

Modular PLC XC-CPU201-...(-XV)

**Hardware, Engineering and
Functional Description**

01/08 AWB2724-1491GB

MOELLER 

We keep power under control.

All brand and product names are trademarks or registered trademarks of the owner concerned.

1st published 2003, edition date 12/03
2nd published 2003, edition date 12/03
3rd published 2004, edition date 06/04
4th published 2004, edition date 08/04
5th published 2004, edition date 11/04
6th published 2005, edition date 03/05
7th published 2005, edition date 11/05,
8th published 2006, edition date 12/06
9th published 2007, edition date 04/07
10th published 2008, edition date 01/08

See revision protocol in the "About this manual" chapter.

© Moeller GmbH, 53105 Bonn

Authors: Peter Roersch
Editor: Thomas Kracht
Translator: OneWord

All rights reserved, including those of the translation.

No part of this manual may be reproduced in any form (printed, photocopy, microfilm or any other process) or processed, duplicated or distributed by means of electronic systems without written permission of Moeller GmbH, Bonn.

Subject to alteration without notice.



Warning! **Dangerous electrical voltage!**

Before commencing the installation

- Disconnect the power supply of the device.
- Ensure that devices cannot be accidentally restarted.
- Verify isolation from the supply.
- Earth and short circuit.
- Cover or enclose neighbouring units that are live.
- Follow the engineering instructions (AWA) of the device concerned.
- Only suitably qualified personnel in accordance with EN 50110-1/-2 (VDE 0105 Part 100) may work on this device/system.
- Before installation and before touching the device ensure that you are free of electrostatic charge.
- The functional earth (FE) must be connected to the protective earth (PE) or to the potential equalisation. The system installer is responsible for implementing this connection.
- Connecting cables and signal lines should be installed so that inductive or capacitive interference does not impair the automation functions.
- Install automation devices and related operating elements in such a way that they are well protected against unintentional operation.
- Suitable safety hardware and software measures should be implemented for the I/O interface so that a line or wire breakage on the signal side does not result in undefined states in the automation devices.
- Ensure a reliable electrical isolation of the low voltage for the 24 volt supply. Only use power supply units complying with IEC 60364-4-41 (VDE 0100 Part 410) or HD 384.4.41 S2.
- Deviations of the mains voltage from the rated value must not exceed the tolerance limits given in the specifications, otherwise this may cause malfunction and dangerous operation.
- Emergency stop devices complying with IEC/EN 60204-1 must be effective in all operating modes of the automation devices. Unlatching the emergency-stop devices must not cause restart.
- Devices that are designed for mounting in housings or control cabinets must only be operated and controlled after they have been installed with the housing closed. Desktop or portable units must only be operated and controlled in enclosed housings.
- Measures should be taken to ensure the proper restart of programs interrupted after a voltage dip or failure. This should not cause dangerous operating states even for a short time. If necessary, emergency-stop devices should be implemented.
- Wherever faults in the automation system may cause damage to persons or property, external measures must be implemented to ensure a safe operating state in the event of a fault or malfunction (for example, by means of separate limit switches, mechanical interlocks etc.).

Contents

| | | |
|----------------------------------|---|-----------|
| About this manual | | 5 |
| | List of revisions | 5 |
| | Reading conventions | 6 |
| | Additional documentation | 6 |
| 1 Design of the XC200 PLC | | 7 |
| | Module rack | 7 |
| | CPU with PSU and local inputs/outputs | 7 |
| | – Performance scope of the CPU | 7 |
| | – Functional spans | 7 |
| | – Power supply | 8 |
| | – Local inputs/outputs | 8 |
| | – Processor unit with interfaces | 9 |
| | – Real-time clock | 9 |
| | – Battery | 9 |
| | – Multimedia card (MMC) and USB stick | 9 |
| | – CPU drives | 9 |
| | – ETH232 programming interface | 10 |
| | – CANopen/easyNet interface | 10 |
| | – Supplementary functions of the CPU (local inputs) | 11 |
| 2 CPU installation | | 13 |
| | – Detaching the CPU | 13 |
| 3 Engineering | | 15 |
| | Control panel layout | 15 |
| | – Ventilation | 15 |
| | – Layout of units | 15 |
| | Preventing interference | 15 |
| | – Suppressor circuitry for interference sources | 15 |
| | – Shielding | 15 |
| | Lighting protection | 16 |
| | Installation instructions for ABS enclosure | 16 |
| | Connections | 16 |
| | – Connecting the power supply | 16 |
| | – Connecting inputs/outputs (CPU) | 17 |
| | – Connecting the incremental value encoder | 17 |
| | – Connecting up/down counter | 17 |
| | – Connecting interrupt actuators | 17 |
| | – Connect PC | 18 |
| | Interface assignments | 18 |
| | – USB interface | 18 |
| | – ETH232 programming interface | 18 |
| | – CANopen/easyNet interface | 19 |

| | | |
|----------|---|----|
| 4 | Operation | |
| | Start behaviour | 21 |
| | – Configuring the start-up behaviour with easySoft-DeCoSys | 22 |
| | Program start | 22 |
| | – Program start (STOP → RUN) | 22 |
| | – Program stop (RUN → STOP) | 22 |
| | Power off/Interruption of the power supply | 23 |
| | CPU operating state display | 23 |
| | Test and commissioning (Debugging) | 23 |
| | – Breakpoint/single-step mode | 23 |
| | – Single-cycle mode | 23 |
| | – Forcing | 23 |
| | – Status indicator | 23 |
| | Reset | 24 |
| | Programs and project | 24 |
| | – Loading the program | 24 |
| | – Saving boot project on MMC | 25 |
| | – Erase boot project on MMC | 25 |
| | Updating the operating system | 25 |
| | – Transferring the operating system from the PC to the PLC | 25 |
| | – Transferring the OS from the PC into the MMC | 26 |
| | Erase operating system/boot project from the MMC | 26 |
| 5 | Program processing, multitasking and system times | 27 |
| | Task configuration | 27 |
| | Creating task (example) | 27 |
| | System events | 29 |
| | Multitasking | 29 |
| | – Behaviour of the CAN stack with multitasking | 30 |
| | Task monitoring with the watchdog | 31 |
| | – Watchdog configuration | 31 |
| | – Multiple tasks with the same priority | 32 |
| | Direct peripheral access | 33 |
| | – ReadBitDirect | 34 |
| | – ReadWordDirect | 34 |
| | – ReadDWordDirect | 34 |
| | – Write...Direct | 35 |
| | – WriteBitDirect | 35 |
| | – WriteWordDirect | 35 |
| | – GetSlotPtr | 35 |
| | – Error code with "direct peripheral access" | 36 |
| | Operating states | 36 |
| | Web visualization | 37 |
| | Limit values for memory usage | 37 |
| | Addressing inputs/outputs and markers | 37 |
| | – "Activate Automatic addresses" | 38 |
| | – "Activating Check for overlapping addresses" | 38 |
| | – Uneven word addresses | 38 |
| | – Address range | 38 |
| | – Free assignment or modification of addresses of input/output modules and diagnostic addresses | 39 |
| | – Run "Automatic calculation of addresses" | 39 |
| | Diagnostics | 39 |

| | | |
|-----------|--|----|
| 6 | Establishing a PC – CPU connection | 41 |
| | Connection set-up via RS-232 interface | 41 |
| | – Defining/changing the PC's communication settings | 41 |
| | – Changing the CPU's communication settings | 42 |
| | Connection setup via Ethernet | 42 |
| | Selecting communication channel and address | 42 |
| | Scan/Modify the IP address | 43 |
| 7 | Defining the system parameters via the STARTUP.INI file | 45 |
| | Overview | 45 |
| | Structure of the INI file | 45 |
| | Creating the Startup.INI file | 45 |
| | Entry of the INI file: "HOST_NAME" | 46 |
| | Switch-on of the control with inserted memory card with XCSTARTUP.INI file | 46 |
| | Changing settings | 46 |
| | Deleting the Startup.INI file | 46 |
| 8 | Programming via CAN(open) Network (Routing) | 47 |
| | Prerequisites | 47 |
| | Routing features of the controller | 47 |
| | Notes | 48 |
| | Addressing | 48 |
| | Communication with the target PLC | 49 |
| | PLC combinations for routing | 50 |
| | Number of communication channels | 50 |
| 9 | RS -232 interface in Transparent mode | 51 |
| | Programming of the RS -232 interface in transparent mode | 51 |
| 10 | Configuration and parameterization of the inputs/outputs | 53 |
| | Input/output general | 53 |
| | Local digital inputs/outputs | 53 |
| | Inputs/outputs for additional functions | 55 |
| | – Incremental encoder | 55 |
| | – Counter | 57 |
| | Interrupt processing | 59 |
| | – DisableInterrupt | 59 |
| | – EnableInterrupt | 59 |
| | – Parameterization | 59 |
| | – Example for interrupt processing | 60 |

| | |
|--|----|
| 11 Libraries, function blocks and functions | 61 |
| Using libraries | 61 |
| Installing additional system libraries | 61 |
| XC200 specific functions | 62 |
| – CAN_Uilities. | 62 |
| – XI/OC functions | 62 |
| – Event functions | 63 |
| Additional functions of the XC200_UTIL2.lib | 65 |
| – UT12_GetIPConfig | |
| Issue of the IP address, subnetmask address and IP gateway address | 65 |
| – UT12_GetMacAddress | |
| Issue of the MAC address (MAC=Media Access Control) | 65 |
| – UT12_SetIPConfig | |
| Setting of the IP- and subnetmask address | 65 |
| – UT12_SetIPGateway | |
| Setting of the IP Gateway address | 66 |
| – UT12_Reboot | |
| Restart with registry storage | 66 |
| – UT12_SaveRegistry | |
| Saving of the registry | 66 |
| 12 Browser commands | 67 |
| Calling up browser commands | 68 |
| Accessing communications parameters | 69 |
| Display CPU loading (plcload) | 69 |
| Display the loading of the CAN bus (canload) | 69 |
| – Access to memory objects | 70 |
| Error and event list after calling browser commands | 70 |
| Appendix | 73 |
| Characteristic of the Ethernet cable | 73 |
| Properties of the CANOpen cable | 73 |
| Transparent mode: Text output via RS232 (example) | 74 |
| Access to the CPU drives/memory card | 76 |
| – "SysLibFile.lib" library | 76 |
| – Mode for opening a file | 76 |
| – Examples of the "SysFile..." functions | 76 |
| USB stick types | 77 |
| Dimensions | 77 |
| – XC-CPU201... | 77 |
| – XT-FIL-1 line filter | 77 |
| – Racks | 77 |
| Technical data | 78 |
| Index | 85 |
| <p>→ The previous Chapter 13: "The easyNet network" and Chapter 14: "Programming via easyNet (routing)" are omitted.</p> <p>You will find this information in far greater detail in the 08/07 AWB2786-1593 manual "Data transfer between easy and IEC PLCs (easyNet)".</p> | |

About this manual


List of revisions


The following significant amendments have been introduced since previous issues:


| Edition date | Page | Keyword | New | Modification |
|-----------------|------------|---|-----|--------------|
| 12/03 (Reprint) | 38 | Data remanence, 1st paragraph | | ✓ |
| 04/04 | 37 | Limit values for memory usage | ✓ | |
| | 35 | WriteBitDirect | | ✓ |
| 06/04 | 16, 86, 91 | External 24 V DC line filter for the XC200 power supply | ✓ | ✓ |
| 08/04 | 38 | Data remanence, note | ✓ | |
| | 42 | Download of programs | ✓ | |
| | 65 | RS -232 interface of the XIOC-SER in transparent mode (COM2/3/4/5) | ✓ | |
| | 90 | Electromagnetic compatibility | | ✓ |
| 11/04 | 9, 85 | Multimedia card (MMC) and USB stick | ✓ | |
| | 10 | Splitting of the ETH232 interface | | ✓ |
| | 23 | Status indicator | | |
| | 45 | Connection set-up via RS -232 interface | ✓ | |
| | 62 | XC200 specific functions | ✓ | |
| | 65 | Additional functions of the XC200_UTIL2.lib | | ✓ |
| | 51 | RS -232 interface in Transparent mode | ✓ | |
| 03/05 | 10 | Splitting of the ETH232 interface | | ✓ |
| | 17 | Figure 20 | | ✓ |
| | 37 | Segment size of the XC-CPU201-EC256k | ✓ | |
| | 37 | Addressing inputs/outputs and markers | ✓ | |
| | 39 | Diagnostics | ✓ | |
| | 47 | Programming via CAN(open) Network (Routing) | ✓ | |
| 09/05 | 43 | Set the system parameters via the STARTUP.INI file | ✓ | |
| 11/05 | | Complete revision of the manual | | |
| 09/06 | 45 | chapter "Defining the system parameters via the STARTUP.INI file" | | ✓ |
| | 71 | Chapter "The easyNet network" | ✓ | |
| 12/06 | 27 | chapter "Program processing, multitasking and system times" | | ✓ |
| | 47 | chapter "Programming via CAN(open) Network (Routing)" | | ✓ |
| 04/07 | 71 | Chapter "The easyNet network" | | ✓ |
| 01/08 | | Chapter 13: "The easyNet network" and Chapter 14: "Programming via easyNet (routing)" are omitted. You will find this information in far greater detail in the 08/07 AWB2786-1593 manual "Data transfer between easy and IEC PLCs (easyNet)". | | ✓ |
| 01/08 | 10 | Figure 7 | | ✓ |

Reading conventions

→ Draws your attention to interesting tips and supplementary information.

 **Attention!**
Warns of the risk of material damage

 **Caution!**
Warns of the possibility of serious damage and slight injury.

 **Warning!**
Indicates the risk of major damage to property, or serious or fatal injury.

► indicates instructions to be followed

Select «File → New» means: activate the instruction “New” in the “File” menu.

For clarity of layout, we adhere to the following conventions in this manual: at the top of left-hand pages you will find the Chapter heading, at the top of right-hand pages the current Section heading; exceptions are the first pages of Chapters and empty pages at the end of Chapters.

Additional documentation

At different points in this manual, references are made to more detailed descriptions in other manuals. These are described with their title and documentation number (e.g. AWB2700-1437D). All manuals are available in PDF format. If for some reason they are not supplied on the product CD, they are available for download as PDF files. Go to <http://www.moeller.net/support> and enter the document number in the Quick Search field.

1 Design of the XC200 PLC

The XC200 PLC is designed for use in machine controls and systems. With an RS232/Ethernet interface for connection of a programming device (RS232/Ethernet), the central coupling of XI/OC signal modules and the decentral coupling of CANopen devices, this control forms the basis for the implementation of a comprehensive automation system.

The PLC consists of the:

- Rack → page 7
- CPU with PSU and local inputs/outputs→ page 7
- XI/OC signal modules → separate manual "Hardware and Engineering" (AWB2725-1452GB).

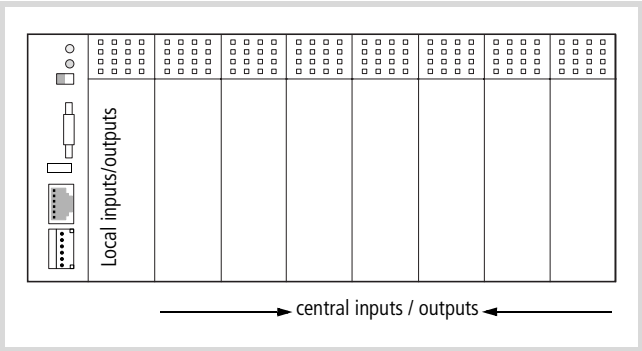


Figure 1: Layout of the XC-CPU201 with XI/OC modules

→ XSoft from Version 200 is required for programming the XC2.3.

Module rack

There are base module racks and expansion racks.

The base module rack XIOC-BP-XC features two slots for the CPU. The XIOC-BP-XC1 provides three slots, so that there is also place available for an XI/OC signal module beside the CPU.

A base module rack can be expanded using several expansion racks. Expansion racks are fitted with XI/OC signal modules.

The rack establishes the connection between the CPU and the modules using an integrated bus rail.

→ Detailed information about the backplanes and XI/OC signal modules can be found in the manual "Hardware and Engineering" (AWB 2725-1452GB).

CPU with PSU and local inputs/outputs

Performance scope of the CPU

In order to better cover the requirements for different applications, the CPU is available with different performance levels. This affects the size of the user memory and function of the integrated web server.

The following part numbers are available:

- XC-CPU201-EC256K-8DI-6DO(-XV)
- XC-CPU201-EC512K-8DI-6DO (-XV).

"EC256K" and "EC512K" are a measure for the size of the user memory. "XV" identifies a visualisation CPU with integrated web server.

According to the size of the application program, the following memory values apply:

| | XC-CPU201-...-8DI-6DO(-XV) | |
|-------------------------|----------------------------|---------------------------------|
| | ...EC256K | ...EC512K |
| Program code | 512 kByte | 2048 kByte From OS V 1.04.01 |
| Program data, of which: | 256 kByte | 512 kByte |
| Markers | 16 kByte | 16 kByte |
| Retain data | 32 kByte | 32 kByte |
| Persistent data | 32 kByte | 32 kByte |

Functional spans

The CPU is arranged into three functional areas:

- Power supply → page 8
- Local inputs and outputs → page 8
- Processor unit with interfaces → page 9



Figure 2: XC200-CPU

- ① Processor unit with interfaces
- ② Power supply with local inputs/outputs

Power supply

Two separate voltage supplies are available for the power supply of the processor unit and the local inputs/outputs: On the one hand a 24 V connection exists for the processor unit (inscription: 24V/0V) and on the other a 24 V connection for the local inputs/outputs (inscription: 24VQ/0VQ).

If there is an interruption break or collapse of the 24 V supply (threshold is about 10 V) then a power-down logic switches of the 5 V supply to the signal modules (central I/O).

Local inputs/outputs

On the right half of the CPU an 18-pole terminal block is located behind the front cover of the CPU. This is used to connect the voltage supply of the CPU and the local inputs/outputs as well as the sensors and actuators.

The eight digital inputs and six semiconductor outputs are designed for 24 V signals and have a common 0VQ/24VQ power supply which is potentially isolated right up to the bus.

The outputs Q0.0 to Q0.5 can be loaded with 500 mA, a duty factor (ED) of 100% and a utilization factor (g) of 1.



Attention!

Please observe the limitations of performance for the outputs with ABS enclosures on → page 16.

The outputs are short-circuit proof. A short-circuit state should, however, not be permitted to exist over a longer period.

See also:

- Supplementary functions of the CPU (local inputs) → page 11

Terminal assignments

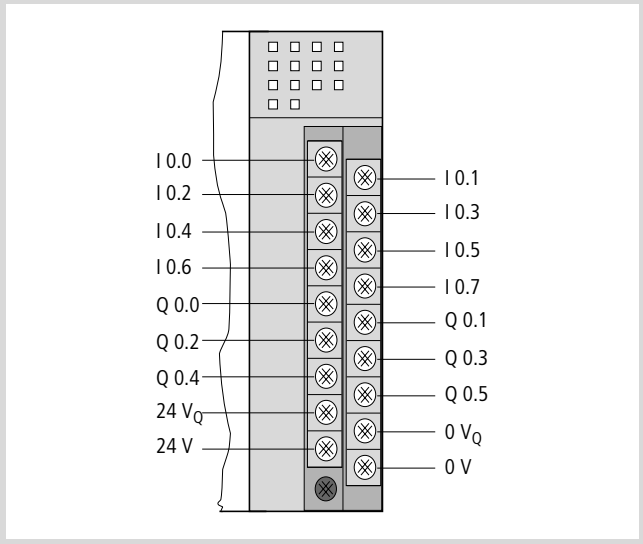


Figure 3: Power supply and local input/output connections

- I0.0 to I0.7: local digital inputs
- Q0.0 to Q0.5: local digital outputs
- 0VQ/+24VQ: supply voltage for the local inputs/outputs
- 0V/+24V: supply voltage to the processor unit

LED displays

The LEDs indicate the signal status for the inputs and outputs. An LED that is ON indicates a H-level signal on the corresponding terminal.

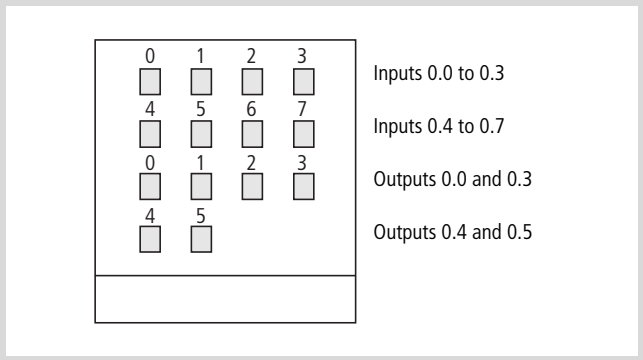


Figure 4: LEDs for the local inputs/outputs

The two upper rows of LEDs show the signal status for the eight digital inputs of the CPU module (I0.0 to I0.7), and the two lower rows show the signal status for the six digital outputs (Q0.0 to Q0.5).

Processor unit with interfaces

Belonging to the processor unit are:

- Real-time clock → page 9
- Battery → page 9
- Multimedia card (MMC) and USB stick → page 9
- CPU drives → page 9
- USB interface → page 18
- ETH232 programming interface → page 10
- CANopen/easyNet interface → page 10
- Supplementary functions of the CPU (local inputs) → page 11.

Real-time clock

The XC200 features a real-time clock, which can be referenced in the user program via the functions from the "SysLibRTC" library. The following functions are possible:

- Display of the battery charge state
- Display mode for hours (12/24 hour display)
- Reading and setting of the real-time clock.

A description of the functions can be found in the "SysLibRTC.pdf" file.

Furthermore, you can set or scan the realtime clock via the following browser commands:

- setrtc (set the real-time clock) → page 67
- getrtc (query the real-time clock) → page 67.

Battery

A Lithium battery of type 1/2 AA (3.6 V) is used for saving of volatile data and for operation of the real-time clock. The battery compartment can be found on the left side of the CPU unit, behind a cover plate. The charge level of the battery is monitored. If the battery voltage falls below a fixed preset level, then a general error message will be generated. The battery backup times are:

- Worst-case: 3 years continuous buffering
- Typical: 5 years of continuous buffering



Attention!

Exchange the battery only when the power supply is switched on. Otherwise data will be lost.

Ordering designation of the battery: XT-CPU-BAT-1.

Multimedia card (MMC) and USB stick

MMC and USB serve as bulk storage. You can load the recipe data, general data and the user program onto them. The operating system (OS) supports memory types with the FAT16 file system.

From operating system version 01.03, the operating system can be transferred to the MMC in order to load it from there into other controls (OS update). The use of a USB stick is also possible from this OS version.



Attention!

The file system of the memory card is not transaction-safe. Make sure that all the files of the program are closed before you plug or un-plug a card or turn off the voltage.

See also:

- Updating the operating system → page 25
- Erase operating system/boot project from the MMC → page 26
- Assignment of the USB interface → page 18
- Usable USB stick types → page 77



Erasing of files is implemented in the same way as erasing the operating system.

CPU drives

The XC200 has the following drives available:

- internal
 - Memory system (disk_sys)
- external, optional
 - Multi Media Card MMC (disk_mmc)
 - USB stick (disk_usb).

The boot system and the operating system are saved in compressed format and protected against failure of the power supply in the transaction safe system memory. In the operating state, the boot project and the relevant sections of operating system are "unpacked" and copied into the working memory. The retentive data are stored in the battery-buffered SRAM memory.



Transaction safe means that if there is a voltage dip when the file is being processed, the file system and the opened file are generally not destroyed. It is possible however, that data which you have written into the file last opened may be lost.

Figure 5 indicates the interaction of the differing XC200-CPU memory systems/drives:

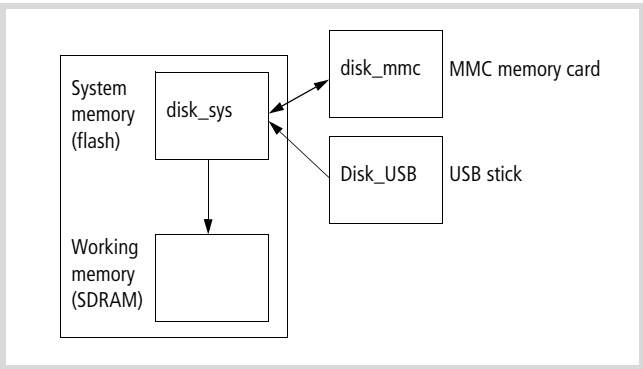


Figure 5: XC200-CPU memory organization

See also:

- Data access to the memory card
 - with the aid of browser commands such as, for example, copyprojtoMMC, to copy the user program onto the MMC → page 67
 - Functions such as "SysFileOpen" or "SysFileRead" → page 76
- Limit values for memory usage → page 37

ETH232 programming interface

Communication between the CPU and the programming device takes place through the programming interface ETH232. It comprises of an Ethernet interface and an RS 232 interface.

The Ethernet interface is used for programming and debugging, as it is processed more quickly by the operating system. This interface features network capabilities and is electrically isolated.

You can also program via the RS-232 interface. From operating system version V01.03 you can also switch the RS232 interface in transparent mode, in order to establish a point-to-point connection without handshake lines.

Splitting of the ETH232 interface

Using a cable splitter XT-RJ45-ETH-RS232 you can communicate simultaneously via the RS-232 and the Ethernet interface. The connection between the CPU and the cable splitter is established using the EASY-NT-30/80/130 cable. Then connect the cable from the "IN" socket of the cable splitter to the ETH232 connector of the CPU.

On the Ethernet interface of the cable splitter you can for example, connect the programming device and the RS-232 interface (in transparent mode) of a printer. The pin assignment of the RS232 and Ethernet connector socket of the cable splitter corresponds with the pin assignment of the ETH232 connector socket of the CPU, → section „Splitting of the ETH232 interface“ on Page 10.

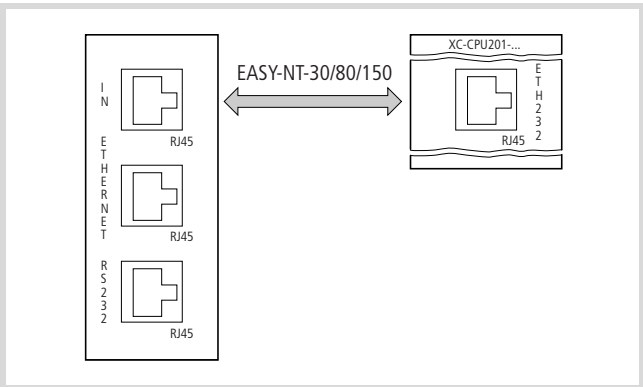


Figure 6: Connection of the XC-CPU201 with the XT-RJ45-ETH-RS232

See also:

- Connect PC → page 18
- Assignment of the programming interface → page 18
- Establishing a PC – CPU connection → page 41.

CANopen/easyNet interface

The CANopen-/easyNET interface is potentially isolated. The connections for both interfaces are identical. The CPU can be operated on the CANopen bus as a network (NMT) master or as a NMT slave (device). The CAN and the easyNET protocol can be executed in parallel by the CPU.

See also:

- Detailed information for engineering and programming CAN stations → Application note AN2700K27.
- Operating manual easy800 control relays (AWB2528-1423)
- Network easyNet → page 73

Bus termination resistors: a bus termination resistor must be installed at the first or last station on the line:

| Stations | Bus termination resistor |
|----------|--------------------------|
| XC200 | 120 Ω |
| easy800 | EASY-NT-R |
| MFD | EASY-NT-R |

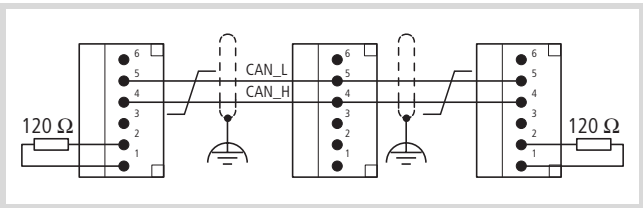


Figure 7: Example: network with bus termination resistor on XC200

Terminals 1 and 4 , 2 and 5 , 3 and 6 are internally connected.

Behaviour of the stations on the CANopen bus

Station/Bus monitoring: CAN telegrams are sent and received directly by the user program. An interruption on the CAN Bus will only be recognised when the respective CAN slave is monitored by the PLC (Nodeguarding function).

Start/Stop reaction: if you set the STOP position on the operating mode selector switch, all outputs of the decentral devices are set at the end of the cycle to "0".

Voltage switch on: The sequence in which the power supply of the individual CAN slaves is connected does not have an effect on the functionality of the CAN bus. Depending on the parametric programming, the PLC "waits" for the non-existent slave or starts it at the time at which the slave is interfaced to the CAN network.

Communication with CAN stations: The communication with the CAN stations and their configuration is described in the following application notes and operating manuals:

- Interfacing of an XION station to the XC100/200 via CANopen (AN27K18D)
- Communication between two controls using network variables via CANopen (AN27K19D)
- Coupling multiple autonomous controls (CAN-Device) via CANopen (AN27K20D)
- Engineering of CAN stations (AN27K27D)
- Library description: CANUser.lib/CANUser_Master.lib (AWB 2786-1554).

See also:

- Assignment of the CANopen/easyNet interface → page 19
- Properties of the CANopen cable → page 73

Supplementary functions of the CPU (local inputs)

The inputs I0.0 to I0.5 can be parameterised as:

- Incremental encoder inputs (I0.0 to I0.3)
- Counter inputs (32 bit I0.0, I0.1)
- Counter inputs (16 bit I0.0, I0.1 and I0.2, I0.3)
- Interrupt inputs (I0.4 and I0.5)

The input signals in the CPU are preprocessed with these functions.

Incremental encoder input (32 Bit)

The function is available once. On inputs I0.0 and I0.1 the incremental signals A and B of the encoder are directed to input I0.2 of the reference signal, which the encoder generates once per revolution. The switch is connected on input I0.3, which maps the reference window in the closed state in which the reference signal I0.2 is processed.

The incremental signals A and B are phase shifted by 90 degrees in order to indicate the count direction. The falling and rising edges are processed (4-fold evaluation). The maximum input frequency is 50 kHz. This results in an overall frequency of 200 kHz.

See also:

- Connecting the incremental value encoder → page 17
- Incremental encoder parameterisation → page 55

Up/down counter (32 Bit)

The function is available once. The counter input I0.0 accepts the impulses with a maximum frequency of 50 kHz. The directional signal on input I0.1 defines if the counter impulse is to be incremented or decremented when the counting pulse arrives. The direction signal is a static signal which must be present before the counting pulses. The count value is incremented/decremented with each counter value until the setpoint value is reached.

