT



- Restart the PLC with "Online" => "Start".
- Logout from the PLC by selecting "Online" => "Logout".

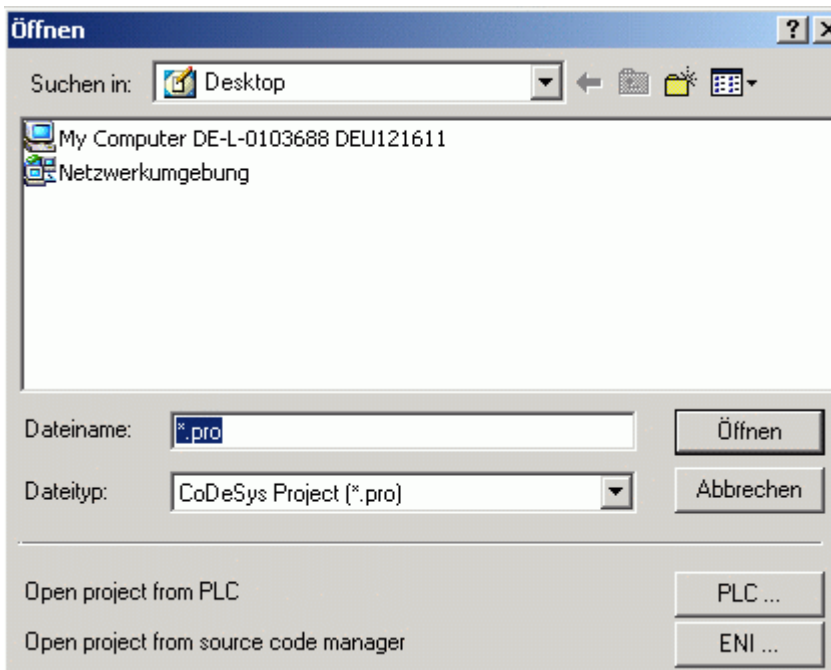
Example of the project sources size:

Project file	Upload_Test_PM581.pro		
Size of built user program	18638	18	kB
	Bytes		
Size of project sources (all files)	321503	314	kB
	Bytes		
Size of project sources (sources only)	90297	89	kB
	Bytes		

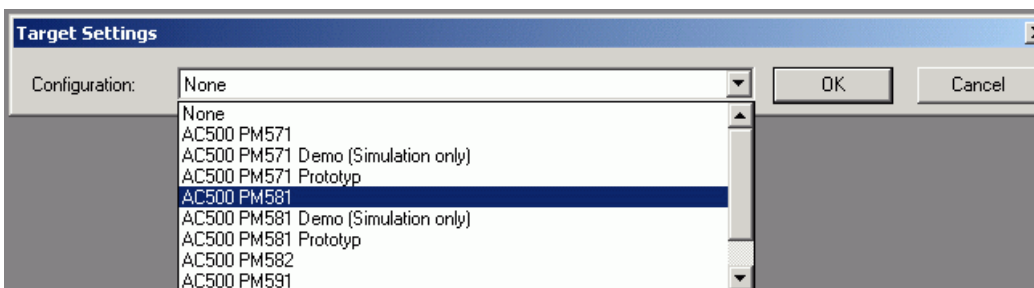
## 6.7.2 Loading the project sources from the PLC's SD card into the PC

Loading the project sources from the SD card into the PC is done as follows:

- Insert the SD card into the CPU (if not already inserted).
- Start the Control Builder PS501 (CoDeSys.exe).
- Select "File" => "Open".
- Click "PLC".

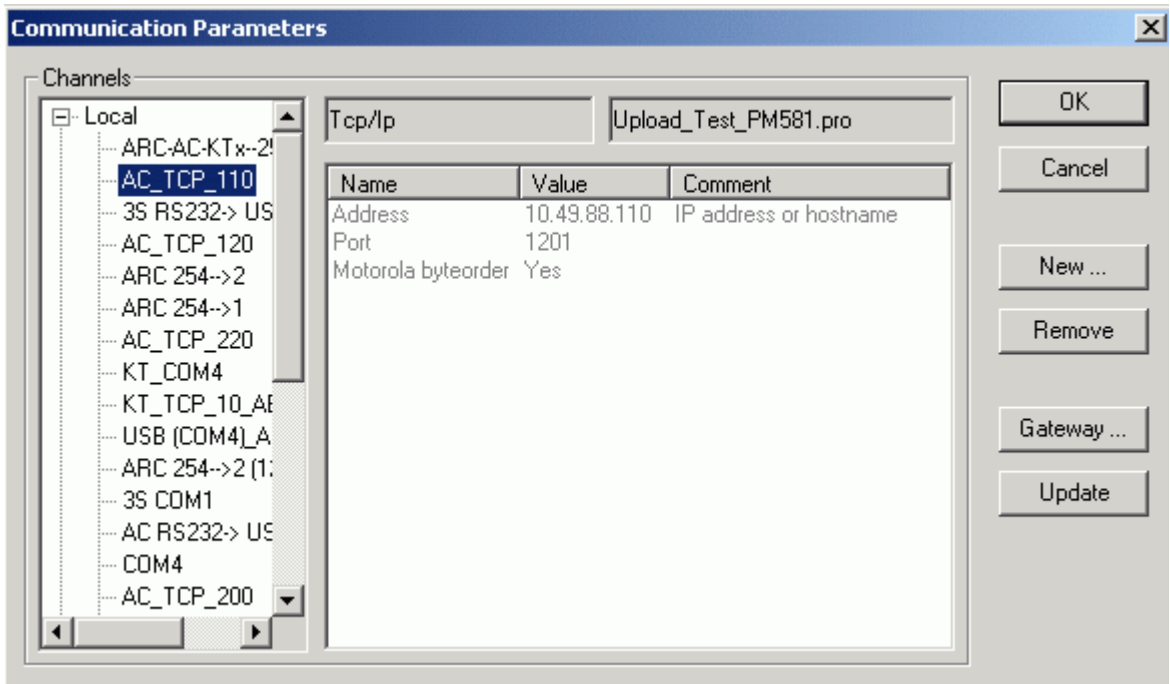


- Select the required CPU type (target), for example PM581.

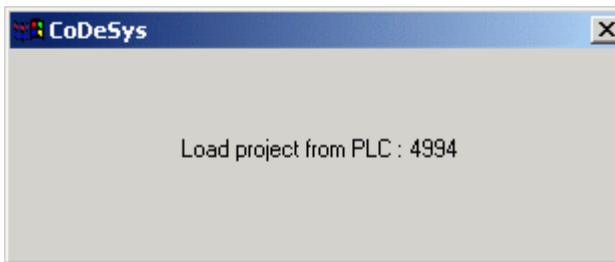


- Select the communication channel, for example an Ethernet channel.

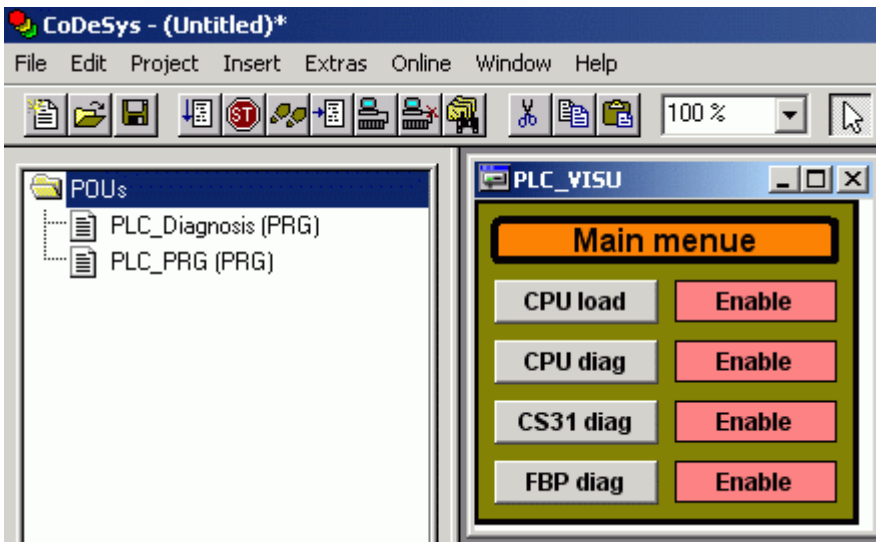




- The project sources are loaded block-wise.



- Once the last block has been loaded, the project is decompressed by the software automatically. Now all project files are available.



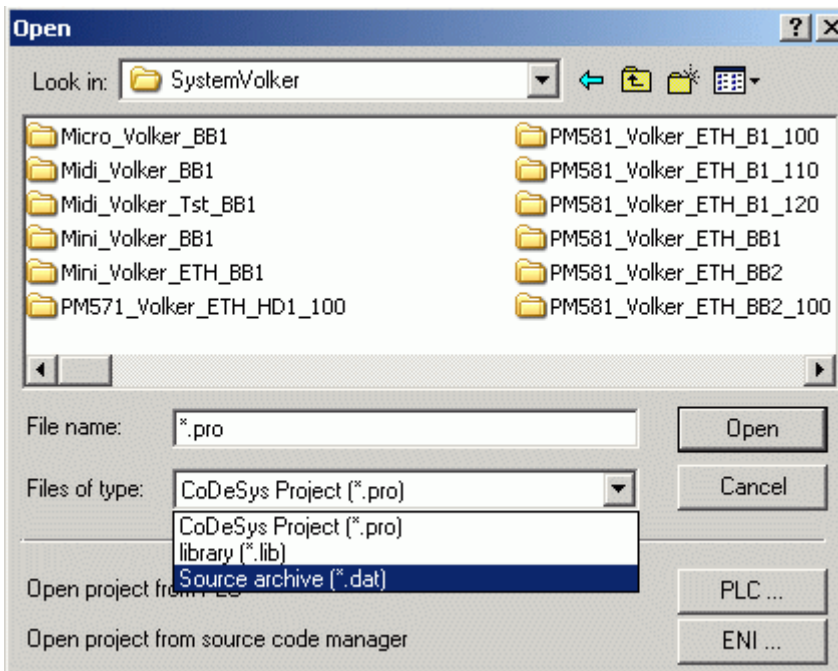
You can use this project to directly login to the PLC (no build required).

- If you want to change the project, save it under a new name, change and build the source code and then download the changed and built code (user program) to the PLC. Reload the boot project and the project sources.

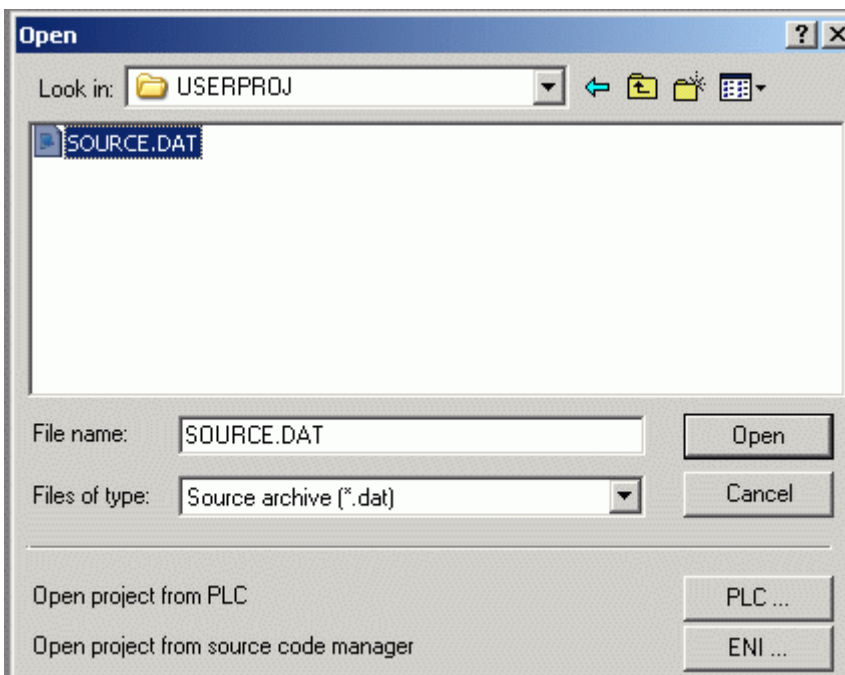
### 6.7.3 Loading the project sources from the SD card using the PC SD card reader

It is also possible to read the file SOURCE.DAT containing the project sources directly using the SD card reader for the PC. Proceed as follows:

- Insert the SD card into the PC SD card reader.
- Start the Control Builder PS501 (CoDeSys.exe).
- Select "File" => "Open".
- "Source archive (\*.dat)" from the "Files of type" list box



- In the "Look in" list box, navigate to the SD card directory [Device]:\USERDATA\PM5x1\USERPROJ\SOURCE.DAT and select the file SOURCE.DAT. Click "Open".



- The project is loaded and then decompressed automatically.  
It is now possible to directly switch to online mode (without saving the project).
- If you want to change the project, save it under a new name, change and build it and then download the built user program with the boot project to the PLC. Insert the SD card into the PLC and rewrite the project sources to the SD card.

## 6.8 SD card error messages

The following table shows the error messages generated by the SD card.

Error number	Class	Description	Effect	Remedy
	E4 Warning	Incorrect directory structure when reading the user program	User program not loaded	Correct the directory structure with the PC
152369165	E4 Warning	Write error or SD card full	Data not written	Check the SD card with the PC and, if necessary, delete some files on the SD card or cut and paste them to the PC or insert another SD card
152369164	E4 Warning	Unable to read the SD card		Check the SD card with the PC
152369233	E4 Warning	Write-protection: Unable to initialize the SD card	Directory structure cannot be created	Remove write-protection
152369297	E4 Warning	Write-protection: Error writing the PLC program to the SD card	Data not written	Remove write-protection
152369361	E4 Warning	Write-protection: Error when storing the PLC firmware to the SD card	Data not written	Remove write-protection
152369425	E4 Warning	Write-protection: Error when storing the retentive variables (RETAIN.BIN) to the SD card	Data not written	Remove write-protection
152369489	E4 Warning	Write-protection: Error when writing the card function to the SD card	Data not written	Remove write-protection

## 7 Data storage in Flash memory

### 7.1 Blocks used for data storage

The library "SysInt\_AC500\_V10.lib" located in the directory "Data storage" / "Flash" contains the following blocks which are used to store data in the Flash memory:

Block	Function
FLASH_DEL	Deletes a data segment in the Flash memory
FLASH_READ	Reads a data segment from the Flash memory
FLASH_WRITE	Writes a data segment to the Flash memory

### 7.2 Example program for data storage

On the PS501 CD-ROM the directory:

[Device]:\CD\_AC500\Examples\Flash\_Data

contains the examples:

- FlashData\_Structure\_v10.pro Saves/loads structured data to/from Flash memory
- PM581\_ST\_Flash.pro Saves/loads data sets

## 8 Real-time clock and battery in the AC500

### 8.1 General notes concerning the real-time clock in the AC500

The real-time clock in the AC500 operates as a PC clock. It saves date and time to a DWORD in DT format (DATE AND TIME FORMAT), i.e., in seconds passed since the start time: 1 January 1970 at 00:00.

If a battery is connected and full, the real-time clock continues to run even if the control voltage is switched off.

If no battery is inserted or the battery is empty, the real-time clocks starts with the value 0 (=1970-01-01, 00:00:00).

When switching on the control voltage, the system clock of the operating system is set to the value of the real-time clock.

### 8.2 Setting and displaying the real-time clock

#### 8.2.1 Setting and displaying the real-time clock with the PLC browser

The PLC browser commands **date** and **time** are used to set the real-time clock.

The commands **date** <ENTER> or **time** <ENTER> display the current date and time of the real-time clock.

The command:

**date yyyy-mm-dd**<ENTER> (year-month-day)  
sets the date.

The command:

**time hh-mm-ss**<ENTER> (hours-minutes-seconds)  
sets the time.

**Example:**

The real-time clock should be set to 22 February 2005, 16:50.

**Enter the date:**

date 2005-02-22<ENTER>

**Display:**

date 2005-02-22  
Clock set to 2005-02-22 08:01:07  
(Remark: the time remains unchanged)

**Enter the time:**

time 16:50<ENTER> (seconds are optional)

**Display:**

time 16:50  
Clock set to 2005-02-22 16:50:00

## 8.2.2 Setting and displaying the real-time clock with the user program

The following blocks located in the folder "Realtime clock" of the system library SysExt\_AC500\_Vxx.LIB can be used to set and display the real-time clock (RTC) with help of the user program:

Block	Function
CLOCK	Sets and displays the real-time clock with values for year, month, day, hours, minutes and seconds. Also the day of week is indicated (Mo=0, Tue=1, Wed=2, Thu=3, Fr=4, Sa=5, Su=6). <b>Note:</b> The week of day cannot be set. It is given by the real-time clock. The input DAY_SET is ignored.
CLOCK_DT	Sets and displays the real-time clock in DT format, for example DT#2005-02-17-17:15:00.

