Manual 06/14 MN05003001Z-EN

Modular PLC
XC-CPU201-…(-XV)
XC-CPU202-…-XV

EATON
Powering Business Worldwide

Emergency On Call Service
Please call your local representative:
http://www.eaton.com/moeller/aftersales
or
Hotline of the After Sales Service:
+49 (0) 180 5 223822 (de, en)
AfterSalesEGBonn@eaton.com

Original Operating Instructions
The German-language edition of this document is the original operating manual.

Translation of the original operating manual
All editions of this document other than those in German language are translations of the original German manual.

1st published 2003, edition date 12/03
2nd edition 2004, edition date 12/03
3rd edition 2004, edition date 04/04
4th edition 2004, edition date 06/04
5th edition 2004, edition date 08/04
6th edition 2004, edition date 11/04
7th edition 2005, edition date 03/05
8th edition 2005, edition date 11/05
9th edition 2006, edition date 09/06
10th edition 2006, edition date 12/06
11th edition 2007, edition date 04/07
12th edition 2008, edition date 01/08
13th edition 2010, edition date 10/10
14th edition 2013, edition date 08/13
15th edition 2014, edition date 06/14
See revision protocol in the “About this manual“ chapter

Production: Antje Panten-Nonnen
Translation: globaldocs GmbH

Danger!
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/IL) 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

0 About this manual

0.1 List of revisions

The following significant amendments have been introduced since previous issues:

Publication date	Page	Subject	New	Modification
12/03 (Reprint)	38	Data remanence, 1st paragraph		✓
04/04	64	Limit values for memory usage	✓	
	62	WriteBitDirect		✓
06/04	23, 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	13, 85	Multi-media card (MMC), secure digital card (SD), USB stick	✓	
	14	Splitting of the ETH232 interface		✓
	35	Status display		
	45	Connection set-up via RS 232 interface		
	100	XC200 specific functions		
	106	Additional functions of the XC200_Util2.lib library for the XC-CPU201		
	85	RS 232 interface in Transparent mode		
03/05	14	Splitting of the ETH232 interface		
	17	Figure 20		
	65	Segment size of the XC-CPU201-EC256k		
	66	Addressing inputs/outputs and markers		
	69	Diagnostics		
	78	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	75	chapter “Setting system parameters via the Startup.ini file”		
	112	Chapter “The easyNet network”		
12/06	48	chapter “Program processing, multitasking and system times”		
	78	chapter “Programming via CAN(open) Network (Routing)”		
04/07	112	Chapter “The easyNet network”		
01/08		Chapter 13: “The easyNet network” and chapter 14: “Programming via easyNet (routing)” are now found in manual MN05006004Z-EN.		
01/08 (publication date unchanged)	16	Fig. 7		
10/10		XC-CPU202 added	✓	
08/13	145	Typical current consumption: ~~1.4~~ 0.85 A		✓
06/14		Firmware update for XC-CPU202		✓

0.2 Writing conventions

→ **Short notation for XC200**
Whenever this manual uses the designation "XC200", it is referring to both the XC-CPU201 and XC-CPU202 device versions. Whenever it needs to refer to the XC-CPU201 or XC-CPU202 version specifically, it will explicitly indicate which version it is referring to.

Symbols with the following meaning are used in this manual:

▶ Indicates instructions to be followed.

0.2.1 Hazard warnings of material damages

NOTICE
Warns about the possibility of material damage.

0.2.2 Hazard warnings of personal injury

CAUTION
Warns of the possibility of hazardous situations that may possibly cause slight injury.

WARNING
Warns of the possibility of hazardous situations that could result in serious injury or even death.

DANGER
Warns of hazardous situations that result in serious injury or death.

0.2.3 Tips

→ Indicates useful tips.

▶ Indicates instructions to be followed

Select ‹File → New› means: activate the instruction "New" in the "File" menu.

0.3 Additional documentation

At different points in this manual, references are made to more detailed descriptions in other manuals. These manuals are described with their title and documentation number (e.g. MN05002002Z-DE). 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.

To find the manuals required go to the following Internet address

http://www.eaton.eu → **Customer Support** → **Download Center Documentation**

and enter the documentation number in the quick search box.

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, the central coupling of XIOC signal modules and the decentral connection of CAN devices, this control forms the basis for the implementation of a comprehensive automation system.

The PLC consists of the:

- Rack → Page 8
- CPU with PSU and local inputs/outputs→ Page 9
- XIOC-signal modules → separate manual "Hardware and Engineering" (MN05002002Z-EN).

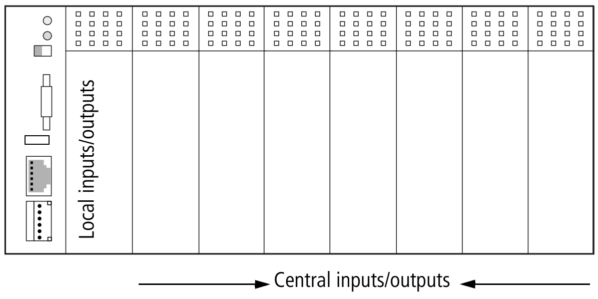


Figure 1: Layout of the XC-CPU201 with XIOC modules

→ CODESYS programming software from Version 2.3 is required for programming the XC200.

1.1 Rack

There are basic backplanes and expansion racks.

The basic backplane XIOC-BP-XC features two slots for the central processing unit. The XIOC-BP-XC1 provides three slots, so that there is also place available for an XIOC signal module beside the central processing unit.

A basic backplane can be expanded using several expansion racks. Expansion backplanes are fitted with XIOC signal modules.

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

→ Detailed information on the module racks and the XIOC signal modules is provided in the manual "XI/OC signal module – Hardware and Engineering" (MN05002002Z-EN).

1.1.1 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 speed, 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),
- XC-CPU202-EC4M-8DI-6DO-XV.

"EC256K", "EC512K" and "EC4M" are a measure for the size of the user memory. "XV" identifies a visualization CPU with integrated web server.

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

	XC-CPU201-...-8DI-6DO(-XV)		**XC-CPU202...**
	...EC256K	**...EC512K**	**...EC4M**
Program code	512 kByte	2048 kByte – from operating system 1.04.01	4 MB
Program data, of which:	256 kByte	512 kByte	2M Bytes
Marker	16 kByte	16 kByte	16 kByte
Retain data	32 KByte	32 KByte	64 kByte

1.1.2 Functional spans

The CPU is arranged into three functional areas:

- Power supply → Page 10,
- Local inputs and outputs → Page 10,
- Processor unit with interfaces → Page 12

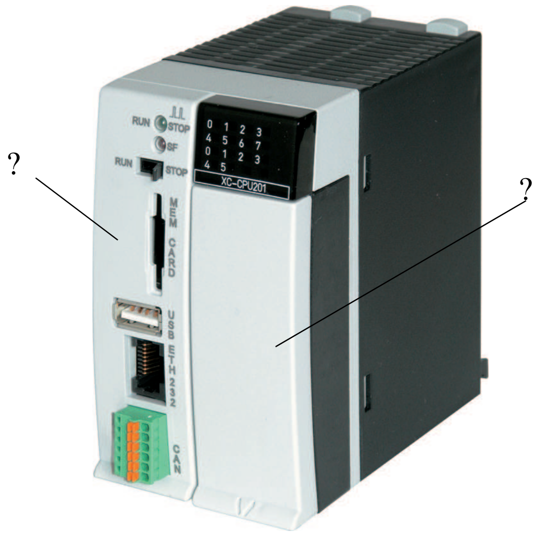


Figure 2: XC-CPU201 – a CPU of the XC200 series

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

1.1.3 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 a voltage dip of the 24 V supply voltage (switching threshold is about 10 V) then a power-down logic switches of the 5 V supply to the signal modules (central I/O).

1.1.4 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 power supply of the CPU and the local inputs/outputs as well as the sensors and actuators.

The eight digital inputs (I0.0 to I0.7) and six semiconductor outputs (Q0.0 to Q0.5) 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.

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

See also:

- Add-on functions of the CPU (local inputs) → Page 17

1.1.4.1 Terminal assignments

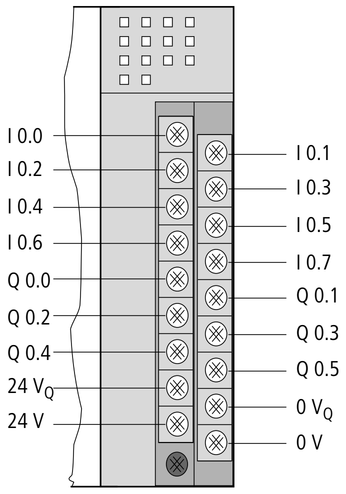


Figure 3: Terminals of the power supply unit and local inputs/outputs
I0.0 to I0.7: local digital inputs
Q0.0 to Q0.5: local digital outputs
$0V_Q$/$+24V_Q$: supply voltage for the local inputs/outputs
0V/+24V: supply voltage to the processor unit

1.1.4.2 LED indicators

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

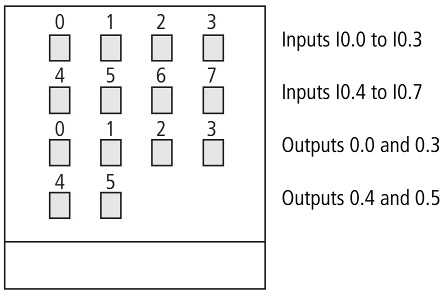


Figure 4: LEDs for the local inputs/outputs

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).

1.1.5 Processor unit with interfaces

Belonging to the processor unit are:

- Real-Time Clock → Page 12,
- Battery → Page 12,
- Multi-media card (MMC), secure digital card (SD), USB stick → Page 13,
- CPU drives → Page 13,
- USB interface → Page 27,
- ETH232 programming interface → Page 14,
- CAN/easyNet interface → Page 15,
- Add-on functions of the CPU (local inputs) → Page 17.

1.1.6 Real-Time Clock

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

- 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 real-time clock via the following browser commands:

- setrtc (set the real-time clock) → Page 129,
- getrtc (query the real-time clock) → Page 128.

1.1.7 Battery

A Lithium battery of type 1/2 AA (3.6 V) is used for saving of retentive data and for operation of the real-time clock. The battery compartment can be found on the left side of the central processing unit unit, behind a blanking plate. The charge level of the battery is monitored. If the battery voltage exceeds a preset fixed limit value, a common group fault is indicated.

The battery backup times are:

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

NOTICE
Only change the batteries when the supply voltage is switched on. Otherwise data may be lost.

The order designation of the battery is: XT-CPU-BAT1

1.1.8 Multi-media card (MMC), secure digital card (SD), USB stick

MMC, SD and USB serve as mass memory. 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.

If you want, you can use an SD memory card instead of an MMC memory card.

From operating system version 01.03. of the XC-CPU201, you can transfer the operating system to the MMC in order to load it from there to other controllers (operating system update). From this operating system version in XC-CPU201 it is also possible to use a USB stick for data storage.

NOTICE
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:

- XC-CPU-202: → Page 40
- Deleting the operating system/boot project from the MMC of the XC-CPU201 → Page 44
- Configuration of the USB interface → Page 27

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

1.1.9 CPU drives

The XC200 has the following drives available:

- internal
  - Memory system (disk_sys)
- external (optional)
  - Multi-media card MMC or secure digital card SD (disk_mmc)
  - USB stick (disk_usb).

The external CPU drive will continue to be called disk_mmc even if you are using an SD card.

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 XC-CPU200 memory systems/drives.

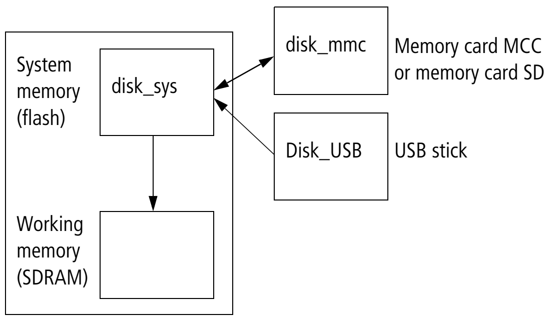


Figure 5: XC-CPU200 memory organization

See also:
- Data access on the memory card
  - with the aid of browser commands such as, for example, copyprojtommc, to copy the user program onto the MMC → Page 128
  - Functions such as SysFileOpen or SysFileRead? Page 139
- Limit values for memory usage → Page 64

1.1.10 ETH232 programming interface

The communication between the PLC and the programming device is implemented via the ETH232 programming interface of the CPU. It consists of an Ethernet interface and an RS232 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.

The XC-CPU202 also allows you to update the operating system of the controller via the Ethernet interface.

You can also execute programming via the RS232 interface.
From operating system version V01.03 of the XC-CPU201 you can also switch the RS232 interface to Transparent mode in order to establish a point-to-point connection without handshake lines.

1.1.11 Splitting of the ETH232 interface

The cable switch XT-RJ45-ETH-RS232 enables you to communicate with the XC200 via the RS232 and Ethernet interfaces simultaneously.

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 connecter of the CPU.

For example you can connect the programming device to the Ethernet interface of the cable switch and the RS232 interface (in Transparent mode) to a printer. The pin assignment of the RS232 and the Ethernet plug socket of the cable switch is the same as that of the ETH232 socket of the central processing unit.

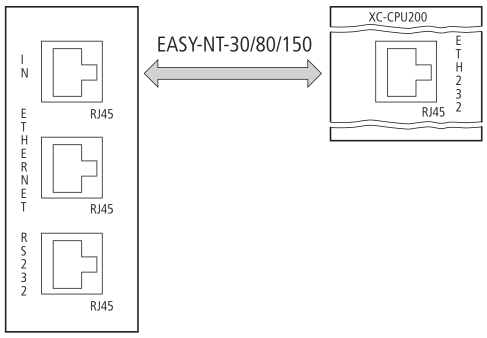


Figure 6: Connection of the XC-CPU200 with the XT-RJ45-ETH-RS232

See also:
- Connect PC → Page 26
- Configuration of the programming interface → Page 27
- Connection set-up PC – XC200 → Page 70.

1.1.12 CAN/easyNet interface

The CAN/easyNet interface is isolated. The connections of the interfaces are the same. The CPU can be run both as a network (NMT) master as well as an NMT slave (device) on the CAN bus.
The CPU can run the CANopen and the easyNet protocol at the same time.

See also:
- Detailed information for engineering and programming CAN stations → Application note AN2700K27.
- easy800 control relay operator manual (MN04902001Z-EN; previously called AWB2528-1423GB)
- Network easyNet → page 73

Bus terminating resistors

Bus termination resistors must be installed at the first or last station on the line.

Module	Bus termination resistor
XC-CPU201, MFD4	120 Ω (external)
easy800, MFD	EASY-NT-R
XC-CPU202	switched neutral

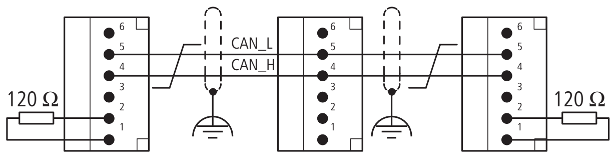


Figure 7: Example: network with bus termination resistor on XC-CPU201

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

The bus terminating resistor on the XC-CPU202 can be switched. This switch is located above the battery switch (factory setting: ON).

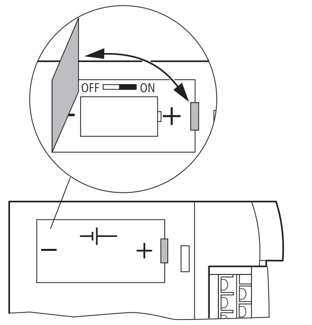


Figure 8: XC-CPU202, XT-CPU-BAT1

1.1.13 Reaction of the station on the CAN bus

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

Start/Stop behavior:

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”.

Switch on voltage:

The order in which you switch on the power supply of the individual CAN stations has no effect on the functioning of the CAN bus. Depending on the parameters set, the controller “waits” for stations that are not present or starts them when the station is connected to the CAN net.

Communication with CAN stations:

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

- Connection of an XION station to the XC100/200 via CAN (AN27K18D)
- Communication between two controls using network variables via CAN (AN27K19D)
- Connection of multiple autonomous controls (CAN-Device) via CAN (AN27K20D)
- Engineering of CAN stations (AN27K27D)

- Library description: CANUser.lib/CANUser_Master.lib (MN05010001Z-EN).

See also:
- Configuration of the CAN/easyNet interface → Page 28
- Properties of the CAN cable → Page 136

1.1.14 Add-on functions of the CPU (local inputs)

The inputs I0.0 to I0.5 can be parameterized 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.

1.1.14.1 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 a total frequency of 200 kHz.

See also:
- Connecting the incremental encoder → Page 24
- Incremental encoder parameterization → Page 90

1.1.14.2 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 value is achieved an interrupt is initiated which is used to branch to a programming routine (POU). The reaction after the setpoint value 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 95
- Input of the setpoint value in the control configuration → Page 93
- Connecting up/down counter → Page 25

1.1.14.3 Up/down counter (16 Bit)

Two of these counters are available. 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

1.1.14.4 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 145
- Programming in the “Interrupt function” is described on Page 95.
- Connecting up/down counter → Page 25

2 CPU installation

→ Detailed information on mounting the module rack and the XIOC signal modules is provided in the manual "XI/OC Signal Modules – Hardware and Engineering" (MN05002002Z-EN).

▶ Insert the loop on the bottom of the CPU module into the hole in the module rack $\boxed{1}$.

▶ Press the top of the CPU module onto the module rack, until you hear it click into position $\boxed{2}$.

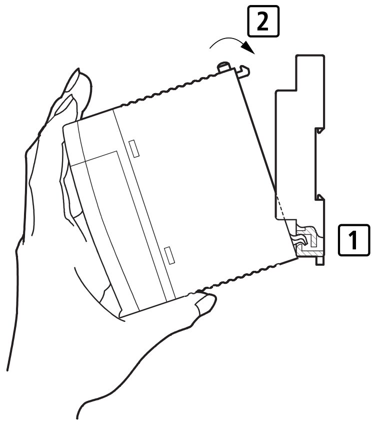


Figure 9: CPU installation

2.1 Detaching the CPU

▶ Press in the catch $\boxed{1}$.

▶ Keep the catch pressed in, and pull the top of the CPU module forwards $\boxed{2}$.

▶ Lift up the CPU module and remove it $\boxed{3}$.

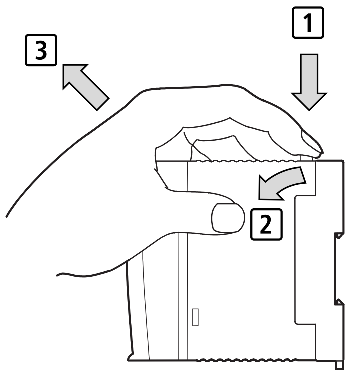


Figure 10:Detaching the modules

3 Engineering

3.1 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,
- Variable frequency drives,
- Current converters.

In order to effectively exclude any electromagnetic interference, it is a good idea to divide the system into sections, according to their power and interference levels. In small control panels it is often enough to provide a sheet steel dividing wall, to reduce interference factors.

3.1.1 Ventilation

In order to ensure sufficient ventilation a minimum clearance of 50 mm to passive components must be observed. If the adjacent components are active elements (e.g. power supplies, transformers) a minimum clearance of 75 mm must be observed. The values that are given in the technical data must be observed.

3.1.2 Layout of units

Build the module racks and the controls into the control panel in a horizontal position.

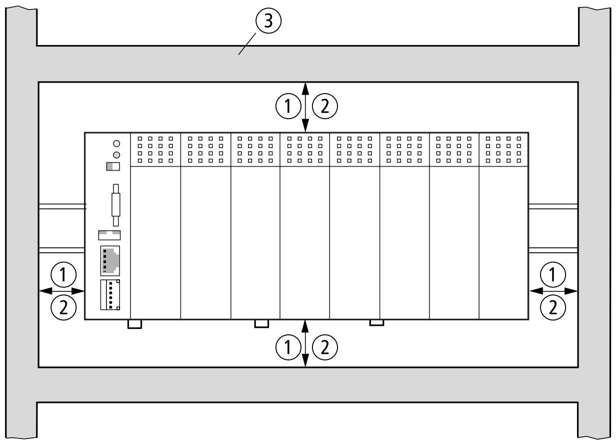


Figure 11:Control panel layout

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

3.2 Preventing interference

3.2.1 Cable routing and wiring

Cables are divided into the following categories:

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

→ Always route power cables and signal cables as far apart as possible. This avoids capacitive and inductive coupling.
If separate cable 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 clearance:
  - 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 opposing currents on this cable pair cause the sum of all currents to equal zero. The generated electromagnetic fields cancel each other out.

3.2.2 Suppressor circuit for interference sources

- ▶ Connect all suppressor circuits as close to the source of interference (contactors, relays, solenoids) as possible.

→ Switched reactors should always have suppressor circuitry fitted.

3.2.3 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.

3.3 Lighting protection

3.3.1 External lightning protection

All cables between buildings must be shielded. Metal conduits are recommended for use here. Fit signal cables with overvoltage protection, such as varistors or other surge arresters. Implement these measures ideally where the cable enters the building and at least at the control panel.

3.3.2 Internal lightning protection

Internal lightning protection covers all those measures taken to reduce the effects of a lightning strike and the resulting electrical and magnetic fields on metallic installation and electrical plant. These measures are:

- Potential equalizing/earthing
- Shielding,
- Using surge protective devices

3.4 Connections

3.4.1 Connecting the power supply

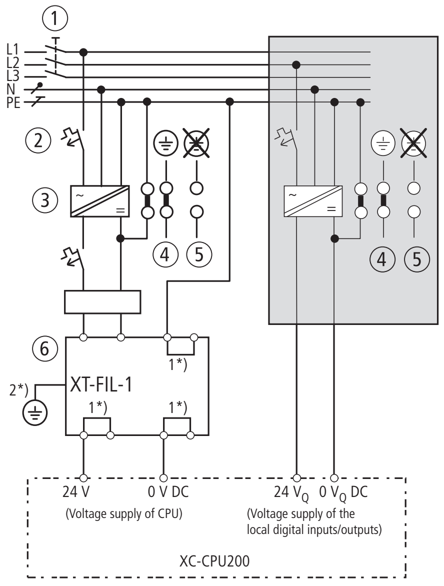


Figure 12:Example of wiring for power supply unit

① Main switch
② Circuit protection device
③ 24 V DC supply voltage
④ Earthed operation
⑤ In floating (i.e. non-earthed) operation, an isolation monitor must be used (IEC 204-1, EN 60204-1, DIN EN 60204-1)
⑥ 24 V DC line filter; ensures that a rated operating voltage of up to 24 V DC (maximum) is available at a rated voltage of 2.2 A. Use of the filter ensures that the EMC stipulations for devices.
Instructions: The filter is not a component of the central processing unit and must therefore be ordered separately:
Type: XT-FIL-1, Article no.: 285316 (Supplier: Eaton Industries GmbH)
→ Dimensions on Page 141
→ Technical data on Page 147.