After the setpoint is achieved an interrupt is initiated which is used to branch to a programming routine/POU. The reaction after the setpoint is reached is determined by the direction of counting:

Incrementing: Count direction "up": If a setpoint value is achieved, the parameterized interrupt is activated. With the next counting pulse the counter begins at 0. The interrupt source is defined in the control configurator.

Decrementing: If a setpoint value is achieved, the parameterized interrupt is activated. When the next count pulse occurs, the counter begins to count at the preselected setpoint value. The interrupt source is defined in the control configurator.

See also:

- Interrupt processing → page 59
- Input of the setpoint value in the control configuration → page 57
- Connecting up/down counter → page 17

Up/down counter (16 Bit)

The function is available twice. It corresponds with the up/down counter (32 bit)

Inputs

Counter 1:

| | |
|------|-------------------|
| I0.0 | Pulse input |
| I0.1 | Directional input |

Counter 2:

| | |
|------|-------------------|
| I0.2 | Pulse input |
| I0.3 | Directional input |

Interrupt inputs

The digital inputs I0.4 and I0.5 can be parameterized as interrupt inputs. The leading or the lagging edge (can be parameterized) of the input signals are evaluated.

→ If an XC100 PLC is replaced by an XC200 PLC, the interrupt inputs are connected to other physical input addresses!

See also:

- Time constraints placed on the interrupt inputs: "Technical data – input delay – fast digital input" → page 81
- Programming in the "Interrupt function" is described on Page 59.
- Connecting up/down counter → page 17

2 CPU installation

→ Detailed information about the installation of the backplanes and XI/OC modules can be found in the manual "Hardware and Engineering XI/OC Signal Modules" (AWB2725-1452GB).

- ▶ Insert the loop on the bottom of the CPU module into the hole in the module rack ①.
- ▶ Press the top of the CPU module onto the module rack, until you hear it click into position ②.

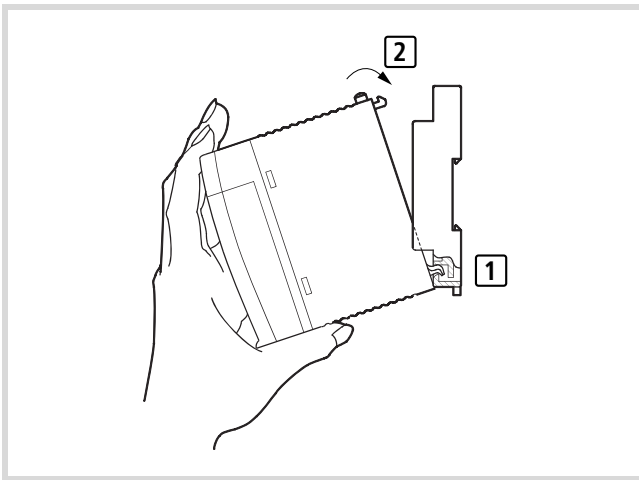


Figure 8: CPU installation

Detaching the CPU

- ▶ Press in the catch ①.
- ▶ Keep the catch pressed in, and pull the top of the CPU module forwards ②.
- ▶ Lift up the CPU module and remove it ③.

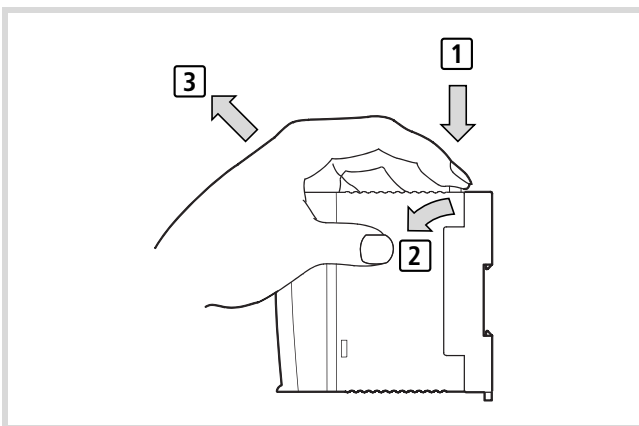


Figure 9: Detaching the modules

3 Engineering

Control panel layout

The layout of the components inside the control panel is a major factor for achieving interference-free functioning of the plant or machinery. During the project planning and design phase, as well as its implementation, care must be taken that the power and control sections are separated. The power section includes:

- Contactors
- Coupling/interfacing components
- Transformers
- Frequency inverters
- Converters

In order to effectively exclude any electromagnetic contamination, it is a good idea to divide the system into sections, according to their power and interference levels.

In small switchgear cabinets it is often enough to provide a sheet steel dividing wall, to reduce interference factors.

Ventilation

In order to ensure sufficient ventilation a minimum clearance of 50 mm to passive components must be observed. If the neighbouring components are active elements, such as power supplies or transformers, then the minimum spacing should be 75 mm. The values that are given in the technical data must be observed.

Layout of units

Build the module racks and the controls into the switchgear cabinet in a horizontal position:

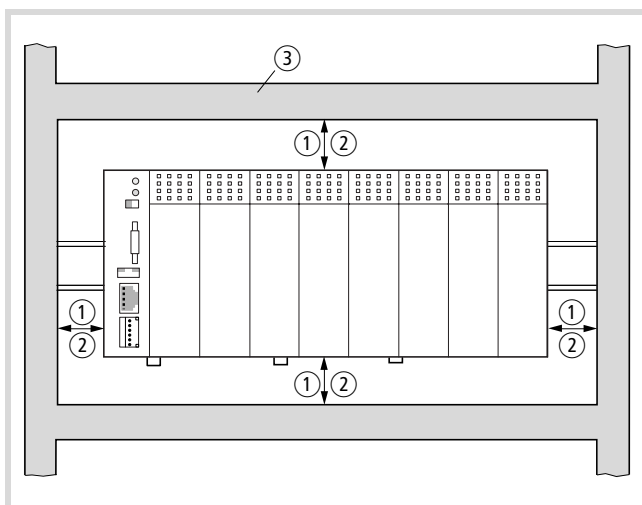


Figure 10: Control panel layout

- ① Spacing > 50 mm
- ② Spacing > 75 mm to active elements
- ③ Cable duct

Preventing interference

Cable routing and wiring

Cables are divided into the following categories:

- Electric power lines (e.g. power lines carrying high currents, or lines to converters, contactors, solenoid valves)
- Control and signal cables (e.g. digital input lines)
- Measurement and signal cables (e.g. fieldbus connections)

→ Always route power cables and control cables as far apart as possible. This avoids capacitive and inductive coupling. If separate routing is not possible, then the first priority must be to shield the cable responsible for the interference.

Take care to implement proper cable routing both inside and outside the control panel, to keep interference as low as possible:

- ▶ Avoid parallel routing of sections of cable in different power categories.
- ▶ As a basis rule, keep AC cable separated from DC cables.
- ▶ Keep to the following minimum spacing:
 - at least 10 cm between power cables and signal cables;
 - at least 30 cm between power cables and data or analog cables.
 - When routing cables, make sure that the outgoing and return leads of a circuit pair are routed together: The currents flowing in opposite directions thus cancel each other out as a summation,

Suppressor circuitry for interference sources

- ▶ All suppressor circuitry should be wired in as close to the source of interference (contactors, relays, solenoids) as possible.

→ Switched inductors should always have suppressor circuitry fitted.

Shielding

- ▶ Use shielded cables for the connections to the data interfaces. The general rule is: the lower the coupling impedance, the better the shielding effect.

Connecting inputs/outputs (CPU)

This example shows the connection of inputs/outputs and their power supply.

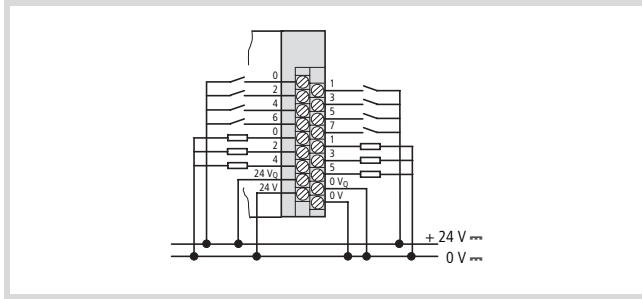


Figure 12: Connecting inputs/outputs to the CPU

→ You can find wiring examples for the XI/OC modules in the manual "Hardware and Engineering, XI/OC signal Modules" (AWB2725-1452GB).

Connecting the incremental value encoder

The incremental encoder is shown in the following figure in the manner in which it is to be connected to the control.

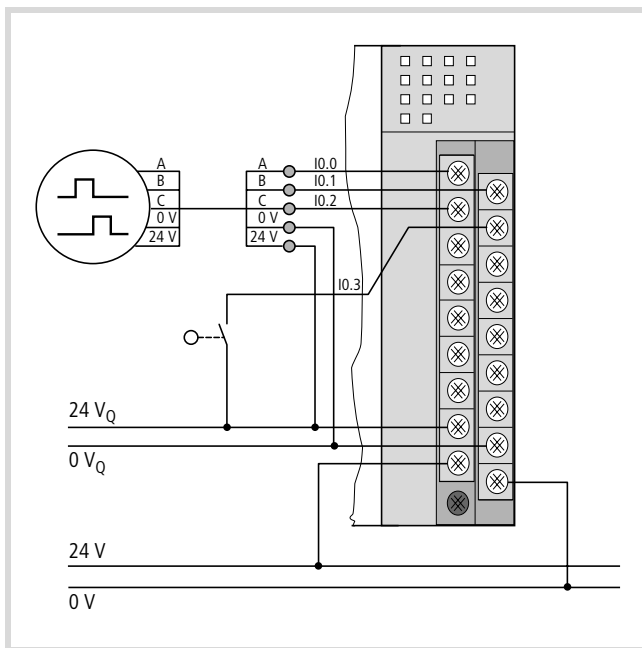


Figure 13: Connection of the incremental value encoder with a reference window switch

Connecting up/down counter

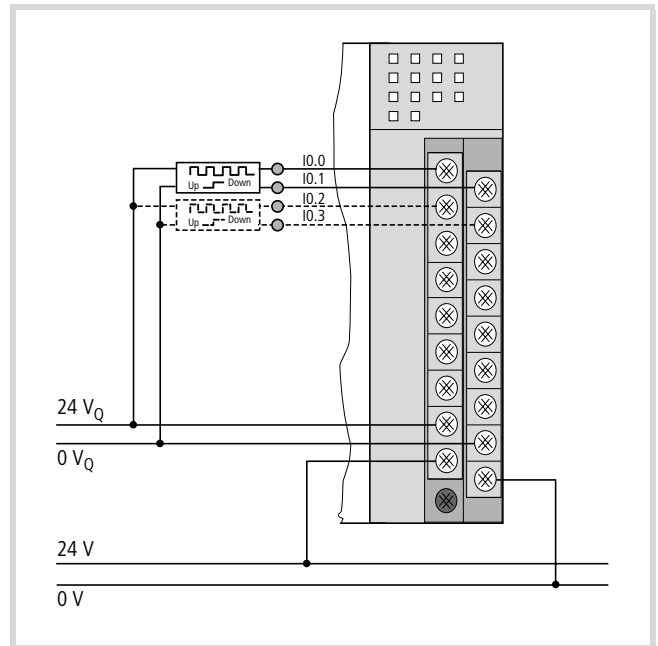


Figure 14: Connection of pulse encoder with signal for incrementing/decrementing

Connecting interrupt actuators

The inputs I0.4 and I0.5 can be parameterized as interrupt inputs.

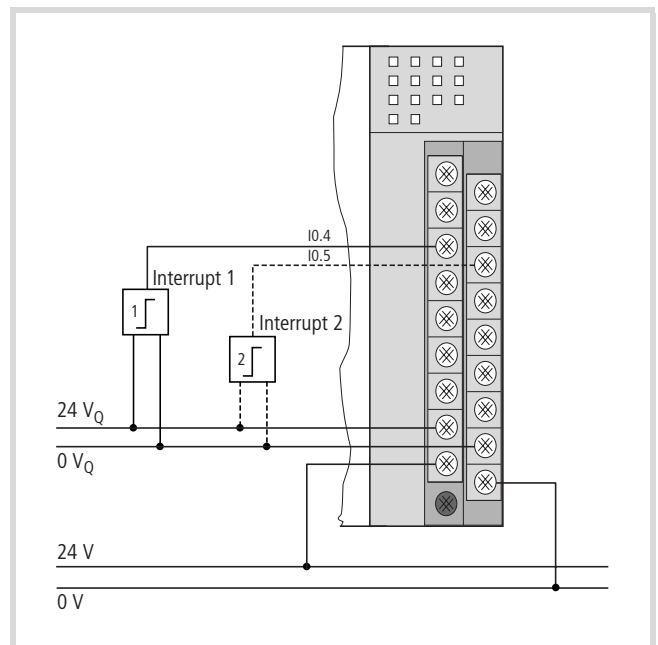


Figure 15: Interrupt input connections

→ Please note that when an XC100 PLC is replaced by an XC200 PLC the interrupt inputs are situated at other physical input addresses!

Connect PC

Ethernet connection

From a purely physical/mechanical point of view the programming devices interface is an RJ-45 interface (socket). Thus conventionally Ethernet (patch) cables with RJ-45 plugs can be used.

- Direct connection PC – XC200:

The XC200 can be connected directly to the (programming) PC via a crossover Ethernet cable, → figure 16, 17.

Crossover cables have the following design features:

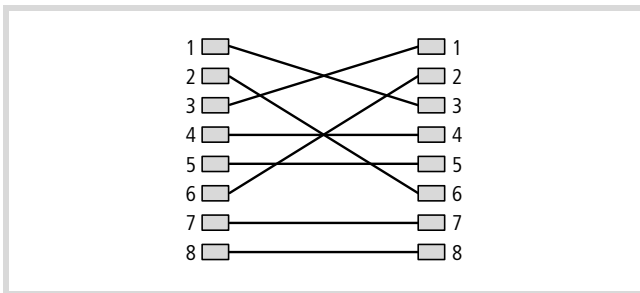


Figure 16: Connection set-up of a 8-pole crossover cable

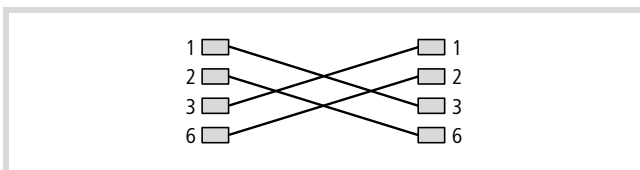


Figure 17: Connection set-up of a 4-pole crossover cable

The following cross-over cables are available:

- XT-CAT5-X-2 2 m long
- XT-CAT5-X-5 5 m long

- PC – XC200 via Hub/Switch connection:

If you use a Hub or a Switch between the PC – XC200 connection, you must use a standard Ethernet cable which is connected 1:1 for the connection between PC – Hub/Switch and Hub/Switch – XC200.

The following standard cables are available:

- CAT5-KG2.0 2 m long
- CAT5-KG5.0 5 m long
- CAT5-KG10.0 10 m long

→ Please note that when there is a double assignment of the RJ 45 interface with the RS 232 and Ethernet, the connections 4 and 7 are connected to "GND potential" because of the RS 232 interface. For this reason, we recommend the use of 4-core cables for the connection of the XC200 to the Ethernet.

See also:

- Characteristic of the Ethernet cable → page 73

RS-232 connection

Please use the XT-SUB-D/RJ45 programming cable to make the physical connection between the XC200 and PC.

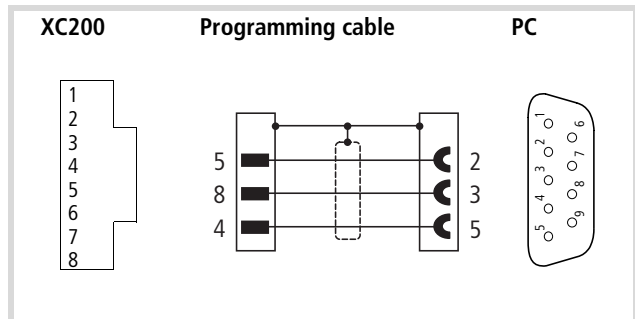


Figure 18: Pin assignment RS-232 programming cable

See also:

- Establishing a PC – CPU connection → page 41
- RS -232 interface in Transparent mode → page 51

Interface assignments

USB interface

Table 2: Assignment of the USB interface

| | Signal |
|---|--------|
| 1 | +5 V |
| 2 | USB– |
| 3 | USB+ |
| 4 | GND |

ETH232 programming interface


Table 3: Assignment of the programming interface

| RJ 45 | | Signal | |
|-------|---|--------|----------|
| | | RS232 | Ethernet |
| 8 | 8 | RxD | – |
| 7 | 7 | GND | 1) |
| 6 | 6 | – | Rx– |
| 5 | 5 | TxD | – |
| 4 | 4 | GND | 1) |
| 3 | 3 | – | Rx+ |
| 2 | 2 | – | Tx– |
| 1 | 1 | – | Tx+ |

1) Pin 4 and 7 may not be used

CANopen/easyNet interface

Table 4: Assignment of the CANopen/easyNet interface

| | Terminal | Signal | |
|---|----------|---------|---------|
| | | CANopen | easyNet |
|  | 6 | GND | GND |
| | 5 | CAN_L | ECAN_L |
| | 4 | CAN_H | ECAN_H |
| | 3 | GND | GND |
| | 2 | CAN_L | ECAN_L |
| | 1 | CAN_H | ECAN_H |

Connector type: 6-pole, plug-in spring-loaded terminal block, conductor cross-section up to 0.5 mm²
Terminals 1 and 4, 2 and 5, 3 and 6 are internally connected.

4 Operation

Start behaviour

Several different user programs/boot projects can be saved on the CPU. They can be located on the MMC as well as on the DISK_SYS system memory. However, the CPU simply runs a user program.

The following flow diagram indicates which program is used. The diagram shows the update of the operating system using the MMC.

After voltage recovery, a boot project saved in the XC200 will be started in accordance with the position of the operating mode selector switch and the programmed start conditions.

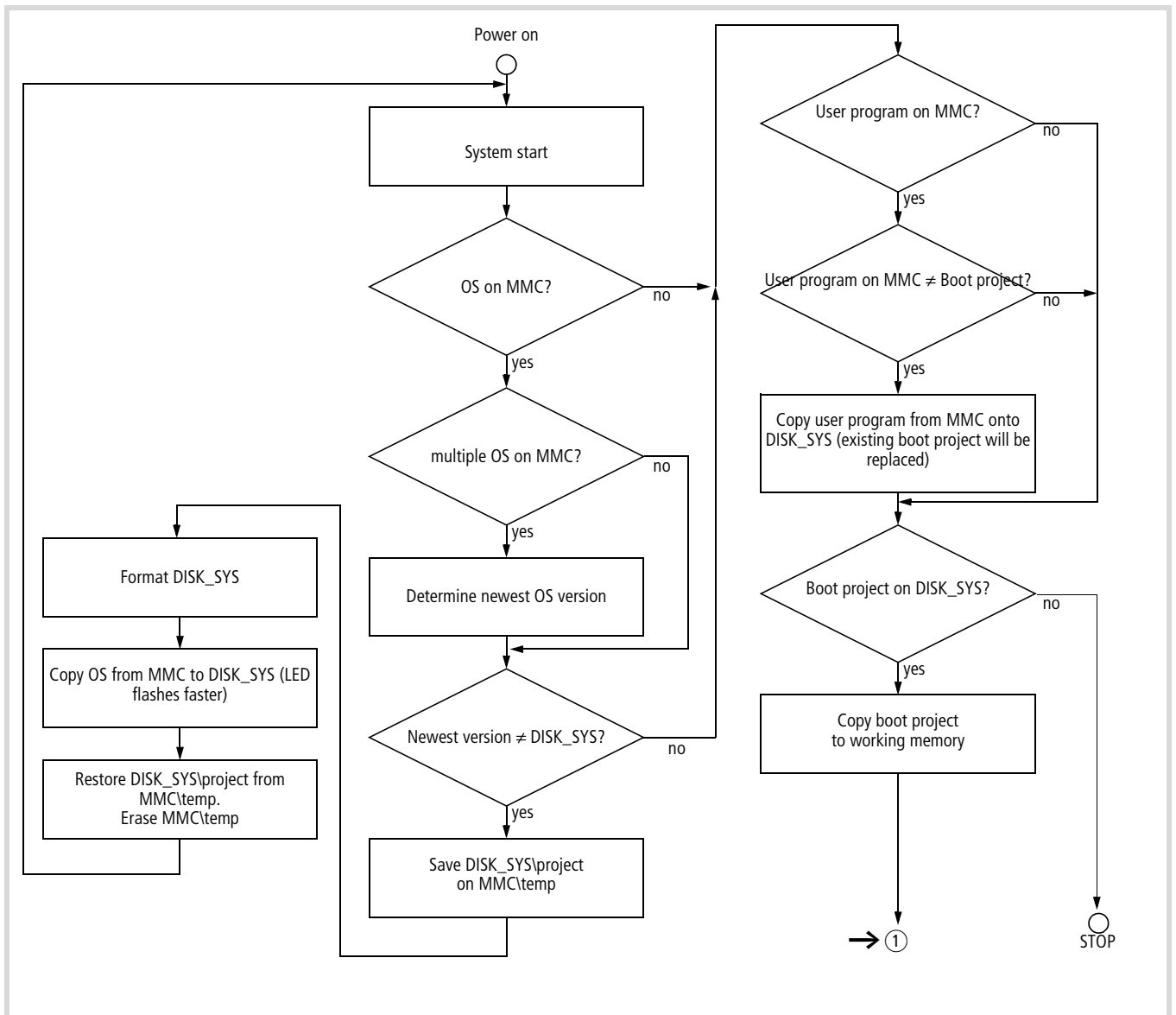
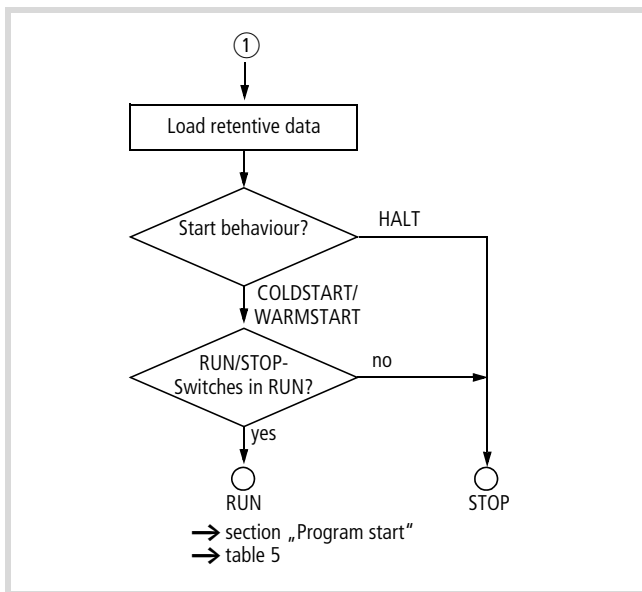


Figure 19: Start behaviour



Configuring the start-up behaviour with easySoft-DeCoSys

The start-up behaviour setting primarily defines the handling of the retentive variables. The following settings are only taken into consideration when the power supply is switched on.

Select one of the following start conditions in the "START BEHAVIOUR" drop-down menu in the "Other Parameters" tab of the PLC configurator.

- HALT
- COLDSTART
- WARMSTART

HALT

The application program is not started independently of the switch position of the RUN/STOP switch.

COLDSTART/WARMSTART

Precondition: The RUN/STOP switch is in the RUN position.

The variables are initialised in accordance with Table 5, before the control starts.

Table 5: Behaviour of the variables after COLDSTART/WARMSTART

| Variable type | Behaviour of the variables after ... | |
|----------------------------|--------------------------------------|----------------------------------|
| | COLDSTART | WARMSTART |
| Non-retentive | Activation of the initial values | Activation of the initial values |
| Retain¹⁾ | Activation of the initial values | Values remain in memory |
| Persistent | Activation of the initial values | Activation of the initial values |
| Retain Persistent | Values remain in memory | Values remain in memory |

1) Physical operands such as I, Q, M cannot be declared as "Retain" variables.

Program start

When a program starts, the CPU checks whether the configured inputs and outputs match the physically present ones. It also checks whether the actual module corresponds with the parameterized module type. If the wrong module type is identified, the CPU changes to NOT READY state.

Program start (STOP → RUN)

You have the following possibilities to start the program:

| | Program exists in main memory | Program should be loaded |
|---------------------------------|---|---|
| Prerequisite | <ul style="list-style-type: none"> • CPU in STOP • RUN/STOP switch in STOP | <ul style="list-style-type: none"> • CPU in STOP • RUN/STOP switch in RUN |
| Action | <ul style="list-style-type: none"> • Switch RUN/STOP switch to RUN or • in online operation, issue the "Start" command. | <ul style="list-style-type: none"> • Load program • in online operation, issue the "Start" command. |
| Result for all variables | CPU in RUN Values are retained at the start | CPU in RUN Initial values are activated |

Program stop (RUN → STOP)

A change of the RUN/STOP switch to the STOP position leads the CPU to the STOP state after completion of the program cycle (ending of all active tasks).

After the task has ended the outputs used by the I/O task are set to 0, → chapter „Program processing, multitasking and system times“ on Page 27.

You can stop the program in one of two ways:

- In online operation, issue the STOP command.
- Set the RUN/STOP switch in the STOP position.

Power off/Interruption of the power supply

The switch off/interruption of the (CPU) voltage during a running program leads to an immediate stop of the program cycle or task so that data integrity is no longer given!

All outputs in which the I/O tasks are used are set to 0 or switched off → chapter „Program processing, multitasking and system times“ to Page 27. The behaviour of retentive variables is shown in can be seen in Table 5.

The remaining program cycle will not be completed when power is reconnected!

If the data integrity is not practical for an application, additional measures should be engineered, such as e.g. the use of an uninterruptible power supply (UPS) with battery backup. The start of the control commences after Figure 19 (Start behaviour).

CPU operating state display

The operating state of the CPU is displayed on the RUN/STOP and SF LEDs:

| CPU status | RUN/STOP LED | SF-LED |
|------------|--------------|--------|
| RUN | ON | OFF |
| STOP | flashes | OFF |
| NOT READY | flashes | ON |

The NOT READY state is indicated by the RUN/STOP and SF LEDs. The PLC goes into this state when an error has occurred during the start. The CPU remains in STOP state. The CPU can be restarted after elimination of the fault.

Test and commissioning (Debugging)

The PLC supports the following test and commissioning features:

- Breakpoint/single-step mode
- Single cycle mode
- Forcing
- Online amendment, → Manual for programming software (AWB2700-1437GB), Section “Online Functions”
- Status indication/Powerflow.

Breakpoint/single-step mode

Breakpoints can be set within the application program. If an instruction has a breakpoint attached, then the program will halt at this point. The following instructions can be executed in single-step mode. Task monitoring is deactivated.



Attention!

Any outputs already set when the program reaches the breakpoint remain set!

Single-cycle mode

In single-cycle operation, one program cycle is performed in real time. The outputs are enabled during the cycle. At the end of the cycle, the output states are cancelled and the outputs are switched off. Task monitoring is active. Task monitoring is active.

Forcing

All variables of the user program can be forcibly set. A local output is only forced if the corresponding variable is forced and the CPU is in the RUN state.

Status indicator

The inputs/outputs are to be referenced in order to visualize the states of the configured inputs/outputs in an interval controlled task in the PLC configurator. The following syntax is sufficient in the ST programming language in order to be able to display individual I/O bits, e.g.:

```
%IB0; (referencing of inputs I0.0 - I0.7)
%QB0; (referencing of outputs Q0.0 - Q0.7)
```

in IL:

```
LD    %IB0
ST    Default byte
LD    Default byte
ST    %QB0
```

Reset

There are three different types of Reset commands:

- Warm reset
- Cold reset
- Full reset

Table 6: The commands also affect the state of the CPU: shows the commands to use for initializing a retentive variable range. The commands also affect the state of the CPU:

Warm reset

The program is stopped. The variables are initialised. The program can be restarted.

Cold reset

The program is stopped. The variables are initialised. The program can be restarted.

Full reset

The program in the PLC and the boot project are deleted. The variables are initialised. The PLC is set into the NOT READY state.

Table 6: Behaviour of the variables after a Reset

| Variable type | Reset command | | |
|----------------------|----------------------------------|----------------------------------|----------------------------------|
| | Warm reset | Cold reset | Full reset ¹⁾ |
| Non-retentive | Activation of the initial values | Activation of the initial values | Activation of the initial values |
| Retain ²⁾ | Values remain in memory | | |
| Persistent | Activation of the initial values | | |
| Retain Persistent | Values remain in memory | Values remain in memory | |

- 1) After a full reset, the program must be reloaded. In online operation, the "Start" command can now be issued.
- 2) Physical operands such as I, Q, M cannot be declared as "Retain" variables.

Programs and project

Loading the program

You must log on in order to load recently created or modified programs. The question "Load the new program?" will appear. The load operation will start once this prompt has been confirmed.

→ Please note that the "Retain" variables are initialised during the load process, but the "PERSISTENT" variables retain their value.

Program download is monitored. After the default transfer time is exceeded, communication ends and the error message: "Communications fault (#0). Logging out."

This happens if the programs are very large or if the number of "Persistent" variables and/or "Retain-Persistent" variables are greater than 5000. The number is independent of the data type.

The transfer time can be extended to 30000 ms to eliminate this problem:

- Open the "ECP" file in the "XSoft V.." folder with an editor and incorporate the following instruction: DownloadWaitTime=30000

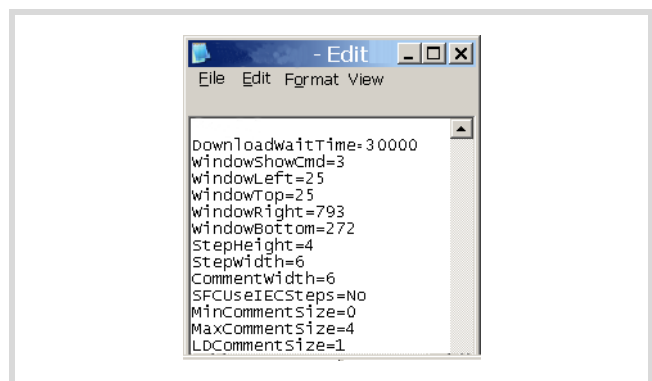


Figure 20: Addition of the DownloadWaitTime

Disadvantage: If a further error message occurs it will also require this time before it is displayed.

The input of the transfer time is made in ms. Here 30000 ms has been entered as an example.

In order to safely store the program, a boot project must be generated by the user program. With the "Create boot project" command the program is loaded from the PC into the system memory and saved as a zero-voltage safe boot project.

The following steps are necessary in order to create a boot project:

- Change over to the "Online" folder.
- Select the "Login" command.
- Select the "Create boot project" command.