The blocks CLOCK and CLOCK\_DT are described in the documentation for the system library SysExt\_AC500\_Vxx.LIB.

## 8.3 The AC500 battery

The AC500 battery buffers the following data in case of "control voltage off":

- Retentive variables in SDRAM (VAR\_RETAIN..END\_VAR)
- File "Persistent.dat" in SDRAM (VAR\_RETAIN PERSISTENT .. END\_VAR) (in version V1.0.x only)
- Persistent data in %R area (as of version V1.2.0)
- Date and time of the real-time clock

If no battery is inserted or if the battery is empty, a warning (E4) is generated and the LED "ERR" lights up.

If no battery is required for the application (and thus no battery is inserted), a warning is generated and the error LED lights up each time the controller is switched on. To avoid this battery error indication, the parameter "Check Battery" is available under "CPU parameters" in the PLC configuration. The default setting of this parameter is "On", i.e., battery check is performed. If this parameter is set to "Off", the battery check is still performed and a corresponding error message is still generated each time the control voltage is switched on, but the system automatically quits this error and therefore the error LED does not light up (provided no further error exists).

The status of the battery can be checked in the PLC browser using the command "batt". The following is output:

```
0 Battery empty
20 Remaining battery charge below 20 %
100 Battery charge OK
```

In the user program, the battery status can be checked with the function "BATT" which is available in the folder "Battery" of the system library SysExt\_AC500\_Vxx.LIB. The following is output:

```
0 Battery empty
20 Remaining battery charge below 20 %, battery must be replaced
100 Battery charge OK
```

## 9 The fast counters in the AC500

### 9.1 Activating the fast counters via the I/O bus

The function "fast counters" is available in S500 I/O modules with firmware version V1.3 and later.

The digital I/O modules on the I/O bus contain two fast counters per module. If the I/O module does not have digital outputs, the corresponding counter modes are not valid. In case of an incorrect parameter setting, a diagnosis message is sent.

The fast counters are activated by setting the counter mode with the parameter "Fast counter" in the PLC configuration for the according I/O device (see also chapter "PLC configuration / I/O bus").

Control of the fast counter(s) is performed via the I/O data contained in the control byte of the submodule "Fast counter".

### 9.2 Counting modes of the fast counters

The counting modes of the fast counters are described in detail in the chapter "The fast counters of S500 I/O devices".

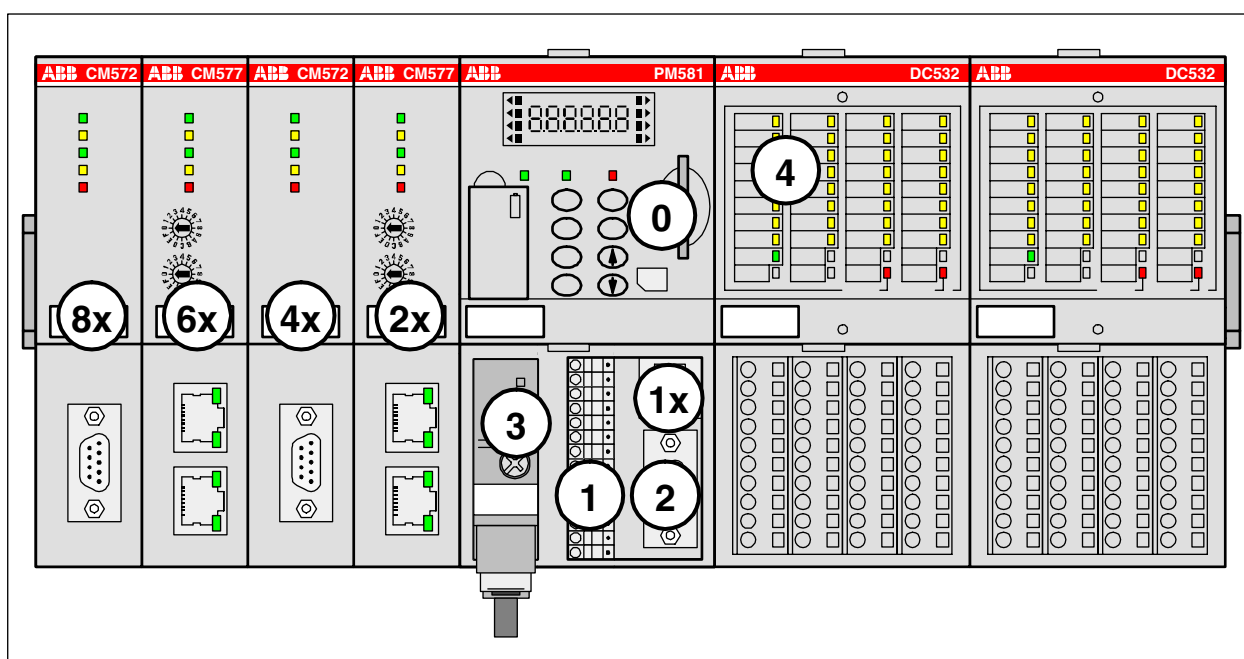
For an easy use, the blocks in the library Counter\_AC500\_V11.LIB can be used. These blocks are described in detail in the library documentation.

## 10 Programming and test

### 10.1 Programming interfaces to the AC500 used by the Control Builder

The AC500 controllers provide the following interfaces for communication with other devices:

No.	Designation	Interface	Programming access
0	CPU	Own CPU	CPU for online operation
1	COM1	Serial interface COM1	yes
2	COM2	Serial interface COM2	yes
3	FBP	FBP slave interface	yes (PM59x as of FW V1.2.0)
4	I/O bus	I/O bus	no
1x	Line 0	Internal coupler with channel 0 x 9	depends on type
2x	Line 1	Coupler inserted in slot 1 with chan. 0 x 19	depends on type
4x	Line 2	Coupler inserted in slot 2 with chan. 0 x 19	depends on type
6x	Line 3	Coupler inserted in slot 3 with chan. 0 x 19	depends on type
8x	Line 4	Coupler inserted in slot 4 with chan. 0 x 19	depends on type



#### Communication drivers:

The following communication drivers are available for programming the AC500:

Serial (RS232)	Serial interface driver
ABB RS232 Route AC	Driver for serial interfaces with routing functionality (as of PS501 V1.2, routing as of firmware V1.3.x)
TCP/IP	Ethernet driver
ABB Tcp/Ip Level 2 AC	Ethernet driver with routing functionality (as of PS501 V1.2, routing as of firmware V1.3.x)
ABB Arcnet Route fast AC	Driver for programming via ARCNET with routing and adjustable block size (as of PS501 V1.2, routing as of firmware V1.3.x)
Serial (Modem)	Modem driver for modem connected to serial interface of the PC and PLC

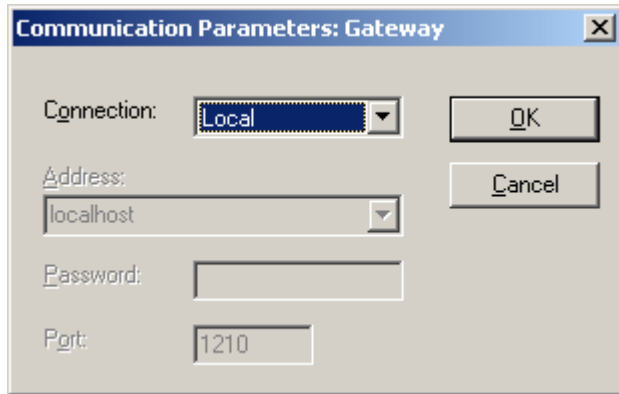




**Note:** Routing is available with version V1.3.x of the Control Builder and PLC firmware.

### **Gateway configuration:**

In the Control Builder, select "Online/Communication Parameters/Gateway" and select "Local" from the "Connection" list box (see also 3S Operation Manual / The Individual Components / Online Communication Parameters):



The following gateway settings apply for the Ethernet driver "ABB Tcp/Ip Level 2 AC":

Connection: Tcp/Ip  
Address: localhost  
Port: 1210

### **Setting the communication parameters:**

The communication parameters and address data are set in the Control Builder by selecting the desired driver and specifying the parameters in the "Communication Parameters" window, which can be opened using the menu item "Online/Communication Parameters".

The following sections describe the drivers listed above and their settings.

## **10.2 Programming via the serial interfaces**

The operation modes of the serial interfaces COM1 and COM2 are described in the chapter "PLC configuration". Both serial interfaces COM1 and COM2 are defined as programming interface by default.

The Installation guide provides information how to set the serial interfaces for the different PC operating systems.

PC and PLC are connected via the system cable TK501.

The interface parameters are set to fixed values and cannot be changed.

Baudrate: 19200 baud  
Parity bit: no  
Data bits: 8  
Stop bits: 1  
Synchronization: none  
Motorola byteorder: yes

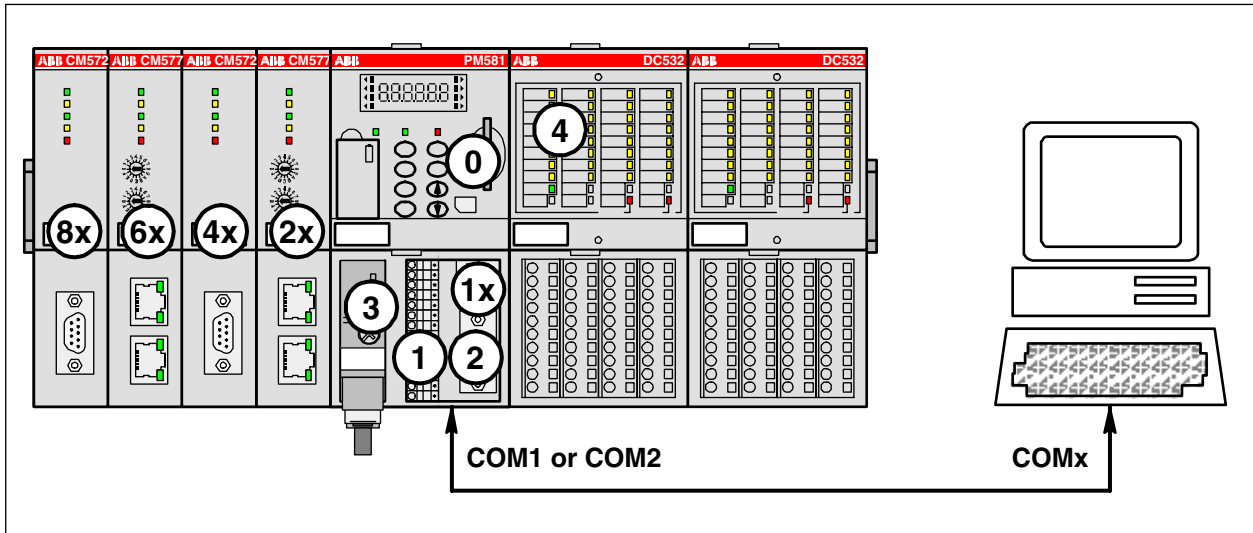
The following drivers are available for programming via the serial interfaces:

- Serial (RS232)
- ABB RS232 Route AC (as of PS501 V1.2, routing as of firmware V1.3.x)

## 10.2.1 Serial driver "Serial (RS232)"

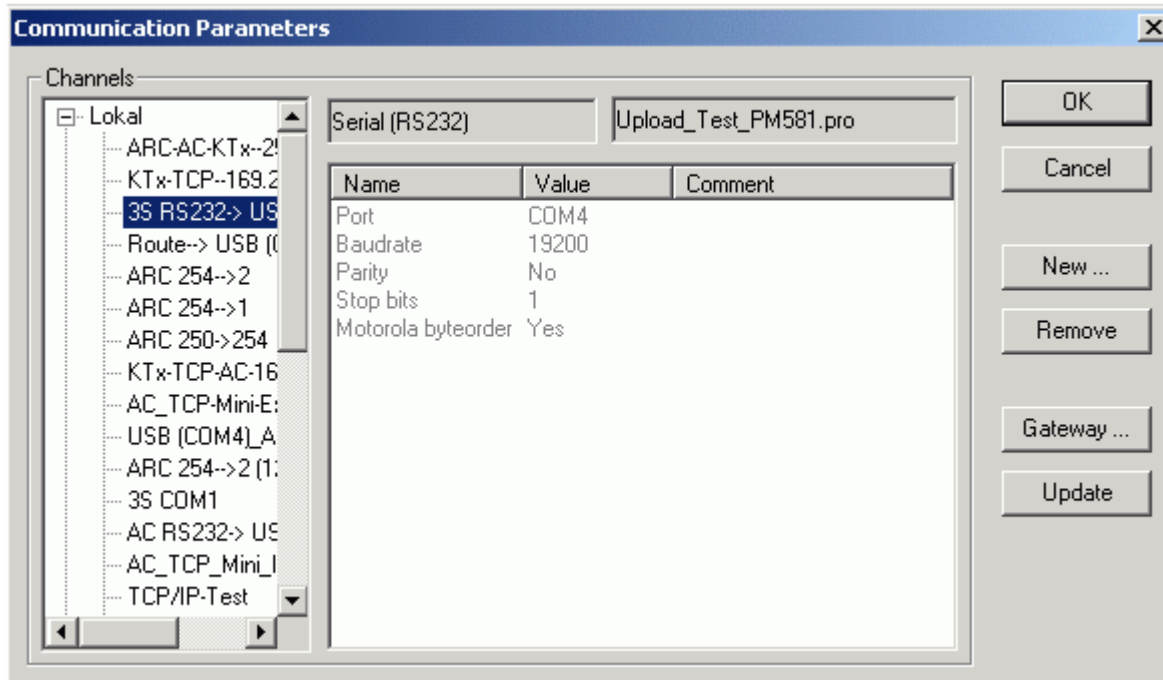
The serial driver "Serial (RS232)" provides the following functions:

- Online operation of the PLC with the Control Builder
- Online operation of the PLC with the fieldbus configurator SYCON.net
- OPC connection with OPC server, as of version V1.0
- Parallel operation of Control Builder and SYCON.net
- Parallel operation of Control Builder and OPC server



To define a new gateway channel for the serial driver, select "Online/Communication Parameters" and press the button "New" in the "Communication Parameters" window. In the appearing window, enter a name for the channel (for example USB->COM4) and select the driver "Serial RS232" from the device list. Select the desired values by **double clicking** the corresponding parameter:

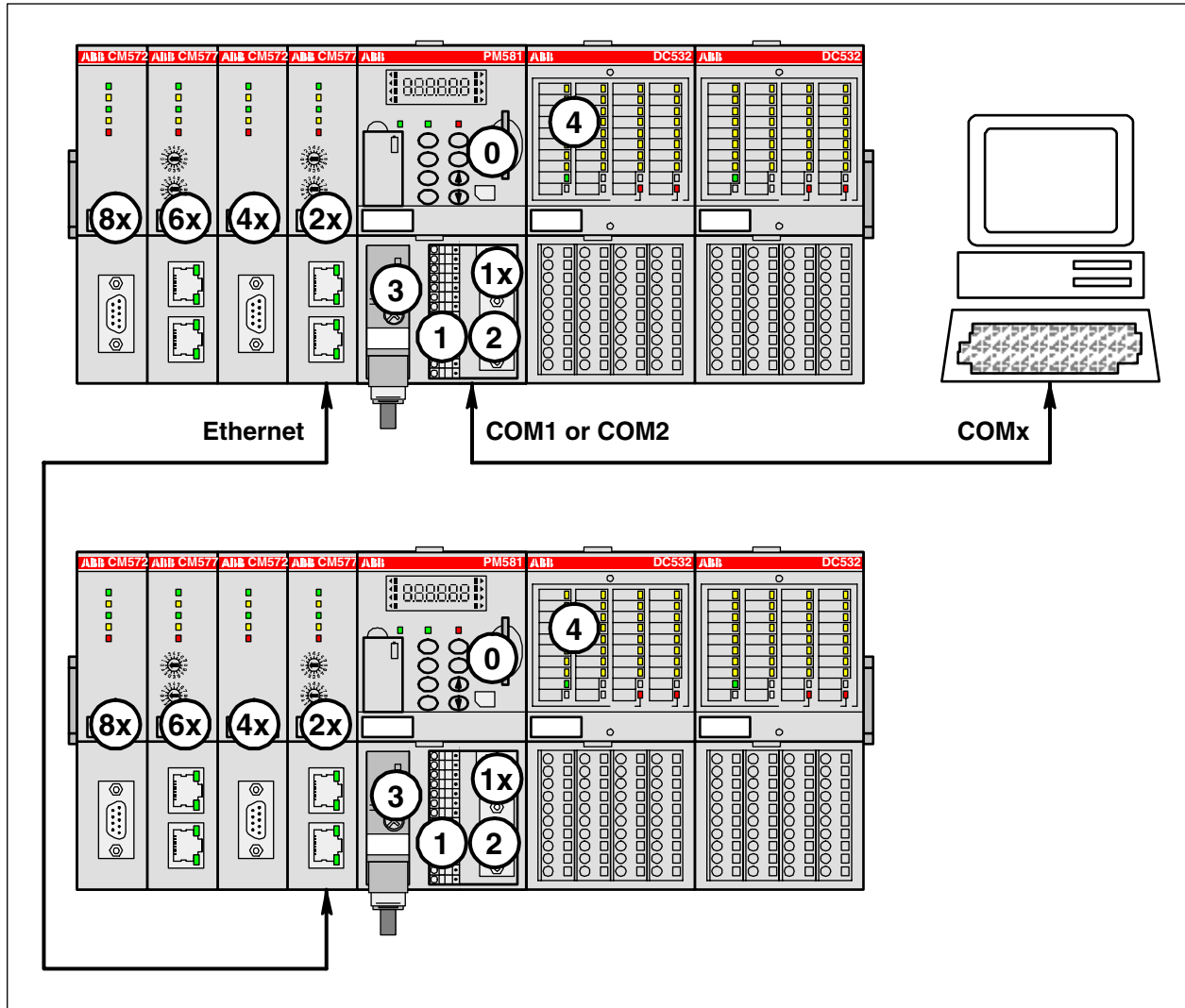
Port: COM port on the PC  
 Baud rate: 19200 Baud  
 Motorola byteorder: Yes



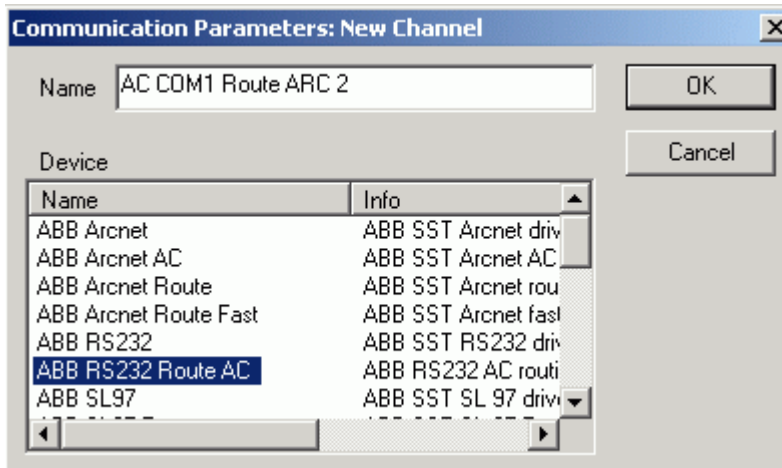
## 10.2.2 Serial driver "ABB RS232 Route AC"

As of version V1.2.0, the serial routing driver "ABB RS232 Route AC" is available in addition to the serial driver. This driver provides the following functions:

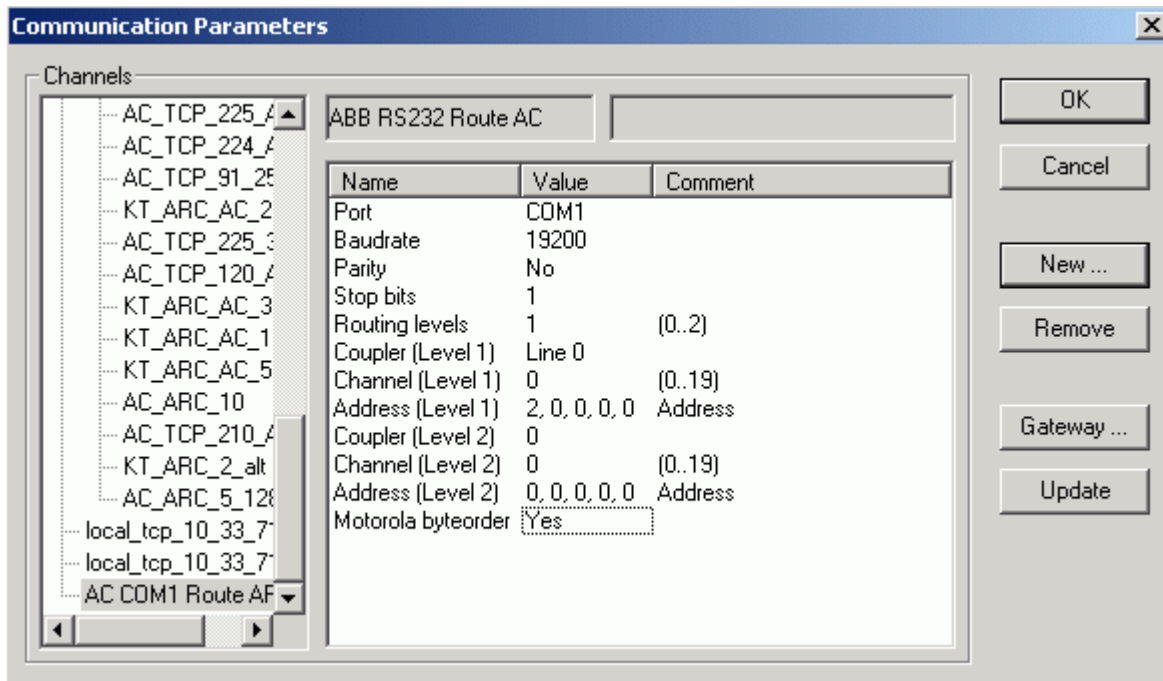
- Online operation of the PLC with the Control Builder
- Online operation of the PLC with the fieldbus configurator SYCON.net
- OPC connection with OPC server, as of version V1.2.0
- Online operation of PLCs connected via Ethernet or ARCNET using the serial interface (Control Builder version V1.3 and later)
- Parallel operation of Control Builder and SYCON.net
- Parallel operation of Control Builder and OPC server
- Online operation of AC31 series 90 controllers (07KT9x)



To define a new gateway channel for the serial routing driver, select "Online/Communication Parameters" and press the button "New" in the "Communication Parameters" window. In the appearing window, enter a name for the channel (for example AC COM1 Route ARC 2) and select the driver "ABB RS232 Route AC" from the device list.



The following communication parameters can be set for the serial routing driver "ABB RS232 Route AC":



Parameter	Possible values	Meaning
Port	COMx (PC dependant)	Serial interface of the PC
Baudrate	19200	Always 19200 baud
Parity	No	Always no parity
Stop bits	1	Always one stop bit
Routing levels	0...2	Routing levels (0 = none)
Coupler (Level 1)	0, line 0...line 4	Coupler for level 1
Channel (Level 1)	0...19	Channel on coupler level 1
Address (Level 1)	0, 0, 0, 0, 0 (max. 5 bytes)	Address in target coupler level 1
Coupler (Level 2)	0, line 0...line 4	Coupler for level 2
Channel (Level 2)	0...19	Channel on coupler level 2
Address (Level 2)	0, 0, 0, 0, 0 (max. 5 bytes)	Address in target coupler level 2
Motorola byteorder	yes	Selection of Motorola or Intel byteorder

If you want to use the serial routing driver in order to directly access the connected CPU, set all routing parameters (parameter Routing levels and following parameters listed in the table above) to 0.



**Note:** Routing is available with version V1.3.x of the Control Builder and PLC firmware.

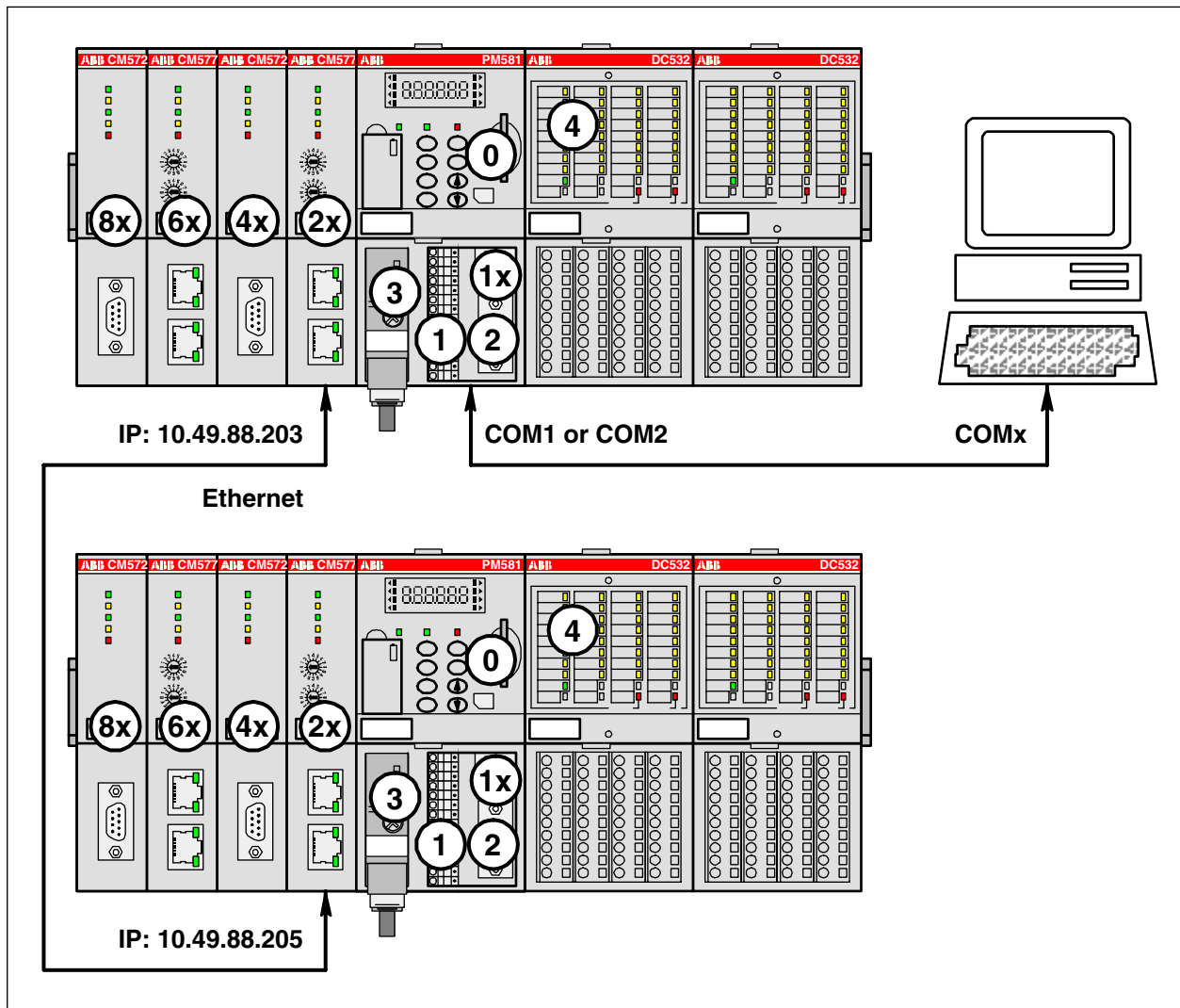
The following applies to the routing levels:

Routing levels = 0 No routing (parameters for Level 1 and Level 2 not set)

Routing levels = 1 Single-level routing (set parameters for Level 1)

Routing levels = 2 Double-level routing (set parameters for Level 1 and Level 2)

**Example:**



Configuration of PLC 2 (IP: 10.49.88.205) shall be performed via the external Ethernet coupler (IP: 10.49.88.203) inserted in slot 1 of PLC 1. The serial PC interface COM2 is connected to the serial interface COM1 of PLC 1:

Parameter	Value	Remark
Port	COM2	PC COM2
Baudrate	19200	
Parity	No	
Stop bits	1	
Routing levels	1	Single-level routing
Coupler (Level 1)	Line 1	Coupler in slot 1
Channel (Level 1)	0	Channel 1
Address (Level 1)	10, 49, 88, 205, 0	Subscriber address of the target PLC (Node 2)
Coupler (Level 2)	0	No level 2
Channel (Level 2)	0	
Address (Level 2)	0, 0, 0, 0, 0	
Motorola byteorder	yes	Motorola byteorder

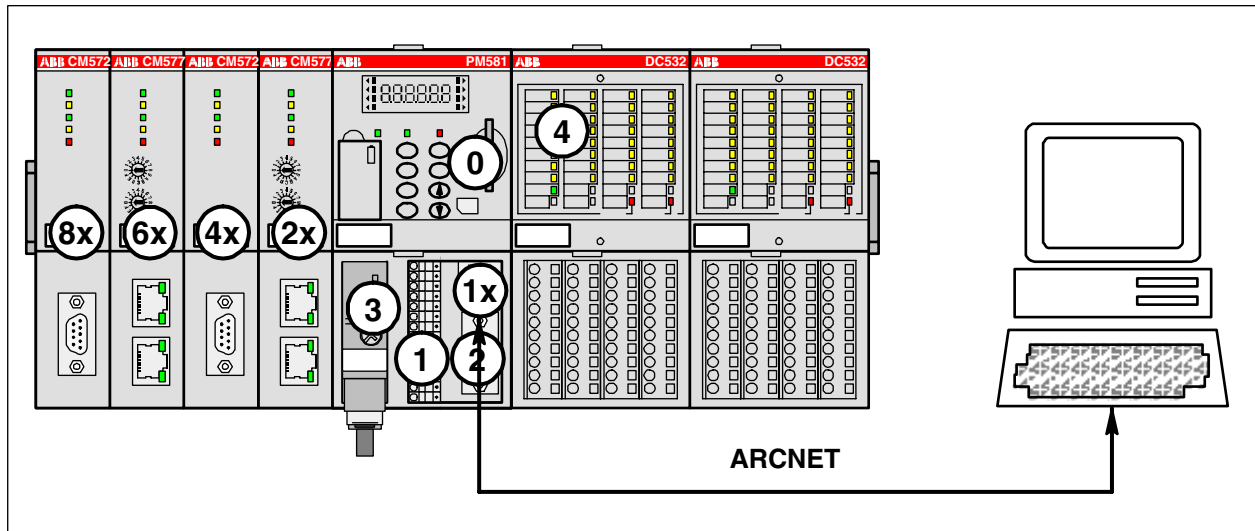
## 10.3 Programming via ARCNET



**Note:** The ARCNET coupler is available as of Control Builder version V1.2 and PLC firmware version V1.2.0.

Programming via ARCNET is only possible on a PC with installed ARCNET board. The installation of the board(s) and drivers is described in the Installation chapter (see also Installation / ARCNET drivers).

When programming via ARCNET, the PC is an ARCNET node.



The "sender node", i.e., the ARCNET subscriber address of the PC is set in the file:

Arcnet\_xx.ini with the parameter NodeID1 = 254.

The file Arcnet\_xx.ini is located in the folder where the PC operating system is installed (for example C:\WINNT for Win2000).