1*) Internal jumper
2*) Additional PE connection via contact spring on rear

3.4.2 Connecting inputs/outputs (central processing unit)

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

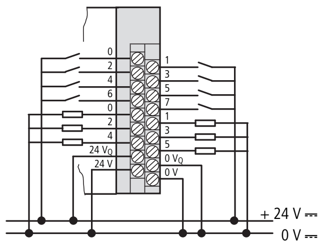


Figure 13:Connecting inputs/outputs to the central processing unit

→ Wiring examples on the XIOC modules are provided in the manual “XI/OC Signal Modules – Hardware and Engineering” (MN05002002Z-EN).

3.4.3 Connecting the incremental encoder

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

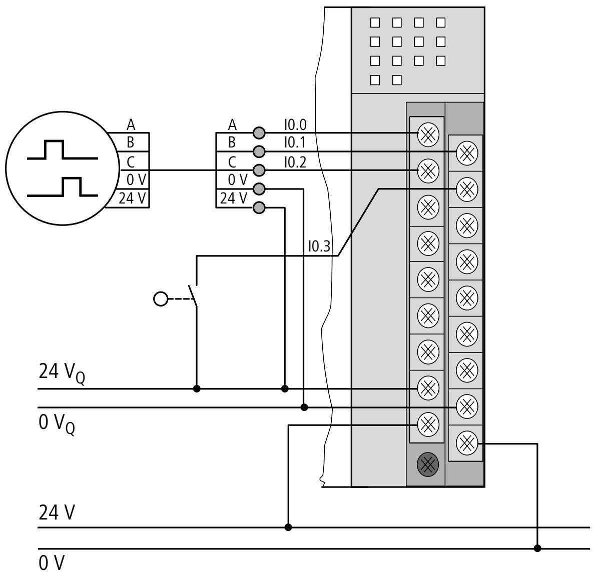


Figure 14:Connection of the incremental value encoder with a reference window switch

3.4.4 Connecting up/down counter

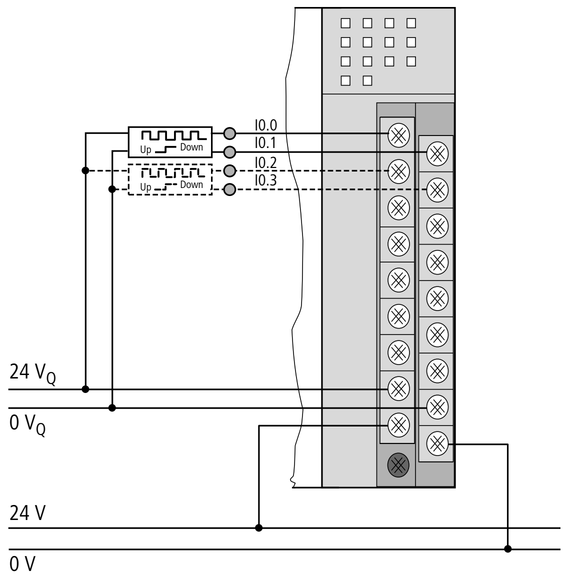


Figure 15:Connection of pulse generator with signal for incrementing/decrementing

3.4.5 Connecting interrupt actuators

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

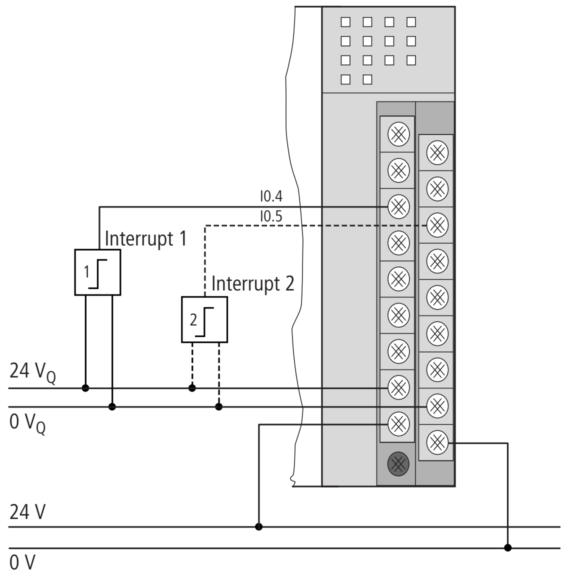


Figure 16:Interrupt input connections

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

3.4.6 Connect PC

3.4.6.1 Ethernet connection

From a purely physical/mechanical point of view the programing devices interface is an RJ45 interface (socket). This means that normal commercial RJ45 connectors or Ethernet patch cables can be used.

Direct connection PC – XC200:

The XC200 can be connected directly to the (programming) PC via a crossover Ethernet cable, → Figure 17, 18.

Crossover cables have the following design features:

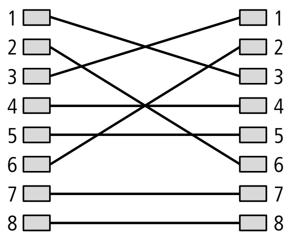


Figure 17:Connection set-up of an 8-pole crossover cable

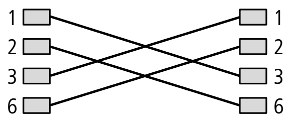


Figure 18:Connection set-up of a 4-pole crossover cable

The following cross-over cables are available:

• XT-CAT5-X-2	2 m long (article no. 256487)
• XT-CAT5-X-5	5 m long (article no. 256488)

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 cable EU4A-RJ45-USB-CAB1 (Art. no. 115735) is provided for programming via the USB interface of a PC.

→ Please note that when there is a double assignment of the RJ45 interface with the RS232 and Ethernet, the connections 4 and 7 are connected to “GND potential” because of the RS232 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 135

3.4.6.2 RS232 connection

Please use the XT-SUB-D/RJ45 (article no. 262186) programming cable to make a connection between the XC200 and PC.

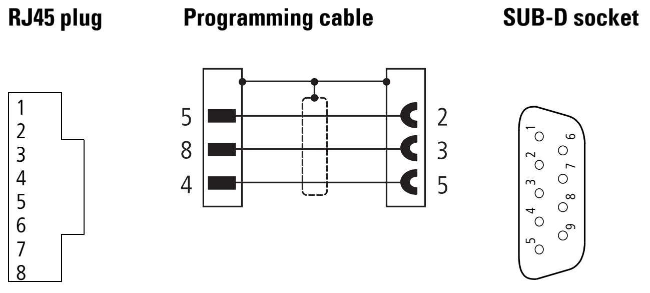


Figure 19:Pin assignment RS232 programming cable

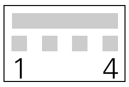
See also:

- Connection set-up PC – XC200 → Page 70
- RS 232 interface in Transparent mode → Page 85

3.5 Interface assignments

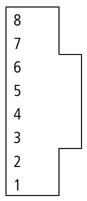
3.5.1 USB interface

Table 1: Configuration of the USB interface

	Signal
1	+5 V ⎓
2	USB–
3	USB+
4	GND

3.5.2 XC200 programming interface

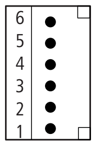
Table 2: Configuration of the programming interface

		Signal	
RJ45 socket		RS232	Ethernet
	8	RxD	–
	7	GND	–
	6	–	Rx–
	5	TxD	–
	4	GND	–
	3	–	Rx+
	2	–	Tx–
	1	–	Tx+

→ The Ethernet socket on the XC-CPU202 is reversed by 180 degrees. However, the pin assignment is identical to that of the XC-CPU201.

3.5.3 CAN/easyNet interface

Table 3: Configuration of the CAN/easyNet interface

	Terminal	Signal	
		CAN	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 mm^2
Terminals 1 and 4 , 2 and 5 as well as 3 and 6 are internally connected.

4 Operation

4.1 Startup behavior

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

The following flow diagrams (Fig. 20 and Fig. 21) show which program is used. The charts also show the updating of the operating system (OS) using the MMC/SD/USB.

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

4.1.1 Startup behavior of the XC-CPU201

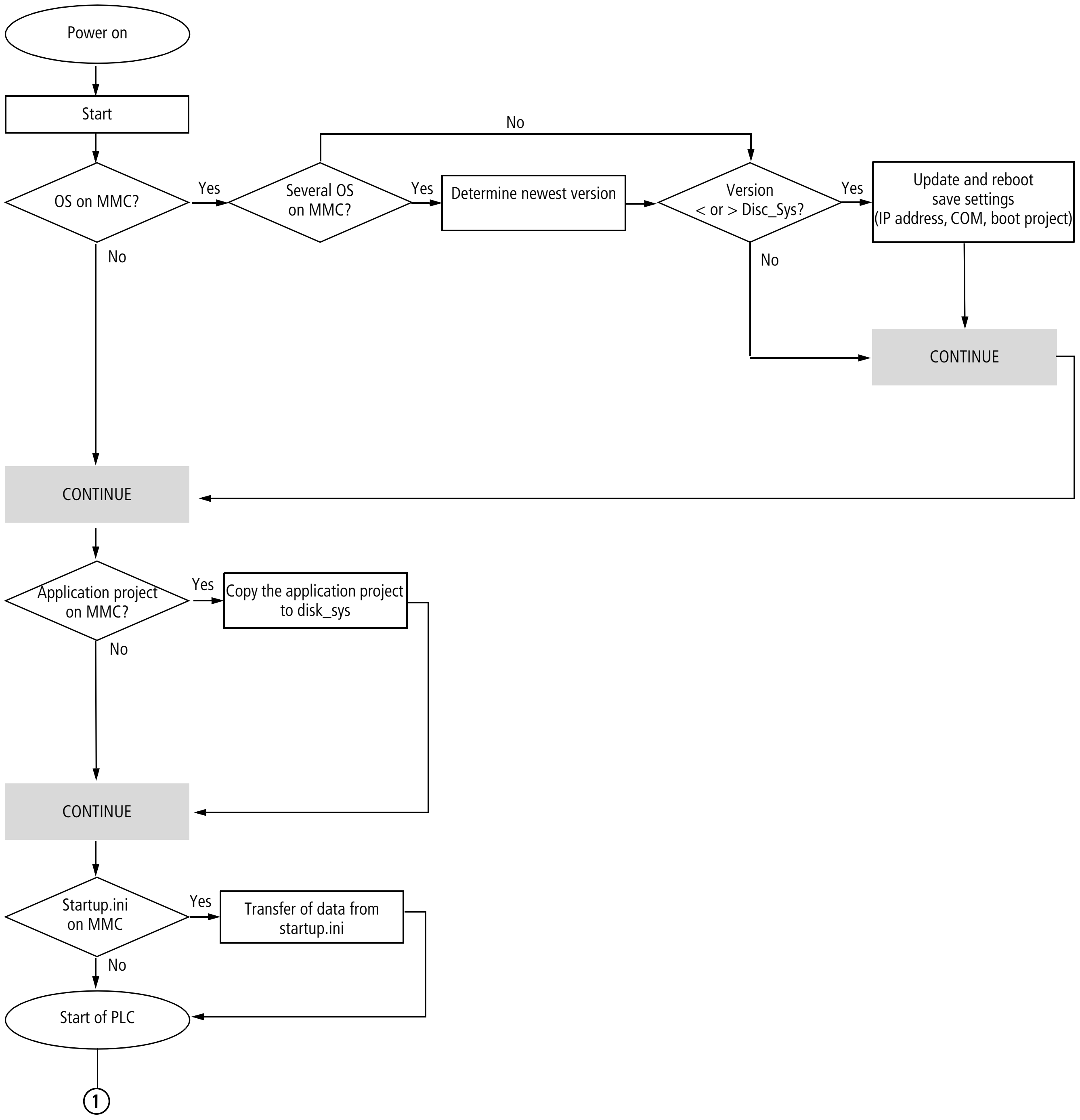
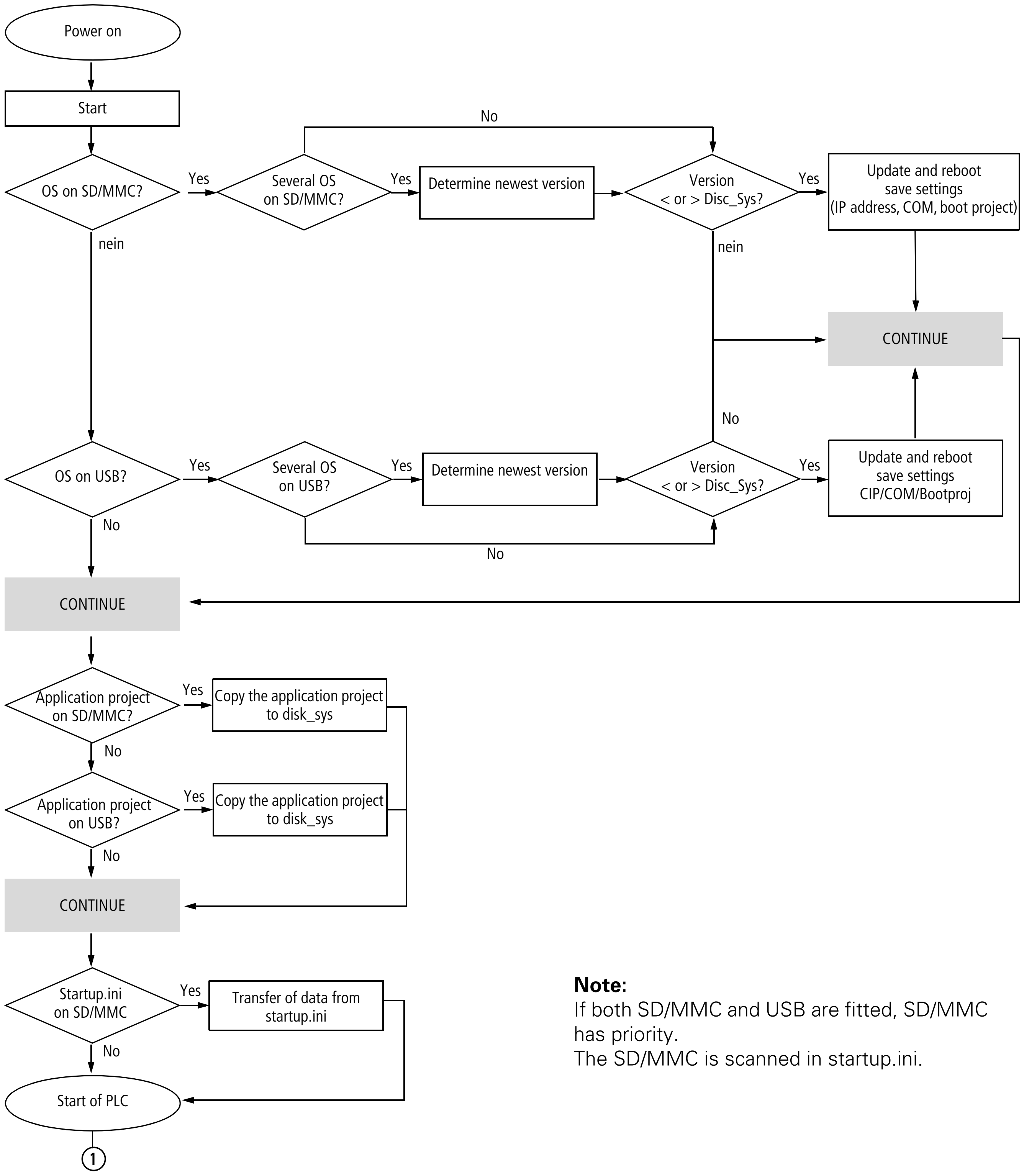


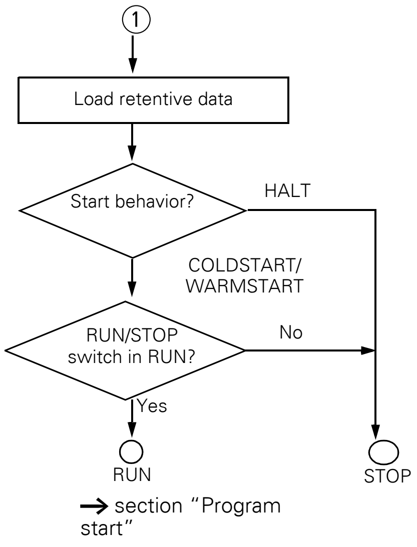
Figure 20:Boot procedure with MMC

4.1.2 Startup behavior of the XC-CPU202


Power on
Start
OS on SD/MMC?
Yes
Several OS on SD/MMC?
Yes
Determine newest version
Version < or > Disc_Sys?
Yes
Update and reboot save settings (IP address, COM, boot project)
No
nein
nein
CONTINUE
No
OS on USB?
Yes
Several OS on USB?
Yes
Determine newest version
Version < or > Disc_Sys?
Yes
Update and reboot save settings CIP/COM/Bootproj
No
No
CONTINUE
Application project on SD/MMC?
Yes
Copy the application project to disk_sys
No
Application project on USB?
Yes
Copy the application project to disk_sys
No
CONTINUE
Startup.ini on SD/MMC
Yes
Transfer of data from startup.ini
No
Start of PLC
1

Note:
If both SD/MMC and USB are fitted, SD/MMC has priority.
The SD/MMC is scanned in startup.ini.

Figure 21:Boot procedure with SD/MMC and USB



4.1.3 Configuring the start-up behavior with CODESYS

The start-up behavior 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 "STARTUP BEHAVIOR" drop-down menu in the "Other Parameters" tab of the PLC configurator:

- STOP
- COLDSTART,
- WARMSTART.

4.1.3.1 HALT

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

4.1.3.2 COLDSTART/WARMSTART

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

The variables are initialized in accordance with Table 4 , before the control starts.

Table 4: Behavior of the variables after COLDSTART/WARMSTART

	Behavior of the variables after …	
Variable type	**COLDSTART**	**WARMSTART**
Non-retentive	Activation of the initial values	Activation of the initial values
Retain[1)]	Activation of the initial values	Values remain in memory

Variable type	Behavior of the variables after … COLDSTART	WARMSTART
Persistent	Values remain in memory	Activation of the initial values
Retain Persistent	Values remain in memory	Values remain in memory

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

4.2 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.

4.2.1 Program start (STOP $\rightarrow$ RUN)

You have the following possibilities to start the program:

	Program exists in main memory	Program should be loaded
Prerequisite	• CPU in STOP • RUN/STOP switch in STOP	• CPU in STOP • RUN/STOP switch in RUN
Action	• Switch RUN/STOP switch to RUN or • in online operation, issue the "Start" command.	• 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

4.2.2 Program stop (RUN $\rightarrow$ STOP)

A change of the RUN/STOP switch to the STOP position leads the central processing unit 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, $\rightarrow$ chapter "Program processing, multitasking and system times" on Page 48.

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.

4.3 Power off/Interruption of the power supply

When the program is running, the switching off or interruption of the (CPU) power supply will cause the program cycle or task to be aborted immediately. The data is no longer consistent!

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 48. The behavior of retentive variables in shown in can be seen in Table 4.

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

If the consistency of the data is absolutely necessary for an application, other measures are required, such as the use of a uninterrupted power supply with battery back-up. The PLC is started as shown in Figure 20 and Figure 21.

4.3.1 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.

4.3.2 Test and commissioning (Debugging)

The PLC supports the following test and commissioning features:

- Breakpoint/Single step mode,
- Single cycle mode,
- Forcing,
- Online modification, → PLC programming with CODESYS manual, Chapter "Online functions",
- Status display/Powerflow.

4.3.3 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.

NOTICE

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

4.3.4 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 images are cancelled and the outputs are switched off. Task monitoring is active. Task monitoring is active.

4.3.5 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 central processing unit is in the RUN state.

4.3.6 Status display

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.

Example:

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

in IL:

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

4.3.7 Reset

There are three different types of Reset commands:

- Warm reset,
- Cold reset,
- Full reset.

Table 5: 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.

4.3.7.1 Warm reset

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

4.3.7.2 Cold reset

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

4.3.7.3 Full reset

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

Table 5: Behavior of the variables after a Reset

	Reset command		
Variable type	**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	Values remain in memory	
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 or M cannot be declared as "retain" variables.

4.4 Programs and project

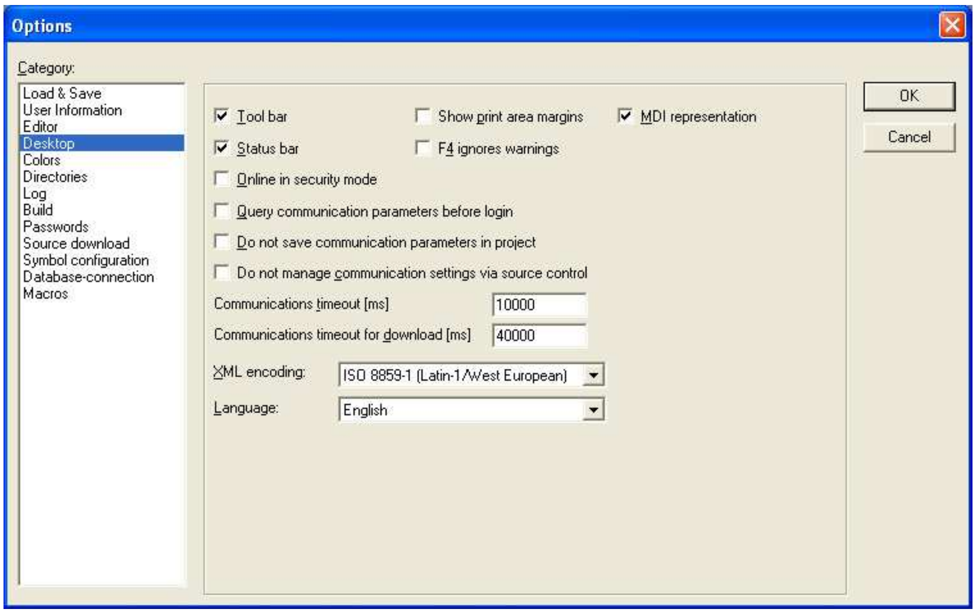
4.4.1 Loading the program

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 initialized 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. The transfer time can be set in CODESYS.