Saving boot project on MMC

- Click on the folder Ressources → PLC Browser and enter the "copyprojtommc" command.

The boot project is stored on the MMC in the sub-directory "project" under the name "Default.prg". Furthermore a "Default.chk" file is generated.

You can copy the boot project with the browser commands "filecopy" or "filerename" (e.g. as a backup copy) and change the name of the file. In the easySoft CoDeSys however only the boot project with the name "Default" is active.

Erase boot project on MMC

Click on the folder Ressources → PLC Browser and enter e.g. for the XC-CPU201-EC256K the command.

```
filedelete \\disk_mmc\\MOELLER\\XC-CPU201-EC256K-8DI-6DO\\
project\\default.t.prg
```

Updating the operating system

On the XC200 it is possible to replace the operating system (BTS) by a more up-to-date operating system. Moeller offers the respective current operating system version on the Internet at: <http://www.moeller.net/support>.

→ If you transfer a current operating system to an older hardware version, it is possible that not all functions of the operating system will be supported by the hardware.

You have two choices available to transfer the operating system (OS).

- Directly from the PC into the PLC (only via RS 232) → page 25
- From the PC via the PLC on the MMC into the "disk_mmc\moeller\XC-CPU201\" folder → page 26

The second variant is only possible for an Ethernet connection! Furthermore, the PLC must feature an OS version 01.03.00 or higher.

Transferring the operating system from the PC to the PLC

→ If an operating system (OS) is loaded into the PLC, the existing operating system (OS) as well as the user program are deleted.

Procedure:

- Establish a serial connection via the RS-232 interface of the PC with the XC200. Information on this is provided in the sections "Connect PC" on Page 18 and "Establishing a PC – CPU connection" on Page 41.

→ The Baud rate is set to a fixed value of 115200 Bit/s for loading the operating system.

- Activate in easySoft-CoDeSys in the "Other Parameters" tab in the "PLC Configuration" window and click on the "Start" button.

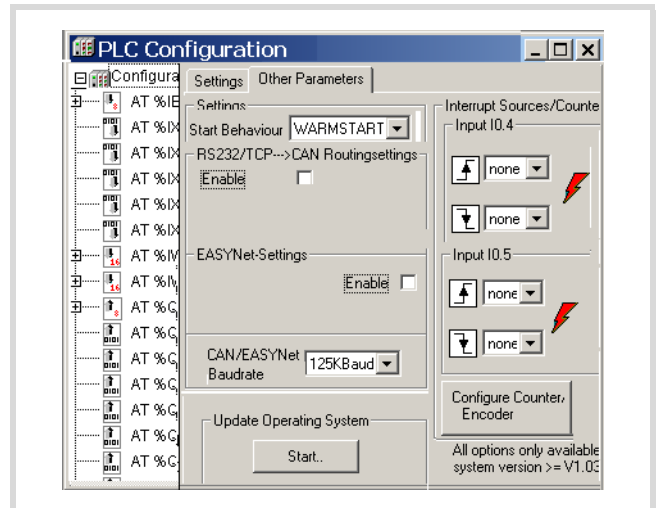


Figure 21: Updating the operating system (OS) of the XC200

The "Download Tool" window opens.

- Click on the "Open" button and enter the path in which the update of the operating system is located.

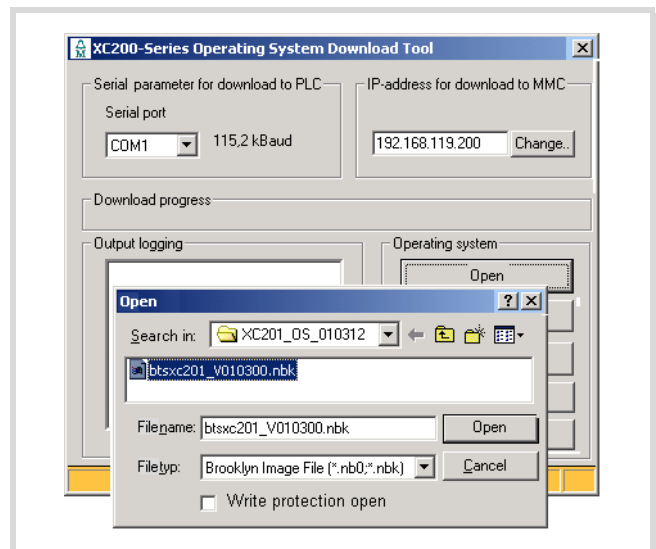


Figure 22: Operating system selection

- Opening of the operating system file to be transferred.

The following window appears:

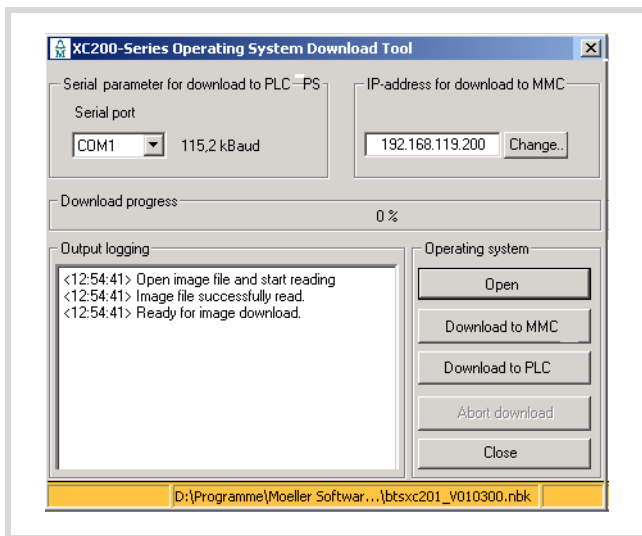


Figure 23: Download of the XC200 operating system

- Click on the "Download image" button.

The "Connect to target PLC Please reboot the PLC now" message appears.

- Switch off the control voltage of the XC200 and wait a few seconds. This will ensure that the residual voltage is discharged.
- Switch the control voltage of the XC200 back on.

The transfer of the operating system into the XC200 is started. It can take a few minutes. Please observe the signal states of the operating LEDs:

The red "SF" LED lights up during the transfer. When the download progress display indicates 100 %, the SF display turns off after a short delay. After about 1 minute it will light up again and the green RUN/STOP LED flashes. The waiting time results from the programming of the internal Flash memory (comparable with booting a PC).

Further inputs appear on the download window. The progress of the download is also indicated by the status bar on the transfer field.

Please do not engage in the download process until the green LED flashes and "Ready for operating system transfer" appears for a second time on the download window. The download is only complete after both attributes have appeared.

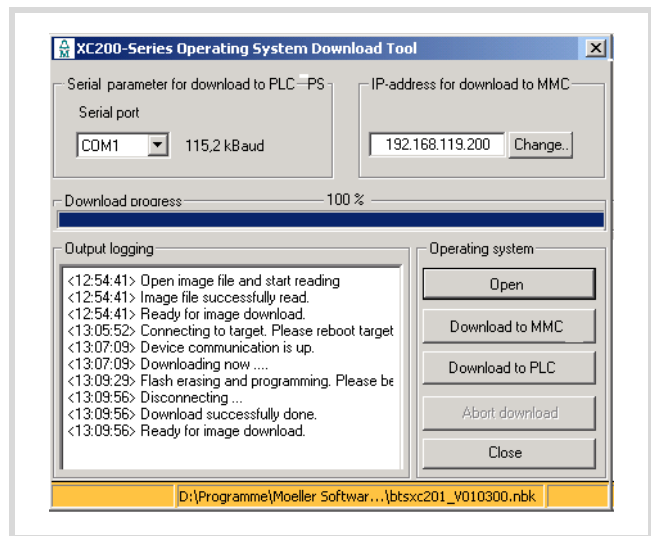


Figure 24: Download of the XC200 operating system ended

- End the download with the "Close" button.

Transferring the OS from the PC into the MMC

PC → MMC

The process operates analogue to the transfer of the OS from the PC to the PLC. Simply click on the button "Transfer OS to MMC" (see Figure 23).

MMC → PLC

If the OS of a PLC is to be updated via the MMC, the PLC must feature an OS with version 01.03.00. The OS is updated during the switch on process → figure 19 on Page 21.

Erase operating system/boot project from the MMC

You can delete the operating/boot project system from the PC, e.g. with Internet Explorer.

- Establish a connection to the XC200 via the default address `ftp://192.168.119.200`.
- Open the "disc_mmc\moeller\XC-CPU201" directory.

All the operating system files are stored in this directory and can be deleted there.

5 Program processing, multitasking and system times

Task configuration

Processing of the project can be controlled via tasks. Each task can be assigned with a range of programs which should be run during execution of the task..

The task is defined by a name, a priority and a type which defines under which conditions a task starts. Task condition and priority determine the sequence in which the tasks are to be processed.

Task priorities can be determined as "cyclic" or "event triggered". A cyclic task will be restarted after the parameterized delay time has expired. An event-triggered task is only started if the event occurs. In addition, you also have the option of coupling system events, such as "start", "stop" or "reset" to the execution of a program.

The task priorities can be parameterized with a value from 0 to 31 where 0 is the highest priority and 31 is the lowest priority.

In principle the output map is written onto the physical outputs before every task is called and the map is read by the inputs (updating of the input/output map). The task is executed thereafter. In addition, all system activities are carried out before or after the task call. This includes for example, communication with the easySoft CoDeSys, Online changes etc....

Updating of the input/output map by multiple tasks is described in the section "Multitasking" on Page 29.

All IEC tasks, including those with the highest priority can be interrupted by an interrupt or an event controlled task.

Time monitoring can be activated for each task (Watchdog).

An extensive description of the task configuration can be found in the programming software manual (AWB2700-1437GB). At the end the control specific settings are explained on the basis of an example.

Creating task (example)

First create the cyclic task "Basic" with the assigned program "Basic_prog".

Then you can add the event controlled task "Param" with the program "Param_prog".

In the program "Basic_prog" an event is programmed which invokes the "Param" task.

The following steps are necessary in order to create a task:

- Add a task
- Define the program call
- Create the program

Creating the "Basic" cyclic task

- Open the "Task configuration" folder in the "Resources" tab
- Click with the right mouse button on the "Task configuration" folder and select the "Add task" command in the popup menu.
- Enter in the Name field a name such as "Basic".
- Parameterise the task in the dialog window to Figure 25.
- Click on the "Task configuration" folder and the configuration is accepted.

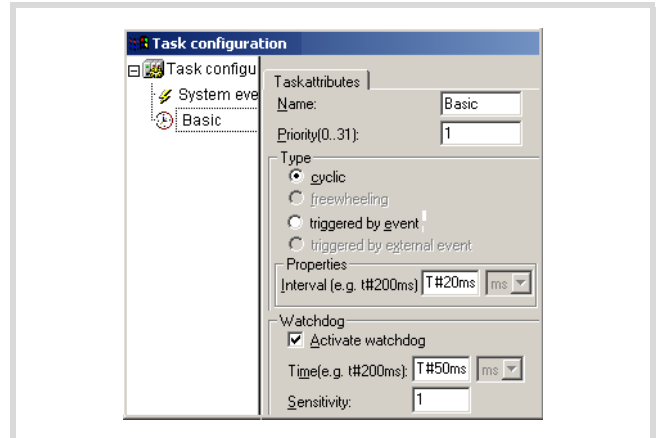


Figure 25: Parameterization of the cyclic task

Define the program call

With the program call you define which program is to be called with the task "Basic".

- Click with the right mouse button on the clock symbol of the "Basic" task created beforehand and select the "Program call" command in the popup menu.
- Enter the name "Basic_prog" in the "Program call" window.

- Click on the button at the end of the input field and confirm the program name in the "Entry help" window.

Writing a program

- Change over to the "Global Variables" tab and click with the right mouse button on the default program element PLC_PRG and select the "Rename object" command. Designate the element as "Basic_Prog".
- You can enter a program. In the programming example (Figure 26) the "count" variable is incremented. On counter status = 9, a = TRUE.

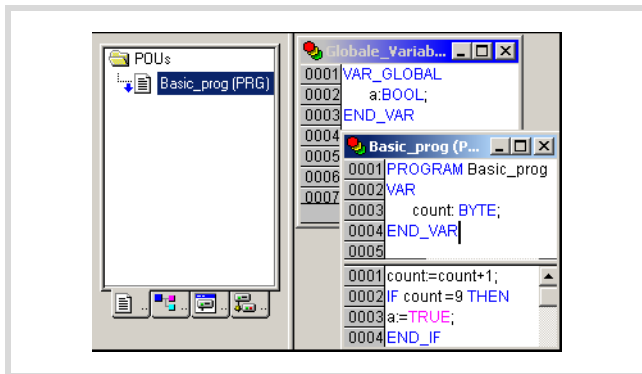


Figure 26: Creating a POU for a cyclic task

Creating event controlled task "Param" and defining the program call

The procedure corresponds to the creation of a cyclic task.

- Create a task of the "event controlled type" with the name "Param" in accordance with Figure 27.
- Define the Boolean variable "a" as the result of the event.
- Enter the program call "Param_prog".

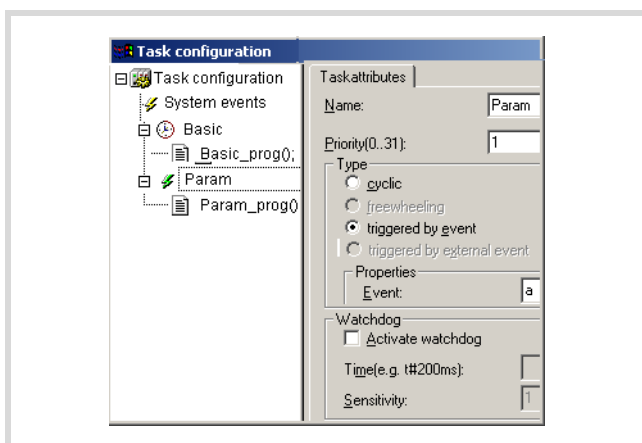


Figure 27: Creating an event controlled task

Writing a program

- Change over to the "Global Variables" tab and insert an object (POU) with the name "Param_prog".
- You can enter a program. In the sample program Param_prog (Figure 28), the variable "value" will be increased by the value 1. The program Param_prog will be executed when variable a = TRUE.

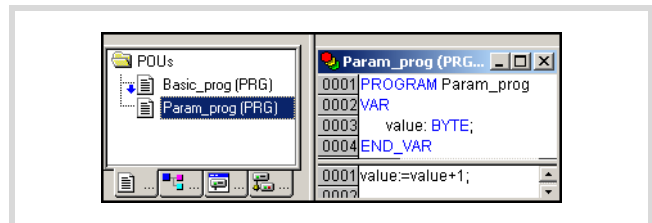


Figure 28: POU for event-triggered task

System events

A POU can be called with the help of a system event (state change of the CPU → STOP/ → START). It can be used when the PLC is started to initialise modules with parameters. The system events are independent of the task!

Assigning a POU to a system event

- Activate the event in task configuration under "System - events", e.g.: "Start" and enter the name of the POU (e.g. Power_prog) that is to be executed.

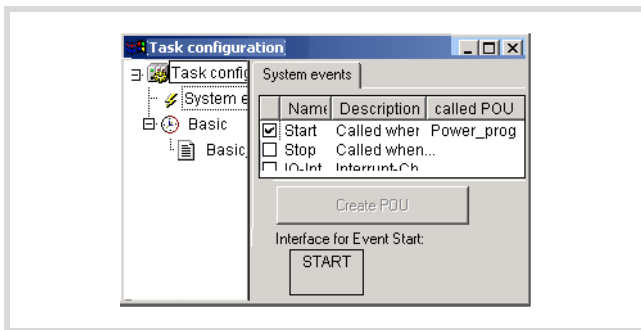


Figure 29: Assigning the POU to a system event

- Change over to the "Resources → Global variables" and add the object (POU) "Power_prog".
- Program the application:

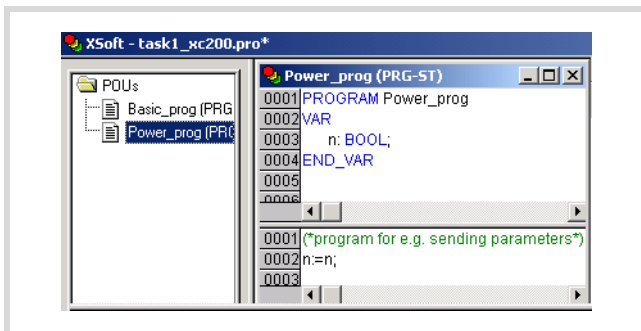


Figure 30: Programming a POU

→ Further information concerning the system events can be found in the programming software manual (AWB2700-1437GB) and in the online help of the programming system.

Multitasking

The XC200 run time system is a multitasking system. This means that multiple-tasks can be practically performed at the same time (in parallel).

Updating the input/output maps

If the local and central inputs/outputs are programmed in several tasks, an update (refresh of the input/output level) of the input/output map is performed according to special rules:

In the first task, the system begins to search for programmed inputs, e.g. after start up. The term "first task" relates to the first task in the task configuration, regardless of the priority and the cycle time of the individual tasks. In Figure 32 the name of the first task is "Prog1". If the system recognises an input coupled with an input module through the configuration, e.g. XIOC-16DI, then all inputs of this module will be updated in the image. If additional inputs are available in this task assigned to other modules, then the inputs of these modules will also be updated (modular update procedure), e.g. inputs %IX6.0 and %IX7.1 of input module "1" addressed by various tasks, the inputs of these modules will only be updated by the 1st task.

Examples

The examples are based on the following configuration:

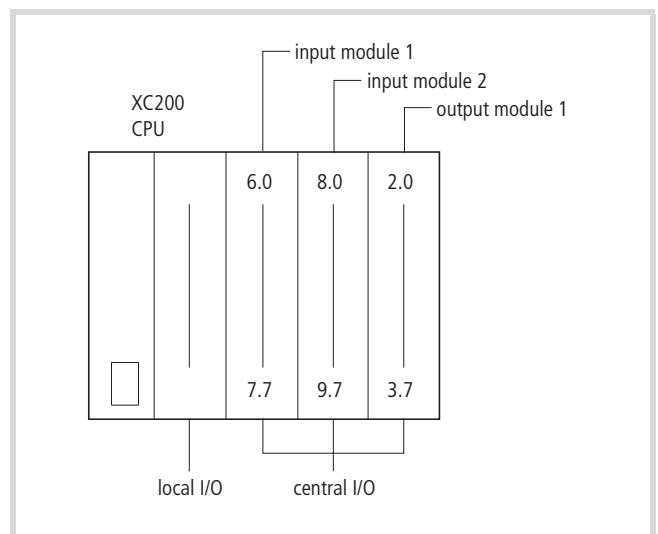


Figure 31: XC200 configuration

The task configuration appears as follows:

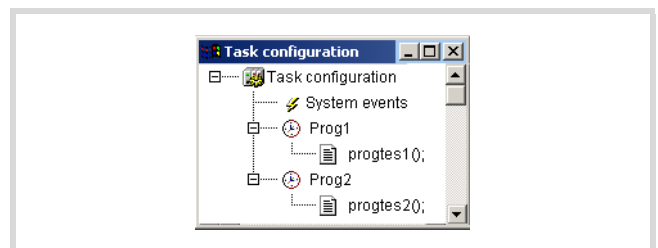


Figure 32: Task configuration for the examples

Example 1:

Table 7: Task details for example 1

| Task name | Priority | Cycle time |
|-----------|----------|------------|
| Prog 1 | 2 | 50 ms |
| Prog 2 | 1 | 10 ms |

In the first task "Prog1", the inputs %IX1 of input module 6.0 and %IX1 of input module 8.3 are programmed in the program "progtes(2)". Before the start of the first task "Prog1" the inputs of these modules are updated.

In the second task "Prog2" the input %IX2 of input module 7.1 is programmed in the program "progtes(1)". Before the start of the 2nd task the inputs of this input module are not updated, as this only occurs in the 1st task.

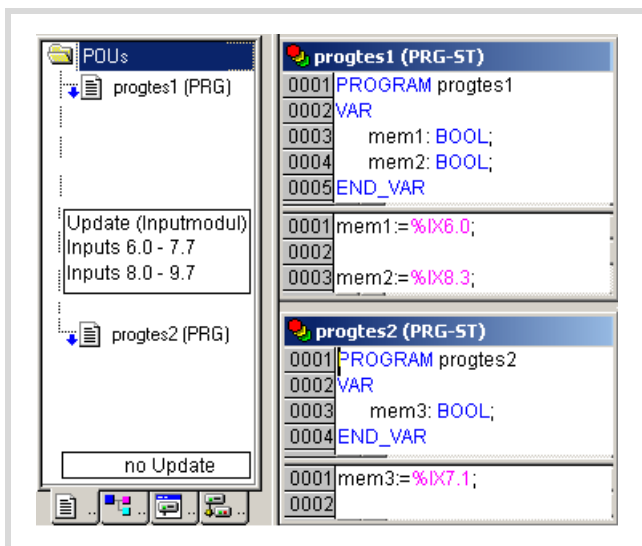


Figure 33: Program for example 1

Example 2:

Table 8: Task details for example 2

| Task name | Priority | Cycle time |
|-----------|----------|------------|
| Prog 1 | 2 | 50 ms |
| Prog 2 | 1 | 20 ms |

In example 2 in the first task the input 6.1 is programmed and in the second task the input 8.4 and output 3.4 is programmed. At the start of the first task an update of the inputs 6.0 to 7.7 of input module 1 occurs.

At the start of the second task, the inputs 8.0 to 9.7 of input module 2 follow as well as the outputs 2.0 to 3.7 of output module 1.

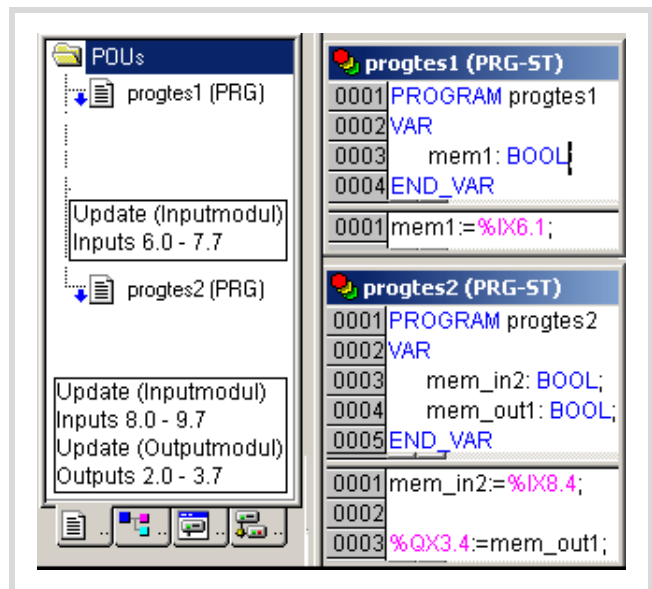


Figure 34: Program for example 2

Creating a task with consistent I/O

Avoid access to the physical outputs from several tasks. In order to guarantee a clear PLC sequence, create for the local/central inputs/outputs a task in which all inputs are copied in global variables and at the end of the interval all outputs of global variables are written to the output module (I/O update task). The I/Os are consistent (data integrity) within this task. The global variables can then be used instead of the I/Os in other tasks.

- On the XC200 PLC a maximum of 10 tasks are possible.
- The parameterisation of a task as "free wheeling" is not supported.
- Note with parametric programming of the watchdog time that the POU called with the interrupt service routines, extends the task run times accordingly.

Behaviour of the CAN stack with multitasking

A CAN stack call occurs before every task in which the CAN variables are used. A multitasking system can contain individual tasks which can be interrupted as required according to their priority. This behaviour can lead to an inconsistency in the CAN stack when it is called by a higher priority task, before the CAN stack has been processed by the interrupted task.

- The CAN stack of the XC200 does not have multitasking capability.
- Only a single user task in which CAN variables are used can be created.

Task monitoring with the watchdog

The processing time of a task can be monitored in terms of time required using a watchdog. The following applies for defining the monitoring time:

Processing time < Interval time of the task < Watchdog(time)

If the processing time exceeds the interval time, the end of the second interval time is awaited until the task is restarted. -> Watchdog deactivated.

The watchdog interrupts the program processing if the processing time of the task exceeds the watchdog time.

Furthermore, the frequency (sensitivity) can be set, which the number of exceeds allows. In this case the outputs of the PLC are switched off and the application program is set to the "Halt" state. Afterwards, the user program must be reset with RESET.

→ If the watchdog is deactivated, task monitoring does not occur!



Warning!

If you want to parameterize a task without a Watchdog or want to deactivate the Watchdog at a later time, all the outputs which have been accessed up to this time can continue to remain active. This is the case for example, when the task can't be ended due to a continuous loop (programming error) and/or missing end condition. These outputs continue to retain their "High potential" until the operating mode is changed from RUN to STOP or until the control voltage for the outputs is switched off.

Watchdog configuration

You can preselect the following settings in the task configuration:

- Watchdog on/off
- Watchdog time
- Watchdog sensitivity

These settings apply for time controlled and event controlled tasks.

Watchdog active

The watchdog is started at the commencement of every processing cycle and reset again at the end of the task.

→ The following rule applies for definition of the watchdog time with several tasks: each watchdog time must be longer than the sum of task interval times.

If the processing time is longer than the watchdog time (sensitivity = 1) – e.g. with a continuous loop in a program – the watchdog becomes active. If the processing cycle is shorter than the watchdog time, the watchdog is not activated.

The triggering of the watchdog continues to be dependant on the watchdog sensitivity. The **watchdog sensitivity** determines when the watchdog will be triggered, after the watchdog time has been exceeded by a determined number of consecutive occasions.

The watchdog is triggered:

- immediately when the watchdog time is exceeded with a watchdog sensitivity of "1".
- immediately after the "x"th consecutive time that the watchdog time is exceeded with a watchdog sensitivity of "x".

For example, a task with a watchdog time of "10 ms" and a watchdog sensitivity of "5" will end at the latest after $10 \text{ ms} \times 5 = 50 \text{ ms}$.

Example: Watchdog active

The interaction of interval time (IZ), task run time (TZ), watchdog time (WT) and watchdog sensitivity are illustrated by the following configuration example:

- Watchdog on
- Watchdog time (WT) = 15 ms
- Watchdog sensitivity = 2

The interval time (IZ) of the task is 10 ms.

Variant ①: The watchdog is not triggered as the task time always remains below the defined watchdog time.

Variant ②: The watchdog is triggered 15 ms after commencement of the second interval ⚡, as both times are longer than the defined watchdog time and occur consecutively.

Variant ③: The watchdog is triggered 15 ms after commencement of the second consecutive task, which is longer than the defined watchdog time.

Variant ④: Endless loop: The watchdog is triggered ⚡, because the task time takes longer than the watchdog time multiplied by the watchdog sensitivity ($15 \times 2 = 30$ ms).

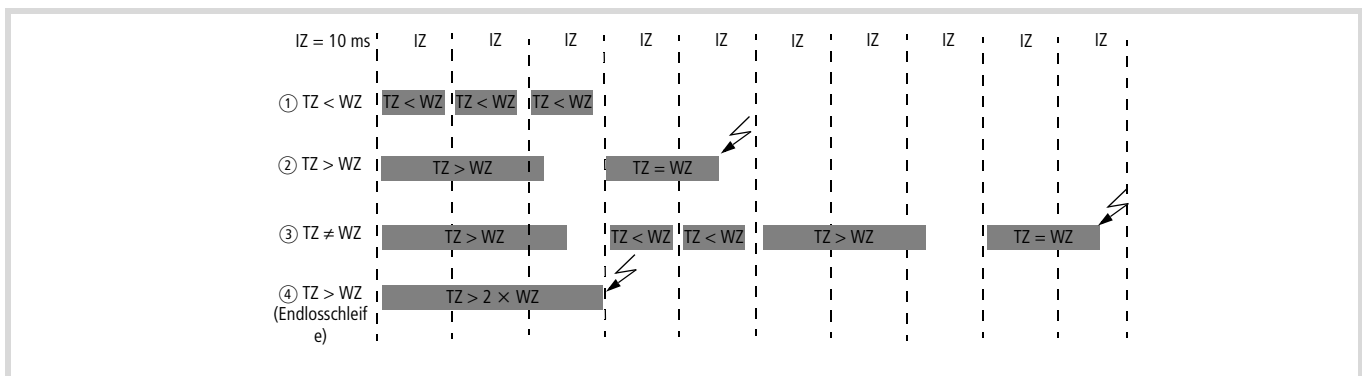


Figure 35: Watchdog active, multiple tasks with differing priority

Watchdog deactivated

The execution time of a task is not monitored when the watchdog is deactivated. If a task has not ended within the preselected interval time when the watchdog is deactivated, this task will not be called or started in the following cycle. A task is only started again if it has been ended in the previous cycle.

Example: Watchdog deactivated

The interval time (IZ) is 10 ms.

Variant ①: The interval time (IT) of a task was set to 10 ms. The actual task time (TT) is 15 ms. The task is started on the first call but is not terminated before the second cycle. Therefore, the task is not started again in the second cycle. Only in the third cycle – after 20 ms – is it possible to restart the task. The task does not run every 10 ms but rather only at a time interval of 2×10 ms.

Variant ②: The running cycle is not ended.

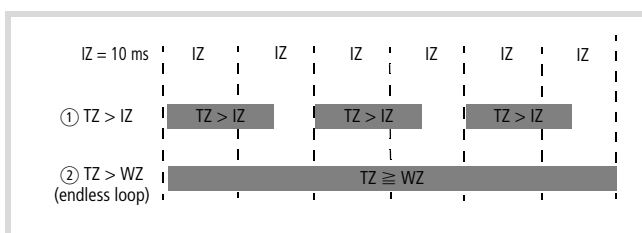


Figure 36: Watchdog deactivated

Multiple tasks with the same priority

You can assign several tasks with the same priority. The tasks are split according to the "Time Slice" principle and are practically executed simultaneously as part intervals (Round Robin).

Direct peripheral access

The "Direct peripheral access" function enables access directly to the local and central input and output signals of the control. The I/O access does not occur via the input/output image. The local and central input and output signals you can find the input and output signals of the CPU and the centrally expanded XC-200 control with the XI/OC signal modules. XI/OC signal modules which can be integrated via a bus system can't be accessed via the "Direct peripheral access".

Addressing is dependent on the slot number "0 to 15" of the signal modules. Further differentiation within the slot exists and relates to bit number "0 to max. 63" of the Inputs/Outputs.

Depending on the functionality of the XI/OC signal modules, access occurs as a bit/word or read/write operation. The access parameter indicates the Table 9.

The inputs/outputs which are required for "Direct peripheral access" are physically connected in the same manner as normal inputs/outputs.