For a PC with installed ARCNET board, the file Arcnet\_xx.ini contains for example the following entries:

```
[ARCNET]
DriverAccessName1=FARC
;Default = Farc
NodeID1 = 254
; Default = 254

;DriverAccessName2=FARC1
; Default = Farc1
;NodeID2 = 253
; Default = 253

;DriverAccessName3=FARC11
; Default = Farc11
;NodeID3 = 252
; Default = 252

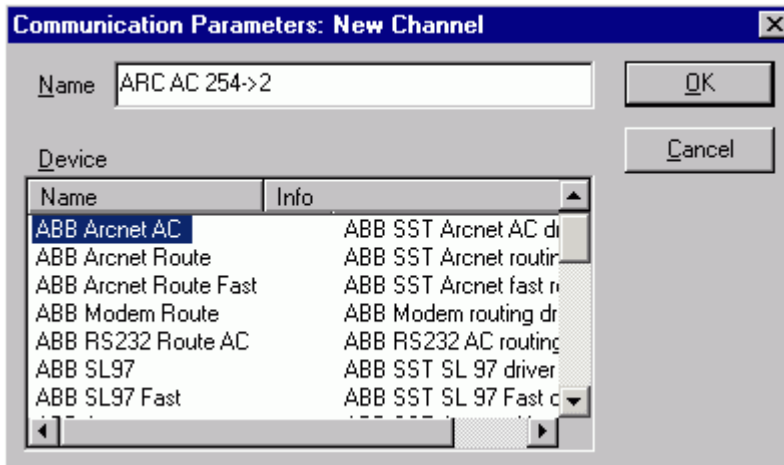
;DriverAccessName4=FARC111
; Default = Farc111
;NodeID4 = 251
; Default = 251
```

### 10.3.1 ARCNET driver "ABB Arcnet AC"

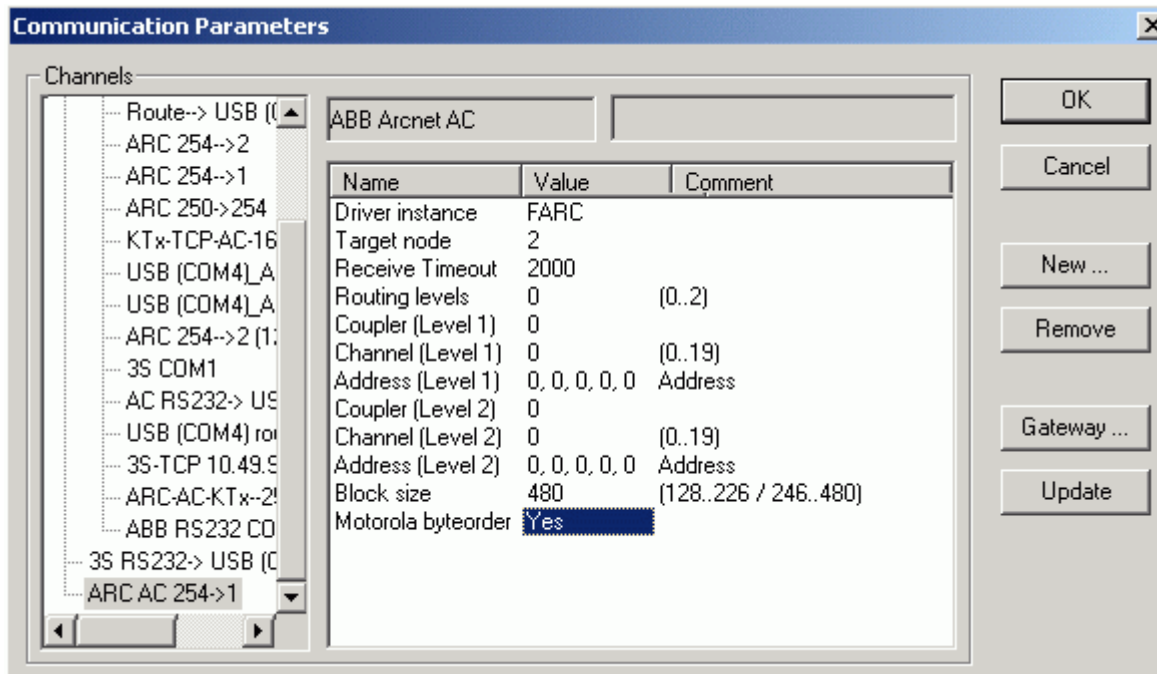
As of PLC runtime system version V1.2.0 and Control Builder version V1.2, the driver "ABB Arcnet AC" is available. This driver provides the following functions:

- Online operation of the PLC with the Control Builder
- Online operation of the PLC with the fieldbus configurator SYCON.net
- OPC connection with OPC server, as of version V1.3
- Parallel operation of Control Builder and SYCON.net
- Parallel operation of Control Builder and OPC server
- Parallel operation of Control Builder instances with several PLCs
- Online operation of AC31 series 90 controllers (07KT9x)

To define a new gateway channel for the ARCNET routing driver, select "Online/Communication Parameters" and press the button "New" in the "Communication Parameters" window. In the appearing window, enter a name for the channel (for example ARC AC 254 -> 2) and select the driver "ABB Arcnet AC" from the device list.



The following communication parameters can be set for the ARCNET routing driver "ABB Arcnet AC":



Parameter	Possible values	Meaning
Driver instance	FARC	DriverAccessName set in Arcnet_xx.ini
Target node	1...255	ARCNET subscriber address of the PLC
Receive Timeout	>= 2000	Timeout [ms] for response
Routing levels	0...2	Routing levels (0 = none)
Coupler (Level 1)	0, line 0...line 4	Coupler for level 1
Channel (Level 1)	0...19	Channel on coupler level 1
Address (Level 1)	0, 0, 0, 0, 0 (max. 5 bytes)	Address in target coupler level 1
Coupler (Level 2)	0, line 0...line 4	Coupler for level 2
Channel (Level 2)	0...19	Channel on coupler level 2
Address (Level 2)	0, 0, 0, 0, 0 (max. 5 bytes)	Address in target coupler level 2
Block size	128...226 / 246...480	User data size
Motorola byteorder	yes/no	Motorola or Intel byteorder

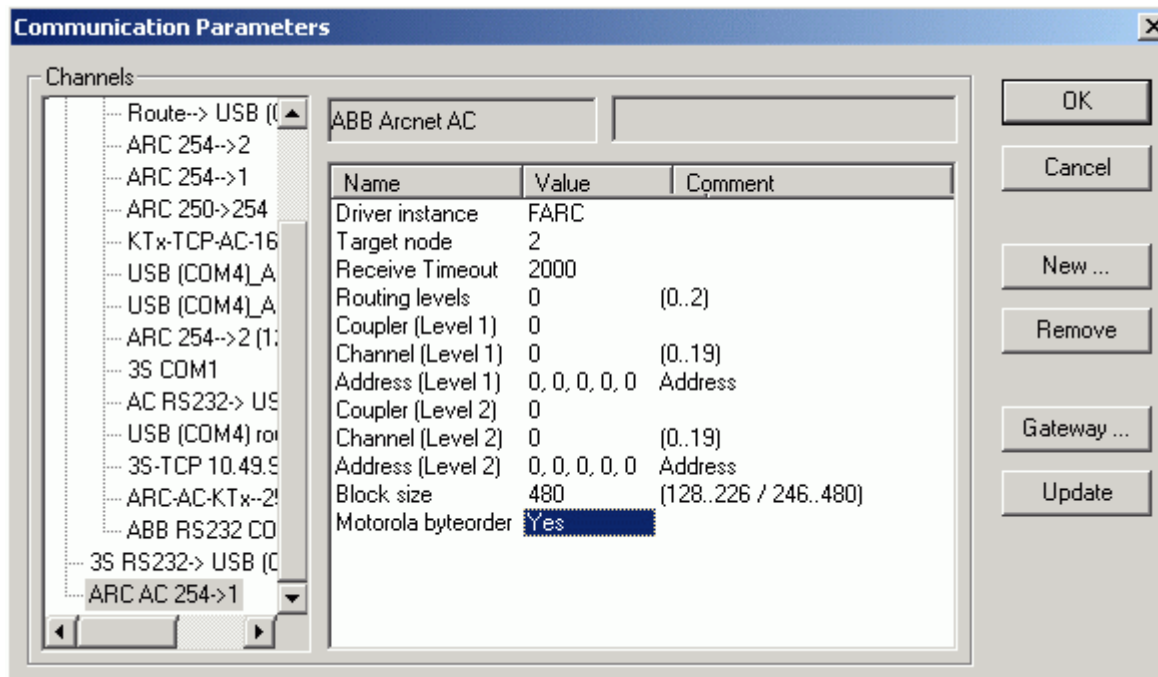
If you want to use the ARCNET routing driver to directly access the connected CPU, set all routing parameters (parameter Routing levels and following parameters listed in the table above) to 0.

The parameter "Block size" (128...480) sets the number of user data within one block. The default value is 480 (this is the maximum allowed block size). **Values in the range of 227 .. 245 are not allowed.**



**Note:** If you want to access a fieldbus coupler using SYCON.net via ARCNET, the parameter "Block size" always must be set to 128!

The parameter "**Motorola byteorder**" must be set to "**Yes**" for AC500 controllers.

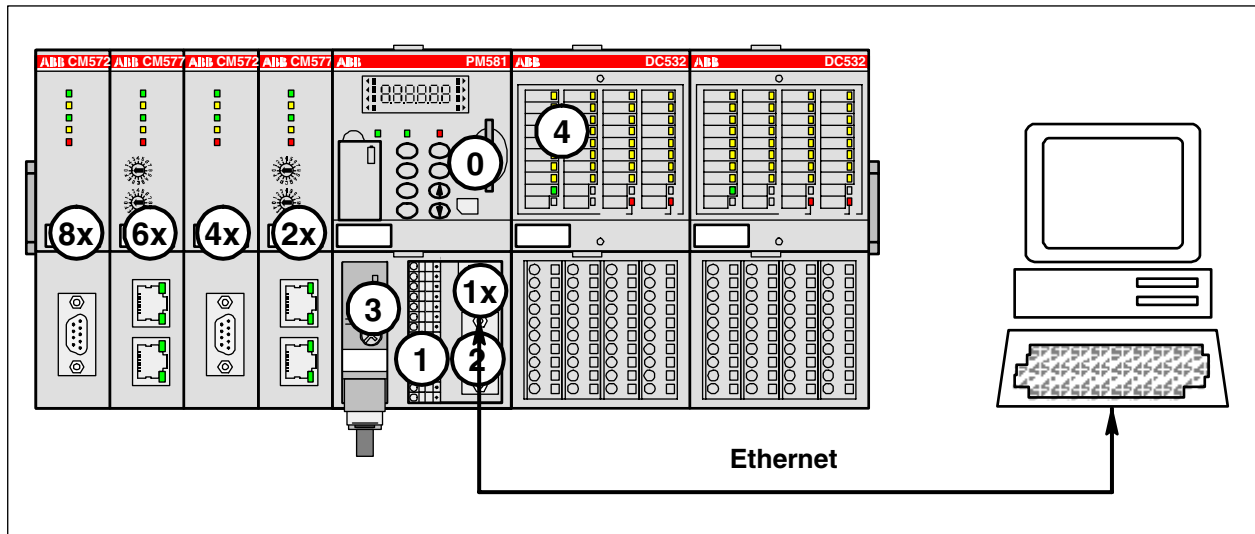




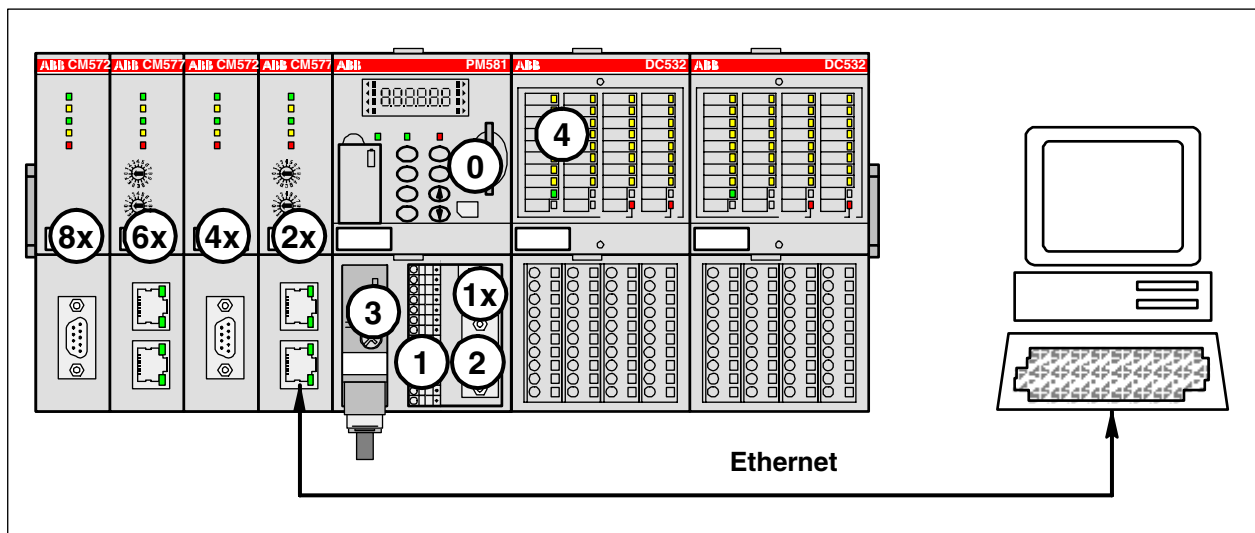
## 10.4 Programming via Ethernet (TCP/IP)

Programming via Ethernet is only possible on a PC with installed Ethernet board and installed network. Programming can be done via the internal and external Ethernet coupler.

Programming via internal Ethernet coupler:



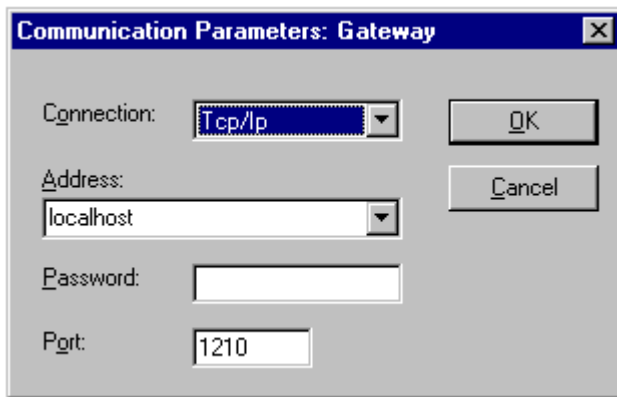
Programming via external Ethernet coupler (in this example coupler 1 in slot 1):



**Note:** Information how to set the IP addresses is available in the section "System Technology" => "System technology of internal couplers" => "The Ethernet coupler".

### Gateway configuration for Ethernet:

In the Control Builder, select "Online/Communication Parameters" and press the button "Gateway" in the "Communication Parameters" window. In the appearing window, select "Tcp/Ip" from the "Connection" list box (see CoDeSys / chapter The Individual Components / Online Communication Parameters).