- [✓] Tool bar
- [✓] Status bar
- [ ] Online in security mode
- [ ] Query communication parameters before login
- [ ] Do not save communication parameters in project
- [ ] Do not manage communication settings via source control
- [ ] Show print area margins
- [ ] F4 ignores warnings
- [✓] MDI representation

Communications timeout [ms]: 10000
Communications timeout for download [ms]: 40000
XML encoding: ISO 8859-1 (Latin-1/West European)
Language: English

Figure 22:Setting the transfer time

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.

4.4.2 General information on RETAIN PERSISTENT

The data of variables declared as RETAIN PERSISTENT are retained (in the memory of the XC200) when a new program is loaded via MMC/SD or CODESYS as long as the following conditions are fulfilled:

- A boot project must be created for the loaded program.
- The names of the variables of the loaded program and the new program must be identical.
- The data types of the variables of the loaded program and the new program must be identical or interconvertible.

The following always apply:

The data for all standard data types will be used 1:1 in the new program. Strings may be truncated depending on the declared string length.

If different data types are assigned to the variable names in the new program, the data is converted automatically by the operating system of the XC200 when the program is loaded.

Normally zeros are filled depending on type (SINT → DWORD) or the higher bytes are truncated (DWORD → BYTE). However, there is some data that is not convertible, e.g. (WORD → UINT). The result for this is always ZERO.

The following table shows the conversions:

			Variable names and data types in the PLC memory (Src)											
	rp_typN	**N =**	**1..11**	**1..11**	**1..11**	**1..11**	**1..11**	**1..11**	**1..11**	**1..11**	**1..11**	**1..11**	**1..11**	
	N =	**Type**	SINT	INT	ENUM	DINT	BYTE	USINT	WORD	UINT	DWOR	UDINT	REAL	LREAL
Variable names and data types in the MMC program (Dest)	1, 11	SINT	=	X	X	X	X	X	X	X	X	X	X	X
	2	INT	X	=	0	X	X	X	X	X	X	X	X	X
		ENUM	X	0	=	X	X	X	X	X	X	X	X	X
	3	DINT	X	X	X	=	X	X	X	X	X	X	X	X
	4	BYTE	X	X	X	X	=	0	0	0	X	X	X	X
	5	USINT	X	X	X	X	0	=	0	0	X	X	X	X
	6	WORD	X	X	0	X	X	X	=	0	X	X	X	X
	7	UINT	X	X	0	X	X	X	0	=	X	X	X	X
	8	DWORD	X	X	X	X	X	X	X	X	=	0	X	X
	9	UDINT	X	X	X	X	X	X	X	X	0	=	X	X
	10	REAL	X	X	X	X	X	X	X	X	X	X	=	X
		LREAL	X	X	X	X	X	X	X	X	X	X	X	=
Program sections			1	2	3	4	5	6	7	8	9	10	11	

rp = RETAIN PERSISTENT

X = Conversion executed. The data is adapted to the new data type (from MMC program).
Preceding zeros are added or or higher bytes are truncated

= No conversion required. Data is identical.

0 = Conversion result is ZERO.

1..11 Blue = Variable names different, but of same type (PLC program)

1.11 yellow = Variable names identical to 1.11 blue, but different type (MMC program)

RETAIN PERSISTENT data is deleted if

- the new program does not contain identical variable names,
- the "Full reset" command is executed,
- the battery was removed.

4.4.3 Storing and deleting the boot project

XC-CPU201

1. Save boot project on MMC

▶ Click on "Resources → PLC Browser" folder and enter the command copyprojtommc.

The boot project is stored on the MMC in the subdirectory "project" under the name "Default.prg". A file is also created with the name "Default.chk". The Browser commands filecopy or filerename can be used to copy the boot project (e.g. for a backup copy) and change the name of the file. In the CODESYS software, however, only the boot project with the name "Default" is active.

2. Delete boot project on MMC

Click "Resources" → PLC Browser and enter the following command for the XC-CPU201-EC256K:

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

XC-CPU-202:

1. Save boot project on MMC

Click on the folder “Resources → PLC Browser” and enter the copyprojtommc command.

The boot project is stored on the MMC in the subdirectory “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 CODESYS software, however, only the boot project with the name “Default” is active.

2. Save boot project on MMC

Click on the folder “Resources → PLC Browser” and enter the copyprojtommc command.

The boot project is stored on the MMC in the subdirectory “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 CODESYS software, however, only the boot project with the name “Default” is active.

3. Delete boot project on SD/MMC

Click on the folder “Resources → PLC Browser” and enter for the XC-CPU202 the following command:

```
filedelete \\disk_mmc\\CONTROL\\XC-CPU202-EC4M-8DI-6DO\\project\\default.prg
```

4. Delete boot project on USB stick

Click on the folder ‹Resources → PLC Browser› and enter for the XC-CPU202 the following command:

```
filedelete \\disk_usb\\CONTROL\\XC-CPU202-EC4M-8DI-6DO\\project\\default.prg
```

4.5 Updating the operating system

With the XC200 it is possible to replace the operating system with the latest version. Eaton offers the latest operating system version on the Internet at: http://www.eaton.com/moeller → **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.

4.5.1 XC-CPU201

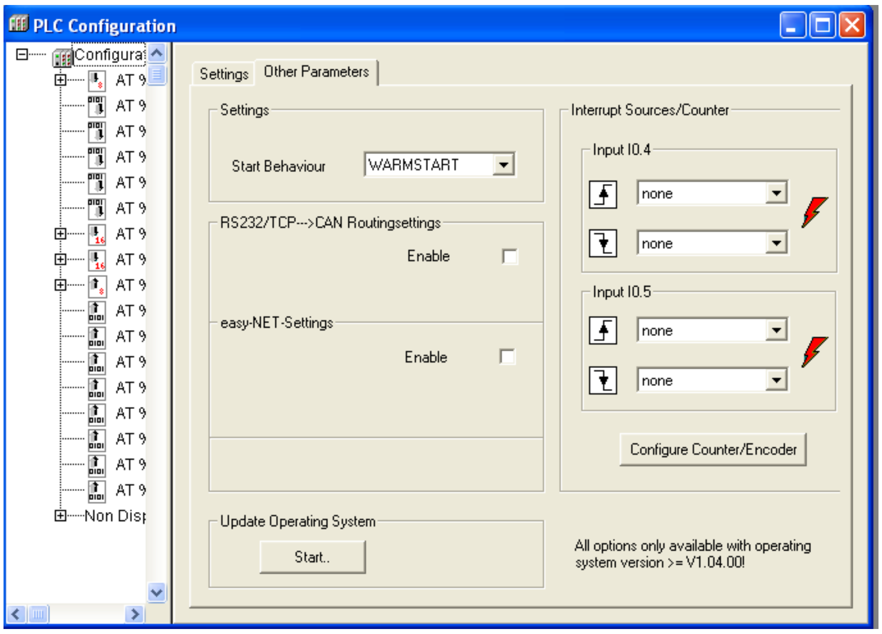
4.5.1.1 Transferring the operating system from the PC to the XC-CPU201

→ When an operating system is loaded onto the PLC, the existing operating system and user program will be deleted!

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

Procedure:

- ▶ Establish a serial connection via the RS232 interface of the PC with the XC201. Information on this is provided in the sections “Connect PC” on Page 26 and “Connection set-up PC – XC200” on Page 70.
- ▶ Activate in the CODESYS software the “Other Parameters” tab in the “PLC Configuration” window and click on the “Start” button.


PLC Configuration

Settings | Other Parameters

Settings
Start Behaviour: WARMSTART

RS232/TCP-->CAN Routingsettings
Enable ☐

easy-NET-Settings
Enable ☐

Update Operating System
Start..

Interrupt Sources/Counter
Input I0.4: none, none
Input I0.5: none, none
Configure Counter/Encoder

All options only available with operating system version >= V1.04.00!

Figure 23:Updating the operating system of the XC-CPU201

The “Download Tool” window opens.

▶ Click on the “Open” button and enter the path in which the update of the operating system is located.

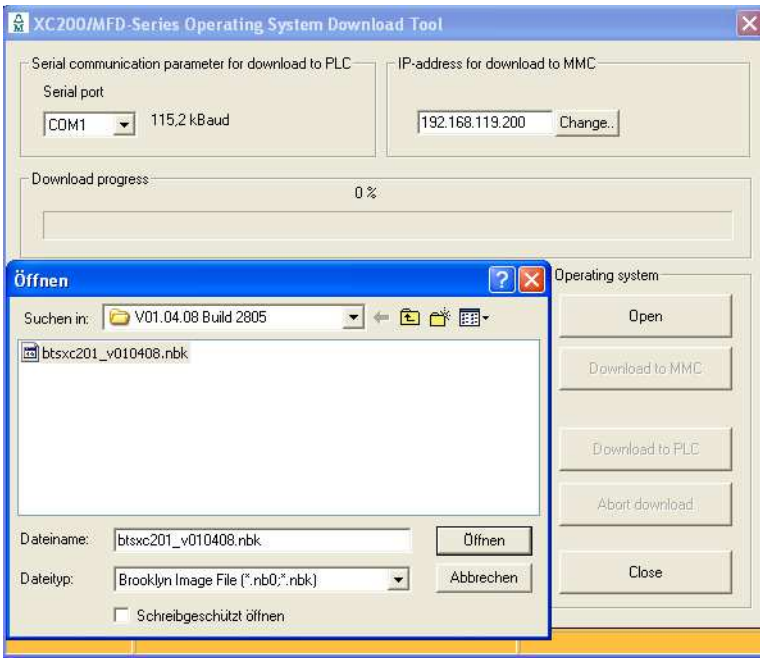
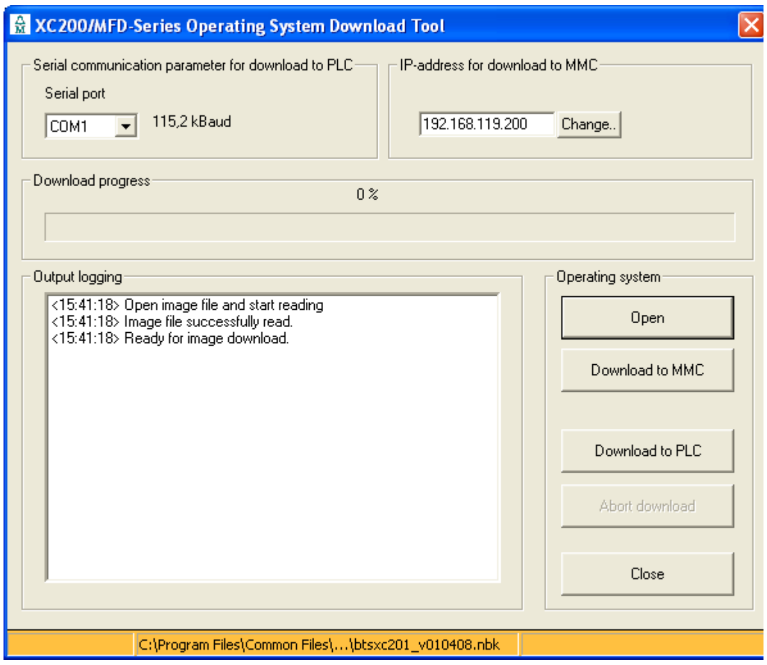


Figure 24:Selecting the operating system for XC-CPU201

▶ Opening of the operating system file to be transferred.

The following window appears:



```
<15:41:18> Open image file and start reading
<15:41:18> Image file successfully read.
<15:41:18> Ready for image download.
```

C:\Program Files\Common Files\...\btsxc201_v010408.nbk

Figure 25:Download of the XC-CPU201 operating system

▶ Click on the “Download to PLC” button.

The “Connecting to target PLC” window entry appears. “Please reboot target now.”

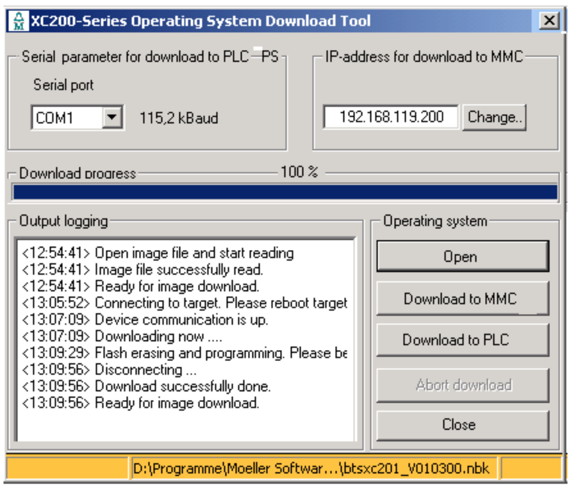
▶ Switch off the control voltage of the XC-CPU201 and wait a few seconds. This will ensure that the residual voltage is discharged.
▶ Switch the control voltage of the XC-CPU201 back on.

The transfer of the operating system into the XC-CPU201 is started.
It can take a few minutes. Please observe the signal states of the operating LEDs:

The read SF LED is lit during the transfer. When the transfer display shows 100 %, the SF LED goes out after a short delay. It will light up later after approx. 1 minute and the green RUN/STOP LED flashes. The waiting time depends on the programming of the internal flash memory (comparable with the booting of 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.


XC200-Series Operating System Download Tool
Serial parameter for download to PLC
PS
IP-address for download to MMC
Change..
Output logging
Operating system
Open
Download to MMC
Download to PLC
Abort download
Close

Serial port: COM1 115,2 kBaud

IP-address for download to MMC: 192.168.119.200

Download progress 100 %

```
<12:54:41> Open image file and start reading
<12:54:41> Image file successfully read.
<12:54:41> Ready for image download.
<13:05:52> Connecting to target. Please reboot target
<13:07:09> Device communication is up.
<13:07:09> Downloading now ....
<13:09:29> Flash erasing and programming. Please be
<13:09:56> Disconnecting ...
<13:09:56> Download successfully done.
<13:09:56> Ready for image download.
```

D:\Programme\Moeller Softwar...\btsxc201_V010300.nbk

Figure 26:Download of the XC-CPU201 operating system ended

▶ End the download with the “Close” button.

4.5.1.2 Transferring the operating system from the PC to the MMC of the XC-CPU201

This is only possible via an Ethernet connection. The XC-CPU201 must contain an operating system from version 01.03.00 or higher.

After the transfer, the operating system is located in the directory: **disc_mmc\moeller\XC-CPU201**

PC ⟶ MMC

The process operates analog to the transfer of the operating system from the PC to the PLC. Simply click on the button “Transfer to MMC” (see Figure 25).

MMC ⟶ PLC

If the operating system of an XC-CPU201 is to be updated via the MMC, the controller must have an operating system from version 01.03.00. The operating system is updated during the startup procedure.

4.5.1.3 Deleting the operating system/boot project from the MMC of the XC-CPU201

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

- ▶ Establish a connection to the XC-CPU201 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.

4.5.2 XC-CPU202

The following description (firmware update) applies to version 2.4.13 and higher (target firmware version).

The operating system of the XC-CPU202 is updated inside the XSOFT-CODESYS-2 programming system.

Procedure:

▶ Establish a serial connection via the Ethernet interface of the PC with the XC202.
Information on this is provided in → Section 3.4.6, "Connect PC", page26 and → Chapter 6, "Connection set-up PC – XC200", Page 70.

▶ Go to the "PLC Configuration" window in XSOFT-CODESYS-2 and open the "Firmware" tab.

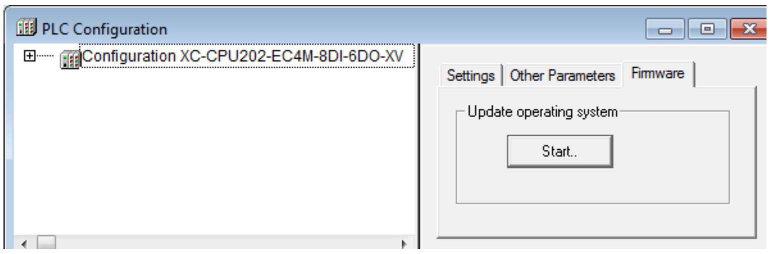
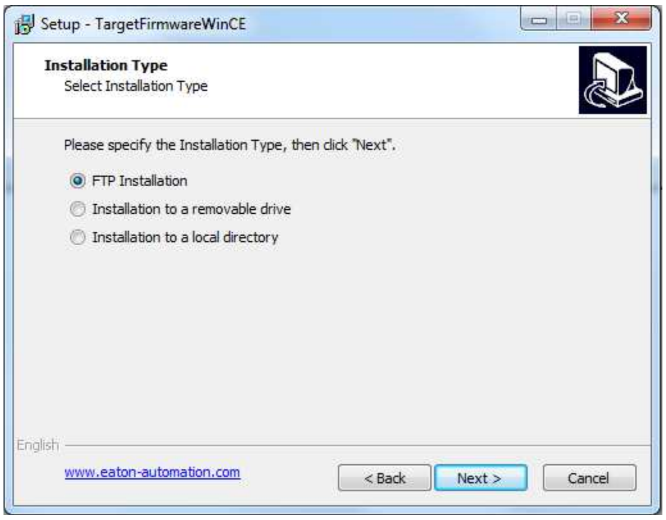


Figure 27:"PLC configuration" window

▶ Click on the "Start" button (under "Update operating system").
▶ Select the firmware file for the XC202 and click on the "Open" button.
▶ Click on "Next" to start the Setup Wizard.
▶ Select the "FTP installation" option and then click on "Next".

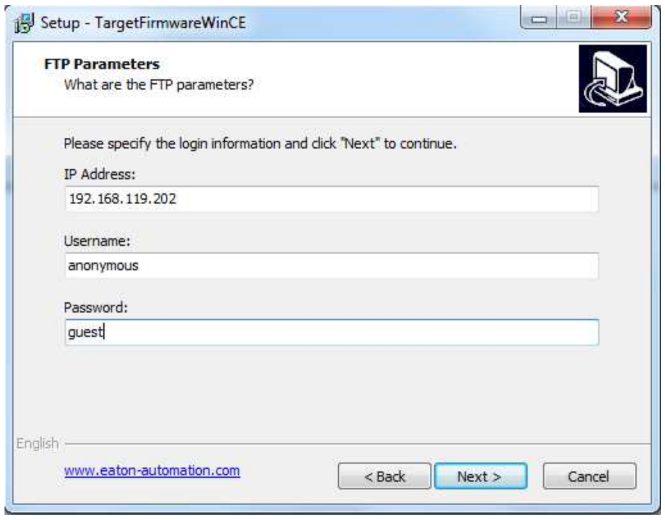


Please specify the Installation Type, then click "Next".

[✓ FTP Installation]
Installation to a removable drive
Installation to a local directory

Figure 28:"FTP installation" window

▶ Enter the PLC's IP address and click on "Next".
Note: The default network setting will be: IP address: 192.168.119.202


Setup - TargetFirmwareWinCE

FTP Parameters
What are the FTP parameters?

Please specify the login information and click "Next" to continue.

IP Address: 192.168.119.202

Username: anonymous

Password: guest

Figure 29:FTP parameters

- ▶ Select the “XC202” device type and click on “Next”.
- ▶ Select the “PLC operating system“ component and click on “Next”.
- ▶ Click on “Install” to start downloading the firmware.
- ▶ Wait until the window shows a message saying “Update finished”. Then close the window with “RETURN”.
- ▶ Click on “Done” to exit the Setup Wizard.

The update process when using removable media or a local directory (→ Figure 28, page 45) is similar to that for FTP installation.

Procedure:

- ▶ Plug the USB flash drive or SD card into the computer.
- ▶ Go to the “Setup - TargetFirmwareWinCE” (→ fig. 28) window and select “Install using removable media”.
- ▶ Select the removable media directory and click on “Next”.
- ▶ Select the “XC202” device type and click on “Next”.
- ▶ Select the “PLC operating system” component and click on “Next”.
- ▶ Click on “Install” to start downloading the firmware.
- ▶ Click on “Done” to exit the Setup Wizard.

The firmware files will be found in a directory on the removable medium called **CONTROL/XC-CPU202**.

- ▶ Plug the USB flash drive or SD card into the XC-CPU202.

The system will check whether there is a different operating system on the removable medium. If one is found, the system will start updating the firmware. When this occurs, the RUN/STOP (green) and SF (red) LEDs will flash alternately.

→ If you are updating a 01.00.xx firmware version to a 2.4.xx firmware version, the filesystem for the local “disk_sys” memory will be changed. This change may take up to 10 minutes.
The XC-CPU202's RUN/STOP (green) and SF (red) LEDs will flash alternately when the system is writing to the PLC's internal memory.

→ The new firmware version will not be enabled until the PLC is restarted.

NOTICE

If you are updating a 01.00.xx firmware version to a 2.4.xx firmware version, the filesystem will be replaced. All data stored on the local “disk_sys” memory (PLC program / registry settings / your own data) will be deleted.
This data will not be deleted if you are updating from a 1.00.xx to another 1.00.xx firmware version or from a 2.4.xx to another 2.4.xx firmware version.

5 Program processing, multitasking and system times

5.1 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.

You can set "Cyclic" or "Event-triggered" as the task condition. A cyclical task is restarted after the set interval time has elapsed. An event-triggered task is only started when the event occurs. You can also link system events such as "Start", "Stop" or "Reset" with 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 image 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 CODESYS or Online changes.

Updating of the input/output image by multiple tasks is described in the section "Multitasking" on Page 52.

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

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

→ For a detailed task configuration description, please refer to the manual for programming PLCs with CODESYS.

At the end the control specific settings are explained on the basis of an example.

Creating task (example)

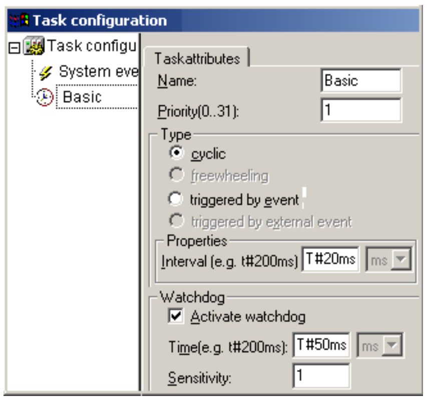
First create the cyclic task "Basic" with the assigned program "Basic_prog". Then you can add the event controlled task "Param" with the program "Param_prog". In the program "Basic_prog" an event is programmed which invokes the "Param" task.