Table 9: "Direct peripheral access" overview

| Module | I/O bit access | | | I/O word access | | | I/O slot |
|-----------------------------|----------------|-------|------------------------|-----------------|-------|-----------------------------|----------|
| | Read | Write | Param./Module | Read | Write | Param./Module | |
| XC-CPU201-EC256K-8DI-6DO | ✓ | ✓ | DI: 0 to 7, DO: 0 to 5 | ✓ | ✓ | 0 | 0 |
| XC-CPU201-EC256K-8DI-6DO-XV | ✓ | ✓ | DI: 0 to 7, DO: 0 to 5 | ✓ | ✓ | 0 | 0 |
| XC-CPU201-EC512K-8DI-6DO | ✓ | ✓ | DI: 0 to 7, DO: 0 to 5 | ✓ | ✓ | 0 | 0 |
| XC-CPU201-EC512K-8DI-6DO-XV | ✓ | ✓ | DI: 0 to 7, DO: 0 to 5 | ✓ | ✓ | 0 | 0 |
| XIOC-8DI | ✓ | – | 0 to 7 | ✓ | – | 0 | 1 to 15 |
| XIOC-16DI | ✓ | – | 0 to 15 | ✓ | – | 0 | 1 to 15 |
| XIOC-16DI-AC | ✓ | – | 0 to 15 | ✓ | – | 0 | 1 to 15 |
| XIOC-8DO | – | ✓ | 0 to 7 | – | ✓ | 0 | 1 to 15 |
| XIOC-16DO | – | ✓ | 0 to 15 | – | ✓ | 0 | 1 to 15 |
| XIOC-16DO-S | – | ✓ | 0 to 15 | – | ✓ | 0 | 1 to 15 |
| XIOC-12DO-R | – | ✓ | 0 to 11 | – | ✓ | 0 | 1 to 15 |
| XIOC-16DX | – | ✓ | 0 to 15 | ✓ | ✓ | 0 | 1 to 15 |
| XIOC-8AI-I2 | – | – | – | ✓ | – | 0 to 7 | 1 to 15 |
| XIOC-8AI-U1 | – | – | – | ✓ | – | 0 to 7 | 1 to 15 |
| XIOC-8AI-U2 | – | – | – | ✓ | – | 0 to 7 | 1 to 15 |
| XIOC-4T-PT | – | – | – | ✓ | – | 0 to 3 | 1 to 15 |
| XIOC-4AI-T | – | – | – | ✓ | – | 0 to 3 | 1 to 15 |
| XIOC-2AO-U1-2AO-I2 | – | – | – | – | ✓ | 0 to 3 | 1 to 15 |
| XIOC-4AO-U1 | – | – | – | – | ✓ | 0 to 3 | 1 to 15 |
| XIOC-4AO-U2 | – | – | – | – | ✓ | 0 to 3 | 1 to 15 |
| XIOC-2AO-U2 | – | – | – | – | ✓ | 0 to 1 | 1 to 15 |
| XIOC-4AI-2AO-U1 | – | – | – | ✓ | ✓ | AI: 0 ... 3/ AO: 0 ... 1 | 1 to 15 |
| XIOC-2AI-1AO-U1 | – | – | – | ✓ | ✓ | AI: 0 ... 1/ AO: 0 | 1 to 15 |

| Module | I/O bit access | | | I/O word access | | | I/O slot Param. |
|-------------------|----------------|-------|---------------|-----------------|-------|---------------|--------------------|
| | Read | Write | Param./Module | Read | Write | Param./Module | |
| XIOC-1CNT-100KHZ | — | — | — | — | — | — | 1 to 15 |
| XIOC-2CNT-100KHZ | — | — | — | — | — | — | 1 to 15 |
| XIOC-2CNT-2AO-INC | — | — | — | ✓ | ✓ | — | 1 to 15 |
| | — | — | — | — | — | — | |
| XIOC-NET-DP-M | — | — | — | — | — | — | 1 to 3 |

ReadBitDirect

A bit of an input module can be read directly with this function. The state of an input bit is stored in the variables, which indicate to the parameterized pointer "ptr_xValue". The pointer variable will not be changed when a fault occurs during processing.

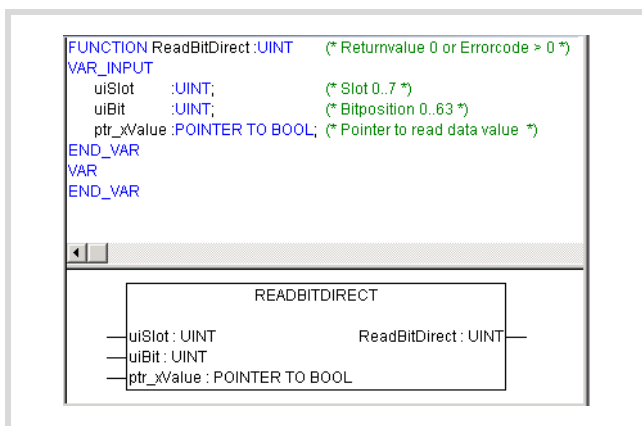


Figure 37: ReadBitDirect function

Parameters

| | |
|---------------|--|
| uiSlot | Slot number of the signal module. For possible parameters see Table 9 on Page 33. |
| uiBit | Bit position within the input value of the signal module. For possible parameters see Table 9 on Page 33 |
| ptr_xValue | Pointer to the variable value |
| ReadBitDirect | Display of the fault code, see Table 10 on Page 36 |

ReadWordDirect

A word of an input module can be read directly with this function. The state of an input word is stored in the variables, which indicate to the parameterized pointer "ptr_wValue".

The pointer variable will not be changed when a fault occurs during processing.

Parameters

| | |
|----------------|--|
| uiSlot | Slot number of the signal module. For possible parameters see Table 9 on Page 33 |
| uiOffset | Word offset within a signal module. For possible parameters see Table 9 on Page 33 |
| ptr_wValue | Pointer to the variable value |
| ReadWordDirect | Display of the fault code, see Table 10 on Page 36 |

ReadDWordDirect

With this function you can directly read a double word of an input module or an input function such as a counter value of the 32 bit counter. The state of the double word is stored in the variables, which point to the parameterized pointer "ptr_dwValue".

The pointer variable will not be changed when a fault occurs during processing.

Parameters

| | |
|-----------------|--|
| uiSlot | Slot number of the signal module. For possible parameters see Table 9 on Page 33 |
| uiOffset | Word offset within a signal module. For possible parameters see Table 9 on Page 33 |
| ptr_dwValue | Pointer to the variable value |
| ReadDWordDirect | Display of the fault code, see Table 10 on Page 36 |

Write...Direct

Fundamentally, the outputs of the PLC should only be modified by a task or an interrupt. Always work within interrupts with direct access functions as the events do not have an image.

If outputs from various tasks or events are modified in an application, the following rules should be observed:

- If an output bit with the "WriteBitDirect" function is processed with an event (interrupt or event task), the "Q-WORD" output word in which the bit is situated, may not be referenced to any other task! The other bits of the output word may still be assigned in other tasks as the "Q-BOOL" output bit.
- If an output bit is modified for fast processing with the "WriteBitDirect" functions, and this bit is also processed at another location (task, event or interrupt), the "WriteBitDirect" function must be used at all locations (no Q-BOOL declaration and no referencing).

Example

Output variable declaration:

| | |
|----------------------|----------------|
| Q-BOOL (e.g. Qbit3) | AT%QX1.2:BOOL; |
| Q-WORD (e.g. Qword0) | AT%QW0:WORD; |

Referencing (assignment in the application):

| |
|------------------|
| Qbit3:=TRUE; |
| Qword0:=16#Test; |

WriteBitDirect

A bit of an output module can be controlled directly with this function. The respective output image is refreshed in addition to the physical output. Writing to the output is possible and not subject to limitation, for only the local 6 outputs of the XC200-CPU with slot "0".

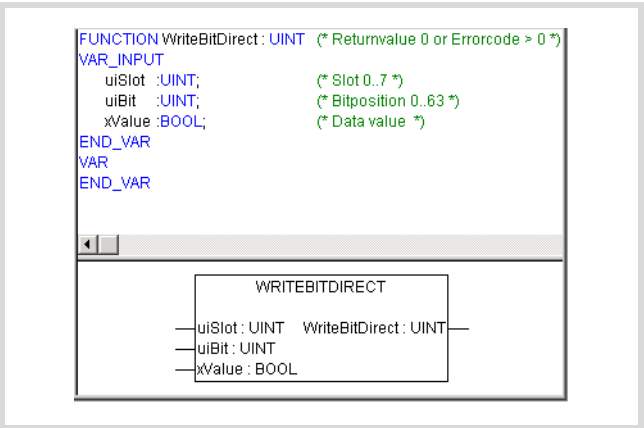


Figure 38: WriteBitDirect function

Parameters

| | |
|----------------|---|
| uiSlot | Slot number of the signal module. For possible parameters see Table 9 on Page 33. |
| uiBit | Output bit within the signal module. For possible parameters see Table 9 on Page 33 |
| xValue | The pointer points to the variable in which the value for the output bit is located |
| WriteBitDirect | Display of the fault code, see Table 10 on Page 36 |

WriteWordDirect

A word of an output module can be written directly with this function. At the time of access, the respective output image is also refreshed in addition to the physical output.

A further refresh of the output word occurs at the end of the cycle.

Parameters

| | |
|-----------------|--|
| uiSlot | Slot number of the signal module. For possible parameters see Table 9 on Page 33 |
| uiOffset | Output word within a signal module. For possible parameters see Table 9 on Page 33 |
| wValue | The pointer points to the variable in which the value for the output word is located |
| WriteWordDirect | Display of the fault code, see Table 10 on Page 36 |

GetSlotPtr

➔ This function is not available!

Error code with "direct peripheral access"

Verify all functions as far as possible for the validity of the call parameters. Verification is undertaken to determine if the access occurs in dependance on the parameterized signal module and the physical existence of the signal module. If a fault is determined, access is not undertaken and an error code is output. The data fields for the value transfer remain unchanged. The "DisableInterrupt" and "EnableInterrupt" functions do not generate an error code.

The following return values are possible:

Table 10: Error codes with direct peripheral access

| | |
|-------------------------------|---|
| IO_ACCESS_NO_ERROR: | No error |
| IO_ACCESS_INVALIDE_SLOTNUMBER | Slot = 0 or greater than 15 |
| IO_ACCESS_INVALIDE_OFFSET | BitWord offset is too large |
| IO_ACCESS_DENIED | Invalid access, e.g. write access to input module, read access to output module or access to non-available address range (offset too large) |
| IO_ACCESS_NO_MODULE | No module available at the parameterized slot |
| IO_ACCESS_INVALIDE_Buffer | No or incorrect pointer to the output variables |
| IO_ACCESS_INVALIDE_Value | Event is not "0" or "1" with "WriteBitDirect" |

Operating states

The following summary provides you with the state definitions for the CPU. The LED indications for the various states are also shown.

Table 11: Definition of the states of the XC200 with LED display

| State | Display RUN/STOP | SF | Definition |
|------------|---------------------------|---------|--|
| Boot | OFF (flashes at Start) | ON | The serial boot loader starts and boots and/or updates the operating system. Windows CE is loaded from Flash memory and copied in unpacked form into memory and started. |
| OS start | OFF (flashes at Start) | OFF | Windows CE system start and system test are carried out. Start of applications <ul style="list-style-type: none"> • HTTP-Server • FTP-Server • Telnet-Server • PLC-Runtime • Webserver |
| STOP | flashes | OFF | PLC in "STOP" state |
| RUN | ON | OFF | PLC in "RUN" state |
| RUN → STOP | flashes | ON | Error in RUN state "cycle time exceeded". The system is set to the Halt state. A "Warm Reset" command is automatically executed. The occurrence of a fault will be protocolled in the "Error List". (read with browser command: "geterrorlist") |
| NOTREADY | flashes | ON | No start possible. A major fault prevents a start (see "Error List") e.g.: <ul style="list-style-type: none"> • No program loaded • Fieldbus error • Configuration error • Checksum error • ... |
| ShutDown | flashes | flashes | Wait for the shut down of the supply voltage (after Browser command: shutdown) |

Web visualization

The description of the web visualization can be found in the programming software manual (AWB2700-1437GB), Section 7.4 Web visualization

The XC200 specific call for the web visualization is as follows:

`http://192.168.119.200:8080/webvisu.htm`

(Prerequisite: You have not changed the default setting of the IP address! Factory default with the XC200)

If you have changed the IP address, replace the IP address in the "http://..." call with the address you have selected.



Attention!

A max. of 10 clients may access the XC200!

Limit values for memory usage

The data memory of the XC200 is divided into memory segments for data. The memory size of the individual segments can be found in Figure 39. The global data utilises multiple segments. The required amount can be specified to suit the size of the loaded program.

The segment size for the different control types can be found under «Resources → Target Settings → Memory Layout»:

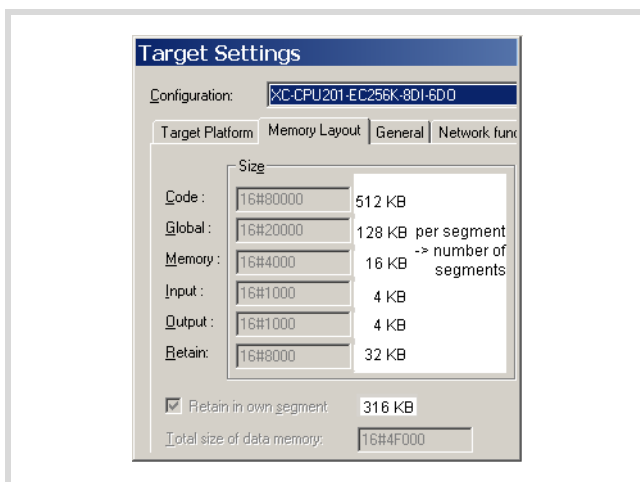


Figure 39: Segment size of the XC-CPU201-EC256k

The hexadecimal values of the other PLC types must be converted to decimal values.

In order to ensure that you use the available memory for the global data in an optimum and efficient manner, we recommend that you make the following settings when a new project is being created:

| PLC type | Number of data segments (global) |
|--------------------------|----------------------------------|
| XC-CPU201-EC256K-8DI-6DO | 2 |
| XC-CPU201-EC512K-8DI-6DO | 4 |

The number of segments is set to 1 by default.

The number of segments is changed as follows:

- Select «Project → Options → Compile options»; select the data segments field and enter the number of segments listed above for the respective control type.

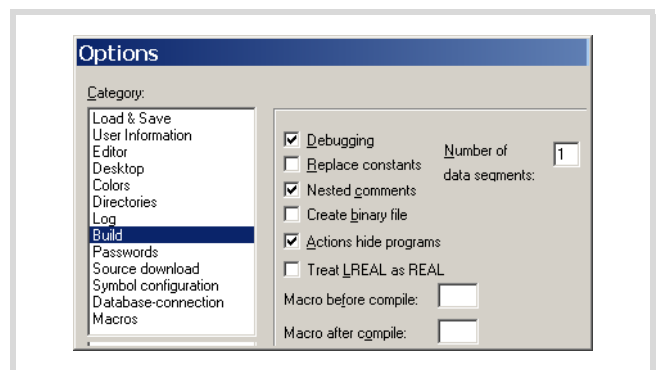


Figure 40: Memory management: Change the number of data segments

Addressing inputs/outputs and markers

If you open the PLC configuration of a new project, you will receive the current view of the default settings of the addressing. In this setting the addresses are automatically assigned and address conflicts (overlaps) are reported.

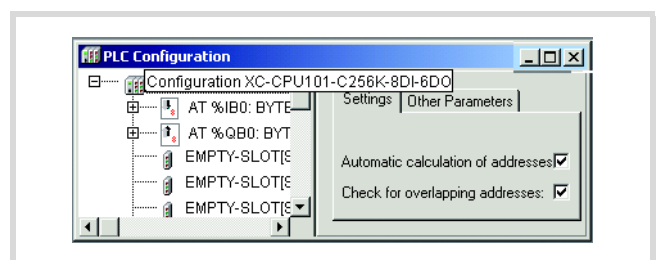


Figure 41: Default setting of the addressing

If you add a module to the PLC in the configurator, the configurator will assign this module with an address. Further modules are assigned with the next addresses in ascending order. You can also assign the addresses freely. However, if you access the "Automatic calculation of addresses" function later, the addresses are shown in reassigned ascending order.

“Activate Automatic addresses”

The addresses are automatically assigned or modified if a module is changed or added. This can occur with a centrally assigned module as well as a module which is a component of a decentral PROFIBUS-DP slave or CAN station.

If you add a module, the addresses of all the subsequent modules (independently of the line) are offset by the address value of the added module, and the added module is assigned with an address. Modules which are located in the configuration before the added module are not changed. If you remove the tick in the “Automatic calculation of addresses” checkbox, the addresses remain unchanged with modifications/expansions.

“Activating Check for overlapping addresses”

If the check for overlapping addresses is activated, addresses which are assigned twice will be detected and an error message is generated during compilation. This setting should not be modified.

Uneven word addresses

(Independent of the “Check for overlapping addresses” setting)

If you assign a word addressable module in field “Input address” to an uneven offset address, e.g. IB3, the PLC configurator automatically displays the following even word address (IW4).

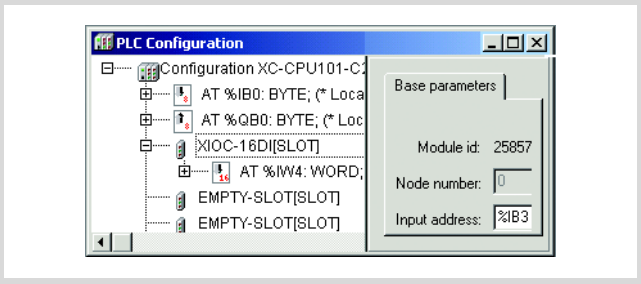


Figure 42: Uneven address

Address range

Addresses can only be assigned within the valid ranges. The range details can be found under <Target Settings → Memory Layout → Size>.

The addresses are checked during compilation. It is essential to ensure that the addresses of the configured module are used (referenced) in the program. If the address exceeds the range, a fault is signalled.

Table 12: Address ranges

| PLC | Input | | | Output | | | Markers | | |
|------------|-------|-------------------|-------------------|--------|-------------------|-------------------|---------|-------------------|-------------------|
| | Size | Max. Byte address | Max. Word address | Size | Max. Byte address | Max. Word address | Size | Max. Byte address | Max. Word address |
| XC100-64k | 2k | 2047 | 2046 | 2k | 2047 | 2046 | 4k | 4095 | 4094 |
| XC100-128k | 4k | 4095 | 4094 | 4k | 4095 | 4094 | 8k | 8191 | 8190 |
| XC100-256k | 16k | 16383 | 16382 | 16k | 16383 | 16382 | 16k | 16383 | 16382 |
| XC200-256k | 4k | 4095 | 4094 | 4k | 4095 | 4094 | 16k | 16383 | 16382 |
| XC200-512k | 4k | 4095 | 4094 | 4k | 4095 | 4094 | 16k | 16383 | 16382 |

Free assignment or modification of addresses of input/output modules and diagnostic addresses

Depending on the module, you can assign/modify the input, output and the diagnostics(marker) addresses:

In order to make the modifications visible in the PLC configurator it is necessary to click once on the PLC Configurator or to select another module after the address has been edited. They will be accepted in all cases during compilation.

Run "Automatic calculation of addresses"

With the "Automatic calculation of addresses" function which you can run either via the context menu or the menu bar, all the respective addresses are recalculated. If you are dealing with a bus

master module, the calculation is also carried out for the modules which are constituents of the slave on the bus line. The freely entered addresses of subordinate modules are overwritten when the address of a higher level module is calculated. If the addresses have changed and you wish to implement the "Automatic calculation of addresses", you must first of all activate the change. Click first of all on the nodes to drop down the structure or set the cursor in the PLC Configuration field and press the left mouse button.

If you mark the "Configuration XC-CPU..." text and call the "Automatic calculation of addresses", all the addresses are recalculated.



Enter the addresses in an ascending order and in continuous blocks.

Diagnostics

You can run diagnostics with the help of the diagnostics function block. The following possibilities are available:

| Type of diagnostics | Function block | Library | Documentation |
|---|--------------------------------------|----------------|---------------|
| Inspection of the XI/OC modules: <ul style="list-style-type: none"> Does the configuration of the hardware correspond with the configurator? Is the module function ok? Note: These tests are undertaken once during switch on or after loading/start of the program | XDiag_SystemDiag | xSysDiagLib | AWB2786-1456 |
| Inspection of the XIOC-NET-DP-M module and the stations on the DP line | XDiag_SystemDiag XDiag_ModuleDiag | XSysDiagLib | AWB2786-1456 |
| | DiagGetState | BusDiag | AWB2725-1452 |
| Inspection of the XIOC-NET-DP-S module | XDiag_SystemDiag XDiag_ModuleDiag | xSysDiagLib | AWB2786-1456 |
| DP slave provides the master with additional diagnostics data | XDPS_SendDiag | xSysNetDPSDiag | AWB2725-1452 |

6 Establishing a PC – CPU connection

The connection between the PC and CPU can be established via:

- the RS -232 interface
- the Ethernet interface

In this chapter you will get to know the settings to be made in the easySoft-CoDeSys.

See also:

- Connect PC → page 18

Connection set-up via RS-232 interface

To establish a connection between PC and CPU, the two devices' communication parameters must be the same.

- To match them, first adjust the PC's communication parameters to the CPU's standard parameters settings to → section „Defining/changing the PC's communication settings“.

The CPU features the following standard parameters:

| | |
|---------------|-------|
| Baud rates | COM1 |
| Parity | 38400 |
| Stop bits | 1 |
| Motorola Byte | No |

→ If you get an error message during login, the CPU's default settings have already been changed. In that case try a baud rate of 57600.

- After logging on the CPU parameters can be redefined (→ section „Changing the CPU's communication settings“).

Defining/changing the PC's communication settings

You can define the communication parameters of the interface in the easySoft-CoDeSys. You can use either the COM1 or the COM2 port of the PC.

- ▶ Select menu point «Online → communicationsparameters».
- ▶ Specify the port (COM1 or COM2 interface), → section „Changing settings“
- ▶ Use the remaining settings as shown in Figure 43.
- ▶ Confirm the settings with OK.
- ▶ Log on to the PLC.

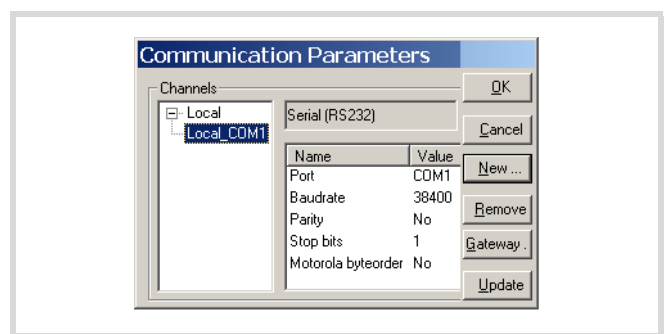


Figure 43: Defining the PC's communication settings

→ Further notes on the communication parameters can be found in the programming software manual (AWB2700-1437GB).

From operating system version V01.03.xx the serial (RS 232) (Level 2 Route) communication channel can be selected and a target ID can be defined. If you enter "0" for the target ID, communication is implemented with the local PLC.

Changing settings

To change settings such as the baud rate or the port, do the following:

- ▶ The field is highlighted in grey.
- ▶ Enter the desired value.

Double-click this field once more to choose the Baud rate, e.g. 57,600 bit/s.

Changing the CPU's communication settings

- ▶ Select "PLC Browser" in the "Resources".
- ▶ Select the "setcomconfig" browser command and add the required baud rate after inserting a space.
- ▶ Acknowledge the selection with RETURN.
- ▶ Select the "save registry" browser command.
- ▶ Select the "reboot" browser command. After reboot has been completed, the new baud rate is activated in the XC200.

Now access the CPU (e.g. by a login), you will receive the following fault message:

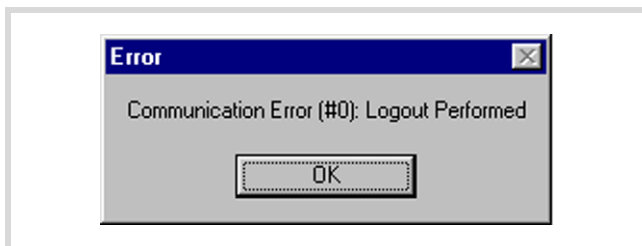


Figure 44: Communications fault

In order to communicate with the CPU, you must adapt the communication settings of the PC, → section „Defining/changing the PC's communication settings“

Connection setup via Ethernet

After you have connected the PC with the CPU using a cable, select the TCP/IP communication channel in the easySoft-CoDeSys and enter the IP address of the CPU. The CPU has the default address 192.168.119.200.

The selection of the data transfer rate of the Ethernet connection is performed in Autosensing (detect) mode. Components with this feature automatically recognise if it is a 10 or 100 MBit connection.

Selecting communication channel and address

- ▶ Access the menu with <Online → Communication parameters>.

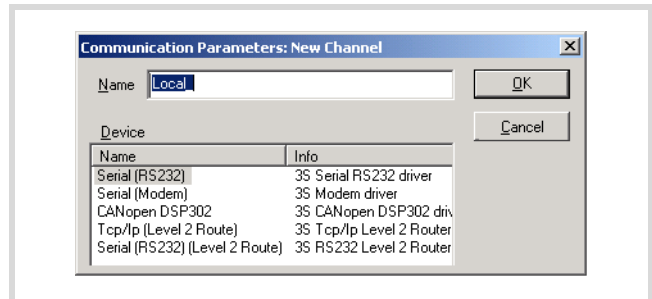


Figure 45: Channel selection

- ▶ Push the "New..." button.
- ▶ Select the overview of the communication channel TCP/IP (Level2Route) and change the name "local" e.g. to "Ethernet-Test"
- ▶ Confirm with OK.

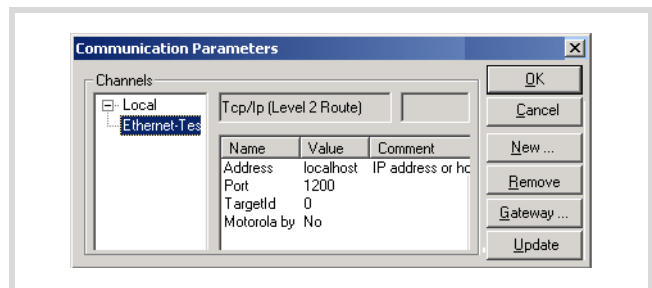


Figure 46: Enter the IP address

- ▶ Perform a double click on the "localhost" field and enter the default address 192,168,119,200
- ▶ Confirm your details, by first pressing on another field and then on OK.

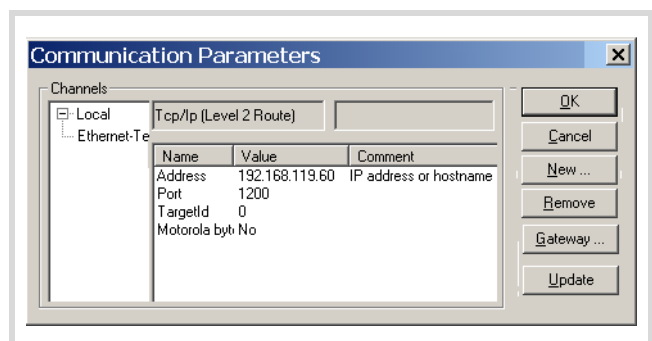


Figure 47: Communication parameters with IP address

- ▶ Compile the program and log out.

Scan/Modify the IP address

The "setipconfig" and "getipconfig" browser commands are available for modifying and scanning the IP address → section „Browser commands“ on Page 67.

Restart the XC200 after you have changed the IP address. The DHCP function (Dynamic Host Configuration Protocol) is not activated.

Ensure that the IP address of the programming device belongs to the same address family. This means, the IP address of the programming device and the XC200 must correspond in the following figure groups:

Example 1

IP address XC200: 192.168.119.xxx
IP address PC: 192.168.119.yyy

Example 2

IP address XC200: 192.168.100.xxx
IP address PC: 192.168.100.yyy

The following conditions apply in example 1 and 2:

- xxx is not equal to yyy
- the addresses must be between the limits 1 and 254.
- The addresses must be part of the same address family.

If a connection is not established, the transfer route can be checked with the "PING" function in order to ensure that the connection has not failed due to a fault on the transmission path. The following steps are necessary:

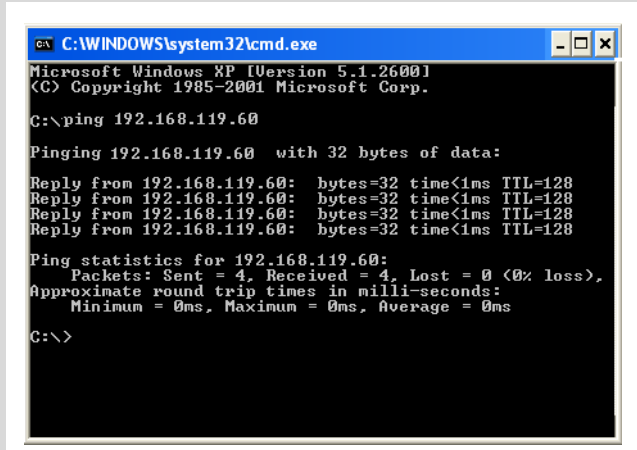
- Open the DOS window via the "Start" field and the "Run" command.
- Enter "CMD" in the input field and confirm with "OK".

You are presented with a window indicating a drive and a flashing cursor behind the drive designator.

- For the example mentioned you would enter the following text: "ping 192,168,119,200" and confirm this with OK.

If the routing is functioning correctly, you will receive a response indicating the response time. Otherwise a time-out will indicate problems with the connection setup.

The following figure indicates the result of a correct connection set-up.



```

C:\WINDOWS\system32\cmd.exe
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\>ping 192.168.119.60

Pinging 192.168.119.60 with 32 bytes of data:

Reply from 192.168.119.60: bytes=32 time<1ms TTL=128
Reply from 192.168.119.60: bytes=32 time<1ms TTL=128
Reply from 192.168.119.60: bytes=32 time<1ms TTL=128
Reply from 192.168.119.60: bytes=32 time<1ms TTL=128

Ping statistics for 192.168.119.60:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 0ms, Maximum = 0ms, Average = 0ms

C:\>
  
```

Figure 48: PING response with a correctly established Ethernet connection

7 Defining the system parameters via the STARTUP.INI file

Overview

System parameters independent of the project can be set by you and saved on the memory card. Here they are contained in the Startup.INI file. The memory card can also be fitted in other controllers. The PLC accepts the parameters during start up. The Startup.INI file is always generated with all control-specific entries (→ table 13).