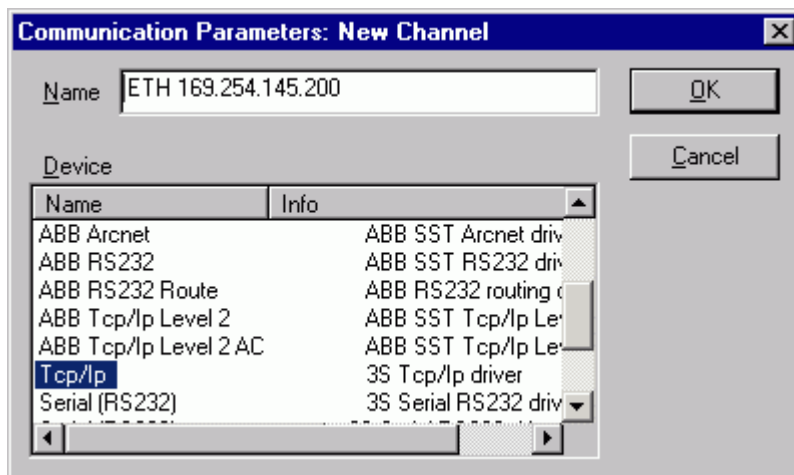


#### 10.4.1 Ethernet driver "Tcp/Ip"

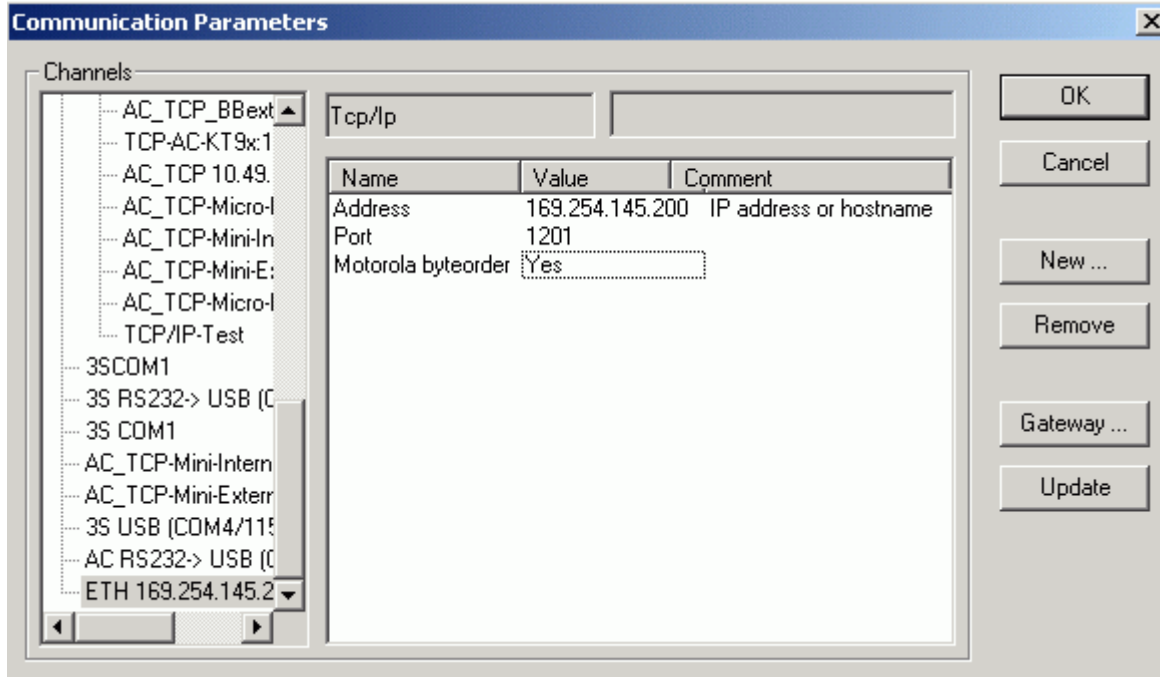
Programming AC500 controllers with internal and/or external Ethernet coupler via Ethernet can be done by using the driver "Tcp/Ip". This driver provides the following functions:

- Online operation of the PLC with the Control Builder
- Online operation of the PLC with the fieldbus configurator SYCON.net
- OPC connection with OPC server, as of version V1.3
- Parallel operation of Control Builder and SYCON.net
- Parallel operation of Control Builder and OPC server
- Parallel operation of Control Builder instances with several PLCs

To define a new gateway channel for the Ethernet interface, select "Online/Communication Parameters" and press the button "New" in the "Communication Parameters" window. In the appearing window, enter a name for the channel (for example ETH 169.254.145.200) and select the driver "Tcp/Ip" from the device list.



The following communication parameters can be set for the Ethernet driver "Tcp/Ip":



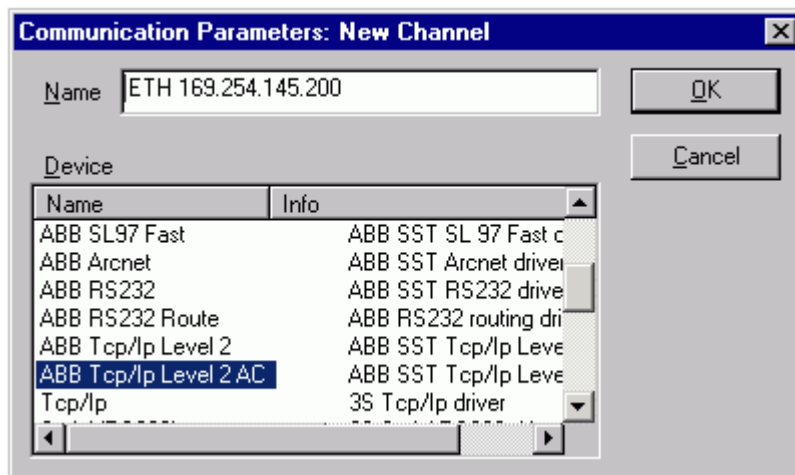
Parameter	Possible values	Meaning
Address	0.0.0.0	IP address or hostname of the PLC
Port	1201	Port 1201
Motorola byteorder	Yes (Yes/No)	Motorola or Intel byteorder (=Yes for AC500)

#### 10.4.2 Ethernet driver "ABB Tcp/Ip Level 2 AC"

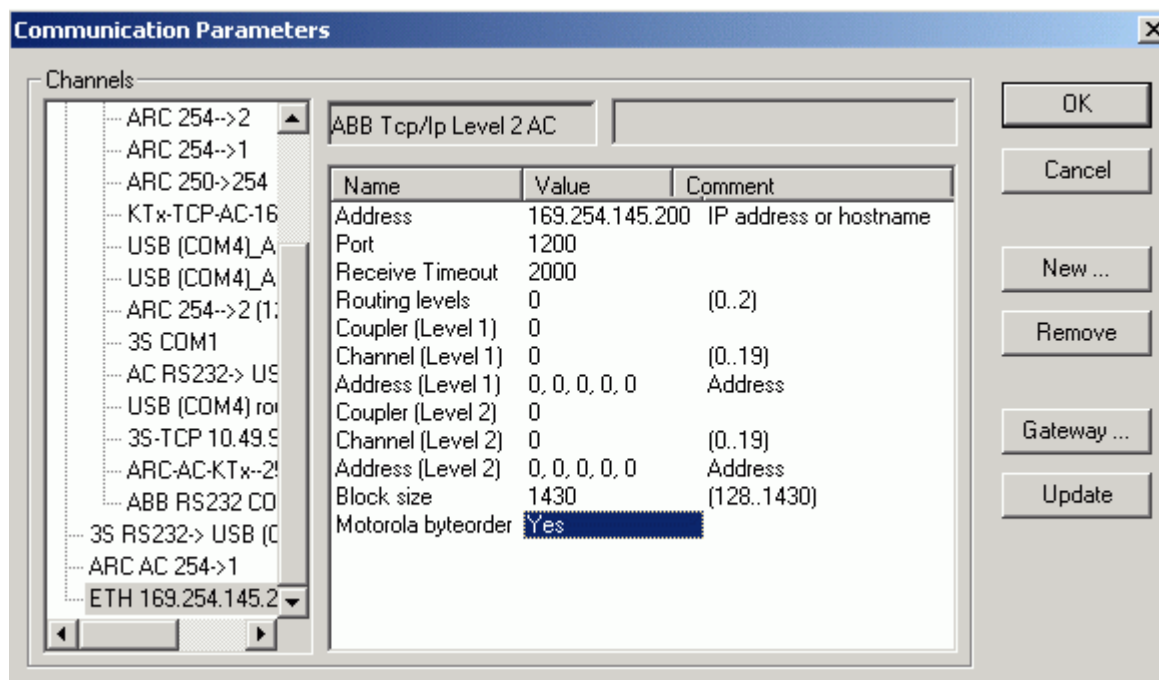
As of version V1.2, the driver "ABB Tcp/Ip Level 2 AC" is available for programming AC500 controllers with internal and/or external Ethernet coupler via Ethernet. This driver provides the following functions:

- Online operation of the PLC with the Control Builder
- Online operation of the PLC with the fieldbus configurator SYCON.net
- OPC connection with OPC server, as of version V1.3
- Parallel operation of Control Builder and SYCON.net
- Parallel operation of Control Builder and OPC server
- Parallel operation of Control Builder instances with several PLCs
- Online operation of PLCs connected via ARCNET. One PLC equipped with Ethernet coupler and one PLC with ARCNET coupler (Routing Ethernet -> ARCNET), as of version V2.x
- Online operation of AC31 series 90 controllers (07KT9x)

To define a new gateway channel for the Ethernet interface, select "Online/Communication Parameters" and press the button "New" in the "Communication Parameters" window. In the appearing window, enter a name for the channel (for example ETH 169.254.145.200) and select the driver "ABB Tcp/Ip Level 2 AC" from the device list.



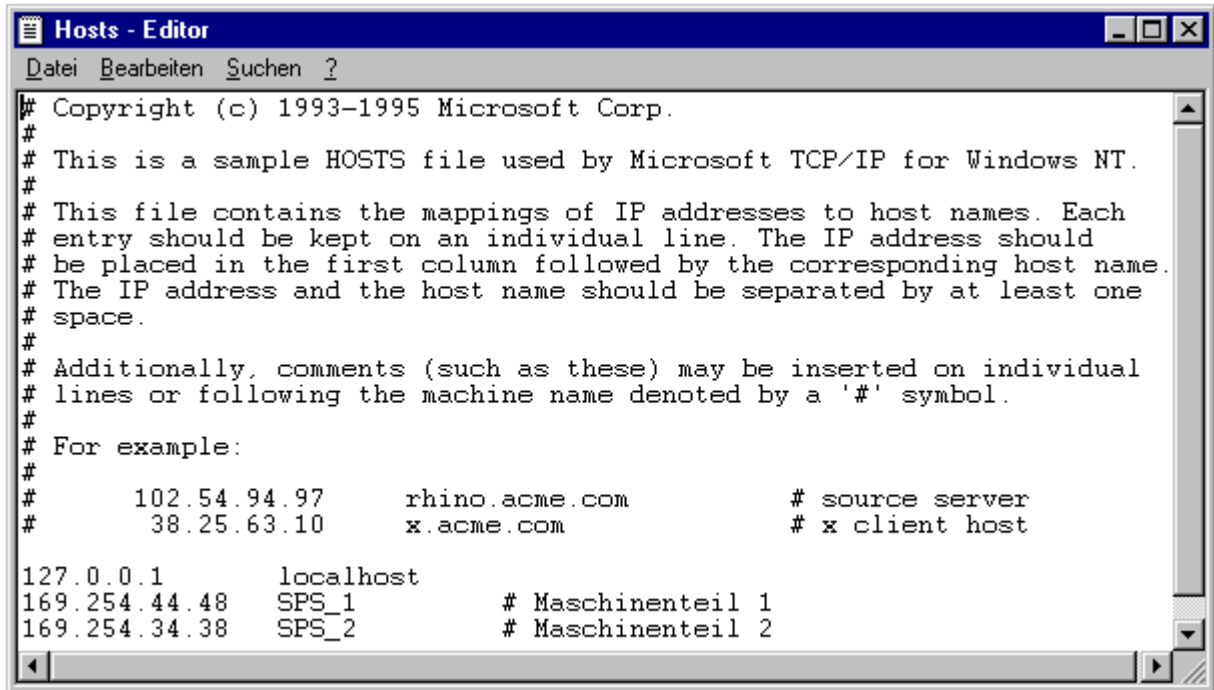
The following communication parameters can be set for the Ethernet driver "ABB Tcp/Ip Level 2 AC":



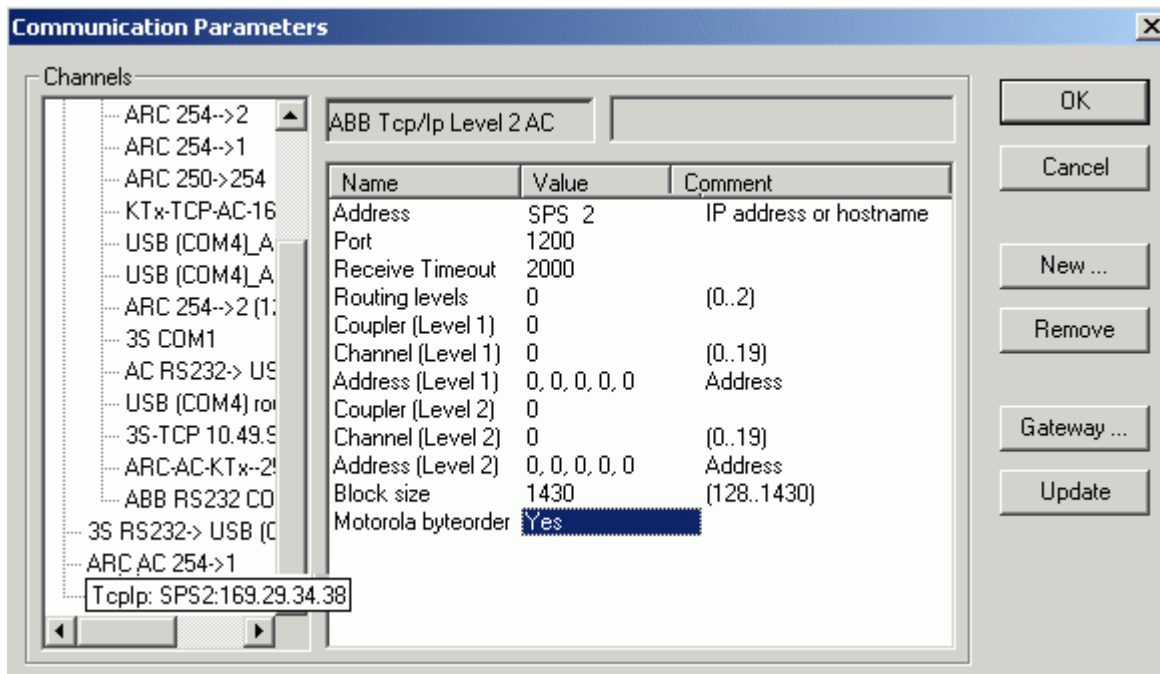
Parameter	Possible values	Meaning
Address	0.0.0.0	IP address or hostname of the PLC
Port	1200	Port 1200
Timeout (ms)	>= 2000	Timeout [ms] for response
Routing levels	0..2	Routing levels (0 = none)
Coupler (Level 1)	0, line 0...line 4	Coupler for level 1
Channel (Level 1)	0...19	Channel on coupler level 1
Address (Level 1)	0, 0, 0, 0, 0 (max. 5 bytes)	Address in target coupler level 1
Coupler (Level 2)	0, line 0...line 4	Coupler for level 2
Channel (Level 2)	0...19	Channel on coupler level 2
Address (Level 2)	0, 0, 0, 0, 0 (max. 5 bytes)	Address in target coupler level 2
Block size	1430 (128...1430)	Bytes per telegram (unallowed 227..245)
Motorola byteorder	Yes (Yes/No)	Motorola or Intel byteorder (=Yes for AC500)

If you want to use the Ethernet driver to directly access the PLC, set all routing parameters (parameter Routing levels and following parameters listed in the table above) to 0.

The "Address" parameter sets the IP address or hostname of the PLC. To be able to use hostnames, the names have to be added to the file "Hosts". Under Win2000, this file is located in the directory "WINNT\System32\drivers\etc".



If you have changed the "Hosts" file accordingly, you can enter the symbolic name for the "Address" parameter instead of the IP address. In the following figure, the IP address "169.254.34.38" is replaced by the hostname "SPS\_2".