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

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

5.1.1 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".
▶ Set the task in the dialog as in Figure 30.
▶ Click on the "Task configuration" folder and the configuration is accepted.



Name: Basic
Priority(0..31): 1
Type: [✓ cyclic] freewheeling triggered by event triggered by external event
Interval (e.g. t#200ms) T#20ms ms
[✓] Activate watchdog
Time(e.g. t#200ms): T#50ms ms
Sensitivity: 1

Figure 30:Parameterization of the cyclic task

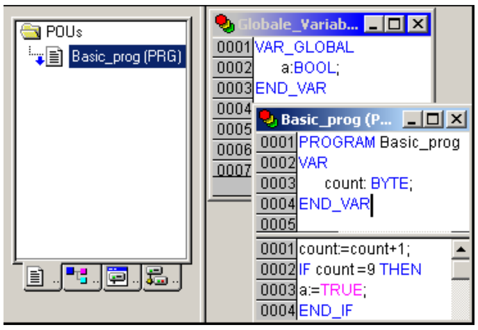
5.1.1.1 Define the program call

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.

5.1.1.2 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 now enter a program. In the program example (Figure 31) the variable “count” is incremented.
  On counter status = 9, a = TRUE.



```
VAR_GLOBAL
    a:BOOL;
END_VAR
```

```
PROGRAM Basic_prog
VAR
    count: BYTE;
END_VAR
```

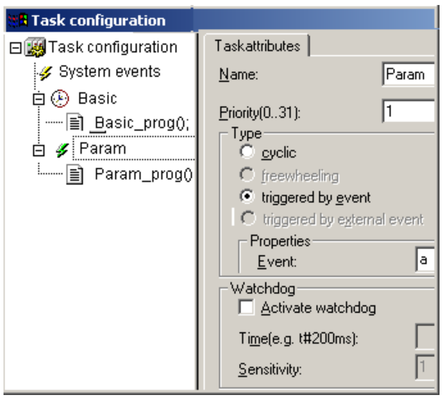
```
count:=count+1;
IF count=9 THEN
a:=TRUE;
END_IF
```

Figure 31:Creating a program element for a cyclic task

5.1.2 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 32.
- ▶ Define the Boolean variable “a” as the result of the event.
- ▶ Enter the program call “Param_prog”.



Name: Param

Priority(0..31): 1

Type
( ) cyclic
( ) freewheeling
(●) triggered by event
( ) triggered by external event

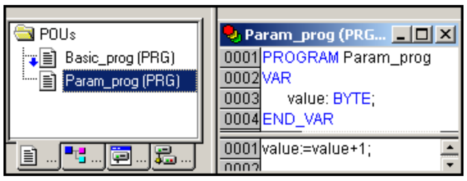
Properties
Event: a

Watchdog
☐ Activate watchdog
Time(e.g. t#200ms):
Sensitivity: 1

Figure 32:Creating an event controlled task

5.1.2.1 Writing a program

- ▶ Change over to the "Modules" tab and insert an object (POU) with the name "Param_prog".
- ▶ You can now enter a program. The program example Param_prog (Figure 33) increments the variable "value" by the value 1.
  The Param_prog is processed if the variable a = TRUE.



```
PROGRAM Param_prog
VAR
    value: BYTE;
END_VAR
```

```
value:=value+1;
```

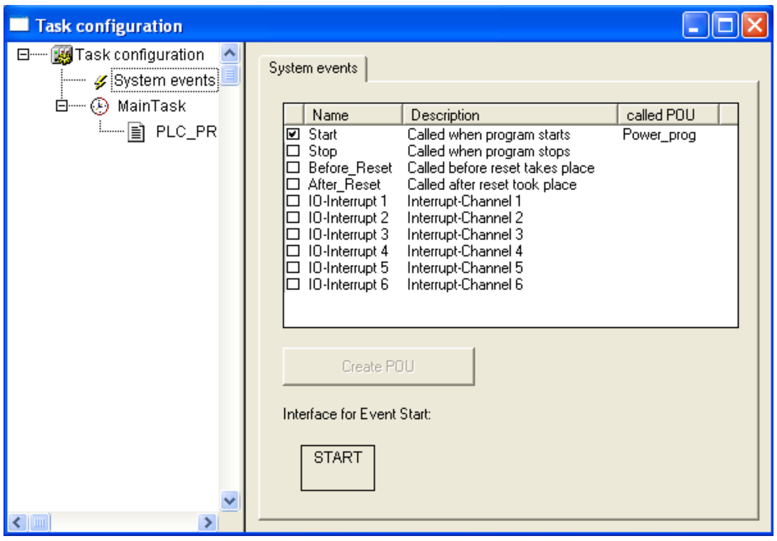
Figure 33:Programmed element for event controlled task

5.2 System events

A POU can be called with the help of a system event. It can be used when the PLC is started to initialize modules with parameters. The system events are independent of the task!

5.2.1 Assigning a POU to a system event

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



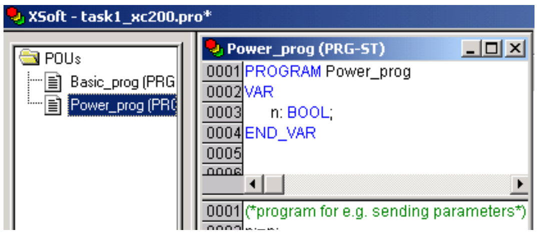
	Name	Description	called POU
☑	Start	Called when program starts	Power_prog
☐	Stop	Called when program stops	
☐	Before_Reset	Called before reset takes place	
☐	After_Reset	Called after reset took place	
☐	IO-Interrupt 1	Interrupt-Channel 1	
☐	IO-Interrupt 2	Interrupt-Channel 2	
☐	IO-Interrupt 3	Interrupt-Channel 3	
☐	IO-Interrupt 4	Interrupt-Channel 4	
☐	IO-Interrupt 5	Interrupt-Channel 5	
☐	IO-Interrupt 6	Interrupt-Channel 6	

Interface for Event Start:

START

Figure 34:Assigning the POU to a system event

- ▶ Change over to the "Resources → Modules" and add the object (POU) "Power_prog".
- ▶ Program the application.



```
PROGRAM Power_prog
VAR
    n: BOOL;
END_VAR
```

```
(*program for e.g. sending parameters*)
```

Figure 35:Programming a POU

→ Further information concerning the system events can be found in the online help of the programming system.

5.3 Multitasking

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

5.3.1 Updating the input/output images

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

The system starts searching the first task for programmed inputs, e.g. after the start. The term “First task” is the first task in the task configuration, irrespective of priority and the cycle time of the individual tasks. The name of the first task “Prog1” is in the Figure 37. If the system detects an input that is connected by the configuration with an input module, e.g. XIOC-16DI, all the inputs of this module are updated in the image. If other inputs are present in this task that are assigned to other modules, the inputs of these modules are also updated (module update procedure). If, for example, the inputs %IX6.0 and %IX7.1 of input module 1 are addressed by different tasks, the inputs of this module are only updated from the first task.

5.3.1.1 Examples

The examples are based on the following configuration:

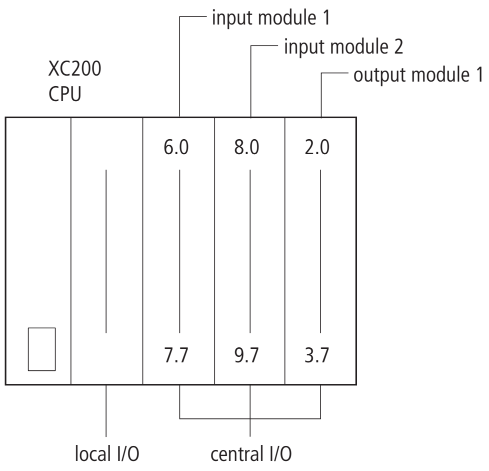


Figure 36:XC200 configuration

The task configuration appears as follows:

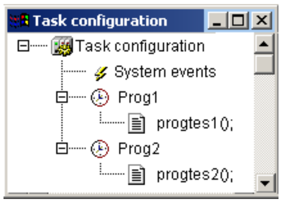


Figure 37:Task configuration for the examples

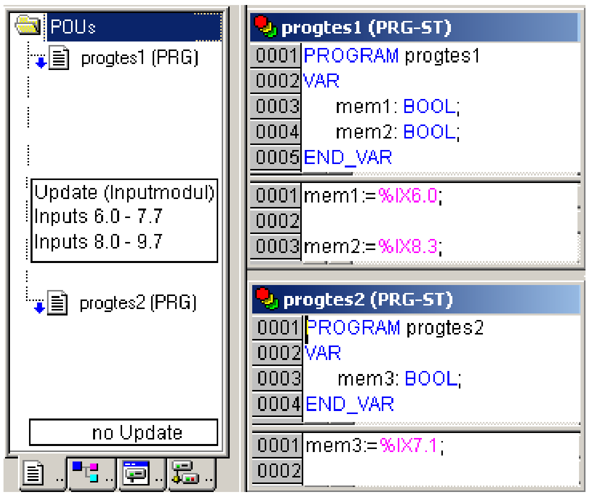
Example 1:

Table 6: 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 %IX7.1 of input module 1 is programmed in the program “progtes(2)”. Before the start of the 2nd task “Prog2” the inputs of this input module are not updated, as this only occurs in the 1st task.



```
PROGRAM progtes1
VAR
    mem1: BOOL;
    mem2: BOOL;
END_VAR
```

```
mem1:=%IX6.0;

mem2:=%IX8.3;
```

```
PROGRAM progtes2
VAR
    mem3: BOOL;
END_VAR
```

```
mem3:=%IX7.1;

```

Figure 38:Program for example 1

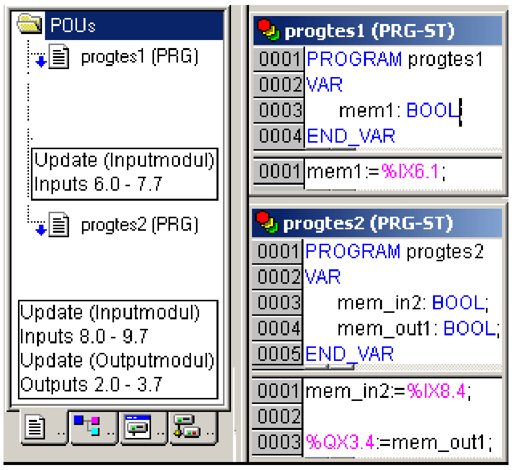
Example 2:

Table 7: 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.



```
PROGRAM progtes1
VAR
    mem1: BOOL
END_VAR
```

```
mem1:=%IX6.1;
```

```
PROGRAM progtes2
VAR
    mem_in2: BOOL;
    mem_out1: BOOL;
END_VAR
```

```
mem_in2:=%IX8.4;

%QX3.4:=mem_out1;
```

Figure 39:Program for example 2

5.3.1.2 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 parameterization 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.

5.3.2 Behavior 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 behavior 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.

5.4 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 user 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!

CAUTION
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 (stepping condition). The 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.

5.4.0.1 Watchdog configuration

You can select 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 5 × 10 ms = 50 ms.

5.4.0.2 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 (2 x 15 ms = 30 ms).

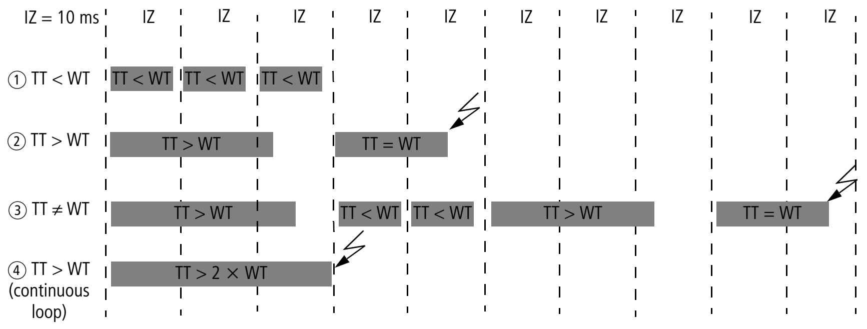


Figure 40:Watchdog active, multiple tasks with differing priority

5.4.0.3 Watchdog deactivated

The cycle 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.

5.4.0.4 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.

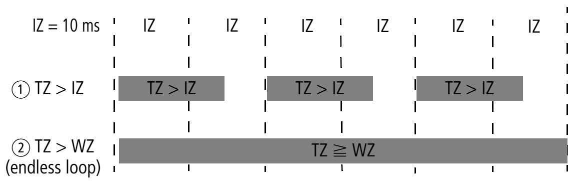


Figure 41:Watchdog deactivated

5.4.1 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).

5.5 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 XIOC signal modules. XIOC signal modules which can be integrated via a bus system cant 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 XIOC signal modules, access occurs as a bit/word or read/write operation. The access parameter indicates the Table 8.

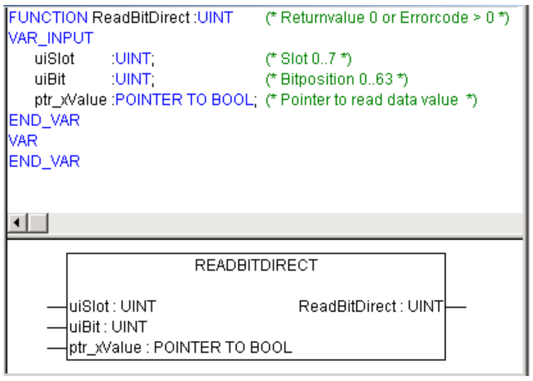
The inputs/outputs which are required for “Direct peripheral access” are physically connected in the same manner as normal inputs/outputs.

Table 8: "Direct peripheral access" overview

Card	I/O bit access			I/O word access			I/O slot
	Read	Write	Parameters/module	Read	Write	Parameters/ module	Parameter
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
XC-CPU202-EC4M-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-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
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

5.5.1 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.



```
FUNCTION ReadBitDirect :UINT      (* Returnvalue 0 or Errorcode > 0 *)
VAR_INPUT
    uiSlot      :UINT;            (* Slot 0..7 *)
    uiBit       :UINT;            (* Bitposition 0..63 *)
    ptr_xValue :POINTER TO BOOL;  (* Pointer to read data value  *)
END_VAR
VAR
END_VAR
```

Figure 42:Function "ReadBitDirect"

5.5.1.1 Parameters of the "ReadBitDirect" function

uiSlot	Slot number of the signal module. For possible parameters see Table 8 on Page 59.
uiBit	Bit position within the input value of the signal module. For possible parameters see Table 8 on Page 59.
ptr_xValue	Pointer to the variable value
ReadBitDirect	Display of the failure code, see Table 9 on Page 63

5.5.2 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.

5.5.2.1 Parameters of the "ReadWordDirect" function

uiSlot	Slot number of the signal module. For possible parameters see Table 8 on Page 59.
uiOffset	Word offset within a signal module. For possible parameters see Table 8 on Page 59
ptr_wValue	Pointer to the variable value
ReadWordDirect	Display of the failure code, see Table 9 on Page 63

5.5.3 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.

5.5.3.1 Parameters of the "ReadDWordDirect" function

uiSlot	Slot number of the signal module. For possible parameters see Table 8 on Page 59
uiOffset	Word offset within a signal module. For possible parameters see Table 8 on Page 59
ptr_dwValue	Pointer to the variable value
ReadDWordDirect	Display of the failure code, see Table 9 on Page 63

5.5.4 Write…Direct

Fundamentally, the outputs of the PLC should only be modified by a task or an interrupt. Please 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 function, 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).

5.5.4.1 Example

Output variable declaration:

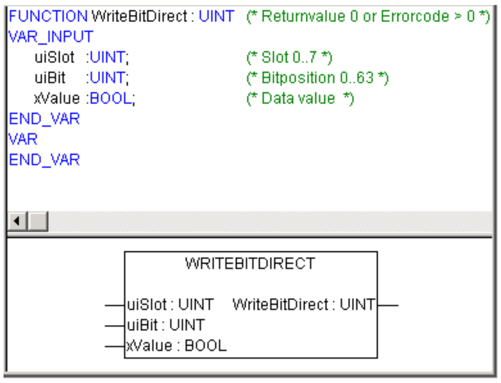
```
Q-BOOL (e.g. Qbit3)        AT%QX1.2:BOOL;
Q-WORD (e.g. Qword0)       AT%QW0:WORD;
```

Referencing (assignment in the application):

```
Qbit3:=TRUE;
Qword0:=16#Test;
```

5.5.5 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”.



```
FUNCTION WriteBitDirect : UINT  (* Returnvalue 0 or Errorcode > 0 *)
VAR_INPUT
    uiSlot  :UINT;              (* Slot 0..7 *)
    uiBit   :UINT;              (* Bitposition 0..63 *)
    xValue  :BOOL;              (* Data value  *)
END_VAR
VAR
END_VAR
```

Figure 43:WriteBitDirect function

5.5.5.1 Parameters of the “WriteBitDirect” function

uiSlot	Slot number of the signal module. For possible parameters see Table 8 on Page 59.
uiBit	Output bit within the signal module. For possible parameters see Table 8 on Page 59.
xValue	The pointer points to the variable in which the value for the output bit is located.
WriteBitDirect	Display of the failure code, see Table 9 on Page 63

5.5.6 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.

5.5.6.1 Parameters of the “WriteWordDirect” function

uiSlot	Slot number of the signal module. For possible parameters see Table 8 on Page 59.
uiOffset	Output word within a signal module. For possible parameters see Table 8 on Page 59.
wValue	The pointer points to the variable in which the value for the output word is located.
WriteWordDirect	Display of the failure code, see Table 9 on Page 63

5.5.7 GetSlotPtr

→ This function is not available!

5.5.8 Failure 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 dependence on the parameterized signal module and the physical existence of the signal module. If a fault is determined, access is not undertaken and a failure code is output. The data fields for the value transfer remain unchanged. The DisableInterrupt and EnableInterrupt functions do not generate a failure code.

The following return values are possible:

Table 9: Failure 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

5.6 Operating states

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

Table 10: Definition of the states of the XC200 with LED display

Status	View		Definition
	RUN/STOP	SF	
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.
Start operating system	off (flashes at Start)	off	Windows CE system start and system test are carried out. Start of applications • HTTP-Server • FTP-Server • Telnet-Server • PLC-Runtime • Web Server
STOP	flashes	off	PLC in STOP state
RUN	on	off	PLC in RUN state

Status	View		Definition
	RUN/STOP	SF	
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 getterrorlist)
NOTREADY	flashes	on	No start possible. A major fault prevents a start (see "Error List") e.g.: • No program loaded • Field bus error • Configuration not OK • Checksum error • ...
ShutDown	flashes	flashes	Wait for the supply voltage to disconnect (after shutdown browser command)

5.7 Web visualization

A description of the web visualization interface can be found in section 7.4, "Web visualization", of the manual for programming PLCs with CODESYS.

The XC-CPU201 specific call for the web visualization is as follows:

http:\\192.168.119.200:8080/webvisu.htm

The XC-CPU202 specific call for the web visualization is as follows:

http:\\192.168.119.202:8080/webvisu.htm

Prerequisite: You have not changed the default setting of the IP address.

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

NOTICE

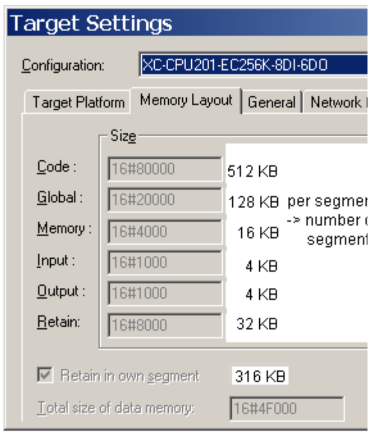
A max. of 10 clients may access the XC200!

5.8 Limit values for memory usage

The data memory of the XC200 is divided into memory segments for data. The segment sizes are shown in Figure 44. The global data utilizes 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›:

Example:



Target Settings

Configuration: XC-CPU201-EC256K-8DI-6DO

Target Platform | Memory Layout | General | Network

Size

Code :	16#80000	512 KB
Global :	16#20000	128 KB per segmer -> number segment
Memory :	16#4000	16 KB
Input :	16#1000	4 KB
Output :	16#1000	4 KB
Retain:	16#8000	32 KB

☑ Retain in own segment 316 KB

Total size of data memory: 16#4F000

Figure 44:Segment size of the XC-CPU201-EC256k

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

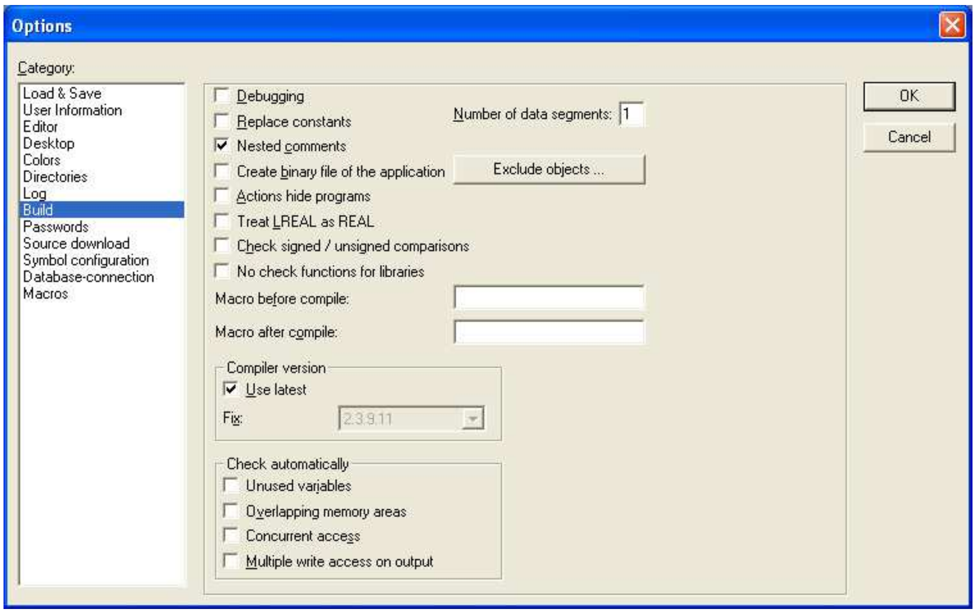
In order to ensure that you use the available memory for the global data in an optimum and efficient manner, we recommend that you make the following settings when a new project is being created:

Control Type	**Number of data segments (global)**
XC-CPU201-EC256K	2
XC-CPU201-EC512K	4
XC-CPU202-EC4M	1

The number of segments is set to 1 by default.