The term STARTUP.INI-file is generally applicable. The file name of the STARTUP.INI-file for the XC200 is XCSTARTUP.INI.

Parameters in the Startup.INI file

Some parameters, e.g. such as the Baud rate of the COM interface have already been entered by the system, to ensure that communication can take place between the PC and PLC. The parameters can be adjusted later.

Table 13: Predefined default parameters in the Startup.INI

```
TARGET=XC-CPU201
HOST_NAME=NoNameSet (from OS V1.04)
IP_ADDRESS=192,168,119,200
IP_SUBNETMASK=255.255.255.0
COM_BAUDRATE=38400
CAN_ROUTING_CHANNEL=1
```

Table 14: Example: contents of the XCSTARTUP.INI file

```
[STARTUP]
TARGET=XC-CPU201
to the Ethernet connection:
HOST_NAME=NoNameSet (from OS V1.04)
IP_ADDRESS=192,168,119,200
IP_SUBNETMASK=255.255.255.0
IP_GATEWAY=
IP_DNS=
IP_WINS=
to the programming interface RS232:
COM_BAUDRATE=4800, 9600, 19200, 38400, 57600
to the CAN interface:
CAN1_BAUDRATE=10, 20, 50,100, 125, 250, 500
CAN1_NODEID=1-127
CAN_ROUTEID=1-127
CAN_ROUTING_CHANNEL=1
for addresses of the PROFIBUS slaves:
NET_DPS1_BUSADDRESS= (DP-S in Slot 1)
NET_DPS2_BUSADDRESS= (DP-S in Slot 2)
NET_DPS3_BUSADDRESS= (DP-S in Slot 3)
```

Structure of the INI file

An INI file is a text file with a fixed data format. The system parameters are listed from a specified section such as [STARTUP], followed by an equals sign and the corresponding value. The line is terminated with CR/LF (Carriage/Return). The line is terminated with CR/LF (Carriage/Return).

```
COM1_BAUDRATE=38400
(Carriage/Return)
```

Lines commencing with a semicolon are interpreted by the PLC as comments and are ignored:

```
; CAN_NODEID=2
```

You can change or create the parameters with a text editor if you fit the memory card in the memory card slot of the PC. The file XCARTUP.INI is saved on the memory card in the directory "MOELLER/XC-201-EC256K-8DI-6DO/PROJECT/".

Creating the Startup.INI file

Generally the control operates when first activated (initial state) with default system parameters, the STARTUP data regardless of if the PLC contains a project or boot project! If you load the project into the PLC which is in the initial state, the PLC will immediately start to operate with the parameters of the project.

With the browser command "createstartupini" you will transfer from the PLC either the STARTUP data – or if a project is contained – the system parameters onto the memory card. This creates the Startup.INI file that contains this data. Precondition: the memory card must be plugged in, formatted and empty, i.e. without Startup.ini file.

A file which already exists cannot be changed or overwritten by the browser command "createstartupini". If you still enter the command, a warning appears. In order to create a new file the existing file must be deleted first. → section „Deleting the Startup.INI file“ on Page 46

Entry of the INI file: "HOST_NAME"

With the help of the parameter "Device name (HOST_NAME)" the PLC can be contacted via the Ethernet with this device name. Furthermore, it can also be contacted with its IP address. The device name is available from operating system 1.04 or higher. It receives the "NoNameSet" entry from the system. If it is not changed the device connected to the Ethernet can be contacted via its IP address. Using the browser command "settargetname ..." you enter a new device name. The device name must be unique for each device. It can also be used as a communication parameter (in the Figure 49: atlas) if it is defined in the programming software in the menu <Online → Communication parameter>. The parameters define the properties of the programming connection between PC and PLC.

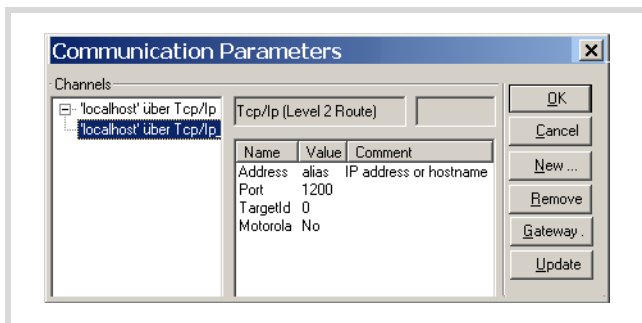


Figure 49: Communication parameters

The device name can be read with the browser command "gettargetname".

Switch-on of the control with inserted memory card with XCSTARTUP.INI file

When the controller is started up, the data from the Startup.INI file on the memory card is transferred to the controller. These system parameters are also active after a new program is loaded.

Changing settings

The parameters are retained until you enter the browser command "removestartupini" and then switch the controller off and on again. The controller will now operate with the parameters of the project.

Deleting the Startup.INI file

The following browser commands are available to access the memory card.

- **removestartupini:**
Always deletes the controller system parameters. If a memory card is fitted, the INI file is also deleted on the memory card. The parameters from the project is accepted next time the device is switched on.
- **removeprojfrommmc:**
Deletes the boot project and the INI file on the memory card. The system parameters are retained in the controller.

The reaction of the Startup.ini tile with the menu command "Full reset", with the command "Factory default" in the PLC menu as well as in the "factoryset" browser command is described in section "Reset" on Page 24.

If you execute the "Full reset" command in online mode, the operating system and the project on the Disk_sys are deleted. TheXCSTARTUP.INI file is retained.

8 Programming via CAN(open) Network (Routing)

“Routing” is the capability to establish an Online connection from a programming device (PC) to any desired (routing capable) control in a CAN network, without having to directly connect the programming device with the target PLC. Es kann an eine andere Steuerung im Netzwerk angeschlossen werden. All actions that are available through a direct PC–PLC connection can also be implemented through the routing connection:

- Program download
- Online modifications
- Program test (Debugging)
- Generation of boot projects
- Writing files in the PLC
- Reading files from the PLC

Routing has the advantage that a PLC connected to the programming PC can access all routing capable PLCs on the CAN bus. You select the control with which you want to communicate by the project selection. This provides an easy way of controlling remote PLCs.

Allerdings ist die Datenübertragung von Routing-Verbindungen deutlich langsamer als bei Direktverbindungen (Seriell oder TCP/IP). This results, for example, in slower display refresh rates of variables and longer download times.

Prerequisites

The following prerequisites must be fulfilled to use routing:

- The routing PLC and the target PLC must both support routing.
- Both PLCs must be connected via the CAN bus.
- The PLCs must both have the same active CAN baud rate.
- The valid routing node ID must be set on both PLCs.
- Routing with the XC200 is possible from BTS version 1.03.02.

Routing features of the controller

The controller supports routing via the CAN bus.

Routing can be implemented without prior download of a user program (default: 125 kBaud, Node-Id 127). The target PLC must not be configured as a CAN Master or CAN Device for this purpose.

You can for example load a program from the PC via a PLC of the XC device series into the XC200. Assign a Routing Node-Id to the XC200 (target PLC) in this case. Assign a Routing Node-Id to the XC200 (target PLC) in this case..

Routing through XC200

To perform a program transfer or routing using TCP/IP through a connection between XC200 and PC, you must first set the block size for the transferred data. The packet size (4 KByte or 128 KByte) depends on the transfer type (program transfer or routing) and the operating system, → table 15.

Table 15: Block size for data transfer

| | Program/file transfer | | Routing | |
|-------------------------------------|-----------------------|---------------|-------------------------|---------------|
| | OS < V1.03.02 | OS ≥ V1.03.02 | OS < V1.03.02 | OS ≥ V1.03.02 |
| Block size Default: 128 KByte | 128 KByte | 128/4 KByte | Routing not possible | 4 KByte |



Attention!

The program download with a block size of 4 KByte to a PLC with an operating system version earlier than V1.03.02 will cause faulty behaviour!

If a program download is performed, the progress bar on the programming device monitor will only change erratically (about every 10 seconds).

The block size can be changed only directly in the Windows Registry.



You can change this setting only if you have administrator rights on your PC.

Changing the block size

- Close all easySoft-CoDeSys applications.
- Close the CoDeSys gateway server.

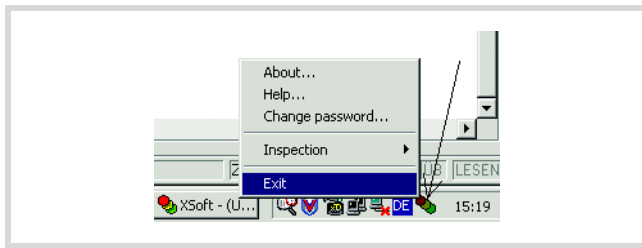


Figure 50: Closing the CoDeSys gateway server

- Change the block size to the required value.

The following *.reg files are available in the easySoft CoDeSys installation directory to enter the block size in the registry:

| | |
|----------------------|--|
| BlockSizeDefault.reg | Enters a block size of 20000 _{hex} = 128 KByte (default value) in the Registry. |
| BlockSizeRout.reg | Enters a block size of 1000 _{hex} = 4 KByte in the Registry. |

Alternatively, you can use the BlockSizeEditor application to change the block size.

The download block size is defined in the following registry key:

```
[HKEY_LOCAL_MACHINE\SOFTWARE\3S-Smart Software Solutions
GmbH\Gateway Server\Drivers\Standard\Settings\Tcp/Ip (Level
2 Route)]
"BlockSize"=dword:00020000
```

The default block size is 20000_{hex} (=128 Kbyte), the block size for routing is 1000_{hex} (= 4 Kbyte).

Notes

- If large files are written to the target PLC or read from the PLC, it is possible that the online connection will be interrupted after the transfer process has been completed. Renewed connection is possible.
- If a program with a modified routing node ID is loaded into the target PLC, the target PLC accepts the modified routing node ID; however, the communication connection will be interrupted. Reconnection with a corrected routing Node ID is possible.
- If a PLC receives a program without valid routing parameters (baud rate/node ID), this PLC cannot be accessed via a routing connection. Erasing of the parameters can for example be implemented via a FULL RESET if the PC with the programming software was directly connected with the target PLC. The parameters are retained if the FULL RESET is implemented via the "routing PLC".
- The routing is independent of the configuration (master/slave): a target PLC that has not been configured as a master or as a slave can be accessed. It must only receive the basic parameters such as Node ID and baud rate, as well as a simple program.

Addressing

PLCs on the CAN-Bus can be configured as a master or as a slave. The PLCs are assigned with a Node ID/node number (address) in order to uniquely identify them (with the basis communication). To use the routing function to access a target PLC, you must assign a further routing ID to the routing and target PLC. An RS200 or Ethernet interface can be used as a connection between the PC and XC232.

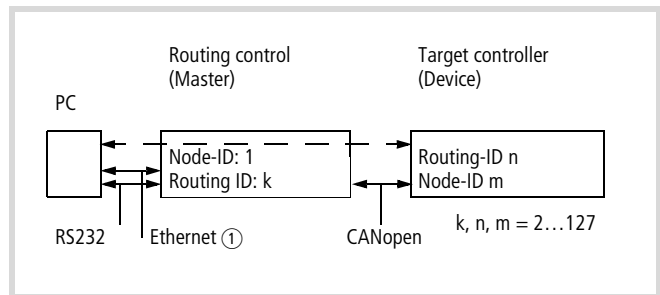


Figure 51: Routing via XC..., EC4-200

① Ethernet connection only possible with XC200

Table 16: Example for setting the Node Id, Baud rate

| PLC | Function | Node ID | Routing ID | Baud rates | → Fig. |
|--------------------|----------|---------|------------|------------|--------|
| Routing controller | Master | 1 | 127 | 125 KB | 53 |
| Target controller | Device | 3 | 54 | 125 KB | 54 |



The following applies for device PLCs: The Routing-ID must **not be equal** to the Node-ID (Basis communication)!

The exception is the XC100 with operating system \geq V2.0: the Routing-ID **must be equal** to the Node-ID!

The Routing-ID with the master can be set in the PLC configurator in the "Other parameters" tab:

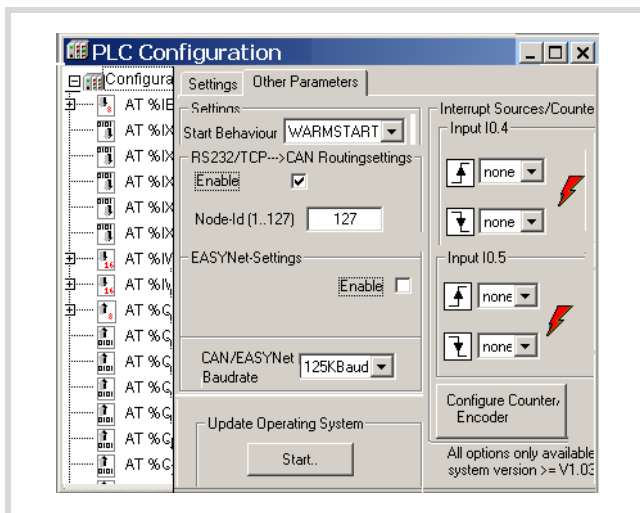


Figure 52: CAN Master routing settings

The ID for basis communication is defined in the "CanMaster" folder in the "CAN parameters" tab (Figure 53).

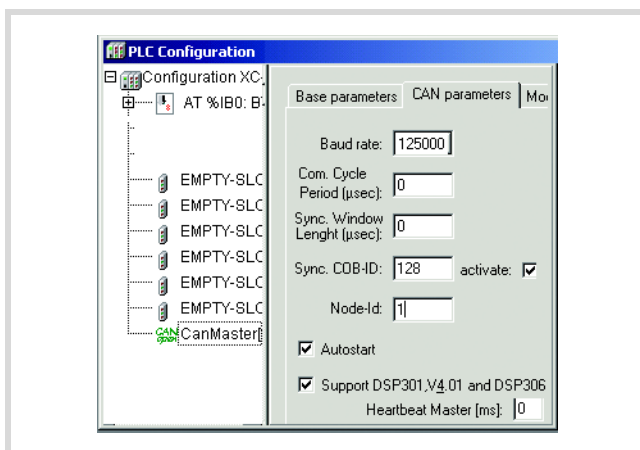


Figure 53: CAN Master: Node ID for basis communication

Communication with the target PLC

- ▶ Connect the PC to the routing PLC.
- ▶ Select the target PLC with which you want to communicate for the project.
- ▶ First of all determine the communication parameters for the connection between the PC and the PLC which is connected to the PC.
- ▶ Enter the target ID (Target ID = Node ID!) of the target PLC, as in the example, and log on.

You can run the following functions:

- Program download
- Online modification
- Program test (Debugging)
- Create boot project
- Filing source code.

Note for project creation

Assign two Node-IDs to the target PLC:

- ▶ One ID for basic communication
- ▶ One ID for routing

The routing ID and the baud rate of the target PLC (e.g. XC 200) to the routing function can be defined in the "Additional parameters" window in the PLC Configuration. → figure 52. First click on the "Activate" box in the "RS232/TCP → CAN Routing" field. This activation is necessary to ensure that the PLC can communicate via the CAN bus. Then enter the routing ID/node number and the baud rate on the bus in the appropriate entry fields.

→ To guarantee a fast data transfer, the routing should be performed only with a CAN baud rate of at least 125 KBit/s.

The ID for basis communication is defined in the "CanDevice" in the "CAN setting" tab → figure 54.

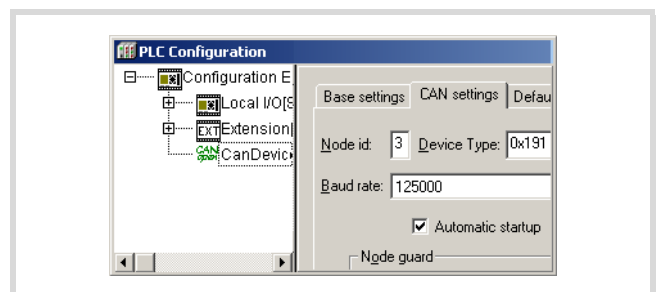


Figure 54: CAN device parameters

ID and baud rate are transferred with the project download.

Example

In the following example which is based on Figure 55 the procedure for access to a PLC program is explained.

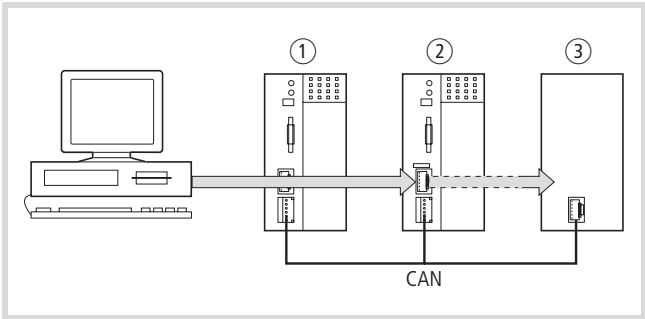


Figure 55: Diagnostics possibilities

- ① XC100 with Node ID 1
- ② XC200 with node ID 2, routing ID 127
- ③ PLC (e.g. XC..., EC4-200) with node ID 3 and routing ID 54

You have connected the PC to the PLC with node ID "2" and want to access the target PLC with routing ID "54".

- Open the project of the target PLC whose program you wish to edit or test.
- First configure the parameters for the hardware connection PC ↔ PLC (Node ID 2).
- From the Online menu select Communication Parameters....
- Click the New button under Local channels.

The "New Channel" window appears.

- Select the channel in the "Device" window:
Serial [RS 232] [Level 2 Route] or TCP/IP [Level 2 Route].
- You can assign a new name in the Name field, e.g. "Rout_232".
- Confirm with OK and return to the original window.

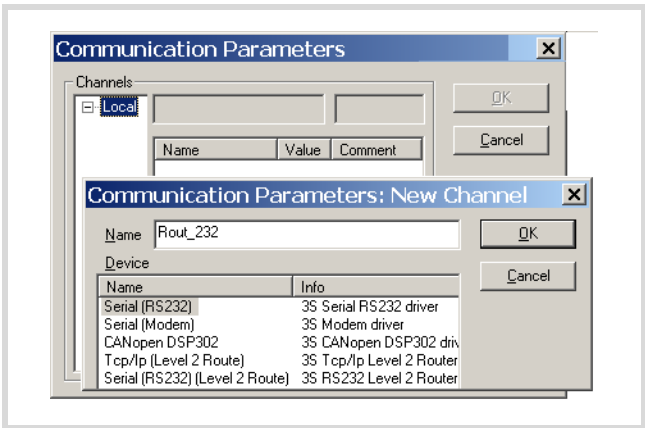


Figure 56: Channel parameter setting

You have now determined the parameters for the hardware connection between the PC and the PLC (node ID 2).

- Call up the communications parameters in the "Online" menu once again and select the control which you want to program/test.

- Enter the number 54 as the target ID in the example. The target ID is identical to the routing ID!
To enter the target ID, click on the field in the Value column, to the right of the term Target ID. Enter the number 54 there and confirm with OK.
- Log on and carry out the action.

PLC combinations for routing

The following PLC support routing:

| From → | XC100 XC121 | XC200 ¹⁾ | EC4-200 |
|---------------------|----------------|---------------------|---------|
| To ↓ | | | |
| XC100, XC121 | × | × | × |
| XC200 ¹⁾ | × | × | × |
| EC4-200 | x | × | × |

1) XC200 from version V2001-03-01

Number of communication channels

Several communication channels can be opened, e.g. PC ↔ PLC 2, PC ↔ PLC 3 in dependence on the PLC (communication channel) which is connected to the PC. The status display of control 2 and 3 can be implemented simultaneously.

Table 17: Type and number of communication channels

| Communications channel | PLC | Max. channel count |
|--------------------------|---------------|--------------------|
| TCP/IP Level2Route | XC200 | 5 |
| Serial RS232 Level2Route | XC.../EC4-200 | 1 |

9 RS -232 interface in Transparent mode

In Transparent mode, data is exchanged between the EC1-1 and data terminal devices (e.g. terminals, printers, PCs, measuring devices) without any interpretation of the data. CAN. For this purpose, the serial RS 232 (COM1) interface of the XIOC-SER modules (COM2/3/4/5) is to be switched using the user program in the transparent mode. This applies from operating system version 01.03.xx for the RS 232 interface (COM1) of the CPU.

→ If the RS-232 interface of the CPU is in transparent mode, programming via this interface is not possible. However, the program can be tested via the Ethernet interface.

Character formats in transparent mode are: 8E1, 8O1, 8N1, 8N2.

This functionality is provided with the XC200 via the "xSysCom200.lib" or "SysLibCom.lib" libraries. Thus, one of these libraries must be integrated into the library manager.

The "SysLibCom.lib" library is introduced (from version 01.03.xx) in order to guarantee the compatibility between the XC200 and other XControl devices such as XC600, HPG....

Both libraries contain functions for opening and closing the interface, for sending and receiving the data and for setting the interface parameters.

The control lines of the RS 232 of the XIOC-SER modules are controlled with the "SysComWriteControl" function from the "xSysCom200.lib" library and monitored with the "SysComReadControl" function.

In contrast to the RS232 interface of the XIOC-SER module, the RS232 interface of the CPU does not feature control lines.

The data types of the libraries are not identical. The baud rate selection differs:

xSysCom200.lib: 300,...,115200

SysLibCom.lib: 4800,...,115200

The RS-232 interface (COM1) of the CPU is addressed (in contrast to the interface of the XIOC-SER module) via the operating system! Therefore, execution of the interface functions can take up to 50 ms. The task, in which the RS-232 interface is contacted should have an interval time of at least 50 ms and be assigned with a low priority (high value) in multitasking mode, so that time critical tasks are not hindered.

With the functions (x)SysComRead/Write, only the parts of the required data length are processed. For complete transfer of data blocks, repeated calls with matched offset values are to be executed in multiple task intervals. The number of calls required depends on the baud rate and quantities of data involved!

The performance of COM1 depends on the load on the PLC (PLCLoad) and on the selected baud rate. They may be hindered by time-critical tasks due to the high interval times of the COM1 task. Character loss is possible when data is received with high baud rates.

Programming of the RS -232 interface in transparent mode

You can access data of the RS232 interface using the user program. The functions of the "xSysCom200.lib" or "SysLibCom.lib" libraries are available for these functions. Only one of the two libraries can be integrated into the library manager. In both libraries there are functions e.g. for opening and closing the interface. In order to simplify the overview, the functions of the libraries are represented beside one another in the following figures. The functions of the "xSysCom200.lib" are on the left and the functions of the "SysLibCom.lib" are on the right.

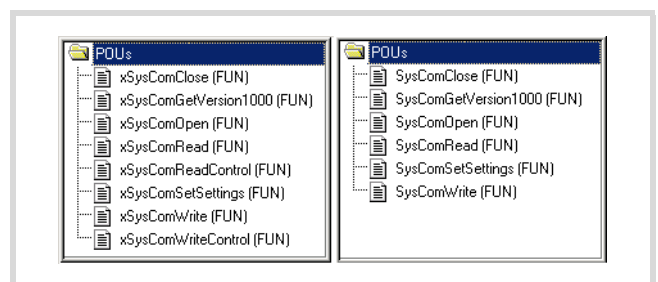


Figure 57: Overview (left: xSysCom200.lib, right: SysLibCom.lib)

The individual functions are described in manual "Function Blocks for easySoft CoDeSys" (AWB2786-1452GB).

See also:

- Transparent mode: Text output via RS232 (example) → page 74

10 Configuration and parameterization of the inputs/outputs

Input/output general

In the PLC configuration the local inputs/outputs IX0.0 to IX0.7, QX0.0 to QX0.5 and the inputs/outputs IX1.0 to IW4 and QX1.0 to QX1.7 indicate the additional functions such as e.g. the counters.

The inputs and outputs of the additional functions only become active after you have selected a function in the "Other parameters" tab.

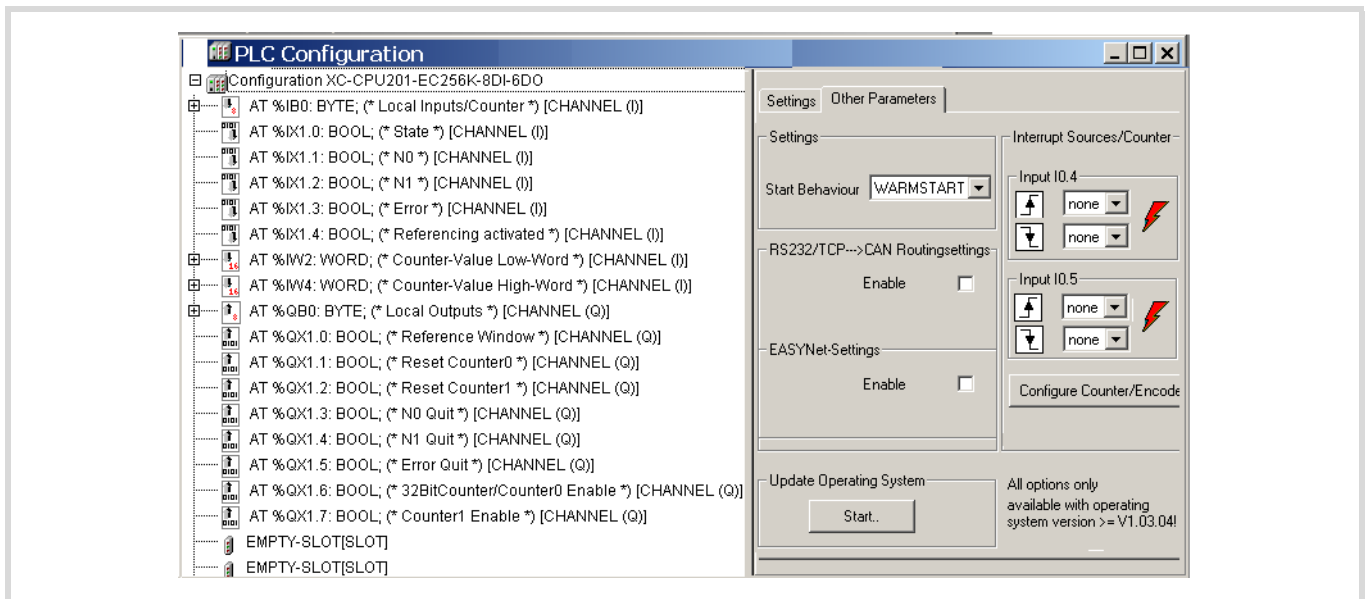


Figure 58: PLC configuration

The processing unit transfers states and events to the virtual input. The required I/Os for the incremental encoder function are shown in Figure 64 or on Page 55, the necessary I/Os for the counter functions can be found on Page 58.

In order to expand the local I/Os simply add XIOC modules by clicking on the "EMPTY SLOT" folder. With the "replace element" command select a module from the list. The new module name is indicated instead of the EMPTY-SLOT.

Local digital inputs/outputs

Each physical change of the modules in the slots of the backplane are recognised by the CPU (change of slot or exchange with another function), as the input/output offset is changed and the assignment to an input or output parameter can lead to an access fault. If you have reserved free slots for later optimization in the configuration, and these free slots are then occupied later, this will cause a difference and change in the input/output offset between the configuration and program.



Attention!

- Match the inputs and outputs in the program each time you make a change to the configuration.
- If the configuration and program do not match or if an unavailable module is configured, the PLC can't change over to the RUN mode.

A difference between the configuration and the physical existence/non-existence of signal modules is entered as a "Fault event" in the buffered memory range. The "geterrorlist" browser command issues this fault as a "General IO access error". A unique slot assignment is not possible here.

The following illustrations indicate the changes of assignment of the input and output parameters when exchanging or adding/removing signal modules.

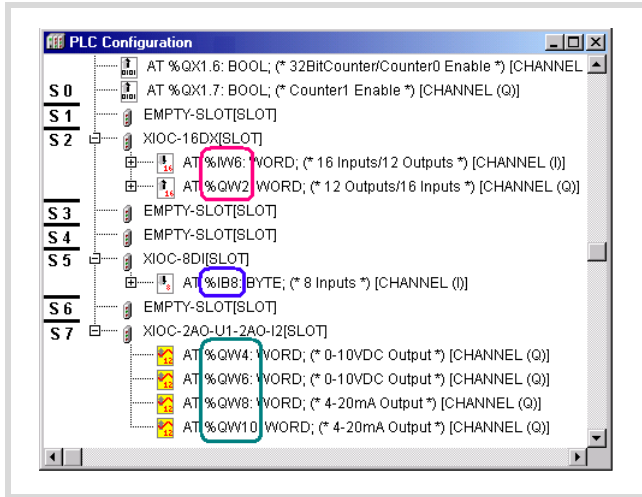


Figure 59: Current configuration
S0, S1, ..., S7 = slot number on backplane

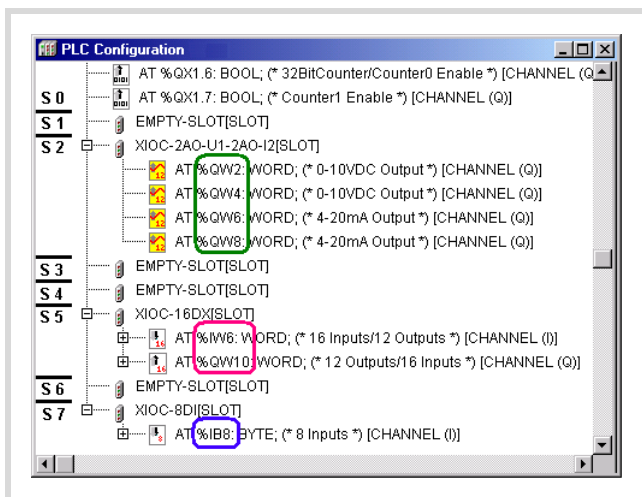


Figure 60: Configuration change through exchange of the modules
(S0, S1, ..., S7 = slot number on backplane)

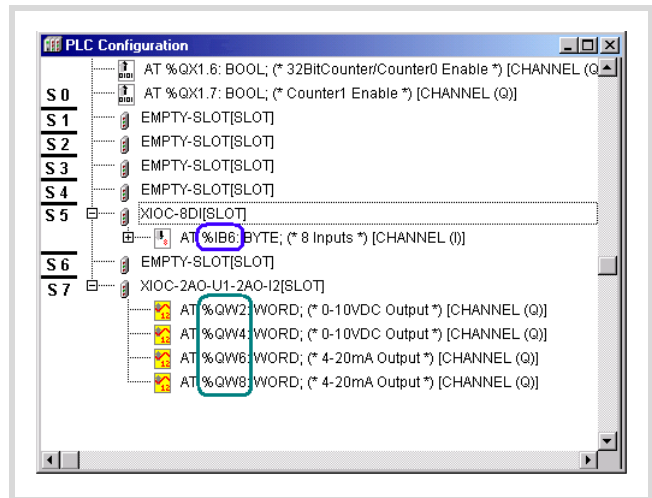


Figure 61: Configuration change through removal of the module
S0, S1, ..., S7 = slot number on backplane

Figures 59 to 61 indicate the changes to the input/output parameters of the signal modules in dependence on the slots and are compiled in Table 18.