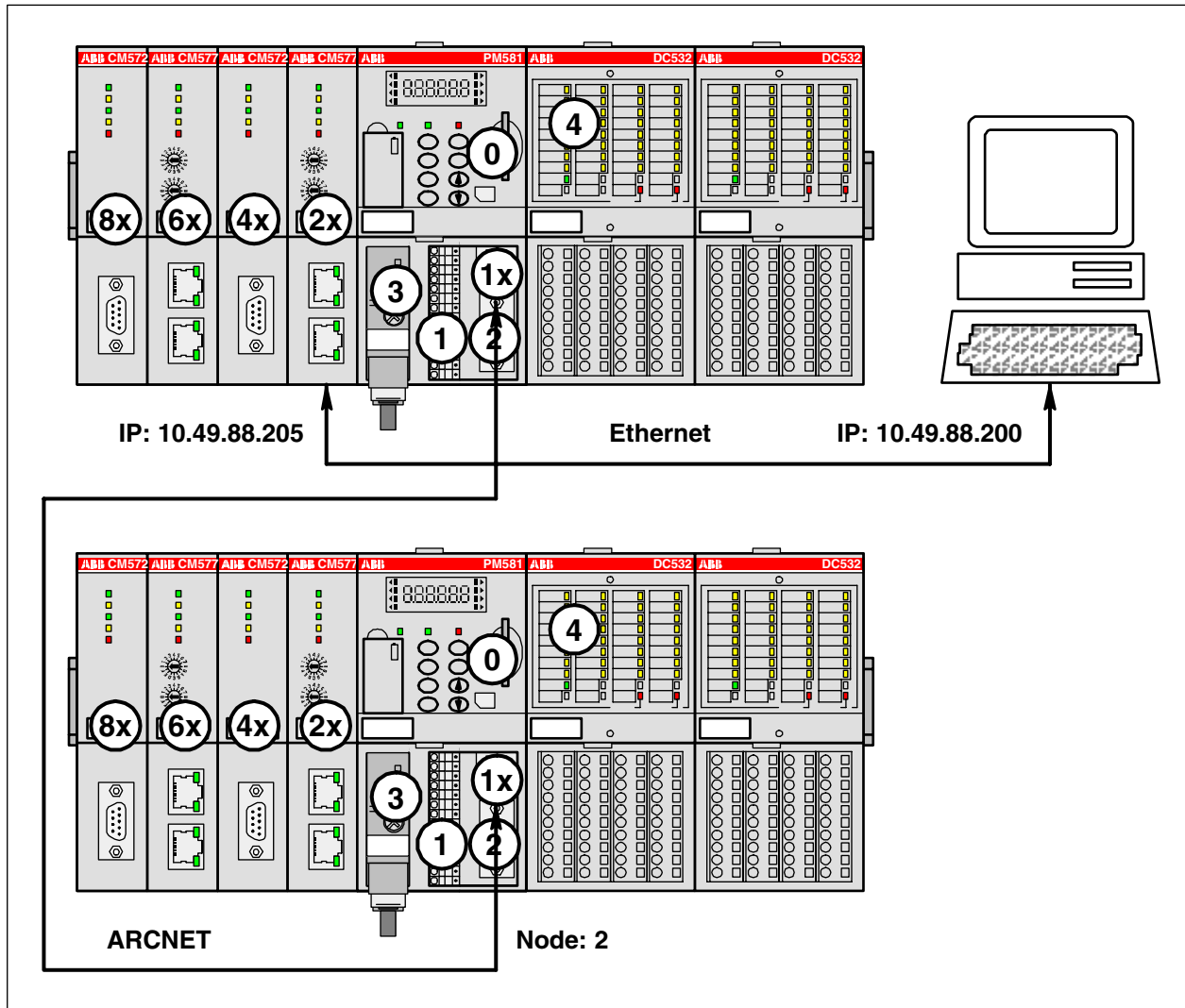


### 10.4.3 Ethernet ARCNET routing



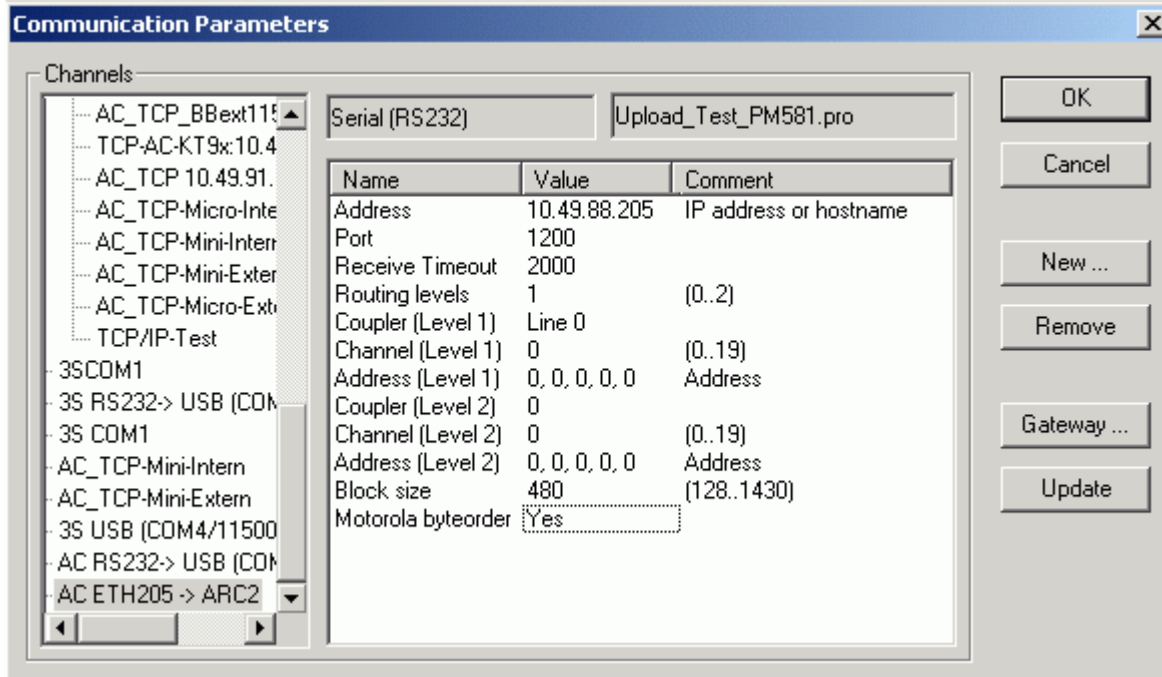
**Note:** Routing is available as of PLC firmware version V1.3.

For controllers with Ethernet and ARCNET coupler, the PLCs connected via ARCNET can be programmed using the PLC Ethernet interface.



For each PLC connected via ARCNET, one gateway channel has to be defined. To do this, select "Online/Communication Parameters" and press the button "New" in the "Communication Parameters" window. In the appearing window, enter a name for the channel (for example Tcplp: PLC1:169.29.44.48 -> ARC\_2) and select the driver "ABB Tcp/Ip Level 2 AC" from the device list.

For example, set the communication parameters as follows for the configuration shown above:



Parameter	Possible values	Meaning
Address	10.49.88.205	IP address of PLC 1
Port	1200	Port 1200
Timeout (ms)	2000	Timeout [ms] for response
Routing levels	1	Single-level routing
Coupler (Level 1)	Line 0	Coupler for level 1 (internal: ARCNET)
Channel (Level 1)	0	Channel on coupler level 1
Address (Level 1)	2, 0, 0, 0, 0	ARCNET node of the target PLC (Node 2)
Coupler (Level 2)	0	No level 2
Channel (Level 2)	0	
Address (Level 2)	0, 0, 0, 0, 0	
Block size	480	Bytes per block: 128...1430
Motorola byteorder	Yes	

For the parameter "Coupler (Level 1)", enter the slot where the ARCNET coupler "Line 0" is inserted (the ARCNET coupler is always the internal coupler).

The ARCNET coupler has only one communication channel. Thus, the "Channel" value must always be 0.

For the ARCNET coupler, 1 byte is required for the subscriber address (node). The address (Node=2) of the target PLC is entered to the first byte of the address byte.

The default value for the block size is 1430. If routing on ARCNET is required (and "large ARCNET packages" are enabled for the target PLC), the block size can be increased to 480 bytes. **Values in the range of 227 .. 245 are not allowed.**

# 11 Communication with Modbus RTU

## 11.1 Protocol description

The Modbus protocol is used worldwide. The **MODICON Modbus® RTU** protocol is implemented in the AC500 CPU.

Numerous automation devices, such as PLC installations, displays, variable-frequency inverters or monitoring systems have a Modbus® RTU interface by default or as an option and can therefore communicate with AC500 basic units without any problems.

Modbus® is a master-slave protocol. The master sends a request to the slave and receives its response.

### Modbus master

In operating mode MODBUS master, the telegram traffic with the slave(s) is handled via the function block MODMAST. The function block MODMAST sends Modbus request telegrams to the slave via the set interface and receives Modbus response telegrams from the slave via this interface.

For Modbus on TCP/IP, the function block ETH\_MODMAST is used and for serial interfaces the function block COM\_MODMAST (link to function blocks: ETH\_MODMAST in library Ethernet\_AC500\_Vxx.lib and COM\_MODMAST in library Modbus\_AC500\_Vxx.lib).

The Modbus® blocks transferred by the master contain the following information:

- Modbus® address of the interrogated slave (1 byte)
- Function code that defines the request of the master (1 byte)
- Data to be exchanged (n bytes)
- CRC16 control code (2 bytes)

### Modbus slave

In operating mode MODBUS slave, no function block is required for Modbus communication. Sending and receiving Modbus telegrams is performed automatically.

**The AC500 CPUs process only the following Modbus® operation codes:**

Function code		Description
DEC	HEX	
01 or 02	01 or 02	read n bits
03 or 04	03 or 04	read n words
05	05	write one bit
06	06	write one word
07	07	fast reading the status byte of the CPU
15	0F	write n bits
16	10	write n words



The following restrictions apply to the length of the data to be sent:

Function code		Max. length	
DEC	HEX	Serial	Modbus on TCP/IP
01 or 02	01 or 02	2000 bits	255 bits (up to coupler FW V01.033) xxx bits (as of coupler FW V01.041)
03 or 04	03 or 04	125 words / 62 double words	100 words / 50 double words
05	05	1 bit	1 bit
06	06	1 word	1 word
07	07	8 bits	8 bits
15	0F	1968 bits	255 bits (up to coupler FW V01.033) xxx bits (as of coupler FW V01.041)
16	10	123 words / 61 double words	100 words / 50 double words

## 11.2 Modbus RTU with the serial interfaces COM1 and COM2

### 11.2.1 Modbus operating modes of the serial interfaces

Both serial interfaces of the AC500 CPUs can be operated simultaneously as Modbus interfaces and can operate as Modbus master as well as Modbus slave.

The Modbus operating mode and the interface parameters are set in the PLC Configuration (see also Controller configuration / Modbus).

#### Description of the Modbus® protocol:

Supported standard	EIA RS-232 / RS-485
Number of connection points	1 master max. 1 slave with RS 232 interface max. 31 slaves with RS 485
Protocol	Modbus® (Master/Slave)
Data transmission control	CRC16
Data transmission speed	up to 187500 baud
Encoding	1 start bit 8 data bits 1 parity bit, even or odd (optional) 1 or 2 stop bits
Max. cable length	for RS 485: 1200 m at 19200 baud

## 11.3 Modbus on TCP/IP via Ethernet

Modbus on TCP/IP is described in the chapter System Technology Coupler / The Ethernet coupler (see also System Technology Ethernet Coupler / Modbus on TCP/IP).

## 11.4 Modbus addresses

### 11.4.1 Modbus address table

A range of 128 kbytes is allowed for the access via Modbus, i.e., the segments line 0 and line 1 of the addressable flag area (%M area) can be accessed. Thus, the complete address range 0000<sub>hex</sub> up to FFFF<sub>hex</sub> is available for Modbus.

The availability of the segments depends on the CPU. The size of the %M area can be found in the technical data of the CPUs (see Technical data of the CPUs) and in the target system settings (see Target Support Package).

Inputs and outputs cannot be directly accessed using Modbus.

The address assignment for word and double word accesses is done according to the following table:

Modbus address		Byte BYTE	Bit (byte-oriented) BOOL	Word WORD	Double word DWORD
HEX	DEC				
<b>Line 0</b>					
0000	0	%MB0.0	%MX0.0.0...%MX0.0.7	%MW0.0	%MD0.0
		%MB0.1	%MX0.1.0...%MX0.1.7		
0001	1	%MB0.2	%MX0.2.0...%MX0.2.7	%MW0.1	
		%MB0.3	%MX0.3.0...%MX0.3.7		
0002	2	%MB0.4	%MX0.4.0...%MX0.4.7	%MW0.2	%MD0.1
		%MB0.5	%MX0.5.0...%MX0.5.7		
0003	3	%MB0.6	%MX0.6.0...%MX0.6.7	%MW0.3	
		%MB0.7	%MX0.7.0...%MX0.7.7		
...					
7FFE	32766	%MB0.65532	%MX0.65532.0 ...%MX0.65532.7	%MW0.32766	%MD0.16383
		%MB0.65533	%MX0.65533.0 ...%MX0.65533.7		
7FFF	32767	%MB0.65534	%MX0.65534.0 ...%MX0.65534.7	%MW0.32767	
		%MB0.65535	%MX0.65535.0 ...%MX0.65535.7		
<b>Line 1</b>					
8000	32768	%MB1.0	%MX1.0.0...%MX1.0.7	%MW1.0	%MD1.0
		%MB1.1	%MX1.1.0...%MX1.1.7		
8001	32769	%MB1.2	%MX1.2.0...%MX1.2.7	%MW1.1	
		%MB1.3	%MX1.3.0...%MX1.3.7		
8002	32770	%MB1.4	%MX1.4.0...%MX1.4.7	%MW1.2	%MD1.1
		%MB1.5	%MX1.5.0...%MX1.5.7		
8003	32771	%MB1.6	%MX1.6.0...%MX1.6.7	%MW1.3	
		%MB1.7	%MX1.7.0...%MX1.7.7		
...					
FFFE	65534	%MB1.65532	%MX1.65532.0 ...%MX1.65532.7	%MW1.32766	%MD1.16383
		%MB1.65533	%MX1.65533.0 ...%MX1.65533.7		
FFFF	65535	%MB1.65534	%MX1.65534.0 ...%MX1.65534.7	%MW1.32767	
		%MB1.65535	%MX1.65535.0 ...%MX1.65535.7		

The **address assignment for bit accesses** is done according to the following table:

Modbus address		Byte BYTE	Bit (byte-oriented) BOOL	Word WORD	Double word DWORD
HEX	DEC				
<b>Line 0</b>					
0000	0	%MB0.0	%MX0.0.0	%MW0.0	%MD0.0
0001	1		%MX0.0.1		
0002	2		%MX0.0.2		
0003	3		%MX0.0.3		
0004	4		%MX0.0.4		
0005	5		%MX0.0.5		
0006	6		%MX0.0.6		
0007	7		%MX0.0.7		
0008	8	%MB0.1	%MX0.1.0		
0009	9		%MX0.1.1		
000A	10		%MX0.1.2		
000B	11		%MX0.1.3		
000C	12		%MX0.1.4		
000D	13		%MX0.1.5		
000E	14		%MX0.1.6		
000F	15		%MX0.1.7		
0010	16	%MB0.2	%MX0.2.0	%MW0.1	
0011	17		%MX0.2.1		
0012	18		%MX0.2.2		
0013	19		%MX0.2.3		
0014	20		%MX0.2.4		
0015	21		%MX0.2.5		
0016	22		%MX0.2.6		
0017	23		%MX0.2.7		
0018	24	%MB0.3	%MX0.3.0		
0019	25		%MX0.3.1		
001A	26		%MX0.3.2		
001B	27		%MX0.3.3		
001C	28		%MX0.3.4		
001D	29		%MX0.3.5		
001E	30		%MX0.3.6		
001F	31		%MX0.3.7		
0020	32	%MB0.4	%MX0.4.0	%MW0.2	%MD0.1
0021	33		%MX0.4.1		
0022	34		%MX0.4.2		
...	...	...	...	...	...
0FFF	4095	%MB0.511	%MX0.511.7	%MW0.255	%MD0.127
1000	4096	%MB0.512	%MX0.512.0	%MW0.256	%MD0.128
...	...	...	...	...	...
7FFF	32767	%MB0.4095	%MX0.4095.7	%MW0.2047	%MD0.1023
8000	32768	%MB0.4096	%MX0.4096.0	%MW0.2048	%MD0.1024
...	...	...	...	...	...
FFFF	65535	%MB0.8191	%MX0.8191.7	%MW0.4095	%MD0.2047

## Calculation of the bit variable from the hexadecimal address:

Formula:			
	<b>Bit variable (BOOL) := %MX0.BYTE.BIT</b>		
where:	DEC	Decimal address	
	BYTE	DEC / 8	
	BIT	DEC mod 8	(Modulo division)

### Examples:

Address hexadecimal = 16#2002  
DEC := HEX2DEC(16#2002) := 8194  
BYTE := 8194 / 8 := 1024  
BIT := 8194 mod 8 := 2  
Bit variable: %MX0.1024.2

Address hexadecimal = 16#3016  
DEC := HEX2DEC(16#3016) := 12310  
BYTE := 12310 / 8 := 1538,75 -> 1538  
BIT := 12310 mod 8 := 6  
Bit variable: %MX0.1538.6

Address hexadecimal = 16#55AA  
DEC := HEX2DEC(16#55AA) := 21930  
BYTE := 21930 / 8 := 2741,25 -> 2741  
BIT := 21930 mod 8 := 2  
Bit variable: %MX0.2741.2

## Calculation of the hexadecimal address from the bit variable:

### Formula:

**Address hexadecimal := DEC2HEX( BYTE \* 8 + BIT )**

### Examples:

Bit variable := %MX0.515.4  
Address hex := DEC2HEX( 515 \* 8 + 4 ) := DEC2HEX( 4124 ) := 16#101C

Bit variable := %MX0.3.3  
Address hex := DEC2HEX( 3 \* 8 + 3 ) := DEC2HEX( 27 ) := 16#001B

Bit variable := %MX0.6666.2  
Address hex := DEC2HEX( 6666 \* 8 + 2 ) := DEC2HEX( 53330 ) := 16#D052

## 11.4.2 Peculiarities for accessing Modbus addresses

### *Peculiarities for bit access:*

- As you can see in the address table, a WORD in the %M area is assigned to each Modbus address 0000hex .. FFFFhex
- Bit addresses 0000hex .. FFFFhex are contained in the word range %MW0.0 .. %MW0.4095

### *Write/read-protected areas for the Modbus slave:*

As described in the PLC configuration, one write-protected and one read-protected area can be defined for each segment line 0 and line 1. (see also Controller configuration / The setting 'COMx - Modbus'). If you try to write to a write-protected area or to read from a read-protected area, an error message is generated.