The number of segments is changed as follows:

▶ Choose ‹Project → Options → Conversion Options› and then the Number of data segments field, and enter the appropriate number of segments shown above for the selected controller type.


Options

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

- [ ] Debugging
- [ ] Replace constants
- [x] Nested comments
- [ ] Create binary file of the application
- [ ] Actions hide programs
- [ ] Treat LREAL as REAL
- [ ] Check signed / unsigned comparisons
- [ ] No check functions for libraries

Number of data segments: 1

Macro before compile:

Macro after compile:

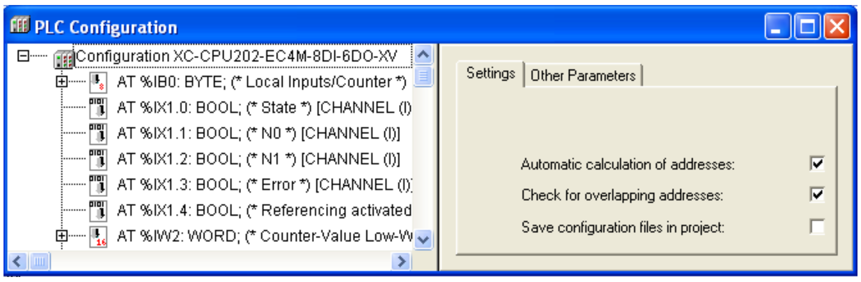
Compiler version
- [x] Use latest
- Fix: 2.3.9.11

Check automatically
- [ ] Unused variables
- [ ] Overlapping memory areas
- [ ] Concurrent access
- [ ] Multiple write access on output

Figure 45:Memory management: Change the number of data segments

5.9 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.


PLC Configuration

```
Configuration XC-CPU202-EC4M-8DI-6DO-XV
  AT %IB0: BYTE; (* Local Inputs/Counter *)
  AT %IX1.0: BOOL; (* State *) [CHANNEL (I)
  AT %IX1.1: BOOL; (* N0 *) [CHANNEL (I)]
  AT %IX1.2: BOOL; (* N1 *) [CHANNEL (I)]
  AT %IX1.3: BOOL; (* Error *) [CHANNEL (I)
  AT %IX1.4: BOOL; (* Referencing activated
  AT %IW2: WORD; (* Counter-Value Low-W
```

Settings | Other Parameters

- Automatic calculation of addresses: [x]
- Check for overlapping addresses: [x]
- Save configuration files in project: [ ]

Figure 46: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.

5.9.1 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.

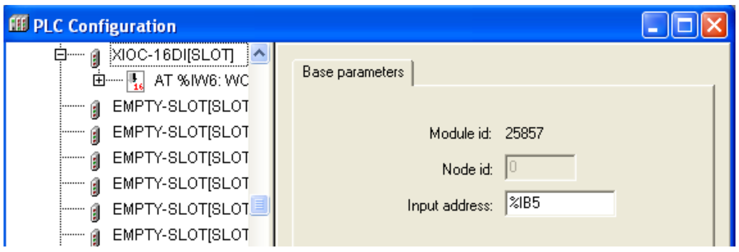
If you add a module, the addresses of all the subsequent modules (independently of the line) are offset by the address value of the added module, and the added module is assigned with an address. Modules which are located in the configuration before the added module are not changed. If you remove the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

5.9.2 “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.

5.9.3 Uneven word addresses

If you assign an odd offset address (e.g. IB5) to a word addressable module in the “Entry address” field, the next even word address (IW6) automatically appears in the PLC configurator. This is completed automatically and is not controlled by the “Check address overlap” setting.



Module id: 25857
Node id: 0
Input address: %IB5

Figure 47:Uneven address

5.9.4 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 11: Address ranges

Control	Input			Output			Marker		
	Size [kByte]	Max. Byte address	Max. Word address	Size [kByte]	Max. Byte address	Max. Word address	Size [kByte]	Max. Byte address	Max. Word address
XC101-64k	2	2047	2046	2	2047	2046	4	4095	4094
XC101-128k	4	4095	4094	4	4095	4094	8	8191	8190
XC101-256k	16	16383	16382	16	16383	16382	16	16383	16382

Control	Input			Output			Marker		
	Size [kByte]	Max. Byte address	Max. Word address	Size [kByte]	Max. Byte address	Max. Word address	Size [kByte]	Max. Byte address	Max. Word address
XC201-256k	4	4095	4094	4	4095	4094	16	16383	16382
XC201-512k	4	4095	4094	4	4095	4094	16	16383	16382
XC202-EC4M	4	4095	4094	4	4095	4094	16	16383	16382

5.9.5 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.

5.9.6 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 node 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.

5.10 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 XIOC modules: • 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 or start of the program.	XDiag_SystemDiag	xSysDiag.lib	MN05010002Z-EN (previously called AWB2786-1456)
Inspection of the XIOC-NET-DP-M module and the stations on the DP line	XDiag_SystemDiag XDiag_ModuleDiag	XSysDiag.lib	MN05010002Z-EN (previously called AWB2786-1456)
	DiagGetState	BusDiag.lib	MN05002002Z-EN (previously called AWB2725-1452)
Inspection of the XIOC-NET-DP-S module	XDiag_SystemDiag XDiag_ModuleDiag	xSysDiag.lib	MN05010002Z-EN (previously called AWB2786-1456)
DP slave provides the master with additional diagnostics data.	XDPS_SendDiag	xSysNetDPSDiag.lib	MN05002002Z-EN (previously called AWB2725-1452)

6 Connection set-up PC – XC200

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

- the RS232 interface
- the Ethernet interface

In this chapter you will get to know the settings to be made in the CODESYS software.

See also:
- Connect PC → Page 26

6.1 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 Rate	38400
Parity	No
Stop bits	1
Motorola Byte	No

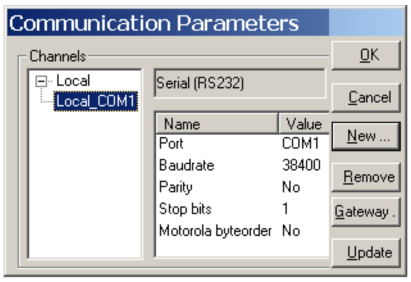
→ If you get an error message during login, the CPU's default settings have already been changed.In that case try a baud rate of 57600 Bit/s.

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

6.2 Defining/changing the PC's communication settings

Define the communication parameters of the interface in the CODESYS software. 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 48.
▶ Confirm the settings with OK.
▶ Log on to the PLC.


Communication Parameters

Channels: Local / Local_COM1

Serial (RS232)

Name	Value
Port	COM1
Baudrate	38400
Parity	No
Stop bits	1
Motorola byteorder	No

Figure 48:Defining the PC's communication settings

→ For more information on these communication settings, please refer to the programming system's online help.

From operating system version V01.03.xx of the XC-CPU201 the serial (RS232) (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.

6.2.0.1 Changing settings

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

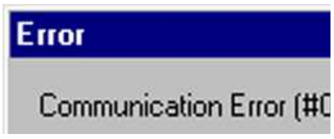
- ▶ Double-click the appropriate value, e.g. 38400.
  The field is dimmed.
- ▶ Enter the desired value.

Double-click this field once more to choose the Baud rate, e.g. 57600 Bit/s.

6.2.1 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:


Error

Communication Error (#C

Figure 49: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".

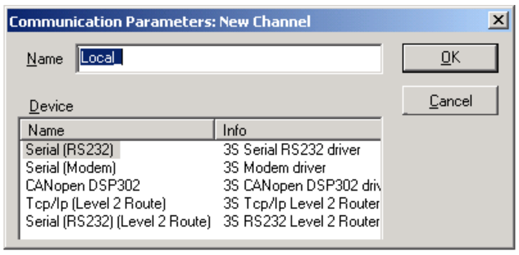
6.3 Connection set-up with Ethernet

After you have connected the PC with the CPU using a cable, select the TCP/IP communication channel in the CODESYS software and enter the IP address of the CPU. The XC-CPU201 has the default address 192.168.119.200, the XC-CPU202 has the default address 192.168.119.202.

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

6.3.1 Selecting communication channel and address

▶ Access the menu with ‹Online → Communication parameters›.



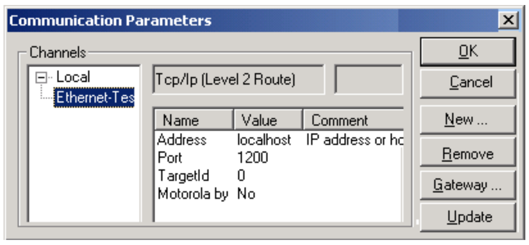
Name: Local

Device

Name	Info
Serial (RS232)	3S Serial RS232 driver
Serial (Modem)	3S Modem driver
CANopen DSP302	3S CANopen DSP302 driv
Tcp/Ip (Level 2 Route)	3S Tcp/Ip Level 2 Router
Serial (RS232) (Level 2 Route)	3S RS232 Level 2 Router

Figure 50:Channel selection

▶ Push the “New…” button.
▶ Select the overview of the communication channel TCP/IP (Level 2 Route) and change the name “local” e.g. to “Ethernet-Test”.
▶ Confirm with OK.

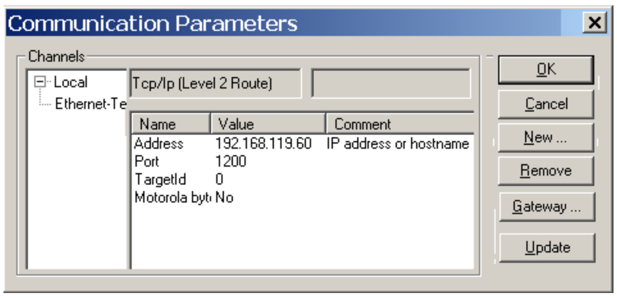


Tcp/Ip (Level 2 Route)

Name	Value	Comment
Address	localhost	IP address or hc
Port	1200	
TargetId	0	
Motorola by	No	

Figure 51:Enter the IP address

▶ Perform a double click on the “localhost” field and enter the default address 192.168.119.200 or 192.168.119.202.
▶ Confirm your details, by first pressing on another field and then on OK.



Tcp/Ip (Level 2 Route)

Name	Value	Comment
Address	192.168.119.60	IP address or hostname
Port	1200	
TargetId	0	
Motorola byt	No	

Figure 52:Communication parameters with IP address

▶ Compile the program and log in.

6.4 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 128).

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

Ensure that the IP address of the programming device (= PC) belongs to the same address family as the PLC. This means that the IP address of the programming device and the XC200 match in the following number groups:

6.4.0.1 Example 1

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

6.4.0.2 Example 2

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

The following conditions apply in examples 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:

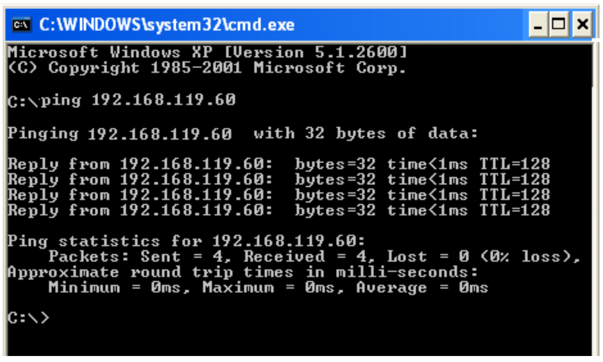
▶ Open the DOS window via the "Start" field and the "Run" command.
▶ Enter "CMD" in the input field and confirm with OK.

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 for XC-CPU201 or ping 192.168.119.202 for XC-CPU202. 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 set-up.

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 53:PING response with a correctly established Ethernet connection

7 Setting system parameters via the Startup.ini file

7.1 Overview

System parameters independent of the project can be set by you and saved on the memory card. They are compiled there to a Startup.ini file. The memory card can also be plugged into other PLCs. The PLC accepts the parameters during start up. The Startup.ini file is always created will all controller parameters (→ Table 12). The term Startup.ini file is generally applicable. The file name of the Startup.ini file for the XC200 is XCSTARTUP.ini.

7.1.1 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 12: Predefined default parameters in the XCSTARTUP.ini file

```
TARGET=XC-CPU202
HOST_NAME=NoNameSet
IP_ENABLE_DHCP=0[1)]
IP_ADDRESS=192168119202
IP_SUBNETMASK=255.255.255.0
COM_BAUDRATE=38400
CAN_ROUTING_CHANNEL=1
```

1) Note: The parameter IP_ENABLE_DHCP only exists on an XC-CPU202.

Table 13: Example: contents of the XCSTARTUP.ini file

```
[STARTUP]
TARGET=XC-CPU202
to the Ethernet connection:
HOST_NAME=NoNameSet
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)
```

7.2 Structure of the ini file

An ini file is a text file with a defined data format. From a named section such as [STARTUP], followed by an equals sign and the corresponding value. 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
```

The parameters can be changed or created with a text editor if you insert the memory card into the memory card slot of a PC.

- The file XCSTARTUP.ini is stored on the memory card of the XC-CPU201 in the directory:
  disk_mmc\MOELLER\XC-CPU201-EC256K-8DI-6DO\PROJEKT
- The file XCSTARTUP.ini is stored on the memory card of the XC-CPU202 in the directory:
  disk_mmc\CONTROL\XC-CPU202-EC4M-8DI-6DO-XV\PROJEKT

7.3 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.

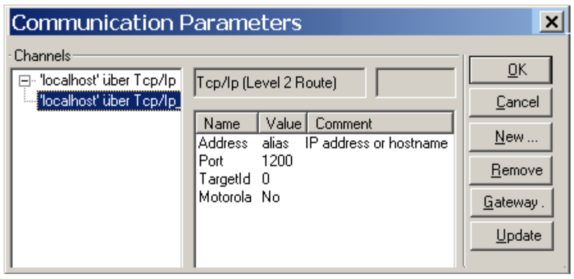
With the browser command createstartupini you will transfer from the PLC either the STARTUP data – or if a project is contained – the system parameters onto the memory card. This creates the Startup.INI file which contains this data. Precondition: the memory card must be plugged in, formatted and empty,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

It is not possible to overwrite or change an already existing file with the createstartupini browser command. 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 77.

7.4 Entry of the ini file HOST_NAME

The parameter HOST_NAME enables the controller to be addressed via the Ethernet with this device name. It can also be addressed with its IP address. The device name for the XC-CPU201 is available from operating system version 1.04. It receives from the system the entry "NoNameSet". If this is not changed, the device linked to the Ethernet can only be addressed via its IP address. You can enter a new device name using the browser command settargetname. The device name must be assigned uniquely for each device.

It can also be used as a communication parameter (in Figure 54: alias), if the parameters are defined in the programming software in the menu ‹Online I Communication parameters›.
The parameters define the properties of the programming connection between the PC and the PLC.



Name	Value	Comment
Address	alias	IP address or hostname
Port	1200	
TargetId	0	
Motorola	No	

Figure 54:Communication parameters

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

7.4.1 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.

7.4.2 Alter parameters

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.

7.4.3 Deleting the Startup.ini file

The following browser commands can be used to access the memory card.

- removestartupini: Always deletes the controller system parameters. If a memory card is plugged in, the INI file on the memory card is deleted. 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 in the controller are retained.

The behavior of the Startup.ini file with the “Hard Reset” and “Default Settings” menu commands on the controller and with the factoryset browser command is described in section “Reset” on Page 36. If you execute the “Full reset” command in online mode, the operating system and the project on the Disk_sys are deleted. The XCSTARTUP.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 directly with the target PLC. It can instead be connected to any other PLC in the network. The routing connection enables you to carry out all the operations that are possible with a direct online connection between the programming device and the controller:

- Program Download
- Online changes
- 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 device can access all routing capable PLCs on the CAN bus. You can determine in the project selection which controller you wish to communicate with. This provides an easy way of controlling remote PLCs.

However, the data transfer from routing connections is significantly slower than with direct (serial or TCP/IP) connections. This results, for example, in slower display refresh rates of variables and longer download times.

8.1 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.
- The routing with the XC-CPU201 is possible from operating system version V1.03.02.

8.2 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..

8.2.0.1 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 block size (4 kByte or 128 kByte) depends on the transfer type (program transfer or routing) and the operating system → Table 14.

Table 14: Block size for data transfer

	Program/file transfer		**Routing**		
	XC-CPU201 OS version < V1.03.02	**XC-CPU201 OS version ≧ V1.03.02**	**XC-CPU201 OS version < V1.03.02**	**XC-CPU201 OS version ≧ V1.03.02**	**XC-CPU202 OS version ≦ V1.00.07**
Block size Default 128 kByte	128 kByte	128/4 kByte	Routing not possible	4 kByte	4 kByte

BTS = operating system

NOTICE
The program download with a block size of 4 KByte to an XC-CPU201 with an operating system version < V1.03.02 causes a malfunction!
If a program download is performed, the progress bar on the programming device monitor will only change erratically (about every 10 seconds).

The setting of the block size (change of the value in the registry) is explained as follows.

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

Setting the block size:

- ▶ Close all CoDeSys applications.
- ▶ Close the CODESYS gateway server.

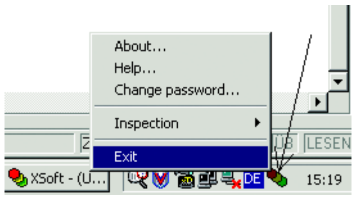


Figure 55:Closing the CODESYS gateway server

▶ Change the block size to the required value.

The following *.reg files are available in the 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).

8.2.1 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/Device): it is possible to access a target PLC which has not been configured as a master or as a device. It must only receive the basic parameters such as node ID and baud rate, as well as a simple program.

8.2.2 Addressing

Controllers on the CAN bus can be configured a master or as a slave (device). 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 RS232 or Ethernet interface can be used as a connection between the PC and XC200.

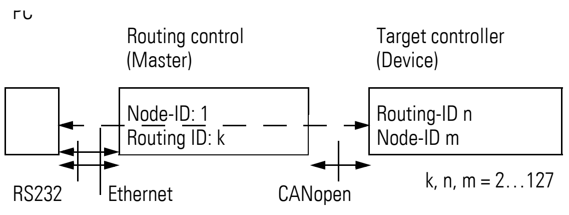


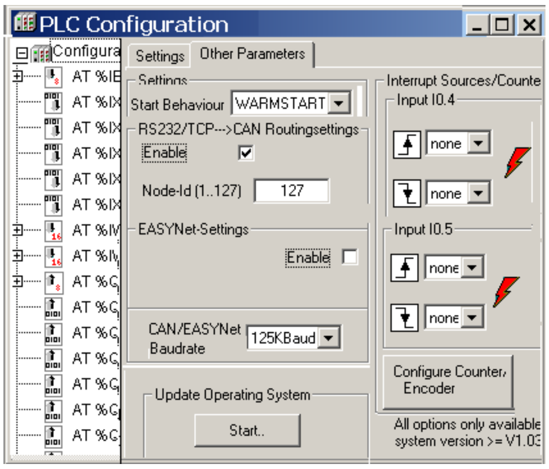
Figure 56:Routing via XC…, EC4P, …

Table 15: Example for setting the Node Id, Baud rate

Control	Function	Node ID	Routing ID	Baud Rate	→ Fig.
Routing controller	master	1	127	125 KB	58
Target controller	Device	3	54	125 KB	59

→ 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 ≧ V2.0: the Routing-ID must be **equal** to the Node-ID!

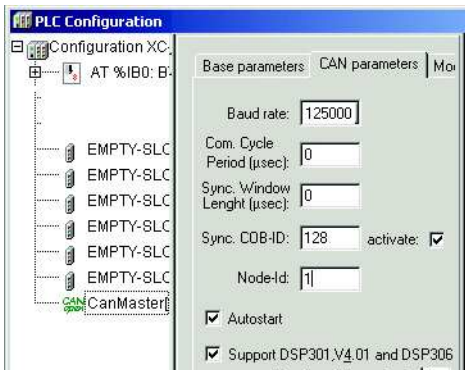
The Routing-ID with the master can be set in the PLC configurator in the “Other parameters” tab:



Settings
Start Behaviour: WARMSTART
RS232/TCP--->CAN Routingsettings
Enable ☑
Node-Id (1..127): 127
EASYNet-Settings
Enable ☐
CAN/EASYNet Baudrate: 125KBaud
All options only available system version >= V1.03

Figure 57:CAN Master routing settings

The ID for basis communication is defined in the "CanMaster" folder in the "CAN parameters" tab (Figure 58).



Baud rate: 125000
Com. Cycle Period (µsec): 0
Sync. Window Lenght (µsec): 0
Sync. COB-ID: 128 activate: ☑
Node-Id: 1
☑ Autostart
☑ Support DSP301,V4.01 and DSP306

Figure 58:CAN Master: Node ID for basis communication

8.2.3 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.
▶ Determine the communication settings for the PC and the PLC 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 change
- Program test (Debugging)
- Create boot project
- Source code storing

Note for project creation

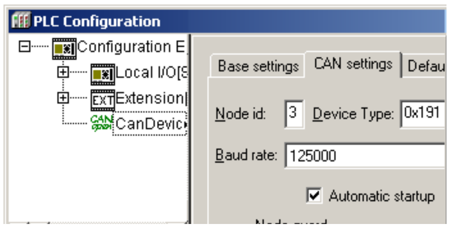
Assign two Node-IDs to the target PLC:

- One ID for basic communication
- One ID for routing

You set the routing ID (node ID, e.g. 54) and the baud rate of the target PLC (e.g. XC200) in the PLC configuration in the Other parameters window → Figure 57. First click the Activate box in the "RS232/TCP → CAN Routing" field. The activation is required so that the PLC can communicate via the CAN bus. Then enter the Node ID/node number and the baud rate on the CAN 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 Kbits/s.

The ID for basis communication is defined in the “CanDevice” in the “CAN setting” tab → Figure 59.



Node id: 3 Device Type: 0x191

Baud rate: 125000

☑ Automatic startup

Figure 59:CAN device parameters

ID and baud rate are transferred with the project download to the PLC.

Example

The following example based on Figure 60 shows the access to a PLC program.

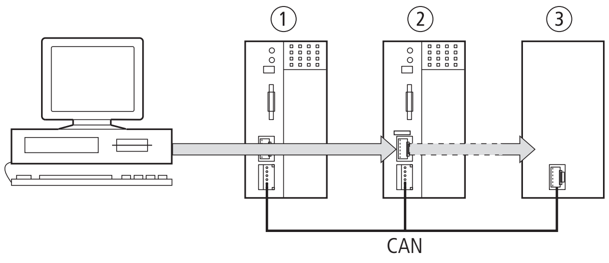


Figure 60:Diagnostics possibilities

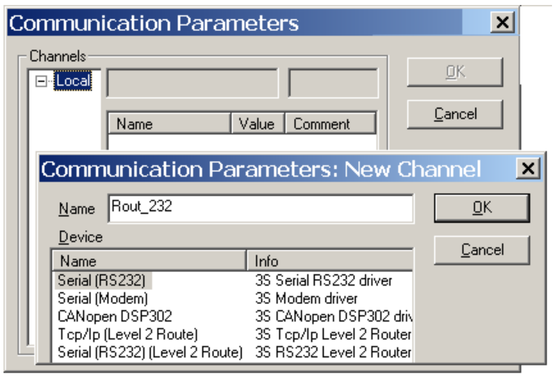
① XC100 with Node ID 1
② XC200 with node ID 2, routing ID 127
③ PLC (e.g . XC..., EC4P...) with node ID 3 and routing ID 54

You have connected the PC to the controller with node ID 2 and wish 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).
- ▶ Choose ‹Online → Communication parameters› menu.
- ▶ Click the “New” button under “local” channels.

The “New Channel” window appears.

- ▶ Select the channel in the “Device” window:Serial [RS232] [Level 2 Route] or TCP/IP [Level 2 Route].
- ▶ You can assign a new name (e. g. „Rout_232“) in the “Name” field.
- ▶ Confirm with OK. You will return to the initial window.


Communication Parameters

Channels: Local

Name	Value	Comment

Communication Parameters: New Channel

Name: Rout_232

Device

Name	Info
Serial (RS232)	3S Serial RS232 driver
Serial (Modem)	3S Modem driver
CANopen DSP302	3S CANopen DSP302 driv
Tcp/Ip (Level 2 Route)	3S Tcp/Ip Level 2 Router
Serial (RS232) (Level 2 Route)	3S RS232 Level 2 Router

Figure 61: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 PLC which you want to program or 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 the field in the Value column to the right of the term Target ID. Enter there the number 54 and confirm with “OK”.

▶ Log in and complete the operation.

8.2.4 PLC combinations for routing

The following PLC support routing:

From → **To ↓**	**XC100, XC121**	**XC-CPU201**[1)]	**EC4P**	**XC-CPU202**
XC100, XC121	×	×	×	×
XC-CPU201[1)]	×	×	×	×
XC-CPU202	×	×	×	×
EC4P	×	×	×	×

1) XC-CPU201 from operating system version V01.03.01

8.2.5 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. This therefore enables the status of PLC 2 and PLC 3 to be displayed at the same time.

Table 16: Type and number of communication channels

Communication Channel	**Control**	**Max. channel number**
TCP/IP Level 2 Route	XC200	5
Serial RS232 Level2 Route	XC.../EC4P	1

9 RS 232 interface in Transparent mode

In Transparent mode, data is exchanged between the XC200 and data terminal devices (e.g. terminals, printers, PCs, measuring devices) without any interpretation of the data. For this purpose, the serial RS232 interface of the CPU or XIOC-SER modules is to be switched using the user program in the transparent mode. This applies from operating system version 01.03.xx for the RS232 interface of the XC-CPU201.

	XC-CPU201	**XC-CPU202**
RS232 of the CPU	COM1	LocalCOM
RS232 of the XIOC-SER and XIOC-TC1	COM2…5	COM2…5
xSysCom200.lib for XIOC-SER and XIOC-TC1 SyslibCom.lib for RS232 of CPUs		

→ If the RS232 interface of the XC-CPU201 is in Transparent mode, programming via this interface not possible. However, you can test the program via the Ethernet interface (so-called joint operation).

→ This type of joint operation is not possible on the XC-CPU202. The RS232 is set by default for programming mode. The Serial Programming Off and Serial Programming On browser command or the library functions (FUN) Disable Com Programming and Enable Com programming enable the RS232 interface to be switched from programming mode to Transparent mode.

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.

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 RS232 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

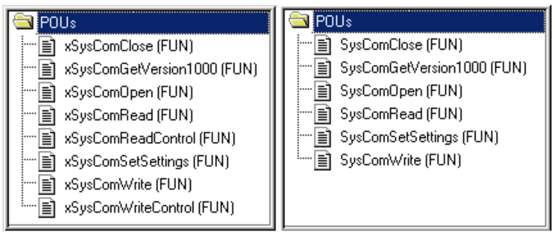
The RS232 interface of the CPU is addressed (in contrast to the interface of the XIOC-SER module) via the operating system! Therefore, execution of the interface functions can take up to 50 ms. The task, in which the RS232 interface is contacted should have an interval time of at least 50 ms and be assigned with a low priority (high value) in multitasking mode, so that time critical tasks are not hindered.

The functions (x)SysComRead/Write therefore only process parts of the required data length. To transfer data blocks completely, repeated calls with adjusted offset values must be carried out in several task intervals. The number of calls depends on the baud rate and the data volume.

The performance of the RS232 of the CPU depends on the load of the PLC (PLCLoad) and the selected baud rate. Due to the high interval times of the COM1 task, it may be displaced by time-critical tasks. When data is received at high baud rates, characters may be lost!

9.1 Programming of the RS 232 interface in transparent mode

You can access the data of the RS232 interface using the user program. The libraries xSysCom200.lib or SysLibCom.lib are provided for this purpose. Note that only one of the two libraries can be incorporated in the Library Manager! Both libraries provided a large number of functions, such as for opening and closing the interface. The library functions are shown here next to each other: the functions of the xSysCom200.lib library on the left and the SysLibCom.lib library.



POUs
- xSysComClose (FUN)
- xSysComGetVersion1000 (FUN)
- xSysComOpen (FUN)
- xSysComRead (FUN)
- xSysComReadControl (FUN)
- xSysComSetSettings (FUN)
- xSysComWrite (FUN)
- xSysComWriteControl (FUN)

POUs
- SysComClose (FUN)
- SysComGetVersion1000 (FUN)
- SysComOpen (FUN)
- SysComRead (FUN)
- SysComSetSettings (FUN)
- SysComWrite (FUN)

Figure 62:Overview of functions (left: xSysCom200.lib, right: SysLibCom.lib)

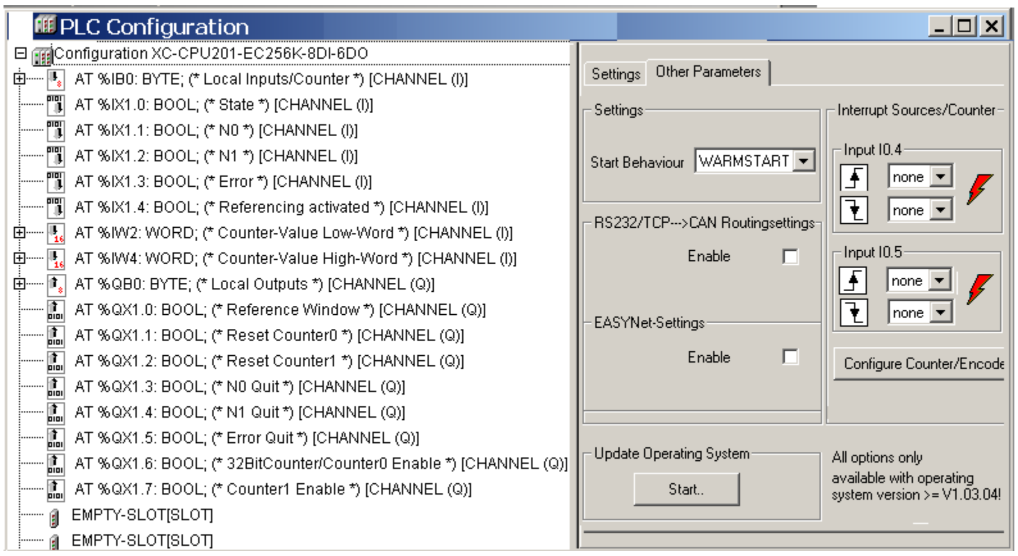
The functions are described in the manual “Function blocks of easySoft-CoDeSys” (MN05002002Z-EN).

See also: Transparent mode: Text output via RS232 (example) → Page 137

10 Configuration and parameterization of the inputs/outputs

10.1 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 add-on functions such as e.g. the counters. The inputs and outputs of the add-on functions only become active after you have selected a function in the "Other parameters" tab.



PLC Configuration