Table 18: Input/output parameters with a change of configuration

| Figure | Slot | Module type | Input parameter | Output parameter |
|--------|------|--------------------|-----------------|-----------------------------------|
| 59 | 2 | XIOC-16DX | %IW 6 | %QW 2 |
| | 5 | XIOC-8DI | %IB 8 | — |
| | 7 | XIOC-2AO-U1-2AO-I2 | — | %QW 4 %QW 6 %QW 8 %QW 10 |
| 60 | 2 | XIOC-2AO-U1-2AO-I2 | — | %QW 2 %QW 4 %QW 6 %QW 8 |
| | 5 | XIOC-16DX | %IW 6 | %QW 10 |
| | 7 | XIOC-8DI | %IB 8 | — |
| 61 | 2 | Slot not used | — | — |
| | 5 | XIOC-8DI | %IB 6 | — |
| | 7 | XIOC-2AO-U1-2AO-I2 | — | %QW 2 %QW 4 %QW 6 %QW 8 |

Inputs/outputs for additional functions

Incremental encoder

Parameterization occurs in the "PLC Configuration".

- Activate the "Other Parameters" tab in the "PLC Configuration" window and click on the "Configure Counter/Encoder" button".

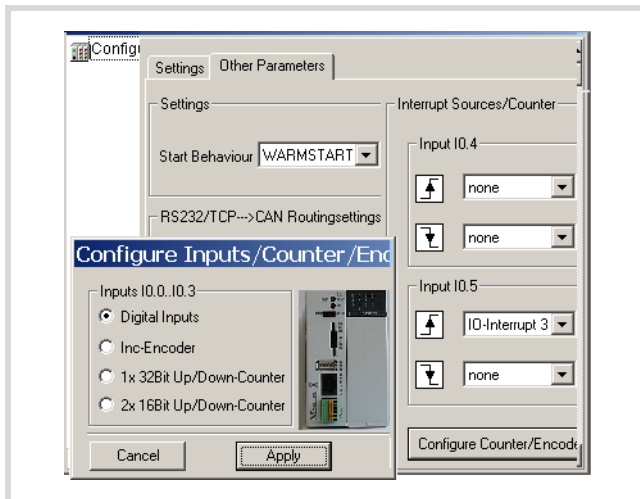


Figure 62: Incremental encoder preselection

- Select "Inc-Encoder". The window changes its appearance:

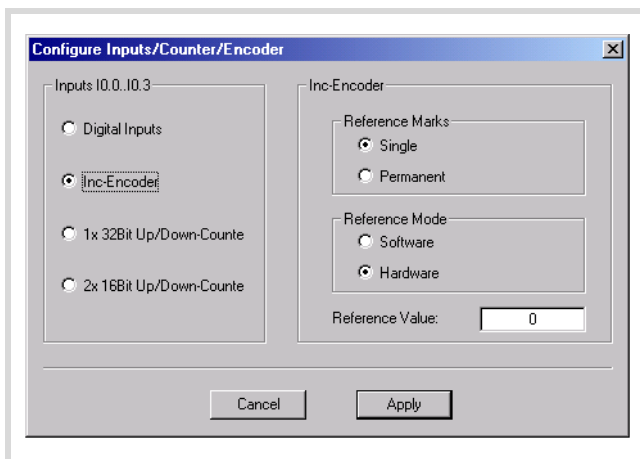


Figure 63: Incremental encoder parametric programming

When the configuration is complete, press the "Apply" button.

Functionality of the inputs/outputs

If you have selected the "Incremental encoder" additional function, the inputs I0.0 to I0.3 are assigned with a new function. The inputs I0.4 to I0.7 retain their standard function. The functions of the virtual inputs/outputs for the incremental encoder can be seen in the following illustrations.

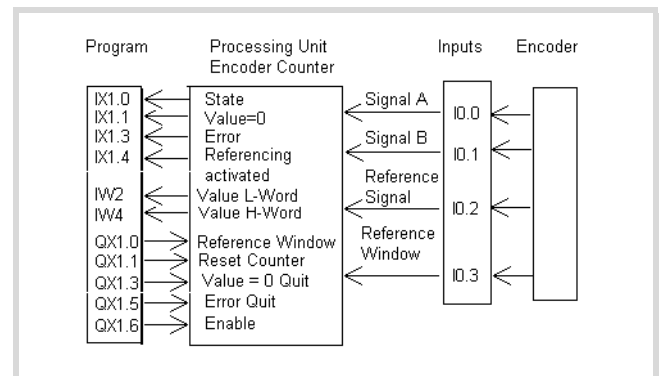


Figure 64: I/Os for incremental encoder

Reference position control:

Many position controls first move to reference point. The referencing process can be controlled via the hardware (reference window-signal from transducer to I0.3) or via the software (QX1.0). The selection can be made in the PLC configuration. If one of the signals has an H-level, then this will be shown on input IX1.4. If an impulse is given to I0.2 in this state (referencing signal from transducer), then the counter state will be set to the reference value as entered in the PLC configuration.

- ➔ Set the reference window large enough for the reference signal to be present only once and still be evaluated reliably.

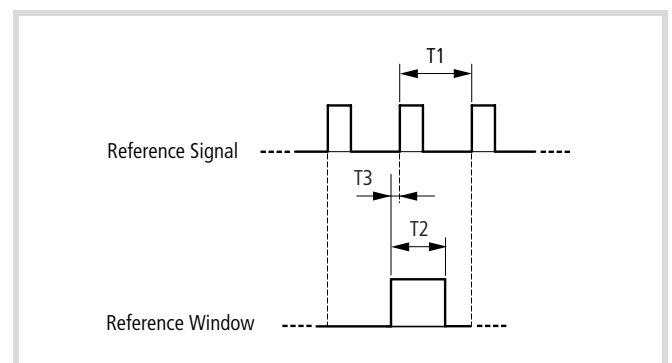


Figure 65: Relationship between reference signal and reference window

- T1 Impulse repeat time of 2 successive marker signals with a single rotation of the incremental encoder
- T2 Maximum permissible duration of the reference window. Must be sufficiently less than T1 to ensure that a second marker pulse is not detected.
- T3 Must be long enough to ensure that the L/H edge of the marker pulse is safely detected.
- T2 and T3 are dependant on the pulse train frequency of the marker pulse and may need to be determined experimentally to suit the application.

Representation of the inputs/outputs of the incremental encoder

Real inputs

| | |
|---------------------------|----------------------------------|
| AT %IX0.0: BOOL; (*Bit0*) | Signal A |
| AT %IX0.1: BOOL; (*Bit1*) | Signal B |
| AT %IX0.2: BOOL; (*Bit2*) | Reference signal |
| AT %IX0.3: BOOL; (*Bit3*) | Enable referencing ¹⁾ |
| AT %IX0.4: BOOL; (*Bit4*) | Local input |
| AT %IX0.5: BOOL; (*Bit5*) | Local input |
| AT %IX0.6: BOOL; (*Bit6*) | Local input |
| AT %IX0.7: BOOL; (*Bit7*) | Local input |

Representation of the virtual inputs/outputs in the PLC configuration

| | |
|--|---|
| AT %IX1.0: BOOL; (*State*) [CHANNEL(I)] | H = referencing implemented |
| AT %IX1.1: BOOL; (*N0*) [CHANNEL(I)] | L = no zero crossover, H = zero crossover of the counter level |
| AT %IX1.2: BOOL; (*N1*) [CHANNEL(I)] | |
| AT %IX1.3: BOOL; (*Error*) [CHANNEL(I)] | L = no fault H = internal error (A and B edges occur simultaneously) |
| AT %IX1.4: BOOL; (*Referencing activated*) | H = referencing has been enabled |
| AT %IW2: WORD; (*Counter-Value Low-Word*) [CHANNEL(I)] | Counter state Low Word |
| AT %IW4: WORD; (*Counter-Value High-Word*) [CHANNEL(I)] | Counter state High Word |
| AT %QX1.0: BOOL; (*Reference Window*) [CHANNEL(Q)] | Enable referencing ²⁾ |
| AT %QX1.1: BOOL; (*Reset Counter0*) [CHANNEL(Q)] | Reset to reference value |
| AT %QX1.2: BOOL; (*Reset Counter1*) [CHANNEL(Q)] | |
| AT %QX1.3: BOOL; (*NO Quit*) [CHANNEL(Q)] | Acknowledgement zero crossover |
| AT %QX1.4: BOOL; (*N1 Quit *) [CHANNEL(Q)] | |
| AT %QX1.5: BOOL; (*Error Quit*) [CHANNEL(Q)] | Error acknowledge |
| AT %QX1.6: BOOL; (*32BitCounter/Counter0 Enable*) [CHANNEL(Q)] | L = inhibit input pulse H = enable input pulse |
| AT %QX1.7: BOOL; (*Counter1 Enable*) [CHANNEL(Q)] | |

1) Precondition: The "Hardware" configuration type has been selected in the configurator.

2) Precondition: The "Software" configuration type has been selected in the configurator.

Counter

Select between the following functions for the detection of counter pulses:

- 1 × 32 Bit up/down counter or
- 2 × 16 Bit up/down counter.

Parameterization occurs in the "PLC Configuration".

- ▶ Activate the "Other Parameters" tab in the "PLC Configuration" window and click on the "Configure Counter/Encoder" button".
- ▶ Select "1 × 32 Bit Up/Down-Counter or 2 × 16 Bit Up/Down-Counter" and click on the "Apply" button.

Another window opens for the configuration.

- ▶ State the "Interrupt Source" and the "Setpoint Value" here.

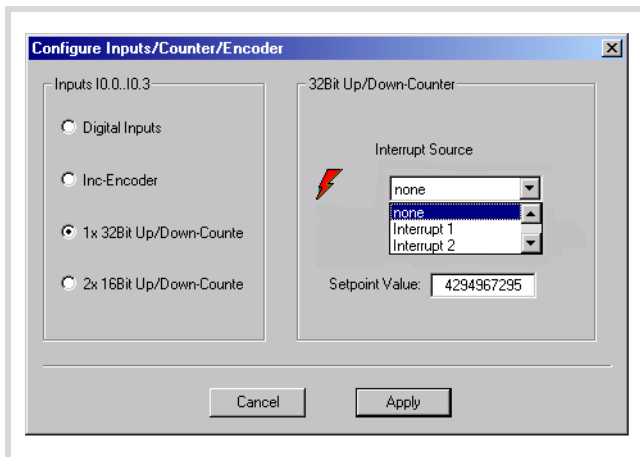


Figure 66: Parameterization of 1 × 32 Bit counter input

- ▶ When the configuration is complete, press the "Apply" button.

See also:

- Interrupt processing → page 59

Representation of the inputs/outputs of the 32 bit counter

Real inputs

| | |
|---------------------------|----------------------------|
| AT %IX0.0: BOOL; (*Bit0*) | Input for counter pulses |
| AT %IX0.1: BOOL; (*Bit1*) | Input for direction signal |

Representation of the virtual I/Os in the PLC configurator

| | |
|--|---|
| AT %IX1.0: BOOL; (*State*) [CHANNEL(I)] | |
| AT %IX1.1: BOOL; (*N0*) [CHANNEL(I)] | L = no zero crossover, H = zero crossover |
| AT %IX1.2: BOOL; (*N1*) [CHANNEL(I)] | |
| AT %IX1.3: BOOL; (*Error*) [CHANNEL(I)] | H = Error |
| AT %IW2: WORD; (*Counter-Value Low-Word*) [CHANNEL(I)] | Counter state Low Word |
| AT %IW4: WORD; (*Counter-Value High-Word*) [CHANNEL(I)] | Counter state High Word |
| AT %QX1.0: BOOL; (*Reference Window*) [CHANNEL(Q)] | |
| AT %QX1.1: BOOL; (*Reset Counter0*) [CHANNEL(Q)] | Reset to 0 |
| AT %QX1.2: BOOL; (*Reset Counter1*) [CHANNEL(Q)] | |
| AT %QX1.3: BOOL; (*NO Quit*) [CHANNEL(Q)] | Acknowledgement zero crossover |
| AT %QX1.4: BOOL; (*N1 Quit *) [CHANNEL(Q)] | |
| AT %QX1.5: BOOL; (*Error Quit*) [CHANNEL(Q)] | Error acknowledge |
| AT %QX1.6: BOOL; (*32BitCounter/Counter0 Enable*) [CHANNEL(Q)] | L = inhibit count pulse, H = enable count pulse |
| AT %QX1.7: BOOL; (*Counter1 Enable*) [CHANNEL(Q)] | |

Representation of the inputs/outputs of two 16 bit counters

Real inputs

| | |
|---------------------------|--|
| AT %IX0.0: BOOL; (*Bit0*) | Input for counter pulses (counter 0) |
| AT %IX0.1: BOOL; (*Bit1*) | Input for direction signal (counter 0) |
| AT %IX0.2: BOOL; (*Bit2*) | Input for counter pulses (counter 1) |
| AT %IX0.3: BOOL; (*Bit3*) | Input for direction signal (counter 1) |

Representation of the virtual I/Os in the PLC configurator

| | |
|---|--|
| AT %IX1.0: BOOL; (*State*) [CHANNEL(I)] | |
| AT %IX1.1: BOOL; (*N0*) [CHANNEL(I)] | L = no zero crossover, H = zero crossover |
| AT %IX1.2: BOOL; (*N1*) [CHANNEL(I)] | L = no zero crossover, H = zero crossover |
| AT %IX1.3: BOOL; (*Error*) [CHANNEL(I)] | H = Error |
| AT %IW2: WORD; (*Counter-Value Low-Word*) [CHANNEL(I)] | Counter status counter 0 |
| AT %IW4: WORD; (*Counter-Value High-Word*) [CHANNEL(I)] | Counter status counter 1 |
| AT %QX1.0: BOOL; (*Reference Window*) [CHANNEL(Q)] | |
| AT %QX1.1: BOOL; (*Reset Counter0*) [CHANNEL(Q)] | Reset to 0 counter 0 |
| AT %QX1.2: BOOL; (*Reset Counter1*) [CHANNEL(Q)] | Reset to 0 counter 1 |
| AT %QX1.3: BOOL; (*NO Quit*) [CHANNEL(Q)] | Acknowledgement zero for counter 0 |
| AT %QX1.4: BOOL; (*N1 Quit *) [CHANNEL(Q)] | Acknowledgement zero for counter 1 |
| AT %QX1.5: BOOL; (*Error Quit*) [CHANNEL(Q)] | Error acknowledge |
| AT %QX1.6: BOOL; (*Counter0 Enable*) [CHANNEL(Q)] | 0: L = inhibit count pulse, H = enable count pulse |
| AT %QX1.7: BOOL; (*Counter1 Enable*) [CHANNEL(Q)] | 1: L = inhibit count pulse, H = enable count pulse |

Interrupt processing

If an interrupt occurs, the operating system executes the program organisational unit (POU) which is linked to the interrupt source.



Caution!

The execution of the interrupt POU is **not** time monitored. Inadvertently programmed endless loops can't be exited.

A maximum of six interrupt sources IO Interrupt1...6) are supported, which differentiate only by the number at the end of the name.

Interrupt generators are:

- Input I0.4 L → H edge
- Input I0.4 H → L edge
- Input I0.5 L → H edge
- Input I0.5 H → L edge
- 32 bit counter, actual value = setpoint value or
- 16 bit counter (1), actual value = setpoint value
- 16 bit counter (2), actual value = setpoint value

The POU initiated by the interrupt is always run to completion and cannot be interrupted by a new interrupt. A new interrupt is only carried out after the current interrupt has ended.



Attention!

All the outputs controlled (H signals) up to this point remain active and can't be switched off.

The interrupts are enabled in the RUN state of the CPU and inhibited in the STOP state. Interrupt sources which are not enabled in the configuration do not initiate an interrupt. If a POU is not assigned to an enabled interrupt source, the interrupt is recognised and executed but without running a POU.

Frequent occurrence of an interrupt during program execution can cause the programmed task time to time-out and result in a RESET being initiated by the Watchdog.

User interrupts can be inhibited and re-enabled from the program. The functions "DisableInterrupt" and "EnableInterrupt" are provided for this purpose. A call parameter in the easySoft CoDeSys determines if an individual interrupt or all interrupts are enabled or inhibited. A disabled interrupt must be enabled with the same parameter that was used to disable it.

Both the "DisableInterrupt" and "EnableInterrupt" functions are components of the "XC200_Util.lib" library. This library must – if not already done so – be integrated into the library manager of the easySoft CoDeSys.

DisableInterrupt

With this function, you disable (deactivate) a parameterized physical interrupt by accessing it from the user program.

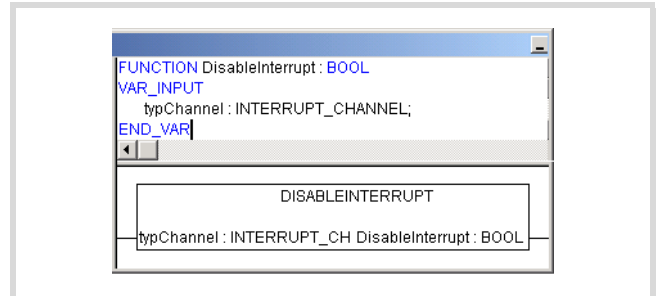


Figure 67: DisableInterrupt function

EnableInterrupt

With this function, the physical interrupt which was deactivated beforehand can now be re-enabled as an active interrupt.

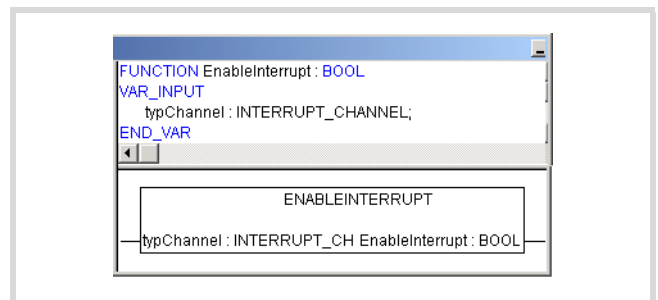


Figure 68: EnableInterrupt function

Parameterization

The parameterization and prioritisation of the interrupt occurs in the "PLC and Task Configuration" of the easySoft CoDeSys (activate the "Resources" tab and call up the "Task configuration → system events" folder). Each interrupt can be assigned with a POU here.

Example for interrupt processing

A "Basic" task contains a POU "PLC_PRG". A further POU "Fastprog" should be processed if an L → H rising edge on the input IO.5 generates an interrupt.

- Create the POU "PLC_PRG" and "Fastprog" as shown in Figure 69.

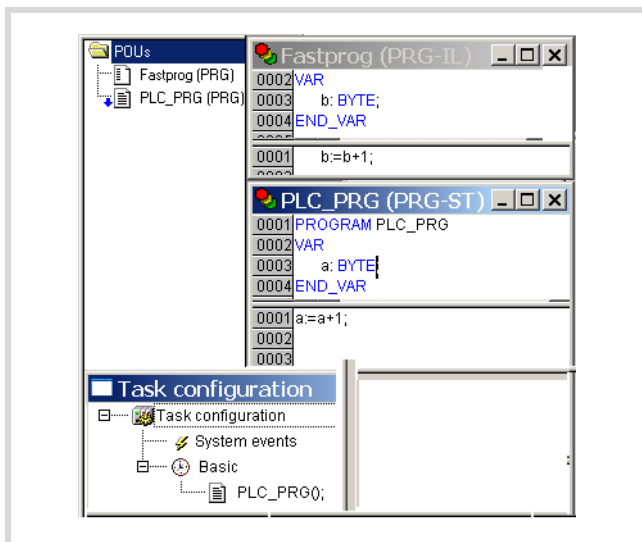


Figure 69: PLC and Task configuration

- Changeover to the PLC configuration and assign input IO.5 (L → H edge) e.g. the interrupt source "IO-Interrupt3" from the drop-down menu.

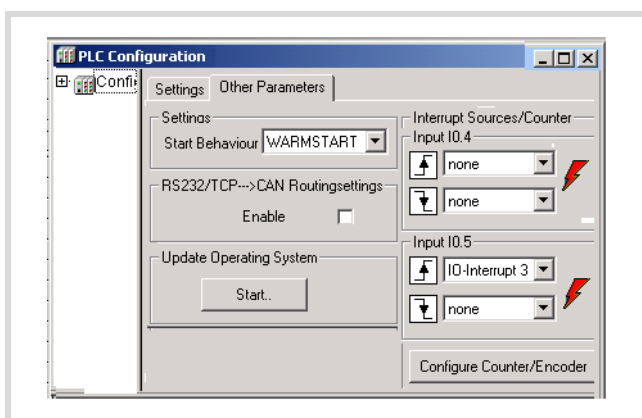


Figure 70: Allocation of IO.5 → interrupt source

Change over to the Task configuration and open the System Events folder.

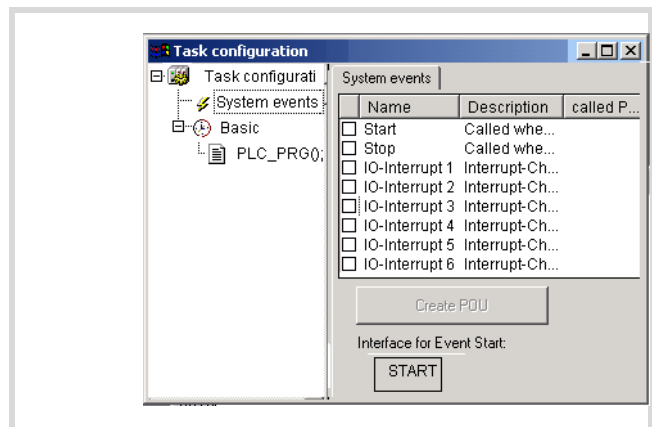


Figure 71: System events

- Enable the IO interrupt3 by clicking in the checkbox beside the name "IO-Interrupt3".
The box is checked to indicate that it has been activated.
- Mark the area of column "Called POU" and the area and the line "IO-Interrupt 3".
- Set the cursor on the marked area and press the function key F2.

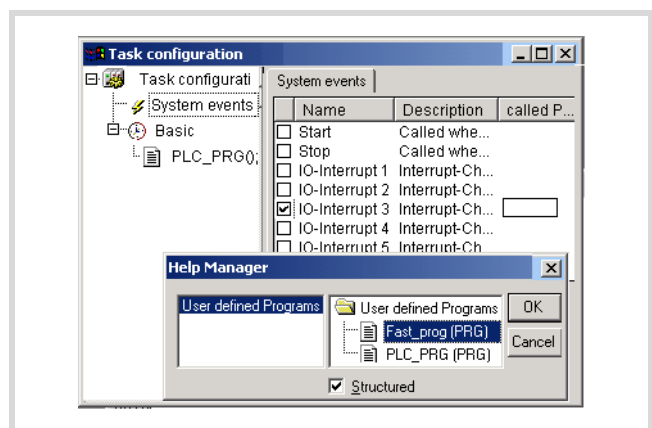


Figure 72: Allocation of Interrupt source → POU

The "Help Manager" window opens in which all predefined programs are listed.

- Select the "Fastprog" POU and confirm with OK.
- Save the project. You can now test it.

The variable "b" is incremented by one with every rising edge on input IO.5.

11 Libraries, function blocks and functions

The libraries contain IEC function blocks and functions that you can use, for example, for the following tasks:

- Data exchange through the CANopen bus
- Controlling the real-time clock
- Determining bus load of the CANopen bus
- Triggering interrupts
- Sending/receiving data through the interfaces

The libraries are located in the following folders:

- Lib_Common for all PLCs
- Lib_CPUxxx for PLCs XC100 and XC200
- Lib_XN_PLC_CANopen for PLC XN-PLC.

Using libraries

When you open a project, the Standard.lib and SYSLIBCALLBACK.lib libraries are copied into the Library Manager. If you need further libraries for your application, you have to install these manually.

The libraries in the Library Manager are assigned to the project after saving. When you open the project, the libraries are then automatically called up as well.

The following overview lists the documents in which the function blocks and functions are described.

| Document | Library |
|--------------------------|---|
| AWB2700-1437 | Standard.lib Util.lib |
| AWB 2724-1453 | XC100_Util. Lib |
| AWB 2724-1491 | XC200_Util. Lib |
| AWB 2724-1566 | XN_PLC_Util. Lib |
| Online help or PDF files | SysLib... lib |
| AWB 2786-1456 | XS40_MoellerFB. Lib/ Visu. Lib/... |
| AN2700K20 | 3S_CANopenDevice. Lib 3S_CANopenManager. Lib |
| AN2700K19 | 3S_CANopenNetVar. Lib |
| AN2700K27 | XC_SysLibCan. Lib XN_PLC_SysLibCan. Lib |
| AWB 2786-1554 | CANUserLib. Lib CANUser_Master. Lib |

Installing additional system libraries

You can install libraries manually as follows:

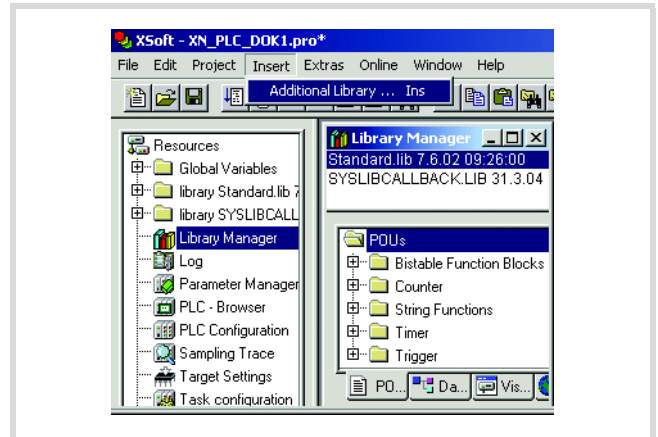


Figure 73: Libraries, installing manually

- In your project, click the “Resources” tab in the object organiser.
- Double-click the “Library Manager” element.
- Click Insert → Additional Library... Ins.

The new window will show the libraries available, depending on the target system.

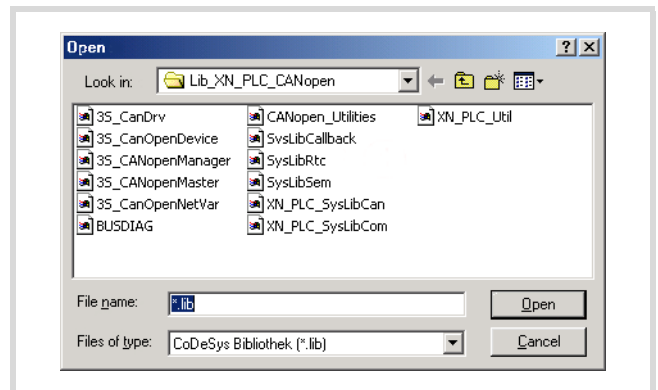


Figure 74: Selecting a library

- Select the library to install and click Open.

The library now appears in the Library Manager.

XC200 specific functions

The XC200 specific functions are contained in the XC200_UTIL.lib library. From operating system version V01.03.xx the XC200_UTIL2.lib library with additional functions has been introduced. The additional functions are described from Page 65.

The functions of the XC200_UTIL.lib library are divided into the following groups:

- CAN_Uilities.
- XI/OC functions
- Event functions

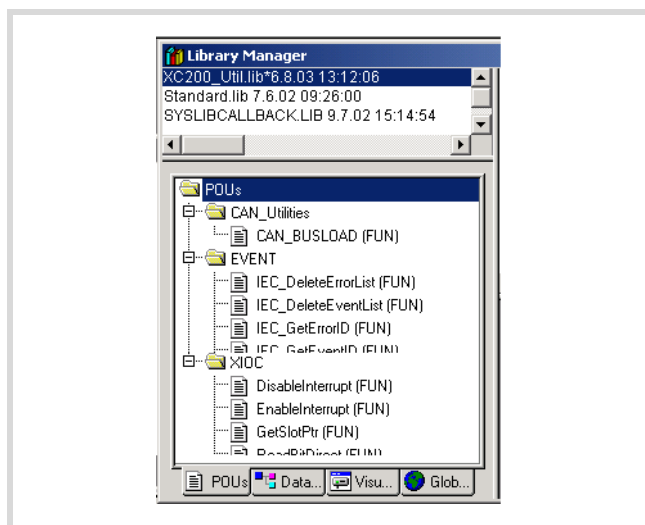


Figure 75: XC200 specific functions of the XC200-UTIL.lib library

CAN_Uilities.

The "CAN_BUSLOAD" function is contained in the "XC200_Util.lib" in the "CAN_Uilities" folder.

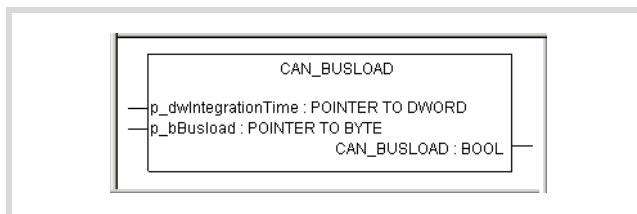


Figure 76: Function CAN_BUSLOAD

This function can be called cyclically in a user program. If a read cycle has been ended successfully, the function returns with TRUE and writes the determined values for the integration time and the bus loading to the transferred addresses.

If the calculation of the bus load is not yet completed or the CAN controller is not yet initialised, the function returns with the FALSE function as a return value.

A read cycle is 500 ms long.

See also:

- Display the loading of the CAN bus (canload) → page 69.

XI/OC functions

The XI/OC functions include functions for processing interrupts (→ page 59) and for programming of the direct peripheral access (→ page 33).

Event functions

Events are special occurrences from the operating system or application. These events are stored in a ring buffer. The following functions allow read and write access to this event (ring) buffer.

IEC_DeleteErrorList

This function erases all error messages listed in the error list.

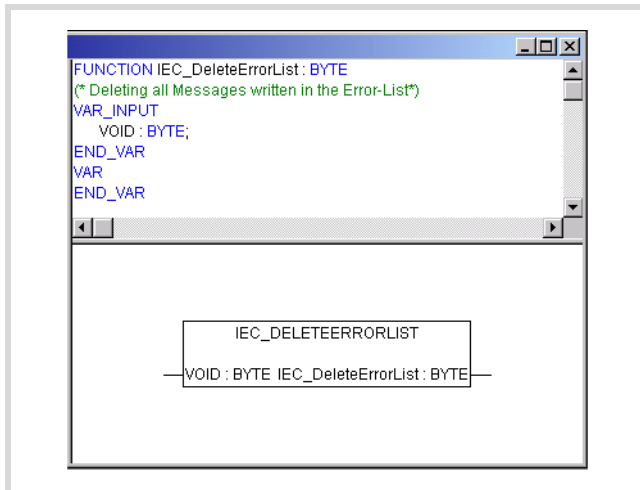


Figure 77: IEC_DeleteErrorList with declaration section function

IEC_DeleteEventList

This function erases all error messages listed in the event list.