### **Segment exceedance for line 0 and line 1:**

A write- or read-protected area that lies in both segments, line 0 and line 1, cannot be accessed with a write/read operation. In case of a segment exceedance, an error message is generated.

#### **Example:**

Read 10 words beginning at address := 7FFE<sub>hex</sub>

This includes the addresses: 7FFE<sub>hex</sub>...8007<sub>hex</sub> with the operands %MW0.32766...%MW1.7. Because line 0 is exceeded in this case, an error message is generated.

Due to this, two telegrams have to be generated here:

1. Read 2 words beginning at address := 7FFE<sub>hex</sub> and
2. Read 8 words beginning at address := 8000<sub>hex</sub>.

### **Valid data areas for reading/writing the Modbus master:**

If the AC500 control system operates as Modbus master, the data exchange with the Modbus slaves is controlled using a MODMAST block (ETH\_MOD\_MAST for Modbus on TCP/IP and COM\_MOD\_MAST for serial interfaces). (Link to blocks: ETH\_MODMAST in library Ethernet\_AC500\_Vxx.lib and COM\_MODMAST in library Modbus\_AC500\_Vxx.lib).

The address of the area from which data are to be read or to which data are to be written is specified at block input "Data" via the ADR operator.

For the AC500, the following areas can be accessed using the ADR operator:

- Inputs area (%I area)
- Outputs area (%Q area)
- Area of non-buffered variables (VAR .. END\_VAR or VAR\_GLOBAL END\_VAR)
- Addressable flag area (also protected areas for %M area)
- Area of buffered variables (VAR RETAIN .. END\_VAR or VAR\_GLOBAL RETAIN .. END\_VAR)

### **11.4.3 Comparison between AC500 and AC31/S90 Modbus addresses**

The following table shows the addresses for AC500 controllers and its predecessor AC31 / S90

Address HEX	FCT HEX	AC1131 operand	FCT HEX	AC500 operand
<b>Bit accesses</b>				
0000...0FFF	01, 02	%IX0.0...%IX255.15	01, 02, 05, 07, 0F	%MX0.0.0...%MX0.511.7
0000		%IX0.0		%MX0.0.0
0001		%IX0.1		%MX0.0.1
0002		%IX0.2		%MX0.0.2
...		...		...
0010		%IX1.0		%MX0.2.0
...		...		...
0FFF		%IX255.15		%MX0.511.7
1000...1FFF	01, 02, 05, 0F	%QX0.0...%QX255.15	01, 02, 05, 07, 0F	%MX0.512.0...%MX0.1023.7
1000		%QX0.0		%MX0.512.0
1001		%QX0.1		%MX0.512.1
1002		%QX0.2		%MX0.512.2
...		...		...
1010		%QX1.0		%MX0.514.0
...		...		...
1FFF		%QX255.15		%MX0.1023.7
2000...2FFF	01, 02, 05, 07, 0F	%MX0.0...%MX255.15	01, 02, 05, 07, 0F	%MX0.1024.0...%MX0.1535.7
2000		%MX0.0		%MX0.1024.0
2001		%MX0.1		%MX0.1024.1
2002		%MX0.2		%MX0.1024.2
...		...		...
2010		%MX1.0		%MX0.1026.0
...		...		...
2FFF		%MX255.15		%MX0.1535.7
3000...3FFF	01, 02, 05, 07, 0F	%MX5000.0...%MX5255.15	01, 02, 05, 07, 0F	%MX0.1536.0...%MX0.2047.7
3000		%MX5000.0		%MX0.1536.0
3001		%MX5000.1		%MX0.1536.1
3002		%MX5000.2		%MX0.1536.2
...		...		...
3010		%MX5001.0		%MX0.1538.0
...		...		...
3FFF		%MX5255.15		%MX0.2047.7
4000...FFFF		No access	01, 02, 05, 07, 0F	%MX0.2048.0...%MX0.8191.7
<b>Word accesses</b>				
0000...0CFF	03, 04	%IW1000.0...%IW1207.15	03, 04, 06, 10	%MW0.0...%MW0.3327
0D00...0FFF	03, 04	No access	03, 04, 06, 10	%MW0.3328...%MW0.4095
1000...1CFF	03, 04, 06, 10	%QW1000.0...%QW1207.15	03, 04, 06, 10	%MW0.4096...%MW0.7423
1D00...1FFF		No access	03, 04, 06, 10	%MW0.7424...%MW0.8191
2000...2FFF	03, 04, 06, 10	%MW1000.0...%MW1255.15	03, 04, 06, 10	%MW0.8192...%MW0.12287
3000...359F	03, 04, 06, 10	%MW3000.0...%MW3089.15	03, 04, 06, 10	%MW0.12288...%MW0.13727
35A0...3FFF		No access	03, 04, 06, 10	%MW0.13728...%MW0.16383
4000...47FF		%MW2000.0.0...%MW2063.15.1 No access	03, 04, 06, 10	%MW0.16384...%MW18431
4800...4FFF		No access	03, 04, 06, 10	%MW0.18432...%MW0.20479
5000...517F		%MW4000.0.0...%MW4023.15.1 No access	03, 04, 06, 10	%MW0.20480...%MW0.21247
5180...FFFF		No access	03, 04, 06, 10	%MW0.21248...%MW1.32767
<b>Double word accesses</b>				
0000...3FFF		No access	03, 04, 06, 10	%MD0.0...%MD0.8191
4000...47FF	03, 04, 06, 10	%MD2000.0...%MD2063.15	03, 04, 06, 10	%MD0.8192...%MD0.9215
4800...4FFF		No access	03, 04, 06, 10	%MD0.9216...%MD0.10239
5000...537F	03, 04, 06, 10	%MD4000.0...%MD4023.15	03, 04, 06, 10	%MD0.1240...%MD0.10815
5480...FFFF		No access	03, 04, 06, 10	%MD0.10816...%MD1.16383

## 11.5 Modbus telegrams

The send and receive telegrams shown in this section are not visible in the PLC. However, the complete telegrams can be made visible using a serial data analyzer connected to the connection line between master and slave, if required.

The amount of user data depends on the properties of the master and slave.

For the following examples, it is assumed that an AC500 Modbus module is used as slave. There may be different properties if modules of other manufacturers are used.

### FCT 1 or 2: Read n bits

#### Master request

Slave address	Function code	Slave operand address		Number of bits		CRC	
		High	Low	High	Low	High	Low

#### Slave response

Slave address	Function code	Number of bytes	...Data...	CRC	
				High	Low

<b>Example:</b>	Modbus interface of the master:	COM1
	Master reads from:	Slave 1
	Data:	%MX0.1026.4 = FALSE; %MX0.1026.5 = TRUE %MX0.1026.6 = FALSE
	Source address at slave:	%MX0.1026.4 : 2014HEX = 8212DEC
	Target address at master:	abReadBit : ARRAY[0..2] OF BOOL;
	The values of the flags %MX0.1026.4..%MX0.1026.6 on the slave are written to the ARRAY abReadBool on the master.	

#### Modbus request of the master

Slave address	Function code	Slave operand address		Number of bits		CRC	
		High	Low	High	Low	High	Low
01HEX	01HEX	20HEX	14HEX	00HEX	03HEX	37HEX	CFHEX

#### Modbus response of the slave

Slave address	Function code	Number of bytes	Data	CRC	
				High	Low
01HEX	01HEX	01HEX	02HEX	D0HEX	49HEX

#### Parameterization of the COM\_MOD\_MAST block inputs

NB = Number of bits

EN	COM	SLAVE	FCT	TIMEOUT	ADDR	NB	DATA
FALSE -> TRUE	1	1	1	Application-specific	8212	3	ADR (abReadBool[0])

### FCT 3 or 4: Read n words

#### Master request

Slave address	Function code	Slave operand address		Number of words		CRC	
		High	Low	High	Low	High	Low

#### Slave response

Slave address	Function code	Number of bytes	...Data...	CRC	
				High	Low

<b>Example:</b>	Modbus interface of the master:	COM1
	Master reads from:	Slave 1
	Data:	%MW0.8196 = 4; %MW0.8197 = 5; %MW0.8198 = 6
	Source address at slave:	%MW0.8196 : 2004HEX = 8196DEC
	Target address at master:	awReadWord : ARRAY[0..2] OF WORD;
	The values of the flag words %MW0.8196..%MW0.8198 on the slave are written to the ARRAY awReadWord on the master.	

#### Modbus request of the master

Slave address	Function code	Slave operand address		Number of words		CRC	
		High	Low	High	Low	High	Low
01HEX	03HEX	20HEX	04HEX	00HEX	03HEX	4FHEX	CAHEX

#### Modbus response of the slave

Slave address	Function code	Number of bytes	Data	Data	Data	CRC	
			High / Low	High / Low	High / Low	High	Low
01HEX	03HEX	06HEX	00HEX /04HEX	00HEX /05HEX	00HEX /06HEX	40HEX	B6HEX

#### Parameterization of the COM\_MOD\_MAST block inputs

NB = Number of words

EN	COM	SLAVE	FCT	TIMEOUT	ADDR	NB	DATA
FALSE -> TRUE	1	1	3	Application-specific	8196	3	ADR (awReadWord[0])

### FCT 3 or 4: Read n double words

The function code "read double word" is not defined in the Modbus RTU standard. This is why the double word is composed of a low word and a high word (depending on the manufacturer).

#### Master request

Slave address	Function code	Slave operand address		Number of words		CRC	
		High	Low	High	Low	High	Low

#### Slave response

Slave address	Function code	Number of bytes	...Data...	CRC	
				High	Low



<b>Example:</b>	Modbus interface of the master:	COM1
	Master reads from:	Slave 1
	Data:	%MD0.8193 = 32DEC = 00000020HEX; %MD0.8194 = 80000DEC = 00013880HEX
	Source address at slave:	%MD0.8193: 4002HEX = 16386DEC
	Target address at master:	adwReadDWord : ARRAY[0..1] OF DWORD
	The values of the flag double words %MD0.8193..%MD0.8194 on the slave are written to the ARRAY adwReadDWord on the master.	

### Modbus request of the master

Slave address	Function code	Slave operand address		Number of words		CRC	
		High	Low	High	Low	High	Low
01HEX	03HEX	40HEX	02HEX	00HEX	04HEX	F0HEX	09HEX

### Modbus response of the slave

Slave address	Function code	Number of bytes	Data	Data	Data	Data	CRC	
			High / Low	High / Low	High / Low	High / Low	High	Low
01HEX	03HEX	08HEX	00HEX / 00HEX	00HEX / 20HEX	00HEX / 01HEX	38HEX / 80HEX	57HEX	B0HEX

### Parameterization of the COM\_MOD\_MAST block inputs

NB = Number of words

EN	COM	SLAVE	FCT	TIMEOUT	ADDR	NB	DATA
FALSE -> TRUE	1	1	31	Application-specific	16386	4	ADR (adwReadDWord[0])

### FCT 5: Write 1 bit

For the function code "write 1 bit", the value of the bit to be written is encoded in one word.

BIT = TRUE -> Data word = FF 00 HEX

BIT = FALSE -> Data word = 00 00 HEX

### Master request

Slave address	Function code	Slave operand address		Number of words		CRC	
		High	Low	High	Low	High	Low

Slave address	Function code	Slave operand address		Data		CRC	
		High	Low	High	Low	High	Low

### Slave response

Slave address	Function code	Slave operand address		Data		CRC	
		High	Low	High	Low	High	Low

<b>Example:</b>	Modbus interface of the master:	COM1
	Master writes to:	Slave 1
	Data:	bBit := TRUE
	Source address at master:	bBit : BOOL;
	Target address at slave:	%MX0.1026.7 : 2017HEX = 8215DEC
	The value of the BOOL variable bBit on the master is written to %MX0.1026.7 on the slave.	

### Modbus request of the master

Slave address	Function code	Slave operand address		Data		CRC	
		High	Low	High	Low	High	Low
01HEX	05HEX	20HEX	17HEX	FFHEX	00HEX	37HEX	FEHEX

### Modbus response of the slave (mirrored)

Slave address	Function code	Slave operand address		Data		CRC	
		High	Low	High	Low	High	Low
01HEX	05HEX	20HEX	17HEX	FFHEX	00HEX	37HEX	FEHEX

### Parameterization of the COM\_MOD\_MAST block inputs

NB = Number of bits

EN	COM	SLAVE	FCT	TIMEOUT	ADDR	NB	DATA
FALSE -> TRUE	1	1	5	Application-specific	8215	1	ADR (bBit)

### FCT 6: Write 1 word

#### Master request

Slave address	Function code	Slave operand address		Data		CRC	
		High	Low	High	Low	High	Low

#### Slave response

Slave address	Function code	Slave operand address		Data		CRC	
		High	Low	High	Low	High	Low

<b>Example:</b>	Modbus interface of the master:	COM1					
	Master writes to:	Slave 1					
	Data:	wData := 7					
	Source address at master:	wData : WORD;					
	Target address at slave:	%MW0.8199 : 2007HEX = 8199DEC					
	The value of the WORD variable bBit on the master is written to %MW0.8199 on the slave.						

### Modbus request of the master

Slave address	Function code	Slave operand address		Data		CRC	
		High	Low	High	Low	High	Low
01HEX	06HEX	20HEX	07HEX	00HEX	07HEX	72HEX	09HEX

### Modbus response of the slave (mirrored)

Slave address	Function code	Slave operand address		Data		CRC	
		High	Low	High	Low	High	Low
01HEX	06HEX	20HEX	07HEX	00HEX	07HEX	72HEX	09HEX

### Parameterization of the COM\_MOD\_MAST block inputs

NB = Number of words

EN	COM	SLAVE	FCT	TIMEOUT	ADDR	NB	DATA
FALSE -> TRUE	1	1	6	Application-specific	8215	1	ADR (wData)

## FCT 7: Fast reading the status byte of the CPU

### Master request

Slave address	Function code	CRC					
		High	Low				

### Slave response

Slave address	Function code	Data byte	CRC			
			High	Low		

<b>Example:</b>	Modbus interface of the master:	COM1
	Master writes to:	Slave 1
	Data:	
	Source address at slave:	
	Target address at slave:	
	In version V1.x, this function always returns 0!	

### Modbus request of the master

Slave address	Function code	CRC					
		High	Low				
01HEX	07HEX	41HEX	E2HEX				

### Modbus response of the slave

Slave address	Function code	Data byte	CRC			
			High	Low		
01HEX	07HEX	00HEX	xxHEX	xxHEX		

### Parameterization of the COM\_MOD\_MAST block inputs

NB = Number of bits

EN	COM	SLAVE	FCT	TIMEOUT	ADDR	NB	DATA
FALSE -> TRUE	1	1	7	Application-specific	0	0	ADR (BoolVar)



**Note:** In version V1.x, function 7 always returns 0!

## FCT 15: Write n bits

### Master request

Slave address	Function code	Slave operand address		Number of bits		Number of bytes	...Data...	CRC	
		High	Low	High	Low			High	Low

### Slave response

Slave address	Function code	Slave operand address		Number of bits		CRC	
		High	Low	High	Low	High	Low

<b>Example:</b>	Modbus interface of the master:	COM1
	Master writes to:	Slave 1
	Data:	abWriteBool[0] := TRUE; abWriteBool[1] := FALSE; abWriteBool[2] := TRUE
	Source address at master:	abWriteBool : ARRAY[0..2] OF BOOL;
	Target address at slave:	%MX0.1026.1 : 2011HEX = 8209DEC
	The values of the BOOL variables abWriteBool[0]..abWriteBool[2] on the master are written to %MX0.1026.1..%MX0.1026.3 on the slave.	