```
Configuration XC-CPU201-EC256K-8DI-6DO
  AT %IB0: BYTE; (* Local Inputs/Counter *) [CHANNEL (I)]
  AT %IX1.0: BOOL; (* State *) [CHANNEL (I)]
  AT %IX1.1: BOOL; (* N0 *) [CHANNEL (I)]
  AT %IX1.2: BOOL; (* N1 *) [CHANNEL (I)]
  AT %IX1.3: BOOL; (* Error *) [CHANNEL (I)]
  AT %IX1.4: BOOL; (* Referencing activated *) [CHANNEL (I)]
  AT %IW2: WORD; (* Counter-Value Low-Word *) [CHANNEL (I)]
  AT %IW4: WORD; (* Counter-Value High-Word *) [CHANNEL (I)]
  AT %QB0: BYTE; (* Local Outputs *) [CHANNEL (Q)]
  AT %QX1.0: BOOL; (* Reference Window *) [CHANNEL (Q)]
  AT %QX1.1: BOOL; (* Reset Counter0 *) [CHANNEL (Q)]
  AT %QX1.2: BOOL; (* Reset Counter1 *) [CHANNEL (Q)]
  AT %QX1.3: BOOL; (* N0 Quit *) [CHANNEL (Q)]
  AT %QX1.4: BOOL; (* N1 Quit *) [CHANNEL (Q)]
  AT %QX1.5: BOOL; (* Error Quit *) [CHANNEL (Q)]
  AT %QX1.6: BOOL; (* 32BitCounter/Counter0 Enable *) [CHANNEL (Q)]
  AT %QX1.7: BOOL; (* Counter1 Enable *) [CHANNEL (Q)]
  EMPTY-SLOT[SLOT]
  EMPTY-SLOT[SLOT]
```

Settings | Other Parameters

Settings
Start Behaviour: WARMSTART

RS232/TCP--->CAN Routingsettings
Enable ☐

EASYNet-Settings
Enable ☐

Interrupt Sources/Counter
Input I0.4: none, none
Input I0.5: none, none

Configure Counter/Encode

Update Operating System
Start..

All options only available with operating system version >= V1.03.04!

Figure 63:PLC configuration

The processing unit transfers states and events to the virtual input.
The required inputs and outputs for the incremental encoder function are shown in Figure 69 or on Page 91, the necessary inputs and outputs for the counter functions can be found on Page 94.

In order to expand the local inputs and outputs simple 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.

10.1.1 Local digital inputs/outputs

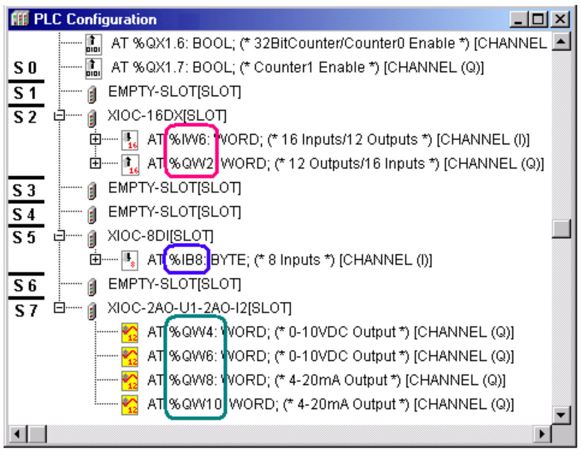
Each physical change of the modules on the slots of the rack (slot exchange or replacement for another function) is detected by the CPU since the input/output offset is changed so that access errors are caused when assigning the input and output parameters. If you have reserved free slots in the configuration for later upgrades, and if these slots are later assigned, this will also cause an imparity and change of the input/output offset between the configuration and the program.

CAUTION

- 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.

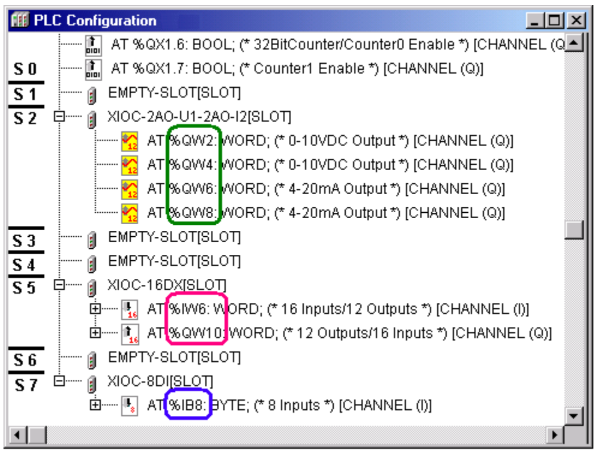
A difference between the configuration and the physical existence/non-existence of signal modules is entered as a "Fault event" in the buffered memory range. The getterrorlist browser command issues this fault as a "General IO access error". A unique slot assignment is not possible here.

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



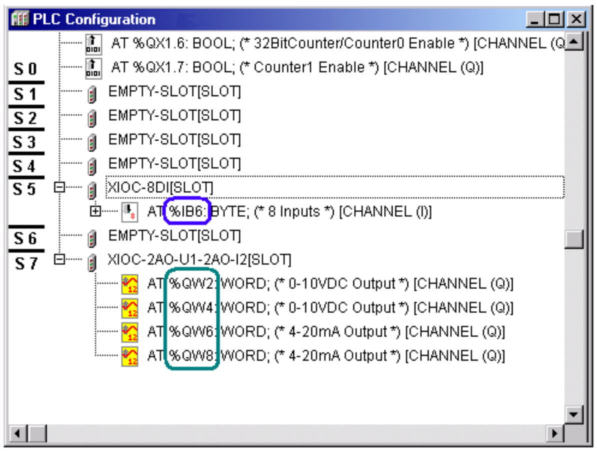
```
PLC Configuration
       AT %QX1.6: BOOL; (* 32BitCounter/Counter0 Enable *) [CHANNEL
S 0    AT %QX1.7: BOOL; (* Counter1 Enable *) [CHANNEL (Q)]
S 1    EMPTY-SLOT[SLOT]
S 2    XIOC-16DX[SLOT]
         AT %IW6: WORD; (* 16 Inputs/12 Outputs *) [CHANNEL (I)]
         AT %QW2: WORD; (* 12 Outputs/16 Inputs *) [CHANNEL (Q)]
S 3    EMPTY-SLOT[SLOT]
S 4    EMPTY-SLOT[SLOT]
S 5    XIOC-8DI[SLOT]
         AT %IB8: BYTE; (* 8 Inputs *) [CHANNEL (I)]
S 6    EMPTY-SLOT[SLOT]
S 7    XIOC-2AO-U1-2AO-I2[SLOT]
         AT %QW4: WORD; (* 0-10VDC Output *) [CHANNEL (Q)]
         AT %QW6: WORD; (* 0-10VDC Output *) [CHANNEL (Q)]
         AT %QW8: WORD; (* 4-20mA Output *) [CHANNEL (Q)]
         AT %QW10: WORD; (* 4-20mA Output *) [CHANNEL (Q)]
```

Figure 64:Current configuration
S0, S1, ..., S7 = slot number on rack



```
PLC Configuration
       AT %QX1.6: BOOL; (* 32BitCounter/Counter0 Enable *) [CHANNEL (Q
S 0    AT %QX1.7: BOOL; (* Counter1 Enable *) [CHANNEL (Q)]
S 1    EMPTY-SLOT[SLOT]
S 2    XIOC-2AO-U1-2AO-I2[SLOT]
         AT %QW2: WORD; (* 0-10VDC Output *) [CHANNEL (Q)]
         AT %QW4: WORD; (* 0-10VDC Output *) [CHANNEL (Q)]
         AT %QW6: WORD; (* 4-20mA Output *) [CHANNEL (Q)]
         AT %QW8: WORD; (* 4-20mA Output *) [CHANNEL (Q)]
S 3    EMPTY-SLOT[SLOT]
S 4    EMPTY-SLOT[SLOT]
S 5    XIOC-16DX[SLOT]
         AT %IW6: WORD; (* 16 Inputs/12 Outputs *) [CHANNEL (I)]
         AT %QW10: WORD; (* 12 Outputs/16 Inputs *) [CHANNEL (Q)]
S 6    EMPTY-SLOT[SLOT]
S 7    XIOC-8DI[SLOT]
         AT %IB8: BYTE; (* 8 Inputs *) [CHANNEL (I)]
```

Figure 65:Configuration change by changing the modules
(S0, S1, …, S7 = slot number of the rack)



```
AT %QX1.6: BOOL; (* 32BitCounter/Counter0 Enable *) [CHANNEL (Q
AT %QX1.7: BOOL; (* Counter1 Enable *) [CHANNEL (Q)]
EMPTY-SLOT[SLOT]
EMPTY-SLOT[SLOT]
EMPTY-SLOT[SLOT]
EMPTY-SLOT[SLOT]
XIOC-8DI[SLOT]
    AT %IB6: BYTE; (* 8 Inputs *) [CHANNEL (I)]
EMPTY-SLOT[SLOT]
XIOC-2AO-U1-2AO-I2[SLOT]
    AT %QW2: WORD; (* 0-10VDC Output *) [CHANNEL (Q)]
    AT %QW4: WORD; (* 0-10VDC Output *) [CHANNEL (Q)]
    AT %QW6: WORD; (* 4-20mA Output *) [CHANNEL (Q)]
    AT %QW8: WORD; (* 4-20mA Output *) [CHANNEL (Q)]
```

Figure 66:Configuration change through removal of the module
(S0, S1, ..., S7 = slot number on rack)

Figures 64 to 66 indicate the changes to the input/output parameters of the signal modules in dependence on the slots and are compiled in Table 17.

Table 17: Input/output parameters with a change of configuration

Figure	Slot	Module type	Input parameter	Output parameter
64	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**
65	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**	–
66	2	Slot not used	–	–
	5	XIOC-8DI	%IB **6**	–
	7	XIOC-2AO-U1-2AO-I2	–	%QW **2** %QW **4** %QW **6** %QW **8**

10.2 Inputs/outputs for additional functions

10.2.1 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".

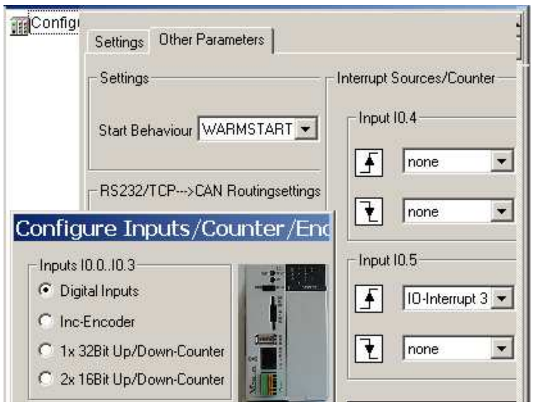


Figure 67:Incremental encoder preselection

▶ Select "Incremental encoder". The window changes its appearance:

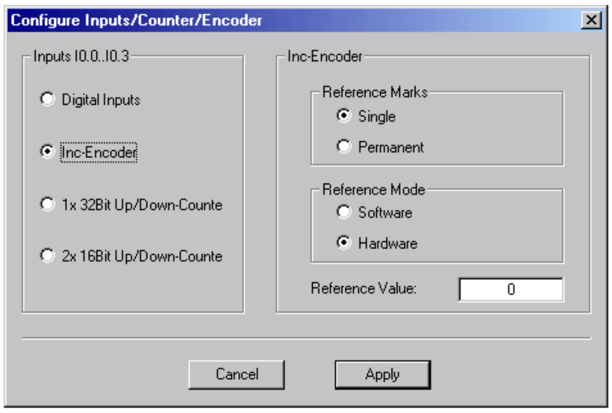


Figure 68:Incremental encoder parametric programming

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

10.2.2 Functionality of the inputs/outputs

If you have selected the "Incremental encoder" add-on 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 and outputs for the incremental encoder can be seen in the following illustrations.

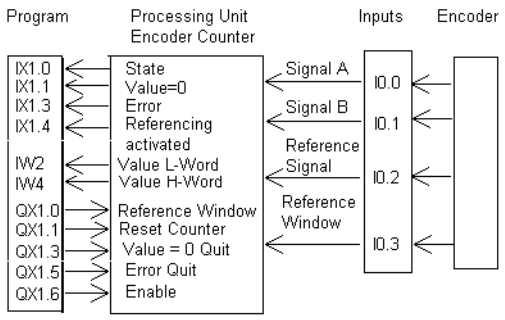


Figure 69:Inputs/outputs for incremental encoders

Referencing

In many positioning controllers, the reference point is approached at the start. The referencing operation can be controlled via the hardware (reference window signal of the encoder on I0.3) or via the software (QX1.0). You can make the selection in the PLC configuration. If one of the two signals is High, this is indicated at input IX1.4. If a pulse is generated in this state at I0.2 (reference signal of the encoder), the counter status is set to the reference value you have stated in the PLC configuration.

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

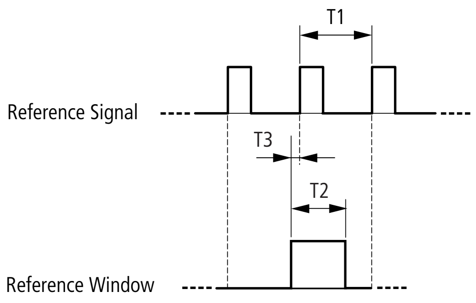


Figure 70:Relationship between reference signal and reference window

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 depend on the frequency of the reference pulse and must be determined for each application by trial and error.

10.2.3 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 referencing1)
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 crossing, H = zero crossing 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 referencing2)
AT %QX1.1: BOOL; (*Reset Counter0*) [CHANNEL(Q)]	Reset to reference value
AT %QX1.2: BOOL; (*Reset Counter1*) [CHANNEL(Q)]	
AT %QX1.3: BOOL; (*N0 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.

10.2.4 Counter

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

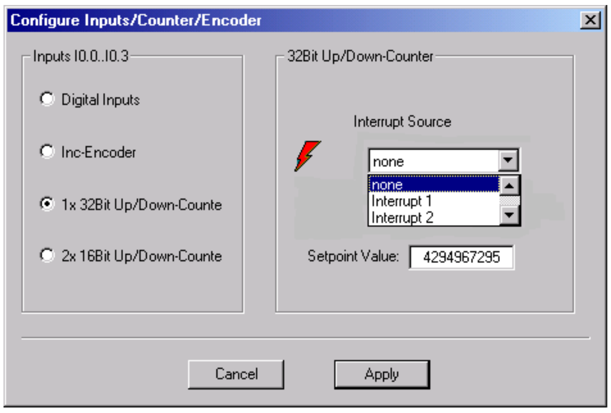
- 1 × 32 Bit up/down counter or
- 2 × 16 Bit up/down counter.

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.


Configure Inputs/Counter/Encoder

Inputs I0.0..I0.3
Digital Inputs
Inc-Encoder
1x 32Bit Up/Down-Counte [✓]
2x 16Bit Up/Down-Counte

32Bit Up/Down-Counter
Interrupt Source: none
none
Interrupt 1
Interrupt 2
Setpoint Value: 4294967295

Figure 71:Parameterization of 1 × 32 Bit counter input

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

See also:
- Interrupt processing → Page 95

10.2.5 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 crossing, H = zero crossing
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; (*N0 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)]	

10.2.6 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 crossing, H = zero crossing
AT %IX1.2: BOOL; (*N1*) [CHANNEL(I)]	L = no zero crossing, H = zero crossing
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 zero counter 0
AT %QX1.2: BOOL; (*Reset Counter1*) [CHANNEL(Q)]	Reset to zero counter 1
AT %QX1.3: BOOL; (*N0 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

10.3 Interrupt processing

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

CAUTION
The execution of the interrupt POU is **not** time monitored. Inadvertently programmed endless loops cant be exited.

A maximum of six interrupt sources (IO Interrupt1, …, IO Interrupt6) are supported, which differentiate only by the number at the end of the name.

Interrupt generators:

- 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.

NOTICE
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 recognized and executed but without running a POU.

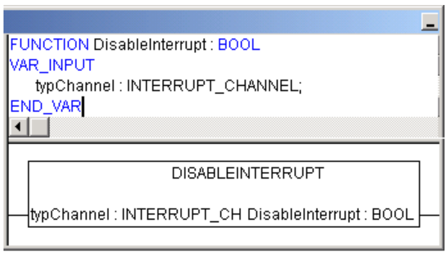
Frequent occurrence of an interrupt during program execution can cause the programmed task time to time-out and result in a RESET being initiated by the Watchdog.

User interrupts can be inhibited and re-enabled from the program. The functions DisableInterrup and EnableInterrupt are provided for this purpose. A call parameter in the CoDeSys software determines if an individual interrupt or all interrupts are enabled or inhibited. Enabling of an inhibited interrupt must be performed with the same parameter used to inhibit 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 CoDeSys.

10.3.0.1 DisableInterrupt

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

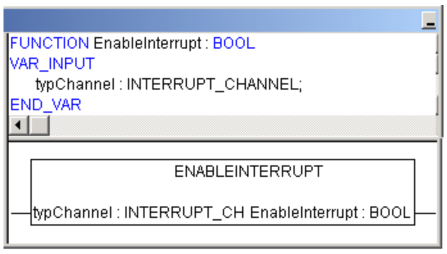