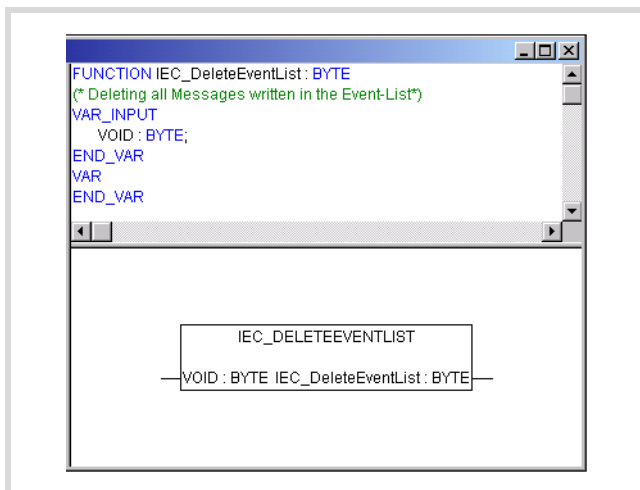


Figure 78: IEC_DeleteErrorList with declaration section function

IEC_GetErrorID

This function returns the Module-ID and Error-ID of the requested error message.

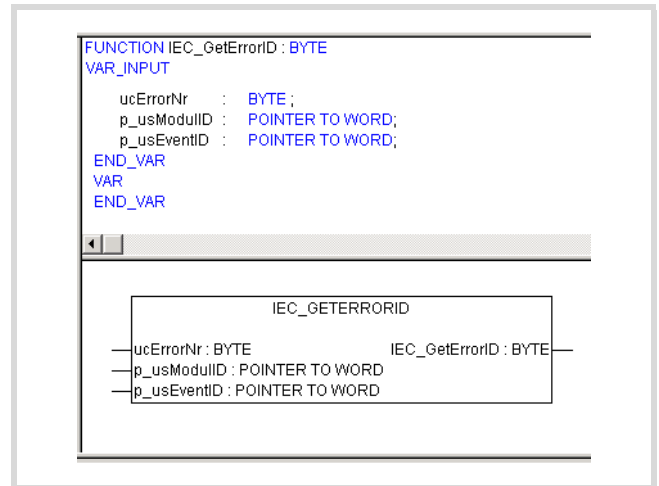


Figure 79: Function IEC_GetErrorID with declaration section

The description of the error messages and the error identity can be found in the Online documentation of the easySoft CoDeSys relating to function IEC_GetErrorID.

IEC_GetEventID

This function returns the Module-ID and Error-ID of the requested event message.

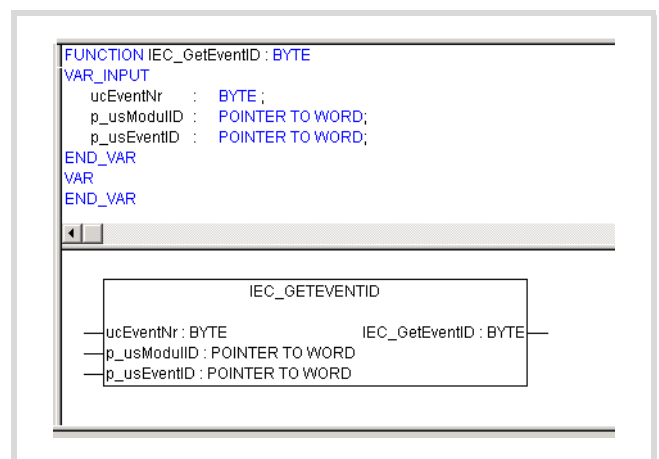


Figure 80: Function IEC_GetEventID with declaration section

The description of the event messages and the event identity can be found in the Online documentation of the easySoft CoDeSys relating to function IEC_GetEventID.

IEC_GetNrOfErrors

This function returns the number of entered error messages.

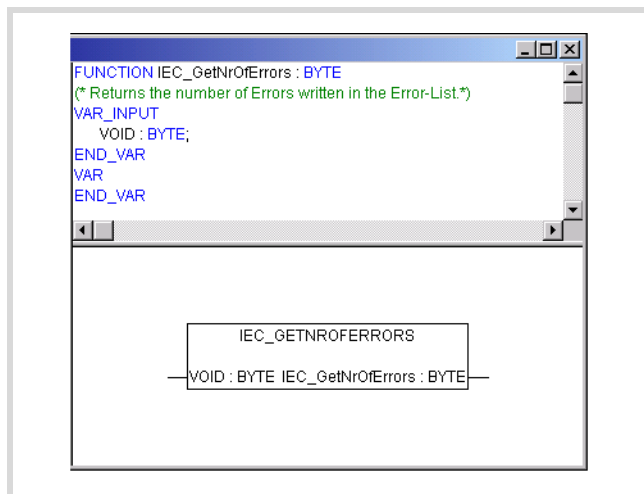


Figure 81: Function IEC_GetNrOfErrors

IEC_WriteError

This function writes an error message into the error list of the control.

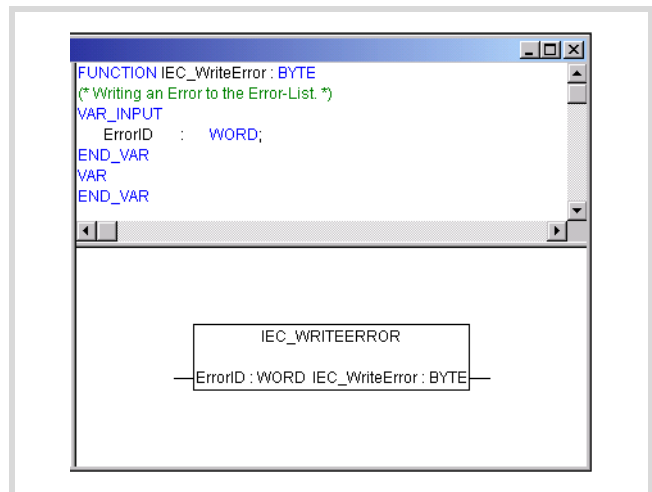


Figure 83: Function IEC_WriteError

IEC_GetNrOfEvents

This function returns the number of entered event messages.

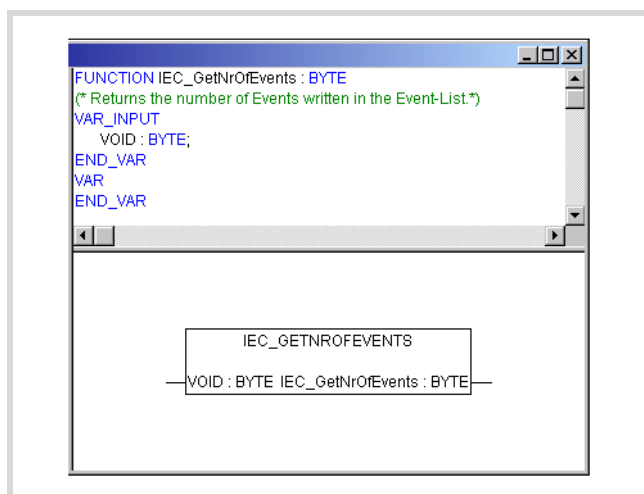


Figure 82: Function IEC_GetNrOfEvents

IEC_WriteEvent

This function writes an event message into the event list of the control.

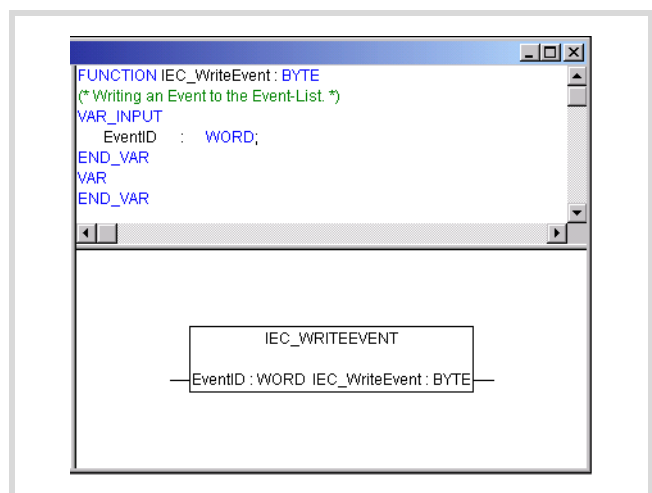


Figure 84: Function IEC_WriteEvent

Additional functions of the XC200_UTIL2.lib

The functions of the XC200_UTIL2.lib can be seen in the following overview:

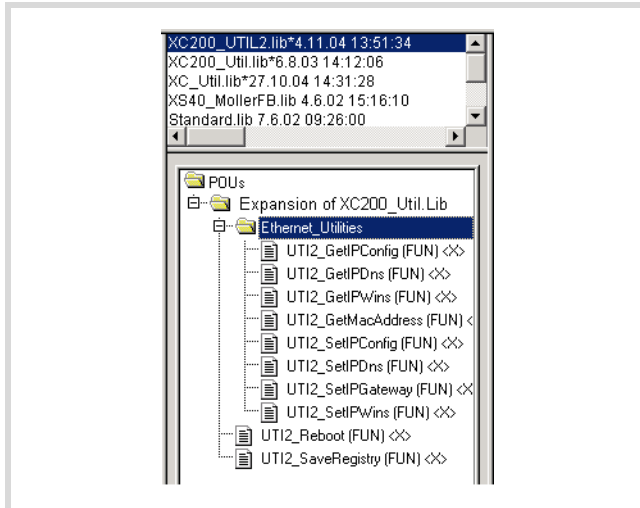


Figure 85: Overview of the XC200_UTIL2.lib

UT12_GetIPConfig

Issue of the IP address, subnetmask address and IP gateway address

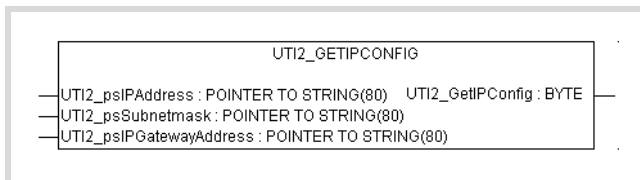


Figure 86: UT12_GetIPConfig

Table 19: Input variables

| Input variables | Meaning |
|--------------------------|---|
| UT12_psIPAddress: | Pointer to a string in which the read IP address is written. |
| UT12_psSubnetmask: | Pointer to a string in which the read address of the subnetmask is written. |
| UT12_psIPGatewayAddress: | Pointer to a string in which the read address of the standard gateway is written. |

Table 20: Return value

| Return value | Meaning |
|--------------|-------------------------------|
| 1 | Read successful. |
| <0 | Read fault (general fault) |
| -4 | No valid pointer transferred. |

UT12_GetMacAddress

Issue of the MAC address (MAC=Media Access Control)

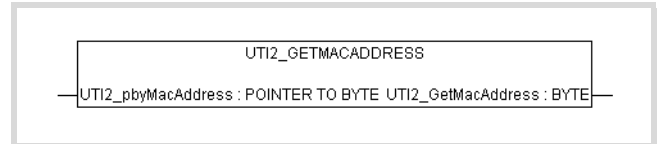


Figure 87: UT12_GetMacAddress

Table 21: Input variables

| Input variables | Meaning |
|--------------------|---|
| UT12_pbyMacAddress | Pointer to an array of 5 byte values, in which the read MAC address is entered. |

Table 22: Return value

| Return value | Meaning |
|--------------|-------------------------------|
| 1 | Read successful. |
| <0 | Read failed (general fault). |
| -4 | No valid pointer transferred. |

UT12_SetIPConfig

Setting of the IP- and subnetmask address

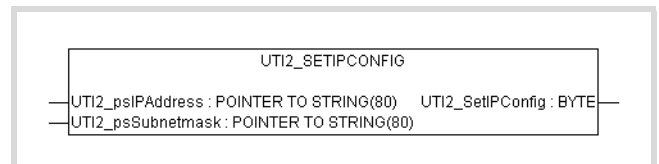


Figure 88: UT12_SetIPConfig



Attention!

A newly entered value must be saved as a non-volatile value by a "SaveRegistry" or a "Reboot" command. The newly entered value is accepted only after a restart of the PLC.

Table 23: Input variables

| Input variables | Meaning |
|-------------------|--|
| UT12_psIPAddress | Pointer to a string variable which contains the IP address to be written. |
| UT12_psSubnetmask | Pointer to a string variable, which contains the value to be entered from the subnet mask. |

Table 24: Return value

| Return value | Meaning |
|--------------|-------------------------------|
| 1 | Write successful. |
| <0 | Write failed (general fault). |
| -4 | No valid pointer transferred. |

UT12_SetIPGateway Setting of the IP Gateway address

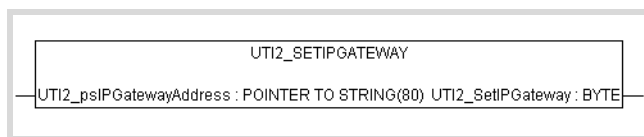


Figure 89: UT12_SetIPGateway



Attention!

A newly entered value must be saved as a non-volatile value by a "SaveRegistry" or a "Reboot" command. The newly entered value is accepted only after a restart of the PLC.

Table 25: Input variables

| Input variables | Meaning |
|-------------------------|--|
| UT12_psiPGatewayAddress | Pointer to a string variable, which contains the value to be entered from the gateway address. |

Table 26: Return value

| Return value | Meaning |
|--------------|-------------------------------|
| 1 | Write successful. |
| <0 | Write failed (general fault). |
| -4 | No valid pointer transferred. |

UT12_Reboot Restart with registry storage

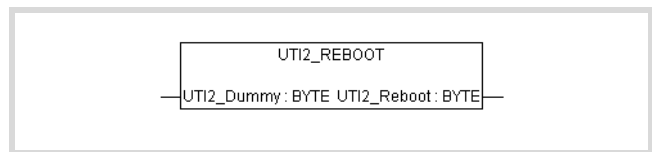


Figure 90: UT12_Reboot

Table 27: Input variables

| Input variables | Meaning |
|-----------------|--|
| UT12_Dummy | A dummy byte which is not evaluated in the function. |

Return value

The value "1" is always returned.

UT12_SaveRegistry Saving of the registry

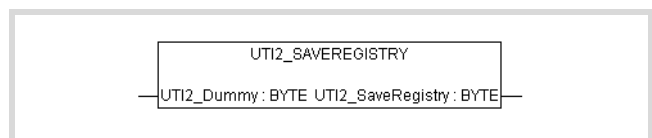


Figure 91: UT12_SaveRegistry

Table 28: Input variables

| Input variables | Meaning |
|-----------------|--|
| UT12_Dummy | A dummy byte which is not evaluated in the function. |

Table 29: Return value

| Return value | Meaning |
|--------------|---|
| 1 | Order successfully entered into the job thread. |
| 2 | Job could not be entered in the job thread. |

12 Browser commands

The PLC browser is a text based control (terminal) monitor. Commands to query certain information from the control are entered in the input line and sent as a string to the control. The answer string is represented in a result window of the browser. This functionality can be used for diagnostic and debugging purposes.

The browser commands which are available for the XC200 target system are listed in the Table 30 alphabetically. They are arranged into two groups:

- Standard PLC 2.3 browser commands (assigned with a grey background in the table)
- Target system specific PLC 2.3 browser commands; these commands are managed in a file and are implemented in the runtime system.

Table 30: Browser commands

| Command | Description |
|-----------------------------|---|
| ? | Get a list of implemented commands. |
| canload | Display of the loading of the CANopen fieldbus |
| clearerrorlist | Erase error list |
| cleareventlist | Erase event list |
| copyprojtommc ¹⁾ | Copy the (boot) project onto a Multi Media Card (incl. directory structure/project directory) |
| copyprojtousb ¹⁾ | Copy the (boot) project onto the USB drive (incl. directory structure/project directory) |
| createstartupini | Create the Startupini file on the disk_sys and disk_mmc |
| delpwd | Erase password for online access |
| dpt | Output data pointer table |
| filecopy ¹⁾ | Copy file |
| filedelete ¹⁾ | Erase file |
| filedir ¹⁾ | Directory list [First folder in the list] |
| filerename ¹⁾ | Rename file |
| getbattery | Display battery status |
| getcomconfig | ...display baud rate of serial interface 1 |
| geterrorlist | Display error list |
| geteventlist | Display event list |
| getipconfig | Display Ethernet address |
| getipdns | Display current DNS address |
| getipgateway | Display Gateway address |
| getipwins | Display current WINS address |
| getlanguage | Display dialog language for the error list |
| getmacaddress | Display MAC address [80-80-99-2-x-x] |
| getprgprop | Read program information |
| getprgstat | Read program status |
| getrtc | Display data and time [YY:MM:DD] [HH:MM:SS] |
| getswitchpos | Display status of the operating switch |
| gettargetname | Display device name |
| getversion | Display version information |
| memdisk_sys | Displays the free memory at "disk_sys" |
| pid | Output project ID |
| pinf | Output project information |

| Command | Description |
|-----------------------------|--|
| plcload ¹⁾ | Display system performance: CPU usage |
| ppt | Output module pointer table |
| reboot | Accept changes (registry save) and restart PLC |
| reflect ¹⁾ | Mirror current command line for test purposes. |
| reload | Reload boot project again |
| removeprojfrommmc | Removes the backup project from the MMC |
| removestartupini | Erases the Startupini file from the disk_sys and disk_mmc |
| resetprg | Reset user program |
| resetprgcold | User program cold reset |
| resetprgorg | Reset user program to original state |
| restoreretain | Restore retentive data from file [File name]" |
| saveregistry | Accept modifications |
| saveretain | Save retentive data in the file [Filename] |
| setcomconfig ¹⁾ | Set the baud rate of the serial interface [setcomconfig 4800,9600,19200, 38400, 57600,115200] e.g.: setcomconfig 38400 |
| setipconfig ¹⁾ | Set Ethernet configuration [setipconfig adr1.adr2.adr3.adr4 mask1.mask2.mask3.mask4] e.g. setipconfig 192.168.119.010 255.255.255.000 |
| setipdns | Set DNS address [setipdns adr1.adr2.adr3.adr4] |
| setipgateway ¹⁾ | Set gateway address [adr1.adr2.adr3.adr4]; e.g.: setipgateway 192.168.119.010 |
| setipwins | Set WINS address [setipwins adr1.adr2.adr3.adr4] |
| setlanguage | Determine dialog language for error list [deu/eng/fra/ita] |
| setpwd | Activate password for online access |
| setrtc ¹⁾ | Set date and time [YY:MM:DD] [HH:MM:SS]; e.g.setrtc 03:07:24 10:46:33 |
| settargetname ¹⁾ | Set device name [devicename]; e.g.: settargetname test |
| shutdown | Accept changes (registry save) and switch off PLC |
| startprg | Start user program |
| stopprg | Stop user program |
| tsk | Output IEC task list with task information |
| updatefrommmc | update windows image from /disk_mmc/MOELLER/XC-CPU201/btsxc201_Vxxxxx.nbk |

1) You can call up help with extended information for these Browser commands in the easySoft CoDeSys. Enter a question mark followed by a space before the command e.g.: ? plcload in the command line of the PLC browser

Calling up browser commands

- Activate the "Resources" tab in the easySoft CoDeSys and select the "PLC browser" folder.
- Click at the top right of the window on the button "..."
- Double-click the required browser command to select it. Add additional parameters to the command if required, such as the Baud rate with "setcomconfig", → table 30.
- The command may require additional parameters.
- Press the Return button.

The result will be displayed.

Accessing communications parameters

Settings of the communication parameters via Browser commands such as device names, Ethernet addresses, gateway addresses or baud rates of the serial interface, are only modified and not directly accepted/saved in the database entry in Windows-CE REGISTRY with the following commands. The function is only accepted after the next Windows CE start.

- setcomconfig
- setipconfig
- setipgateway
- settargetname

After one of these browser commands has been executed, saving of the REGISTRY is necessary. The following Browser commands are available for this purpose.

- saveregistry (saves the registry)
- shutdown (saves the registry and waits for "voltage off")
- reboot (saves the registry and generates a software reset)

The commands "setcomconfig", "setipconfig", "setipgateway", "settargetname" must be supplemented in the command line of the PLC browser, e.g.B. with the Baud rate at "setcomconfig", → table 30. Close the line by pressing RETURN. An answer is received in the window with the grey background.

The "setipconfig" browser command automatically generates a "settargetname". The target name is comprised of a short description of the target system and the last numeric block of the IP address, e.g.: Xc201_Nr010.

The target name is automatically generated according to the IP address and the target system. It can be called via "gettargetname".

Display CPU loading (plcload)

The "plcload" browser command provides information on the current system loading of the CPU.

A loading > 95 % can lead to a failure of the serial and Ethernet communication and/or impairment of the real-time behaviour.

Display the loading of the CAN bus (canload)

The "canload" PLC browser command is a constituent of the "XC200_Util.lib" library. It indicates the loading of the CAN bus.

Examples for display:

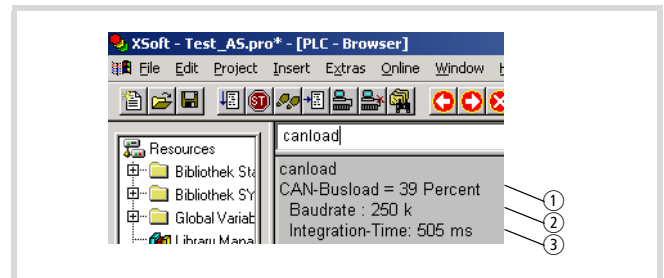


Figure 92: Loading of the CAN bus (Example 1)

- ① Loading of the CAN bus in the last integration interval.
- ② Current baud rate of the CAN bus
- ③ Time via which the loading of the CAN bus has been integrated. The integration time is set by default to 500 ms and can't be changed via the browser.

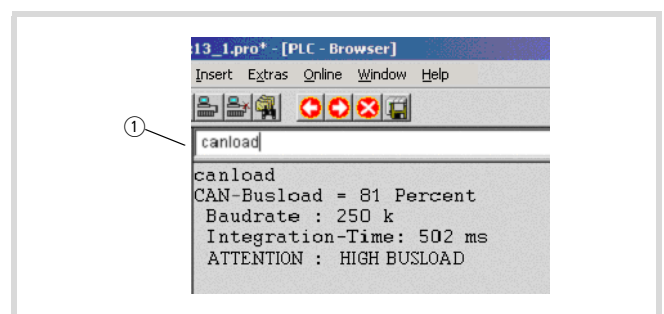


Figure 93: Loading of the CAN bus with warning message (Example 2)

- ① Warning message, → table 31

Table 31: Possible alarm messages

| Alarm message | Meaning |
|-----------------------------------|---------------------------------------|
| ATTENTION: HIGH BUSLOAD | Loading of the CAN bus \geq 75 % |
| CAN bus not activated | The CAN bus is not active |
| CAN-Busload = Invalid Calculation | Monitoring of the bus load has failed |

Access to memory objects

These commands have the name of the memory card, the directory structure and the file names as parameters. Pay close attention to the respective special characters when entering commands.

- filecopy
- FileRename
- FileDelete
- filedir.

Examples:

```
filedir (without parameter details the default setting is: \\disk_sys\\project)
filedir \\disk_sys
filedir \\disk_sys\\project
filedir \\disk_mmc\\MOELLER\\XC-CPU201-EC512k-8DI-6DO
filedir \\disk_mmc\\MOELLER\\XC-CPU201-EC512k-8DI-6DO\\project\\aaa.prg
filedir \\disk_usb\\MOELLER\\XC-CPU201-EC512k-8DI-6DO
filedir \\disk_usb\\MOELLER\\XC-CPU201-EC512k-8DI-6DO\\project\\bbb.prg
filecopy \\disk_sys\\project\\default.prg \\disk_sys\\project\\yyy.prg
filerename \\disk_sys\\project\\yyy.prg \\disk_sys\\project\\xxx.prg
filecopy \\disk_sys\\project\\default.prg \\disk_mmc\\MOELLER\\XC-CPU201-EC512k-8DI-6DO \\project\\default.prg
filedelete \\disk_mmc\\MOELLER\\XC-CPU201-EC512k-8DI-6DO\\project\\default.prg
```

→ If the CPU "XC-CPU201-EC256K-8DI-6DO" is available, the instruction section "512" is replaced by "256".

Error and event list after calling browser commands

The dialog language for error and event lists is available in German, English, French and Italian.

The active language is displayed with "getlanguage", the conversion of the language is implemented with "setlanguage".

Examples for language conversion:

If the error and event list is to be displayed in German, the "setlanguage deu" browser command should be entered. The input is ended with "Return". You receive the following displayed window.

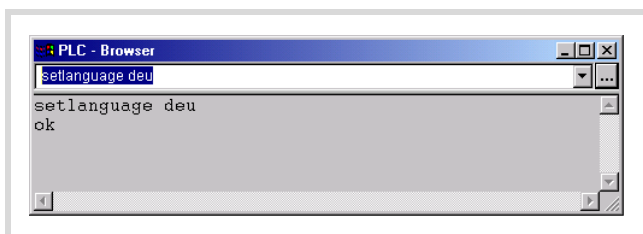


Figure 94: Browser command "setlanguage"

| Module ID | Program type |
|-----------|---------------------------------|
| 1 | RTS (runtime system) |
| 2 | CST (Moeller specific adaption) |
| 3 | XIO (XI/OC) |
| 4 | CAN |
| 5 | IEC |

TheEvent-ID defines the fault number of the program. The error number can start at 0 for every module ID.

The following is an overview of the messages which can occur in the browser error and event lists. The module ID indicates which program type the fault signals:

| Module ID | Event-ID | Fault signal |
|-----------|----------|--|
| 2 | 1 | Stop program |
| 2 | 2 | Start program |
| 2 | 3 | Reset warm |
| 2 | 4 | Cold reset |
| 2 | 5 | Reset Hard |
| 2 | 6 | Battery empty |
| 2 | 7 | No program loaded |
| 2 | 8 | Task monitoring |
| 4 | 10 | CAN controller started |
| 4 | 20 | CAN controller stopped |
| 4 | 30 | Overflow |
| 4 | 31 | Overflow |
| 4 | 40 | Overflow |
| 4 | 41 | Overflow |
| 4 | 42 | Overflow |
| 4 | 50 | Critical CAN fault |
| 4 | 60 | CAN controller in status error warning |
| 4 | 70 | CAN controller in status Bus-Off |
| 1 | 16 | Task monitoring fault |
| 1 | 17 | Hardware monitoring fault |
| 1 | 18 | Bus error |
| 1 | 19 | Checksum error |
| 1 | 20 | Fieldbus error |
| 1 | 21 | I/O update fault |
| 1 | 22 | Cycle time exceeded |
| 1 | 80 | Invalid instruction |
| 1 | 81 | Access violation |
| 1 | 82 | Privileged instruction |
| 1 | 83 | Page fault |
| 1 | 84 | Stack overflow |
| 1 | 85 | Invalid scheduling |
| 1 | 86 | Invalid access Identity |
| 1 | 87 | Access on protected page |
| 1 | 256 | Access to uneven address |
| 1 | 257 | Array limit exceeded |
| 1 | 258 | Division by zero |
| 1 | 259 | Overflow |
| 1 | 260 | Exception cant be overlooked |
| 1 | 336 | Floating decimal point: General fault |
| 1 | 337 | Floating decimal point: Not normalised operand |
| 1 | 338 | Floating decimal point: Division by zero |

| Module ID | Event-ID | Fault signal |
|-----------|----------|---|
| 1 | 339 | Floating decimal point: Inexact result |
| 1 | 340 | Floating decimal point: Invalid instruction |
| 1 | 341 | Floating decimal point: Overflow |
| 1 | 342 | Floating decimal point: Stack verification error |
| 1 | 343 | Floating decimal point: Underflow |

Appendix

Characteristic of the Ethernet cable

Only use the intended cable type for wiring the Ethernet network. The cable must be at least category "CAT-5" compatible. "CAT-5" cables are suitable for data transfer rates of between 10 and 100 MBit/s.

Table 32: Characteristics of the Ethernet cable

| | UTP ¹⁾ | STP ²⁾ | SSTP ³⁾ |
|------------------------|---------------------------|----------------------------|---|
| Transmission medium | Unshielded Twisted Pair | Shielded Twisted Pair | |
| Transmission speed | 10 MBit/s, 100 MBit/s | 10 MBit/s, 100 MBit/s | 10 MBit/s, 100 MBit/s |
| Setup | Stranded every two cores, | | |
| | without screen | with full screen | with full screen, each core pair is additionally screened |
| Flexibility | average | average | average |
| Screen | none | single | double |
| Topology | Point-to-point | Point-to-point, line, star | |
| Maximum segment length | 100 m | 100 m | 100 m |

- 1) Use in industrial environments is not recommended due to poor EMC characteristics.
- 2) The conductor pairs are shrouded in a full shield. The task of the full screen is to prevent external interference. This cable is (conditionally) suitable for industrial use due to the high crosstalk values between the individual conductor pairs.
- 3) Unlike the STP cable, this cable has a separate internal shield for every conductor pair. This significantly reduces the crosstalk values and the cable also demonstrates a good level of protection against EMC. This characteristic makes the SSTP cable particularly good for industrial use.

The maximum segment length is 100 meters. If the network is larger, suitable infrastructure components should be used. Transceivers, Hubs and Switches are particularly suitable.

The cable to be selected depends on the the ambient conditions at the installation location (interference, flexibility, transmission speed).

The installation guidelines for the (Ethernet) wiring are described in ISO/IEC 11801 and EN50173.

Properties of the CANopen cable

Use only cable approved for CANopen applications and with the following characteristics:

- Characteristic impedance 100 to 120 Ω
- Capacitance < 60 pF/m

The demands placed on the cable, connectors and bus termination resistors are specified in ISO 11898. Following you will find some demands and stipulations listed for the CANopen network.

In Table 34, standard parameters for the CANopen network with less than 64 CANopen slaves are listed (table complies with the stipulations of the ISO 11898).

The length of the CANopen bus cable is dependant on the conductor cross-section and the number of bus users connected. The following table includes values for the bus length in dependance on the cross-section and the connected bus users, which guarantee a secure bus connection (table corresponds with the stipulations of the ISO 11898).