### Modbus request of the master

Slave address	Function code	Slave operand address		Number of bits		Number of bytes	Data	CRC	
		High	Low	High	Low			High	Low
01HEX	0FHEX	20HEX	11HEX	00HEX	03HEX	01HEX	05HEX	B4HEX	37HEX

### Modbus response of the slave

Slave address	Function code	Slave operand address		Number of bits		CRC	
		High	Low	High	Low	High	Low
01HEX	0FHEX	20HEX	11HEX	00HEX	03HEX	4EHEX	0FHEX

### Parameterization of the COM\_MOD\_MAST block inputs

NB = Number of bits

EN	COM	SLAVE	FCT	TIMEOUT	ADDR	NB	DATA
FALSE -> TRUE	1	1	15	Application-specific	8209	3	ADR (abWriteBool[0])

### FCT 16: Write n words

#### Master request

Slave address	Function code	Slave operand address		Number of words		Number of bytes	...Data...	CRC	
		High	Low	High	Low			High	Low

#### Slave response

Slave address	Function code	Slave operand address		Number of words		CRC	
		High	Low	High	Low	High	Low

<b>Example:</b>	Modbus interface of the master:	COM1
	Master writes to:	Slave 1
	Data:	awWriteWord[0] := 1; awWriteWord[1] := 2; awWriteWord[2] := 3
	Source address at master:	awWriteWord : ARRAY[0..2] OF WORD;
	Target address at slave:	%MW0.8193 : 2001HEX = 8193DEC
	The values of the WORD variables awWriteWord[0]..awWriteWord[2] on the master are written to %MW0.8193..%MW0.8195 on the slave.	

### Modbus request of the master

Slave address	Function code	Slave operand address		Number of words		Number of bytes		Data	Data	Data	CRC
		High / Low	High / Low	High / Low	High / Low	High / Low	High / Low	High / Low	High / Low	High / Low	
01HEX	10HEX	20HEX / 01HEX	00HEX / 03HEX	06HEX	00HEX / 01HEX	00HEX / 02HEX	00HEX / 03HEX	00HEX / 03HEX	00HEX / 03HEX	00HEX / 03HEX	C0HEX / 84HEX

### Modbus response of the slave

Slave address	Function code	Slave operand address		Number of words		CRC	
		High	Low	High	Low	High	Low
01HEX	10HEX	20HEX	01HEX	00HEX	03HEX	DAHEX	08HEX

### Parameterization of the COM\_MOD\_MAST block inputs

NB = Number of words

EN	COM	SLAVE	FCT	TIMEOUT	ADDR	NB	DATA
FALSE -> TRUE	1	1	16	Application-specific	8193	3	ADR (awWriteWord[0])

### FCT 16: Write n double words

The function code "write double word" is not defined in the Modbus RTU standard. This is why the double word is composed of a low word and a high word (depending on the manufacturer).

### Master request

Slave address	Function code	Slave operand address		Number of words		Number of bytes	...Data...	CRC	
		High	Low	High	Low			High	Low

### Slave response

Slave address	Function code	Slave operand address		Number of words		CRC	
		High	Low	High	Low	High	Low

<b>Example:</b>	Modbus interface of the master:	COM1									
	Master writes to:	Slave 1									
	Data:	adwWriteDWord[0] := 18DEC = 0000012HEX; adwWriteDWord[1] := 65561DEC = 00010019HEX;									
	Source address at master:	adwWriteDWord : ARRAY[0..1] OF WORD;									
	Target address at slave:	%MD0.8192 : 4000HEX = 16384DEC									
	The values of the Double WORD variables adwWriteDWord[0].. adwWriteDWord[1] on the master are written to %MD0.8192..%MD0.8193 on the slave.										

### Modbus request of the master

Slave address	Function code	Slave operand address		Number of words		Number of bytes		Data	Data	Data	Data	CRC
		High / Low	High / Low	High / Low	High / Low	High / Low	High / Low	High / Low	High / Low	High / Low	High / Low	
01HEX	10HEX	40HEX / 00HEX	00HEX / 04HEX	00HEX / 08HEX	00HEX / 00HEX	00HEX / 12HEX	00HEX / 01HEX	00HEX / 19HEX	00HEX / 19HEX	00HEX / 19HEX	00HEX / 19HEX	60HEX / B3HEX

### Modbus response of the slave

Slave address	Function code	Slave operand address		Number of words		CRC	
		High	Low	High	Low	High	Low
01HEX	10HEX	40HEX	00HEX	00HEX	04HEX	DAHEX	0AHEX

## Parameterization of the COM\_MOD\_MAST block inputs

NB = Number of words = 2 x Number of double words

EN	COM	SLAVE	FCT	TIMEOUT	ADDR	NB	DATA
FALSE -> TRUE	1	1	16	Application-specific	16384	4	ADR (adwWriteDWord[0])

### Error telegram

In operating mode Modbus master, the AC500 does only send telegrams, if the parameters at the MODMAST inputs are logically correct. Nevertheless, it can happen that a slave cannot process the request of the master or that the slave cannot interpret the request due to transmission errors. In those cases, the slave returns an error telegram to the master. In order to identify this telegram as an error telegram, the function code returned by the slave is a logical OR interconnection of the function code received from the master and the value 80HEX.

### Slave response

Slave address	Function code OR 80HEX	Error code	CRC	
			High	Low

### Possible error codes of the slave

Code	Meaning
01DEC	The slave does not support the function requested by the master
02DEC	Invalid operand address in the slave
02DEC	Operand area exceeded
03DEC	At least one value is outside the permitted value range
12DEC	The amount of data is higher than the slave can process
13DEC	The telegram contains an odd number of words in case of double word access
10DEC	Length specifications in the telegram do not match
11DEC	The type of operand area and the function do not match

### Example:

Modbus request of the master:			
	Function code:	01	(Read n bits)
	Slave operand address:	4000HEX = 16384DEC	(Area for read access disabled in slave)
Modbus response of the slave:			
	Function code:	81HEX	
	Error code:	03	

## 11.6 Function block COM\_MOD\_MAST

This function block is only required in the operating mode Modbus master. It handles the communication (transmission of telegrams to the slaves and reception of telegrams from the slaves). The function block can be used for the local interfaces COM1 and COM2 of the controller. A separate instance of the function block has to be used for each interface.

COM\_MOD\_MAST is contained in the library **Modbus\_AC500\_V1x.LIB (version V1.0 and later)**.

# Index - System Technology of the CPUs

## A

### 2 AC500 inputs, outputs and flags 26

2.1 AC500 interfaces for inputs and outputs 26

2.1.1 Address scheme for inputs and outputs 27

2.1.2 Example for addressing in BOOL / BYTE / WORD / DWORD 27

2.2 Addressing of inputs and outputs 28

2.3 Processing of inputs and outputs in the multitasking system 29

2.4 Addressable flag area (%M area) in the AC500 30

2.4.1 Allocation of the addressable flag area in the AC500 30

2.4.2 Access to the %M area using the Modbus® Protocol 31

2.4.3 Access to operands in the addressable flag area 31

2.5 Absolute addresses of operands 32

2.5.1 Address operator ADR 32

2.5.2 Bit address operator BITADR 32

2.6 Addressable PERSISTENT area (%R area) in the AC500 34

2.6.1 Special features of the addressable PERSISTENT area in the AC500 34

2.6.2 Segmentation of the addressable PERSISTENT area in the AC500 35

2.6.3 Saving the buffered data of the AC500's %R area 35

2.6.4 Access to operands in the addressable PERSISTENT area (%R area) 37

## C

### 11 Communication with Modbus RTU 166

11.1 Protocol description 166

11.2 Modbus RTU with the serial interfaces COM1 and COM2 167

11.2.1 Modbus operating modes of the serial interfaces 167

11.3 Modbus on TCP/IP via Ethernet 167

11.4 Modbus addresses 168

11.4.1 Modbus address table 168

11.4.2 Peculiarities for accessing Modbus addresses 170

11.4.3 Comparison between AC500 and AC31/90 Modbus addresses 171

11.5 Modbus telegrams 173

11.6 Function block COM\_MOD\_MAST 180

## D

### 7 Data storage in Flash memory 146

7.1 Blocks used for data storage 146

7.2 Example program for data storage 146

## P

### 10 Programming and testing 150

- 10.1 Programming interfaces to the AC500 used by the Control Builder 150
- 10.2 Programming via the serial interfaces 151
  - 10.2.1 Serial driver "Serial (RS232)" 152
  - 10.2.2 Serial driver "ABB RS232 Route AC" 153
- 10.3 Programming via ARCNET 156
  - 10.3.1 ARCNET driver "ABB Arcnet AC" 157
- 10.4 Programming via Ethernet (TCP/IP) 159
  - 10.4.1 Ethernet driver "Tcp/Ip" 160
  - 10.4.2 Ethernet driver "ABB Tcp/Ip Level 2 AC" 161
  - 10.4.3 Ethernet ARCNET routing 164

## R

### 8 Real-time clock and battery in the AC500 147

- 8.1 General notes concerning the real-time clock in the AC500 147
- 8.2 Setting and displaying the real-time clock 147
  - 8.2.1 Setting and displaying the real-time clock with the PLC browser 147
  - 8.2.2 Setting and displaying the real-time clock with the user program 148
- 8.3 The AC500 battery 148

## S

### 4 System start-up / program processing 79

- 4.1 Terms 89
  - Cold start* 89
  - Warm start* 89
  - RUN -> STOP* 89
  - START -> STOP* 89
  - Reset* 89
  - Reset (cold)* 89
  - Reset (original)* 89
  - STOP -> RUN* 89
  - STOP -> START* 90
  - Download* 90
  - Online Change* 90
  - Data buffering* 90
- 4.2 Start of the user program 91
- 4.3 Data backup and initialization 92
  - 4.3.1 Initialization of variables, overview 92
  - 4.3.2 Notes regarding the declaration of retentive variables and constants 94
    - Declaration of retentive internal variables 94
    - Declaration of retentive variables in %M area 94
    - Declaration of constants 94
- 4.4 Processing times 95
  - 4.4.1 Terms 95
  - 4.4.2 Program processing time 95
  - 4.4.3 Set cycle time 95
- 4.5 Task configuration for the AC500 CPU 96



# T

## 1 Target Support Package 7

- 1.1 Introduction 7
  - 1.1.1 Control Builder PS501 versions 7
  - 1.1.2 New functions in PS501 V1.2 8
  - 1.1.3 Compatibility of versions V1.0, V1.1 and V1.2 9
    - File structure of the target system 9
    - Overview on target system files 9
    - Compatibility of CPU bootcode, CPU firmware, target system and CoDeSys 13
    - Conversion of a project created with version V1.0 or V1.1 to version V1.2 14
- 1.2 Selection of the target system - Target support settings 15
- 1.3 CPU parameters in the target support settings 16
  - 1.3.1 "Target Platform" settings 16
  - 1.3.2 "Memory Layout" settings 17
  - 1.3.3 "General" settings 19
  - 1.3.4 "Network Functionality" settings 21
  - 1.3.5 "Visualization" settings 21
- 1.4 Overview on user program size and operands of AC500 CPUs 22
- 1.5 Installation of AC500 targets with the program installTarget.exe 23

## 3 The AC500 PLC configuration 38

- 3.1 Overview on the PLC configuration 38
  - 3.1.1 PLC configuration functions 38
  - 3.1.2 Export and import of configuration data 38
  - 3.1.3 Default settings in the PLC configuration 39
  - 3.1.4 Setting parameters in the PLC configuration 40
- 3.2 Configuration of CPU parameters 40
  - 3.2.1 CPU parameters in PS501 versions V1.0 and V1.1 40
    - Remark 1: Setting the parameters Auto run and MOD using the display/keypad 41
  - 3.2.2 CPU parameters in version PS501 V1.2 42
    - Remark 1: Setting the parameters Auto run and MOD using the display/keypad 44
    - Remark 2: Error LED 44
    - Remark 3: Behaviour of outputs in Stop 44
    - Remark 4: Reaction on floating point exceptions 44
    - Remark 5: Stop on error class 45
    - Remark 6: Warmstart 45
    - Remark 7: Start PERSISTENT %Rsegment.x and End PERSISTENT %Rsegment.x 46
- 3.3 I/O bus configuration 47
  - 3.3.1 Setting the general I/O bus parameters 47
  - 3.3.2 Inserting input and output modules 47
  - 3.3.3 Configuring the input and output modules and channels 48
  - 3.3.4 Module parameter "Ignore module" of S500 I/O devices 51
- 3.4 Configuration of the serial interfaces (Interfaces / COM1 and COM2) 52
  - 3.4.1 Setting the protocol of the serial interfaces 52
  - 3.4.2 The setting 'COMx - Online access' 53
  - 3.4.3 The setting 'COMx - ASCII' 53
    - Remark 1: Enable login 56
    - Remark 2: Usage of modems 56
    - Remark 3: Telegram ending identifier 57
    - Remark 4: Checksum 59
  - 3.4.4 The setting 'COMx - Modbus' 60
  - 3.4.5 The setting 'COM1 - CS31 Bus' 62
    - Connecting the DC551 and S500 I/O devices to the CS31 bus 64
    - Overview on input/output data of S500 I/O devices 68
    - Examples of impossible configurations 69

- 3.4.6 The setting 'COMx - SysLibCom' 71
  - Remark 1: Enable login 73
  - Remark 2: Usage of modems 73
  - Remark 3: Telegram ending identifier 73
  - Example for sending/receiving with "SysLibCom" 74
- 3.4.7 The setting 'COMx - Multi' 77
  - Functions of the block COM\_SET\_PROT 78
- 3.5 FBP slave interface configuration (Interfaces / FBP slave) 79
- 3.6 Coupler configuration (Couplers) 81
  - 3.6.1 Configuring the internal coupler 82
    - 3.6.1.1 The internal Ethernet coupler PM5x1-ETH 82
    - 3.6.1.2 The internal ARCNET coupler PM5x1-ARCNET 84
      - Remark 1: Baudrate of the ARCNET coupler 85
      - Remark 2: Check of DIN identifier on receipt 85
  - 3.6.2 Configuring the external couplers 87

## **5 The diagnosis system in the AC500 97**

- 5.1 Summary of diagnosis possibilities 97
  - 5.1.1 Structure of the diagnosis system 97
  - 5.1.2 Diagnosis directly at the PLC by means of "ERR" LED, keypad and display 98
  - 5.1.3 Plain-text display of error messages in the Control Builder status line during online mode 99
  - 5.1.4 Diagnosis using the PLC browser commands of the Control Builder 99
  - 5.1.5 Diagnosis with help of the user program 99
- 5.2 Organization and structure of error numbers 99
  - 5.2.1 Error classes 100
  - 5.2.2 Error identifiers 100
  - 5.2.3 Possible error numbers 102
  - 5.2.4 Error list 106
  - 5.2.5 Coupler errors 113
- 5.3 Diagnosis blocks for the AC500 118
- 5.4 AC500-specific PLC browser commands 118

## **9 The fast counters in the AC500 149**

- 9.1 Activating the fast counters via the I/O bus 149
- 9.2 Counting modes of the fast counters 149

## **6 The SD memory card in the AC500 122**

- 6.1 SD card functions 122
  - 6.1.1 Summary of memory card functions 122
  - 6.1.2 PLC browser commands for accessing the SD card 122
- 6.2 SD card file system 123
  - 6.2.1 SD card file structure 123
    - File structure in versions V1.0 and V1.1 123
    - File structure as of version V1.2 124
  - 6.2.2 The command file "SDCARD.INI" 126
    - File content in versions V1.0 and V1.1 126
    - File content as of version V1.2 127
  - 6.2.3 Initializing an SD card 129
    - 6.2.3.1 Initializing an SD card using the AC500 129
    - 6.2.3.2 Initializing the SD card using a PC 129
- 6.3 Storing/loading the user program to/from an SD card 130
  - 6.3.1 Storing the user program to an SD card 130
  - 6.3.2 Loading a user program from the SD card to the AC500 130

6.4 Storing/reading user data to/from an SD card	131
6.4.1 Structure of data files stored on the SD card	131
6.4.2 Blocks for storing/reading user data to/from the SD card	132
6.4.3 Deleting a data file stored on the SD card	134
6.4.4 Storing user data to the SD card - data file without sectors	134
6.4.5 Storing user data to the SD card - data file with sectors	135
6.4.6 Loading user data from the SD card - data file without sectors	136
6.4.7 Loading user data from the SD card - data file with sectors	137
6.5 Storing and loading retentive data to/from an SD card	138
6.6 Firmware update from the SD card	138
6.6.1 Storing the firmware to the SD card	138
6.6.2 Updating the firmware of the AC500 CPU from the SD card	138
6.7 Writing and reading the project sources to/from the SD card	139
6.7.1 Writing the project sources from PC to SD card	140
6.7.2 Loading the project sources from the PLC's SD card into the PC	142
6.7.3 Loading the project sources from the SD card using the PC SD card reader	144
6.8 SD card error messages	145