```
FUNCTION DisableInterrupt : BOOL
VAR_INPUT
    typChannel : INTERRUPT_CHANNEL;
END_VAR
```

Figure 72:"DisableInterrupt" function

10.3.1 EnableInterrupt

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



```
FUNCTION EnableInterrupt : BOOL
VAR_INPUT
    typChannel : INTERRUPT_CHANNEL;
END_VAR
```

Figure 73:"EnableInterrupt" function

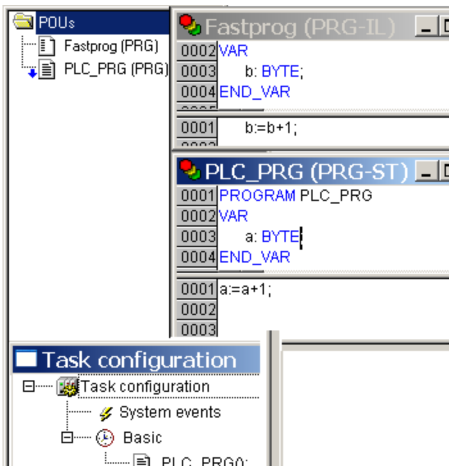
10.3.2 Parameter definition

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

10.3.3 Example for interrupt processing

A “Basic” task contains a POU “PLC_PRG”. A further POU “Fastprog” should be processed if an L $\rightarrow$ H rising edge on the input I0.5 generates an interrupt.

▶ Create the POUs “PLC_PRG” and “Fastprog” as shown in Figure 74.

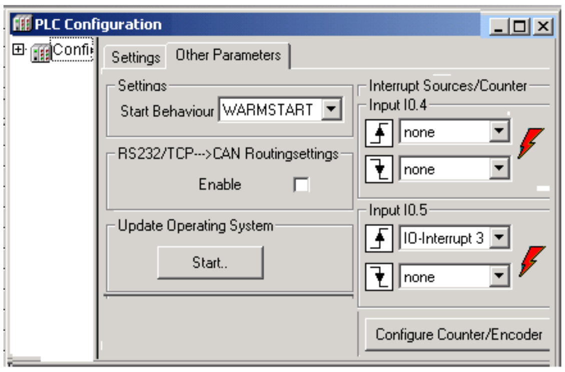


```
VAR
    b: BYTE;
END_VAR

    b:=b+1;
```

```
PROGRAM PLC_PRG
VAR
    a: BYTE
END_VAR

a:=a+1;
```

Figure 74: PLC and Task configuration

▶ Changeover to the PLC configuration and assign input I0.5 (L $\rightarrow$ H edge) e.g. the interrupt source “IO-Interupt3” from the drop-down menu.



Settings
Start Behaviour: WARMSTART

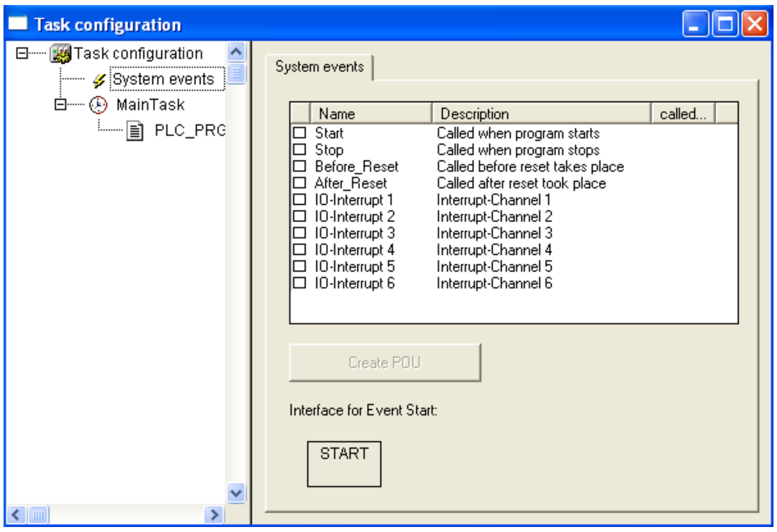
RS232/TCP--->CAN Routingsettings
Enable: ☐

Update Operating System

Interrupt Sources/Counter
Input I0.4: rising edge: none; falling edge: none
Input I0.5: rising edge: IO-Interrupt 3; falling edge: none

Figure 75: Allocation of I0.5 $\rightarrow$ interrupt source

▶ Change over to the Task configuration and open the “System events” folder.



Name	Description	called...
☐ Start	Called when program starts	
☐ Stop	Called when program stops	
☐ Before_Reset	Called before reset takes place	
☐ After_Reset	Called after reset took place	
☐ IO-Interrupt 1	Interrupt-Channel 1	
☐ IO-Interrupt 2	Interrupt-Channel 2	
☐ IO-Interrupt 3	Interrupt-Channel 3	
☐ IO-Interrupt 4	Interrupt-Channel 4	
☐ IO-Interrupt 5	Interrupt-Channel 5	
☐ IO-Interrupt 6	Interrupt-Channel 6	

Interface for Event Start:

Figure 76:System events

- ▶ Enable IO Interrupt 3 by clicking in the check box on the left beside the name “IO Interrupt 3”. 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.

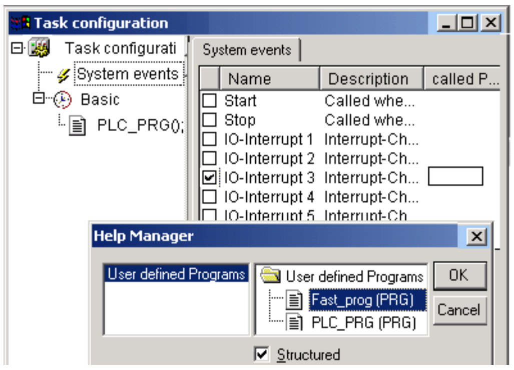


Figure 77: Allocation of Interrupt source $\longrightarrow$ POU

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 I0.5.

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

The libraries are located in the following folders:

- **Lib_Common** for all PLCs
- **Lib_CPU100** and **Lib_CPU200** for PLCs XC100 and XC200
- **Lib_XN_PLC_CANopen** for PLC XN-PLC

11.1 Using libraries

When you open a project, both libraries Standard.lib and SYSLIBCALLBACK.lib are copied in to 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
MN05003004Z-EN (previously called AWB 2724-1453)	XC100_Util. lib
MN05003001Z-EN (previously called AWB 2724-1491)	XC200_Util. lib
Manual 2724-1566	XN_PLC_Util. lib
Online help or PDF files	SysLib… . lib
MN05010002Z-EN (previously called 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
MN05010001Z-EN (previously called AWB 2786-1554)	CANUser.lib CANUser_Master.lib

11.1.1 Installing additional system libraries

You can install libraries manually as follows:

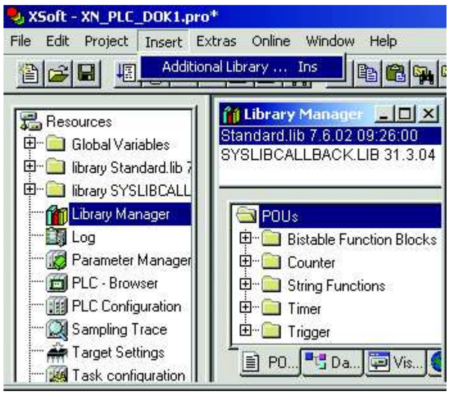


Figure 78:Libraries, installing manually

- ▶ In your project, click the “Resources” tab in the object organizer.
- ▶ Double-click the “Library Manager” element.
- ▶ Click ‹Insert → Additional Library… Ins›.

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

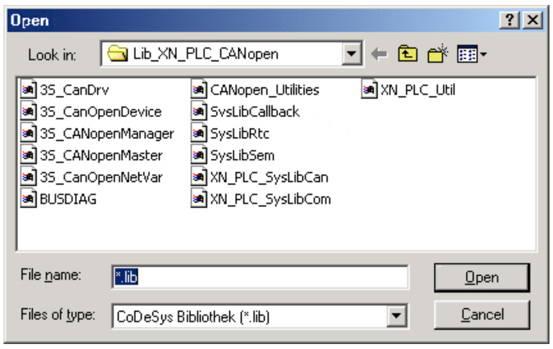


Figure 79:Selecting a library

- ▶ Select the library to install and click Open.

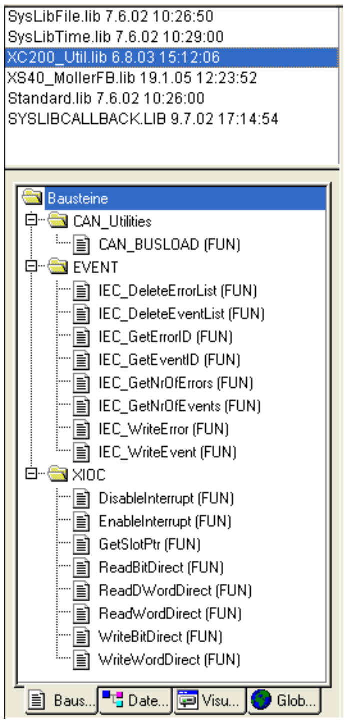
The library now appears in the Library Manager.

11.2 XC200 specific functions

The XC200 specific functions are contained in the XC200_UTIL.lib library. From operating system version V01.03.xx of the XC-CPU201 the XC200_Util2.lib library with additional functions has been introduced. The additional functions are described from Page 106.

The functions of the XC200_Util.lib library are divided into the following groups:

- CAN functions (CAN_Utilities)
- Event functions (EVENT)
- XIOC functions (XIOC)



SysLibFile.lib 7.6.02 10:26:50
SysLibTime.lib 7.6.02 10:29:00
XC200_Util.lib 6.8.03 15:12:06
XS40_MollerFB.lib 19.1.05 12:23:52
Standard.lib 7.6.02 10:26:00
SYSLIBCALLBACK.LIB 9.7.02 17:14:54

- Bausteine
  - CAN_Utilities
    - CAN_BUSLOAD (FUN)
  - EVENT
    - IEC_DeleteErrorList (FUN)
    - IEC_DeleteEventList (FUN)
    - IEC_GetErrorID (FUN)
    - IEC_GetEventID (FUN)
    - IEC_GetNrOfErrors (FUN)
    - IEC_GetNrOfEvents (FUN)
    - IEC_WriteError (FUN)
    - IEC_WriteEvent (FUN)
  - XIOC
    - DisableInterrupt (FUN)
    - EnableInterrupt (FUN)
    - GetSlotPtr (FUN)
    - ReadBitDirect (FUN)
    - ReadDWordDirect (FUN)
    - ReadWordDirect (FUN)
    - WriteBitDirect (FUN)
    - WriteWordDirect (FUN)

Figure 80:XC200 specific functions of the XC200-Util.lib library

11.2.1 CAN_Utilities

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

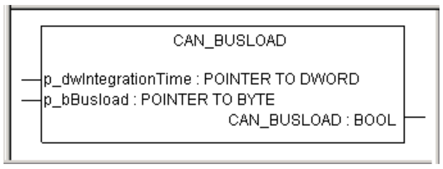

CAN_BUSLOAD
p_dwIntegrationTime : POINTER TO DWORD
p_bBusload : POINTER TO BYTE
CAN_BUSLOAD : BOOL

Figure 81:"CAN_BUSLOAD" function

This function can be called cyclically in a user program. If a read cycle has been completed successfully, the function returns TRUE and writes the determined integration time and the bus utilization values to the passed addresses.
If the bus load calculation is not yet completed or the CAN controller has not yet been initialized, the function returns FALSE. Each read cycle has a duration of 500 ms.

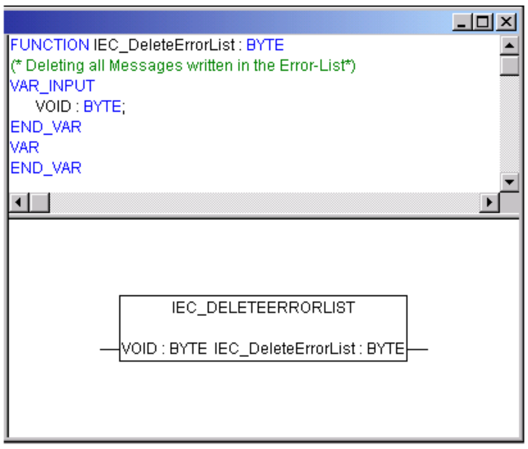
See also: Display the loading of the CAN bus (canload) → Page 131

11.2.2 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.

11.2.2.1 IEC_DeleteErrorList

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

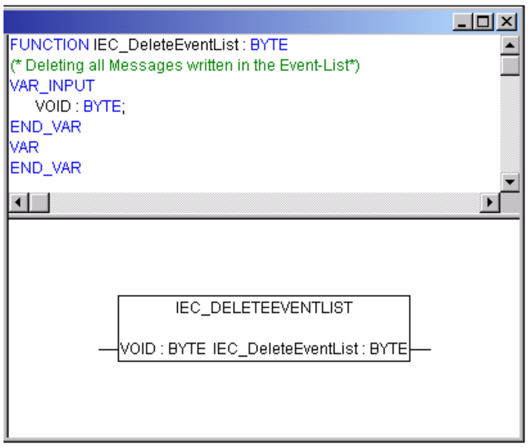


```
FUNCTION IEC_DeleteErrorList : BYTE
(* Deleting all Messages written in the Error-List*)
VAR_INPUT
    VOID : BYTE;
END_VAR
VAR
END_VAR
```

Figure 82:"IEC_DeleteErrorList" with declaration section function

11.2.2.2 IEC_DeleteEventList

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

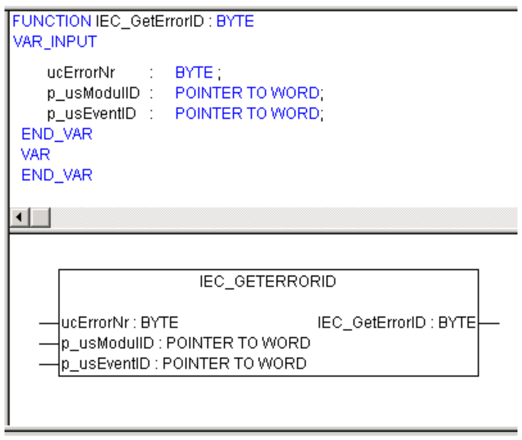


```
FUNCTION IEC_DeleteEventList : BYTE
(* Deleting all Messages written in the Event-List*)
VAR_INPUT
    VOID : BYTE;
END_VAR
VAR
END_VAR
```

Figure 83:"IEC_DeleteEventList" with declaration section function

11.2.2.3 IEC_GetErrorID

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



```
FUNCTION IEC_GetErrorID : BYTE
VAR_INPUT
    ucErrorNr    :   BYTE ;
    p_usModulID  :   POINTER TO WORD;
    p_usEventID  :   POINTER TO WORD;
 END_VAR
 VAR
 END_VAR
```

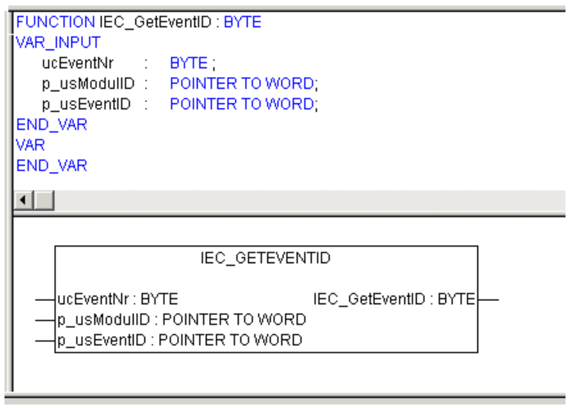
```
                 IEC_GETERRORID
--ucErrorNr : BYTE                 IEC_GetErrorID : BYTE--
--p_usModulID : POINTER TO WORD
--p_usEventID : POINTER TO WORD
```

Figure 84:"IEC_GetErrorID" with declaration section function

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

11.2.2.4 IEC_GetEventID

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



```
FUNCTION IEC_GetEventID : BYTE
VAR_INPUT
   ucEventNr    :   BYTE ;
   p_usModulID  :   POINTER TO WORD;
   p_usEventID  :   POINTER TO WORD;
END_VAR
VAR
END_VAR
```

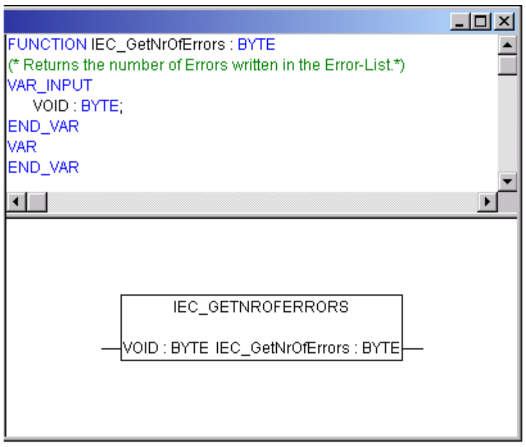
```
                 IEC_GETEVENTID
--ucEventNr : BYTE                 IEC_GetEventID : BYTE--
--p_usModulID : POINTER TO WORD
--p_usEventID : POINTER TO WORD
```

Figure 85:"IEC_GetEventID" with declaration section function

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

11.2.2.5 IEC_GetNrOfErrors

This function returns the number of entered error messages.



```
FUNCTION IEC_GetNrOfErrors : BYTE
(* Returns the number of Errors written in the Error-List.*)
VAR_INPUT
    VOID : BYTE;
END_VAR
VAR
END_VAR
```

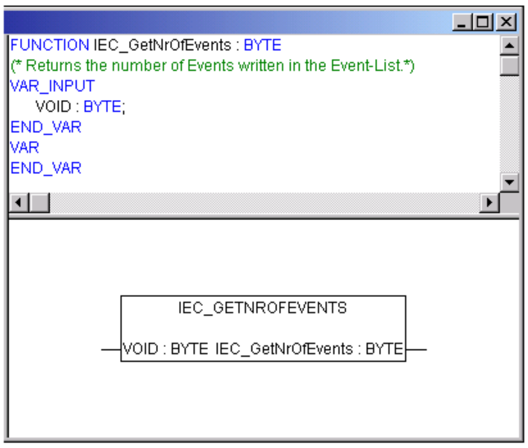
IEC_GETNROFERRORS

VOID : BYTE IEC_GetNrOfErrors : BYTE

Figure 86:"IEC_GetNrOfErrors" function

11.2.2.6 IEC_GetNrOfEvents

This function returns the number of entered event messages.