Table 33: Cable cross-section, bus length and number of bus slaves conform to ISO 11898

| Cable cross-section [mm] | Maximum length [m] | | |
|--------------------------|--------------------|--------|---------|
| | n = 32 | n = 64 | n = 100 |
| 0.25 | 200 | 170 | 150 |
| 0.5 | 360 | 310 | 270 |
| 0.75 | 550 | 470 | 410 |

n = number of connected bus users

If the bus length is greater than 250 m and/or are more than 64 nodes are connected, the ISO 11898 demands a residual ripple of the supply voltage of $\leq 5\%$.

As the bus cable is connected directly to the COMBICON connector of the CPU, additional details concerning stub lines are not required.

The bus users are configured in the "PLC Configuration" window of the CPU in the programming software.

Cable recommendation: LAPP cable, UNITRONIC-BUS LD

Table 34: Standard parameters for CANopen network cable according to the ISO 11898

| Bus length [m] | Loop resistance [mΩ/m] | Core cross-section [mm ²] | Bus termination resistor [Ω] | Transfer rate with cable length [kBit/s] |
|----------------|------------------------|---------------------------------------|------------------------------|--|
| 0 – 40 | 70 | 0.25 – 0.34 | 124 | 1000 at 40 m |
| 40 – 300 | < 60 | 0.34 – 0.6 | 150 – 300 | > 500 at 100 m |
| 300 – 600 | < 40 | 0.5 – 0.6 | 150 – 300 | > 100 at 500 m |
| 600 – 1000 | < 26 | 0.75 – 0.8 | 150 – 300 | > 50 at 1000 m |

Transparent mode: Text output via RS232 (example)

The example shows a text output via the RS-232 interface of the CPU in transparent mode.

```

PROGRAM PLC_PRG
VAR
    BRAKE:TONE;
    STEP:UINT;
    dwSioHandle: DWORD;
    WriteBuffer:STRING(26);
    nWriteLength: DWORD;
    typComSettings:COMSETTINGS;
    typComSetSettings:BOOL;
    out AT %QB0:BYTE;
    INP AT %IX0.0:BOOL;
    STEPERR: UINT;
    Closeserresult: BOOL;
    Coun: DWORD;
    RESET: BOOL;
END_VAR

(*Cycle time/Cycletime: 50ms!*)
CASE STEP OF
0: IF INP =1 THEN (*Start: IX0.0 = TRUE*)
    STEP:=1;
END_IF
1: (*Öffnen/Open*)
    IF dwSioHandle=0 THEN
        dwSioHandle:=xSysComOpen(Port:=Com1);
        IF (dwSioHandle>0) THEN
            typComSettings.typBaudRate           :=Baud_9600;
            typComSettings.typDataLength         :=Data_8Bit;
            typComSettings.typParity              :=NO_PARITY;
            typComSettings.typPort                :=COM1;
            typComSettings.typStopBits            :=ONE_STOPBIT;
            xSysComSetSettings(dwHandle:=dwSioHandle,
                               ComSettings:=ADR(typComSettings));
            STEP :=2;
            RESET:=TRUE;
        ELSE
            STEPERR:=STEP;
            STEP:=99;
        END_IF
        WriteBuffer:='This is the sent text';
    END_IF

```

```

2: (*Ausgabe/Output*)
  IF (dwSioHandle>0) THEN
    nWriteLength:=xSysComWrite(dwHandle:=dwSioHandle,
      dwBufferAddress:=ADR(WriteBuffer),
      dwBytesToWrite:=LEN(WriteBuffer)+1,dwTimeOut:=0);
  END_IF
  IF nWriteLength = LEN(WriteBuffer)+1 THEN
    STEP:=3;
    Coun:=coun+1;
  END_IF
3: (*Schliessen/Shut*)
  Closeresult:=xSysComClose(dwHandle:=dwSioHandle);
  IF (Closeresult = TRUE) THEN
    dwSioHandle:=0;
    STEP:=4;
  ELSE
    STEPERR:=STEP;
    STEP:=99;
  END_IF
4: (*Verzögerung/Delay*)
  BRAKE(IN:=1, PT:=T#2s);
  IF BRAKE.Q = 1 THEN
    STEP :=5;
    BRAKE(IN:=0, PT:=T#2s);
  END_IF
5: (*End*)
  STEP:=0;
99: (*Fehler/Error*)
  STEPERR:=STEPERR;
END_CASE

```

Access to the CPU drives/memory card

“SysLibFile.lib” library

The “SysLibFile” library allows you access to the file system of the XC200, the MMC and the USB stick. The library contains the following functions:

- SysFileClose
- SysFileCopy
- SysFileDelete
- SysFileEOF
- SysFileGetPos
- SysFileGetSize
- SysFileGetTime
- SysFileOpen
- SysFileRead
- SysFileRename
- SysFileSetPos
- SysFileWrite

→ Information about these functions can be found in the online documentation of the programming system under the easySoft-CoDeSys programming system under the “SysFile<Function>” search term.

Attention!

- The PLC may not be switched off when files from the “Multimedia Card” or the “USB memory card” are opened.
- A voltage failure when a file is opened can destroy the memory card
- All the open files must be closed before switch off of the voltage.

Mode for opening a file

“w” mode

The “w” mode opens the file in write mode. An existing file with this name will be overwritten.

Attention!

If you open a file with “w” and close it again, this file is overwritten and a file length of 0 bytes is generated.

“r” mode

The “r” mode opens the file for reading. The file handle which is returned by the “SysFileOpen” function is invalid if this file does not exist. The value “-1” or “16#FFFFFFFF” is then displayed.

The file is opened for sequential reading and with each read access, the read position will be advanced by the number of bytes which have been read.

“a” mode

The “a” mode (append) opens a file in the “w” mode. When data is written to this file, then new text is added to the end of the file.

The “SysFileRead” and “SysFileWrite” functions are each transferred with a buffer and a file handle return value from the “SysFileOpen” function.

In order to close a file, the “SysFileClose” is called with the return value from the “SysFileOpen” function.

Examples of the “SysFile...” functions

The “SysFileOpen” file is used to open a file. The function receives the file names – complete with file path – transferred to it. Furthermore, the function receives the mode in which the file should be opened.

Open in “r” mode

```
OpenFile1 := SysFileOpen('disk_sys\project\File1','r');
```

Open in “w” mode

```
OpenFile2 := SysFileOpen('disk_mmc\MOELLER\XC-CPU201-EC512k-8DI-6DO\Project \File2','w');
```

Open in “a” mode

```
OpenFile3 := SysFileOpen('disk_usb\MOELLER\XC-CPU201-EC512k-8DI-6DO\Project \File3','a');
```

In order to close a file again, the “SysFileClose” function is called.

```
CloseFile:=SysFileClose(OpenFile2);
CloseFile:=SysFileClose(OpenFile3);
```

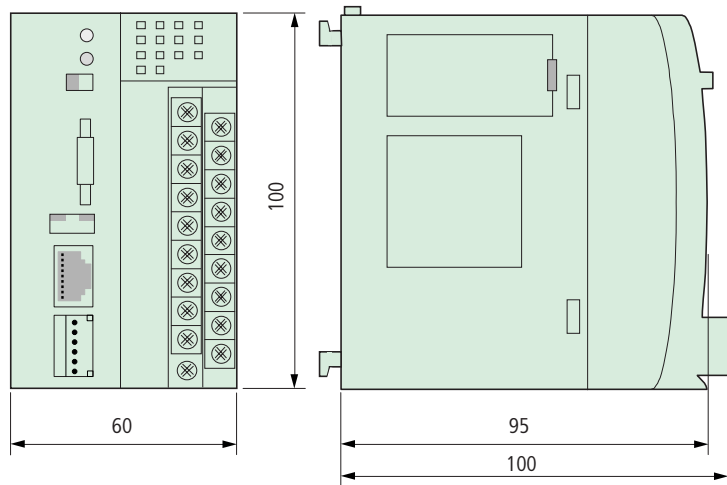

USB stick types

| | | | | | | | | | |
|-------------------------------|-------------------|---------------------|-----------------|-----------------|---------------------------|--------------------|--------------------|---------------------|---------------------|
| Manufacturer | Aiptec | extremem ory | Traxdata | ScanDisk | Kingston | | Fuj./Siemens | ScanDisk | ScanDisk |
| Type designation | MP3/WMA player | | EZ Drive 2.0 | cruzer micro | Data Traveler ELITE | Pen Drive | Memorybird | Cruzer Mini | Cruzer Mini |
| USB specification | 1.1 | 2.0 | 2.0 | 2.0 | 2.0 | 2.0 | 2.0 | 2.0 | 2.0 |
| Memory size [MByte] | 256 | 256 | 256 | 256 | 256 | 256 | 256 | 128 | 256 |
| Mount time [ms] ¹⁾ | 65 | 20463 ²⁾ | 98 | 97 | 7132 ²⁾ | 7732 ²⁾ | 7067 ²⁾ | 20530 ²⁾ | 20564 ²⁾ |

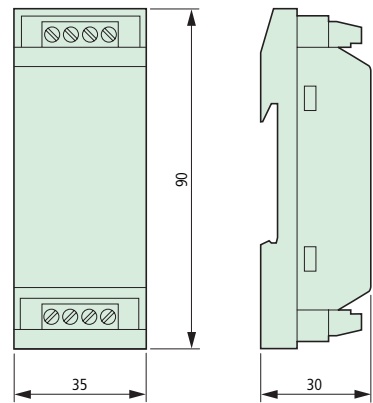
- 1) Time which the operating system requires in order to integrate the USB stick
- 2) Control access of user programs to USB sticks with higher mount times so that the mount time will time out before access to stick/start of the program.

Dimensions

XC-CPU201...



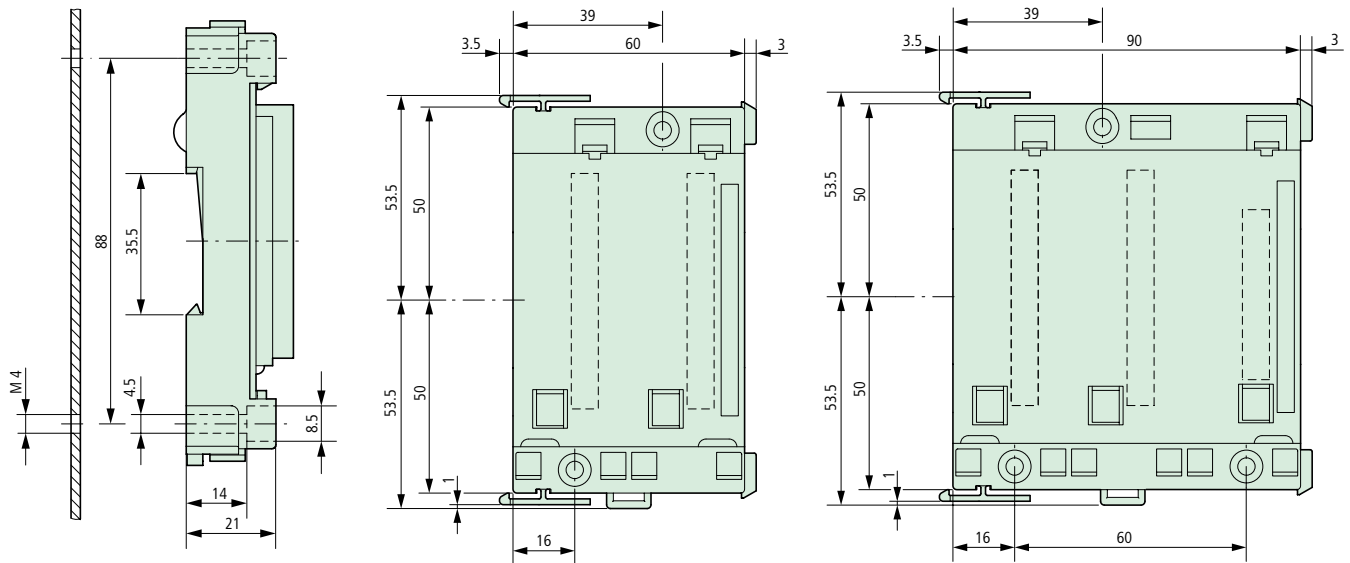
XT-FIL-1 line filter



Racks

XIOC-BP-XC

XIOC-BP-XC1



Technical data

| | | | XC-CPU201-EC256-8DI-6DO(-XV) XC-CPU201-EC512-8DI-6DO(-XV) |
|---|-----------------|--|--|
| General | | | |
| Standards | | | IEC/EN 61131-2 EN 50178 |
| Ambient temperature | °C | | 0 to +55 |
| Storage temperature | °C | | –25 to +70 |
| Mounting position | | | horizontal |
| relative humidity, non-condensing (IEC/EN 60068-2-30) | % | | 10 to 95 |
| Air pressure (operation) | hPa | | 795 to 1080 |
| Vibration resistance | | | 10 to 57 Hz ± 0.075 mm 57 to 150 Hz ± 1.0 g |
| Mechanical shock resistance | | | 15 g/11 ms |
| Overvoltage category | | | II |
| Pollution degree | | | 2 |
| Degree of protection | | | IP20 |
| Rated insulation voltage | V | | 500 |
| Emitted interference | | | EN 50081-2, Class A |
| Noise immunity | | | EN 50082-2 |
| Battery (lifetime) | | | Worst case 3 years, typical 5 years |
| Weight | kg | | 0.23 |
| Dimensions (W × H × D) | mm | | 90 × 100 × 100 |
| Terminals | | | Plug-in terminal block |
| Terminal capacity | | | |
| Screw terminals | | | |
| Flexible with ferrule | mm ² | | 0.5 to 1.5 |
| solid | mm ² | | 0.5 to 2.5 |
| Springloaded terminals | | | |
| stranded | mm ² | | 0.14 to 1.0 |
| solid | mm ² | | 0.34 to 1.0 |
| Electromagnetic compatibility (EMC) | | | → page 82 |

| | | | XC-CPU201-EC256-8DI-6DO(-XV) XC-CPU201-EC512-8DI-6DO(-XV) |
|--|-------|---------|--|
| Supply voltage for the CPU (24 V/0 V) | | | |
| Mains failure bridging | | | |
| Dropout duration | | ms | 10 |
| Repeat rate | | s | 1 |
| Input rated voltage | | V DC | 24 |
| Permissible range | | V DC | 20.4 to 28.8 |
| Current consumption | | A | typ. 1.4 |
| Residual ripple | | % | ≤ 5 |
| Maximum power dissipated (without local I/O) | P_v | W | 6 |
| Overvoltage protection | | | yes |
| Reverse polarity protection | | | yes |
| External supply filter | | | Type: XT-FIL-1, see the technical data on Page 83 |
| Internal supply filter | | | yes |
| Inrush current | | × I_n | No limitation (limited only by upstream 24 V DC power supply unit) |
| Output voltage for the signal modules | | | |
| Output rated voltage | | V DC | 5 |
| Output current | | A | 3.2 |
| Off-load stability | | | yes |
| Short-circuit proof | | | yes |
| Electrically isolated from supply voltage | | | no |

| | | | XC-CPU201-EC256-8DI-6DO(-XV) XC-CPU201-EC512-8DI-6DO(-XV) |
|--|----------|--|---|
| CPU | | | |
| Microprocessor | | | RISC processor |
| Memory | | | |
| Program code (EC256K/EC512K) | kByte | | 512/2048 from OS V1.04.01 |
| Program data (EC256K/EC512K) | kByte | | 256/512 |
| Marker (EC256K/EC512K) | kByte | | 16/16 |
| Retain data (EC256K/EC512K) | kByte | | 32/32 |
| Persistent data (EC256K/EC512K) | kByte | | 32/32 |
| Watchdog | | | yes |
| RTC (real-time clock) | | | yes |
| Interfaces | | | |
| Multimedia card | | | Yes, optional, 16 MB or 32 MB, to be ordered separately |
| Ethernet interface | | | |
| Data transmission rate | MBit/s | | 10/100 |
| Terminations | | | RJ 45 |
| galvanic separation | | | no |
| RS -232 serial interface (without handshake line) | | | |
| Data transmission rate | Bit/s | | 4800, 9600, 19200, 38400, 57600, 115200 |
| Terminations | | | RJ 45 |
| galvanic separation | | | no |
| in the "transparent mode" | | | |
| Data transmission rate | Bit/s | | 300, 600, 1200, 2400, 4800, 9600, 19200, 38400, 57600, 115200 Bit/s |
| Character formats | | | 8E1, 8O1, 8N1, 8N2 |
| CAN(open)/easyNet | | | |
| Data transmission rate | kBits/s | | 20/50/100/125/250/500/800/1000 |
| Potential isolation | | | yes |
| Device profile | | | to DS301V4 |
| PDO type | | | asyn., cyc., acyc. |
| Connection | | | Plug-in spring-loaded terminal block, 6-pole |
| Bus terminating resistors | | | External |
| Stations | Quantity | | max. 126 |
| USB interface, V1.1 | | | |
| Data transfer rate (Autochanging) | MBit/s | | 1.5/12 |
| Potential isolation | | | no |
| Power supply for connected devices: | | | |
| Rated operational voltage | V DC | | 5 |
| max. current | A | | 0.5 |
| Terminations | | | Downstream plug |

| | | | XC-CPU201-EC256-8DI-6DO(-XV) XC-CPU201-EC512-8DI-6DO(-XV) |
|---|----------|--|--|
| Supply voltage for the local inputs/outputs (24 V _Q /0 V _Q) | | | |
| Rated operational voltage | V DC | | 24 |
| Voltage range | V DC | | 20.4 to 28.8 |
| Current consumption | A | | normally 1 |
| Potential isolation | | | |
| Between supply and CPU voltage | | | yes |
| Overvoltage protection | | | yes |
| Reverse polarity protection | | | yes |
| Digital inputs | | | |
| Input rated voltage | V DC | | 24, observe polarity |
| Voltage range | V DC | | 19.2 to 30 |
| Input current per channel at rated voltage | | | |
| Functionality: Normal digital input | mA | | normally 3.5 |
| Functionality: Fast digital input | mA | | normally 7 |
| Power dissipation per channel | | | |
| Functionality: Normal digital input | mW | | normally 85 |
| Functionality: Fast digital input | mW | | normally 168 |
| Switching levels as per EN 61131-2 | | | |
| Limit values type "1" | V DC | | low < 5, high > 15 |
| Input delay | | | |
| Functionality: Normal digital input | | | |
| Off → On | ms | | normally 0.1 |
| On → Off | ms | | normally 0.1 |
| Functionality: Fast digital input | | | |
| Off → On | μs | | normally 7 |
| On → Off | μs | | normally 1 |
| Inputs | Quantity | | 8 |
| Channels with common reference potential | Quantity | | 8 |
| Of which can be used as | | | |
| Interrupt inputs | Quantity | | 2 |
| Counter input 32 Bit or | Quantity | | 1 |
| Counter input 16 Bit or | Quantity | | 2 |
| Incremental encoder input (Track A, B, C) | Quantity | | 1 |
| Status indication | | | LED |

| | | | XC-CPU201-EC256-8DI-6DO(-XV) XC-CPU201-EC512-8DI-6DO(-XV) |
|--|----------|--|--|
| Digital outputs | | | |
| Power dissipation per channel | | | |
| QX0.0 and QX0.5 | W | | 0.08 |
| Load circuits | | | |
| QX0.0 and QX0.5 | A | | 0.5 |
| Output delay | | | |
| Off → On | | | typ 0.1 ms |
| On → Off | | | typ 0.1 ms |
| Channels | Quantity | | 6 |
| Channels with common reference potential | Quantity | | 6 |
| Status indication | | | LED |
| Duty factor | % DF | | 100 |
| Utilization factor | g | | 1 |

| Electromagnetic compatibility | | | | |
|---|---|---------------|--|--------|
| Noise immunity | | | | |
| ESD (IEC/EN 61000-4-2) | Contact discharge | | | 4 kV |
| | Air discharge | | | 8 kV |
| RFI (IEC/EN 61000-4-3) | AM (80 %) | 80 – 1000 MHz | | 10 V/m |
| GSM mobile (IEC/EN 61000-4-3) | PM | 800 – 960 MHz | | 10 V/m |
| Burst (IEC/EN 61000-4-4) | Network/digital I/O (direct) | | | 2 kV |
| | Analog I/O, fieldbus (capacitive coupling) | | | 1 kV |
| Surge (IEC/EN 61000-4-5) | Digital I/O, unsymmetrical | | | 0.5 kV |
| | Analog I/O, unsymmetrical, coupling on the screen | | | 1 kV |
| | Mains DC, unsymmetrical | | | 1 kV |
| | Mains DC, symmetrical | | | 0.5 kV |
| | Mains AC, unsymmetrical | | | 2 kV |
| | Mains AC, symmetrical | | | 1 kV |
| Cable conducted interference, induced by high frequency fields (previously: immunity to line-conducted interference) (IEC/EN 61000-4-6) | | | | 3 V |

| | | | 24 V DC filter XT-FIL-1 |
|---|-----------------|--|--|
| General | | | |
| Standards | | | IEC/EN 61131-2 EN 50178 |
| Ambient temperature | °C | | 0 to +55 |
| Storage | °C | | –25 to +70 |
| Mounting position | | | Horizontal/vertical |
| relative humidity, non-condensing (IEC/EN 60068-2-30) | % | | 10 to 95 |
| Air pressure (operation) | hPa | | 795 to 1080 |
| Vibration resistance | | | 10 to 57 Hz ± 5 mm 57 to 150 Hz ± 1.0 g |
| Mechanical shock resistance | | | 15 g/11 ms |
| Impact resistance | | | 500 g/∅ 50 mm ± 25 g |
| Overvoltage category | | | II |
| Pollution degree | | | 2 |
| Degree of protection | | | IP20 |
| Rated impulse voltage | V | | 850 |
| Emitted interference | | | EN 50081-2, Class A |
| Noise immunity | | | EN 50082-2 |
| Weight | g | | 95 |
| Dimensions (W × H × D) | mm | | 35 × 90 × 30 |
| Terminals | | | Screw terminal |
| Terminal capacity | | | |
| Screw terminals | | | |
| Flexible with ferrule | mm ² | | 0.2 to 2.5 (AWG22-12) |
| solid | mm ² | | 0.2 to 2.5 (AWG22-12) |
| Voltage supply | | | |
| Input voltage | V DC | | 24 |
| Permissible range | V DC | | 20.4 to 28.8 |
| Residual ripple | % | | ≤ 5 |
| Overvoltage protection | | | yes |
| Potential isolation | | | |
| Input voltage to PE | | | yes |
| Input voltage to output voltage | | | no |
| Output voltage to PE | | | yes |
| Output voltage | V DC | | 24 |
| Output current | A | | 2.2 |

Index

| | | |
|----------|---|--------|
| A | ABS enclosure, installation instructions | 16 |
| | Addressing | |
| | Inputs/outputs and markers | 37 |
| | PLC on CAN Bus | 48 |
| B | Backup time, battery | 9 |
| | Battery | 9 |
| | Baud rate specifying/changing | 41 |
| | Baud rates | 48 |
| | Block size, for data transfer | 47 |
| | Boot project | 25 |
| | Breakpoint | 23 |
| | Browser commands | 67 |
| C | Cable routing | 15 |
| | CAN | |
| | -Device parameters | 49 |
| | Interface | 10 |
| | Interface, assignment | 19 |
| | -Master routing settings | 49 |
| | Telegrams, receive/send from user program | 11 |
| | CAN stack | 30 |
| | CANopen cable, properties | 73 |
| | Changing settings | 41 |
| | Character formats | 51 |
| | CoDeSys gateway server | 48 |
| | COLDSTART (start behaviour) | 22 |
| | Commissioning | 23 |
| | Communication | |
| | Channel | 42, 50 |
| | Error | 42 |
| | Parameter | 41 |
| | Communication with the target PLC | 49 |
| | Configuration, inputs/outputs | 53 |
| | Connection | |
| | Incremental encoder | 17 |
| | Inputs/outputs | 17 |
| | Interrupt actuators | 17 |
| | PC | 18 |
| | Power supply and local inputs/outputs | 8 |
| | Up/down counter | 17 |
| | Voltage supply | 16 |
| | Connection establishment | 41 |
| | Control panel layout | 15 |
| | Counter | 57 |
| | CPU | |
| | Display of the operating states | 23 |
| | Functional areas | 7 |
| | Load | 69 |
| | Cyclic task | 27 |
| D | Data transfer, block size | 47 |
| | Data-saving | 9 |
| | Debugging | 23, 67 |
| | Decrementing | 11, 17 |
| | Diagnostics | 39, 67 |
| | via CAN | 50 |
| | Dialog language, for errors and event lists | 70 |
| | Dimensions | 77 |
| | Direct peripheral access | 33 |
| | Error code | 36 |
| | Documentation, online | 6 |
| | Down counter | 57 |
| | Connection | 17 |
| | Download of programs | 24 |
| | DownloadWaitTime | 24 |
| | Drives | 9, 76 |
| E | easyNET interface | 10, 19 |
| | Electromagnetic contamination | 15 |
| | Error code, with direct peripheral access | 36 |
| | Error list | 70 |
| | Ethernet cable, properties | 73 |
| | Ethernet interface | 10 |
| | Event controlled task | 28 |
| | Event list | 70 |
| F | Flash | 10 |
| | Forcing | 23 |
| | Function blocks | 61 |
| | Functional areas, CPU | 7 |
| | Functions | 61 |
| | CAN_BUSLOAD | 62 |
| | DisableInterrupt | 59 |
| | EnableInterrupt | 59 |
| | GetSlotPtr | 35 |
| | IEC_DeleteErrorList | 63 |
| | IEC_DeleteEventList | 63 |
| | IEC_GetErrorID | 63 |
| | IEC_GetEventID | 63 |
| | IEC_GetNrOfErrors | 64 |
| | IEC_GetNrOfEvents | 64 |
| | IEC_WriteError | 64 |
| | IEC_WriteEvent | 64 |
| | Read...Direct | 34 |
| | SysFile... | 76 |
| | UT12_GetIPConfig | 65 |
| | UT12_GetMacAddress | 65 |
| | UT12_Reboot | 66 |
| | UT12_SaveRegistry | 66 |
| | UT12_SetIPConfig | 65 |
| | UT12_SetIPGateway | 66 |
| | Write...Direct | 35 |

| | | | | | |
|----------|--|--------|----------|---|--------|
| H | HALT (start behaviour) | 22 | N | Node ID | 48 |
| | Help for browser comands | 68 | | Node number | 48 |
| I | Incremental encoder | | O | Online dokumentation | 6 |
| | Connection | 17 | | Operating states CPU (LED display) | 23, 36 |
| | Input | 11 | | Operating system | |
| | Incremental encoders | 55 | | Erasing, from Multi Media Card | 26 |
| | Incrementing | 11, 17 | | Transferring, from the PC into the PLC | 25 |
| | Inductors | 15 | | Updating | 25 |
| | Inputs | | | Operation | 21 |
| | Addressing | 37 | | Output MAC address | 65 |
| | Configuration and parameterisation | 53 | | Outputs | |
| | Signal state | 8 | | Addressing | 37 |
| | Wiring example | 17 | | Configuration and parameterisation | 53 |
| | Installation, CPU | 13 | | Signal state | 8 |
| | Interface | | | Wiring example | 17 |
| | CANopen, assignment | 19 | P | Parameterisation, inputs/outputs | 53 |
| | Definition (communication) | 41 | | PC connection | 18 |
| | easyNET, assignment | 19 | | Performance scope, CPU | 7 |
| | ETH232, assignment | 18 | | PING response | 43 |
| | USB, assignment | 18 | | Port assignment | 41 |
| | Interference factors | 15 | | Power down logic | 8 |
| | Interrupt | 59 | | Power off/interruption of the power supply (reaction) | 23 |
| | Actuator connection | 17 | | Power supply | 16 |
| | Inputs | 12 | | For processor unit and local inputs/outputs | 8 |
| | Interrupt, CAN Bus | 11 | | Priority (task) | 27 |
| | IP address | 42, 65 | | Program | |
| | Scan/modify | 43 | | Call (task) | 27 |
| | IP Gateway address | | | Processing | 27 |
| | Issue | 65 | | Start | 22 |
| | | | | Stop | 22 |
| L | Language switchover, error and event list (browser | | | Programming | 7 |
| | commands) | 70 | | Programming interface | 10, 18 |
| | Layout of units | 15 | | Pulse transmitter | 17 |
| | LED display | 8, 23 | R | Real-time clock | 9 |
| | Library | 76 | | Reference signal | 55 |
| | Installation | 61 | | Reference window | 55 |
| | Lightning protection | 16 | | Registry save | 66 |
| | Limit values, for memory usage | 37 | | Reset | 24 |
| | Load program | 24 | | RJ -45 interface | 18 |
| M | Main memory | 10 | | Routing | 47 |
| | Markers Addressing | 37 | | RS -232 interface | 10 |
| | Memory | | | RUN/STOP-LED,SF-LED | 23 |
| | Application program | 7 | S | Segments | 37 |
| | Systems | 9 | | Setting | |
| | Usage, limit values | 37 | | IP Gateway address | 66 |
| | Memory card | 76 | | Shielding | 15 |
| | MMC memory card | 10 | | Signal state inputs/outputs | 8 |
| | Monitoring time, Task | 31 | | Single cycle mode | 23 |
| | Mounting position | 15 | | Single-step mode | 23 |
| | Multimedia card | 9 | | Startup behaviour | 21 |
| | Multitasking | 27, 29 | | Configuring with easySoft-CoDeSys | 22 |

| | |
|---|-----------|
| STARTUP.INI file | 45 |
| Status indication | 23 |
| Subnetmask address | 65 |
| Suppressor circuitry for interference sources | 15 |
| Switching threshold | 8 |
| System | |
| Events | 27 |
| Libraries | 61 |
| Load, CPU | 69 |
| Memory | 10 |
| Parameter predefined (via STARTUP.INI file) | 45 |
| Times | 27 |
| System events | 29 |
| T Task | |
| Condition | 27 |
| Configuration | 27 |
| Monitoring | 31 |
| TCP/IP connection (for routing) | 47 |
| Technical data | 78 |
| Terminal assignment | 8 |
| Test and commissioning | 23 |
| Text output via the RS-232 interface | 74 |
| Transparent mode | 51, 74 |
| Type (task condition) | 27 |
| U Up counter | 57 |
| Connection | 17 |
| Up/down counter | 11 |
| USB interface, assignment | 18 |
| USB stick | 9, 10, 77 |
| User memory, size | 7 |
| User program, memory values | 7 |
| V Ventilation | 15 |
| Voltage dip | 8 |
| Voltage supply | |
| Connection | 16 |
| W WARMSTART (start behaviour) | 22 |
| Watchdog | 31 |
| Web visualization | 37 |
| Web-server | 7 |
| Wiring | 15 |
| Wiring example | |
| Inputs/outputs | 17 |
| Power supply | 16 |
| X XI/OC modules | 7 |