```
FUNCTION IEC_GetNrOfEvents : BYTE
(* Returns the number of Events written in the Event-List.*)
VAR_INPUT
    VOID : BYTE;
END_VAR
VAR
END_VAR
```

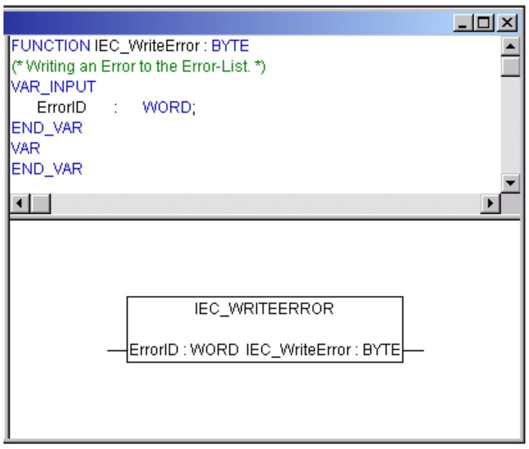
IEC_GETNROFEVENTS

VOID : BYTE IEC_GetNrOfEvents : BYTE

Figure 87:"IEC_GetNrOfEvents" function

11.2.2.7 IEC_WriteError

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

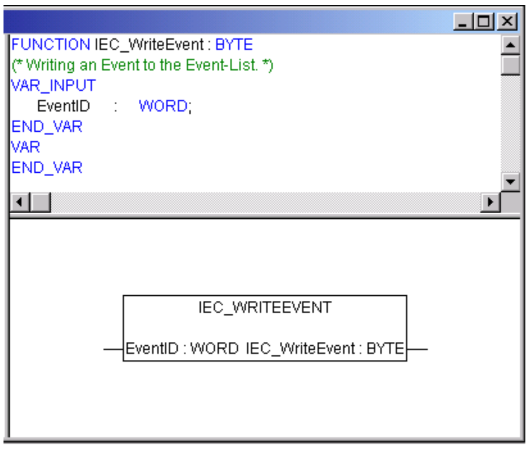


```
FUNCTION IEC_WriteError : BYTE
(* Writing an Error to the Error-List. *)
VAR_INPUT
    ErrorID     :    WORD;
END_VAR
VAR
END_VAR
```

Figure 88:"IEC_WriteError" function

11.2.2.8 IEC_WriteEvent

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



```
FUNCTION IEC_WriteEvent : BYTE
(* Writing an Event to the Event-List. *)
VAR_INPUT
    EventID     :    WORD;
END_VAR
VAR
END_VAR
```

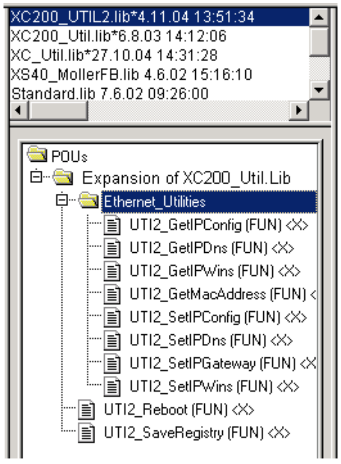
Figure 89:"IEC_WriteEvent" function

11.2.3 XIOC functions

The XIOC functions include functions for processing interrupts (→ Page 95) and for programming of the direct peripheral access (→ Page 58).

11.2.4 Additional functions of the XC200_Util2.lib library for the XC-CPU201

The functions of the XC200_Util2.lib library can be seen in the following overview:



XC200_UTIL2.lib*4.11.04 13:51:34
XC200_Util.lib*6.8.03 14:12:06
XC_Util.lib*27.10.04 14:31:28
XS40_MollerFB.lib 4.6.02 15:16:10
Standard.lib 7.6.02 09:26:00

POUs
Expansion of XC200_Util.Lib
Ethernet_Utilities
UTI2_GetIPConfig (FUN) <X>
UTI2_GetIPDns (FUN) <X>
UTI2_GetIPWins (FUN) <X>
UTI2_GetMacAddress (FUN) <
UTI2_SetIPConfig (FUN) <X>
UTI2_SetIPDns (FUN) <X>
UTI2_SetIPGateway (FUN) <X
UTI2_SetIPWins (FUN) <X>
UTI2_Reboot (FUN) <X>
UTI2_SaveRegistry (FUN) <X>

Figure 90:Overview of the XC200_Util2.lib library for the XC-CPU201

11.2.4.1 Ethernet_Utilities

UTI2_GetIPConfig

Issue of the IP address, subnet mask address and IP gateway address.

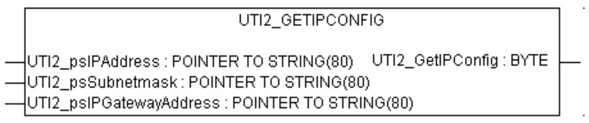

UTI2_GETIPCONFIG
UTI2_psIPAddress : POINTER TO STRING(80)
UTI2_psSubnetmask : POINTER TO STRING(80)
UTI2_psIPGatewayAddress : POINTER TO STRING(80)
UTI2_GetIPConfig : BYTE

Figure 91:UTI2_GetIPConfig

Table 18: Input variables for UTI2_GetIPConfig

Input variables	Meaning
UTI2_psIPAddress	Pointer to a string in which the read IP address is written.
UTI2_psSubnetmask	Pointer to a string in which the read address of the subnet mask is written.
UTI2_psIPGatewayAddress	Pointer to a string in which the read address of the standard gateway is written.

Table 19: Return values for UTI2_GetIPConfig

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

UTI2_GetIPDns

Output of the IP address of the DNS server currently entered in the Registry

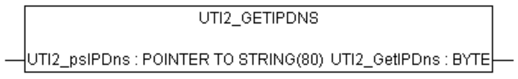


Figure 92:UTI2_GetIPDns

Table 20: Input variables for UTI2_GetIPDns

Input variables	Meaning
UTI2_psIPDns	Pointer to a string in which the read IP address is written.

Table 21: Return values for UTI2_GetIPDns

ReturnVal	Meaning
1	Read successful
< 0	Read failed
-4	No valid pointer transferred.

UTI2_GetIPWins

Output of the IP address of the WINS server currently entered in the Registry

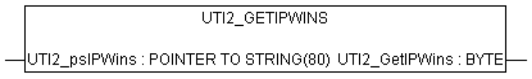


Figure 93:UTI2_GetIPWins

Table 22: Input variables for UTI2_GetIPWins

Input variables	Meaning
UTI2_psIPWins	Pointer to a string in which the read IP address is written.

Table 23: Return values for UTI2_GetIPWins

ReturnVal	Meaning
1	Read successful
< 0	Read failed
-4	No valid pointer transferred.

UTI2_GetMacAddress

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

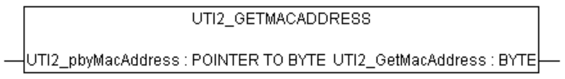

UTI2_GETMACADDRESS
UTI2_pbyMacAddress : POINTER TO BYTE
UTI2_GetMacAddress : BYTE

Figure 94:UTI2_GetMacAddress

Table 24: Input variables for UTI2_GetMacAddress

Input variables	Meaning
UTI2_pbyMacAddress	Pointer to an array of 5 byte values, in which the read MAC address is entered.

Table 25: Return values for UTI2_GetMacAddress

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

UTI2_SetIPConfig

Set IP and subnet mask address

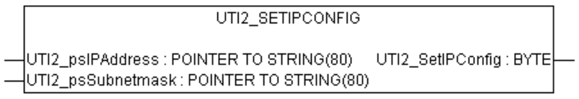

UTI2_SETIPCONFIG
UTI2_psIPAddress : POINTER TO STRING(80)
UTI2_SetIPConfig : BYTE
UTI2_psSubnetmask : POINTER TO STRING(80)

Figure 95:UTI2_SetIPConfig

NOTICE
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 26: Input variables for UTI2_SetIPConfig

Input variables	Meaning
UTI2_psIPAddress	Pointer to a string variable which contains the IP address to be written.
UTI2_psSubnetmask	Pointer to a string variable, which contains the value to be entered from the subnet mask.

Table 27: Return values for UTI2_SetIPConfig

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

UTI2_SetIPDNS

Setting of the IP address of a DNS server in the registry
(if necessary must be saved with UTI2_SaveRegistry)

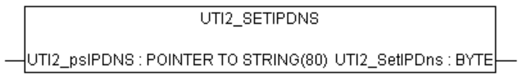


Figure 96:UTI2_SetIPDNS

Table 28: Input variables for UTI2_SetIPDns

Input variables	Meaning
UTI2_psIPDns	Pointer to a string variable which contains the IP address to be written.

Table 29: Return values for UTI2_SetIPDns

ReturnVal	Meaning
1	Write successful
< 0	Write failed
-4	No valid pointer transferred.

UTI2_SetIPGateway

Setting IPGateway address

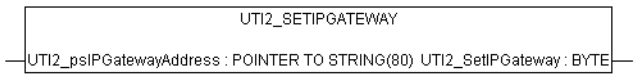


Figure 97:UTI2_SetIPGateway

NOTICE

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 30: Input variables for UTI2_SetIPGateway

Input variables	Meaning
UTI2_psIPGatewayAddress	Pointer to a string variable, which contains the value to be entered from the gateway address.

Table 31: Return values for UTI2_SetIPGateway

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

UTI2_SetIPWins

Setting of the IP address of a WINS server in the registry
(if necessary must be saved with UTI2_SaveRegistry)

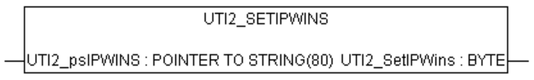


Figure 98:UTI2_SetIPWins

Table 32: Input variables for UTI2_SetIPWins

Input variables	Meaning
UTI2_psIPGatewayAddress	Pointer to a string variable which contains the IP address to be written.

Table 33: Return values for UTI2_SetIPWins

ReturnVal	Meaning
1	Write successful
< 0	Write failed
-4	No valid pointer transferred.

UTI2_Reboot

Restart with registry save

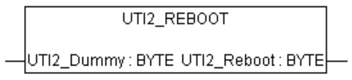


Figure 99:UTI2_Reboot

Table 34: Input variables for UTI2_Reboot

Input variables	Meaning
UTI2_Dummy	A dummy byte which is not evaluated in the function.

Table 35: Return values for UTI2_Reboot

ReturnVal	Meaning
1	Dummy return value / Reboot executed afterwards.

UTI2_SaveRegistry Saving of the registry

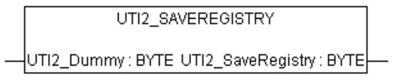


Figure 100:UTI2_SaveRegistry

Table 36: Input variables for UTI2_SaveRegistry

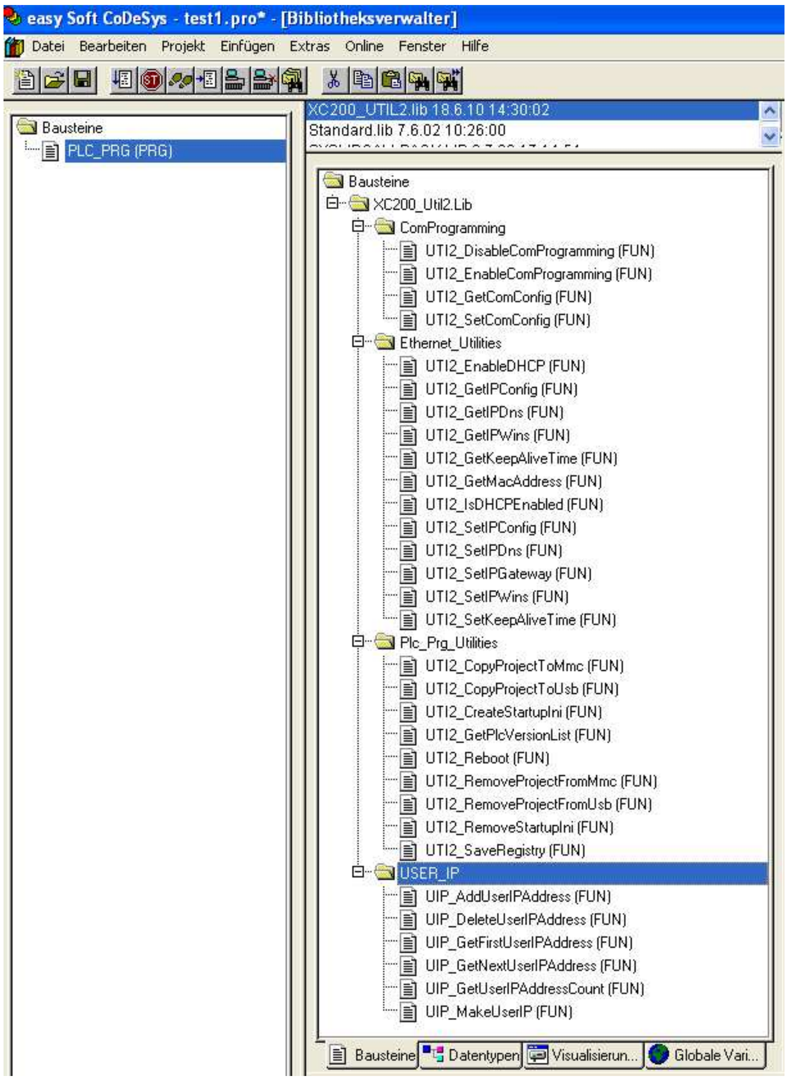
Input variables	Meaning
UTI2_Dummy	A dummy byte which is not evaluated in the function.

Table 37: Return values for UTI2_SaveRegistry

ReturnVal	Meaning
1	Function completed successfully
-1	Errors

11.2.5 Additional functions of the XC200_Util2.lib library for the XC-CPU202

The functions of the XC200_Util2.lib library can be seen in the following overview:


easy Soft CoDeSys - test1.pro* - [Bibliotheksverwalter]
Datei Bearbeiten Projekt Einfügen Extras Online Fenster Hilfe
Bausteine
PLC_PRG (PRG)
Bausteine Datentypen Visualisierun... Globale Vari...

XC200_UTIL2.lib 18.6.10 14:30:02
Standard.lib 7.6.02 10:26:00

- Bausteine
  - XC200_Util2.Lib
    - ComProgramming
      - UTI2_DisableComProgramming (FUN)
      - UTI2_EnableComProgramming (FUN)
      - UTI2_GetComConfig (FUN)
      - UTI2_SetComConfig (FUN)
    - Ethernet_Utilities
      - UTI2_EnableDHCP (FUN)
      - UTI2_GetIPConfig (FUN)
      - UTI2_GetIPDns (FUN)
      - UTI2_GetIPWins (FUN)
      - UTI2_GetKeepAliveTime (FUN)
      - UTI2_GetMacAddress (FUN)
      - UTI2_IsDHCPEnabled (FUN)
      - UTI2_SetIPConfig (FUN)
      - UTI2_SetIPDns (FUN)
      - UTI2_SetIPGateway (FUN)
      - UTI2_SetIPWins (FUN)
      - UTI2_SetKeepAliveTime (FUN)
    - Plc_Prg_Utilities
      - UTI2_CopyProjectToMmc (FUN)
      - UTI2_CopyProjectToUsb (FUN)
      - UTI2_CreateStartupIni (FUN)
      - UTI2_GetPlcVersionList (FUN)
      - UTI2_Reboot (FUN)
      - UTI2_RemoveProjectFromMmc (FUN)
      - UTI2_RemoveProjectFromUsb (FUN)
      - UTI2_RemoveStartupIni (FUN)
      - UTI2_SaveRegistry (FUN)
    - USER_IP
      - UIP_AddUserIPAddress (FUN)
      - UIP_DeleteUserIPAddress (FUN)
      - UIP_GetFirstUserIPAddress (FUN)
      - UIP_GetNextUserIPAddress (FUN)
      - UIP_GetUserIPAddressCount (FUN)
      - UIP_MakeUserIP (FUN)

Figure 101:Overview of the XC200_Util2.lib for the XC-CPU202

11.2.5.1 ComProgramming

UTI2_DisableComProgramming

Deactivates the serial interface as a programming interface (local COM2).

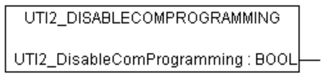


Figure 102:UTI2_DisableComProgramming

Table 38: Input variables for UTI2_DisableComProgramming

Input variables	Meaning
None	–

Table 39: Return values for UTI2_DisableComProgramming

ReturnVal	Meaning
TRUE	Function completed successfully
FALSE	Errors

UTI2_EnableComProgramming

Activates the serial interface as a programming interface (local COM2).

UTI2_ENABLECOMPROGRAMMING
UTI2_EnableComProgramming : BOOL

Figure 103:UTI2_EnableComProgramming

Table 40: Input variables for UTI2_EnableComProgramming

Input variables	Meaning
None	–

Table 41: Return variables for UTI2_EnableComProgramming

ReturnVal	Meaning
TRUE	Function completed successfully
FALSE	Errors

UTI2_GetComConfig

Shows the interface parameters of the local COM interface.

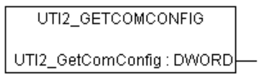


Figure 104:UTI2_GetComConfig

Table 42: Input variables for UTI2_GetComConfig

Input variables	Meaning
None	–

Table 43: Return values for UTI2_GetComConfig

ReturnVal	Meaning
DWORDCOM	Baud rate of the local COM port (4800, 9600, 19200, 38400, 57600, 115200)

UTI2_SetComConfig

Setting of the baud rate of the local serial interface

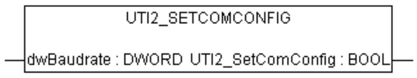


Figure 105:UTI2_SetComConfig

Table 44: Input variables for UTI2_SetComConfig

Input variables	Meaning
dwBaudrate	(4800, 9600, 19200, 38400, 57600, 115200)

Table 45: Return values for UTI2_SetComConfig

ReturnVal	Meaning
TRUE	
FALSE	

11.2.5.2 Ethernet_Utilities

UTI2_EnableDHCP

Activates the DHCP function of the PLC.

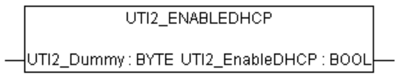


Figure 106:UTI2_EnableDHCP

Table 46: Input variables for UTI2_EnableDHCP

Input variables	Meaning
Dummy BYTE	

Table 47: Return values for UTI2_EnableDHCP

ReturnVal	Meaning
TRUE	always true

To work with DHCP you must reboot the UTI2_SaveRegistry function and the PLC. After booting the controller requests an IP address from the DHCP server.

To deactivate the DHCP function, you must call the function UTI2_SetIPConfig or the browser command setipconfig.

UTI2_GetIPConfig

Issue of the IP address, subnet mask address and IP gateway address

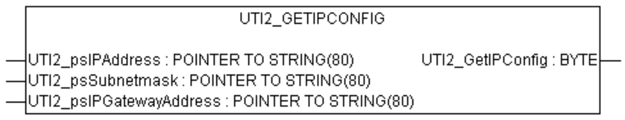

UTI2_GETIPCONFIG
UTI2_psIPAddress : POINTER TO STRING(80)
UTI2_psSubnetmask : POINTER TO STRING(80)
UTI2_psIPGatewayAddress : POINTER TO STRING(80)
UTI2_GetIPConfig : BYTE

Figure 107:UTI2_GetIPConfig

Table 48: Input variables for UTI2_GetIPConfig

Input variables	Meaning
UTI2_psIPAddress	Pointer to a string in which the read IP address is written.
UTI2_psSubnetmask	Pointer to a string in which the read address of the subnet mask is written.
UTI2_psIPGatewayAddress	Pointer to a string in which the read address of the standard gateway is written.

Table 49: Return values for UTI2_GetIPConfig

ReturnVal	Meaning
1	Read successful
< 0	Read failed
-4	No valid pointer transferred.

UTI2_GetIPDns

Output of the IP address of the DNS server currently entered in the Registry

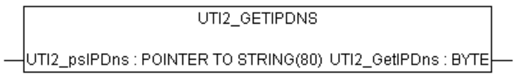


Figure 108:UTI2_GetIPDns

Table 50: Input variables for UTI2_GetIPDns

Input variables	Meaning
UTI2_psIPDns	Pointer to a string in which the read IP address is written.

Table 51: Return values for UTI2_GetIPDns

ReturnVal	Meaning
1	Read successful
< 0	Read failed
-4	No valid pointer transferred.

UTI2_GetIPWins

Output of the IP address of the WINS server currently entered in the Registry

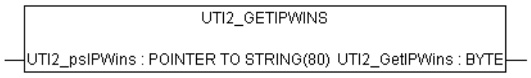


Figure 109:UTI2_GetIPWins

Table 52: Input variables for UTI2_GetIPWins

Input variables	Meaning
UTI2_psIPWins	Pointer to a string in which the read IP address is written.

Table 53: Return values for UTI2_GetIPWins

ReturnVal	Meaning
1	Read successful
< 0	Read failed
-4	No valid pointer transferred.

UTI2_GetKeepAliveTime

Output of KeepAliveTime in seconds

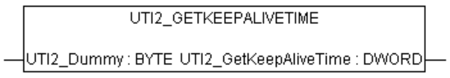


Figure 110:UTI2_GetKeepAliveTime

Table 54: Input variables for UTI2_GetKeepAliveTime

Input variables	Meaning
Dummy BYTE	Not evaluated in the function.

Table 55: Return values for UTI2_GetKeepAliveTime

ReturnVal	Meaning
KeepAliveTime	KeepAliveTime in seconds

UTI2_GetMacAddress

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

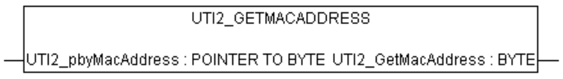

UTI2_GETMACADDRESS
UTI2_pbyMacAddress : POINTER TO BYTE
UTI2_GetMacAddress : BYTE

Figure 111:UTI2_GetMacAddress

Table 56: Input variables for UTI2_GetMacAddress

Input variables	Meaning
UTI2_pbyMacAddress	Pointer to an array of 5 byte values, in which the read MAC address is entered.

Table 57: Return values for UTI2_GetMacAddress

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

UTI2_IsDHCPEnabled

Read DHCP status

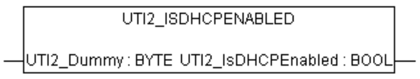


Figure 112:UTI2_IsDHCPEnabled

Table 58: Input variables for UTI2_IsDHCPEnabled

Input variables	Meaning
Dummy BYTE	Not evaluated in the function.

Table 59: Return values for UTI2_IsDHCPEnabled

ReturnVal	Meaning
TRUE	DHCP is active.
FALSE	DHCP inactive

UTI2_SetIPConfig

Set IP and subnet mask address

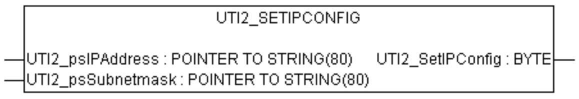

UTI2_SETIPCONFIG
UTI2_psIPAddress : POINTER TO STRING(80)
UTI2_psSubnetmask : POINTER TO STRING(80)
UTI2_SetIPConfig : BYTE

Figure 113:UTI2_SetIPConfig

Table 60: Input variables for UTI2_SetIPConfig

Input variables	Meaning
UTI2_psIPAddress	Pointer to a string variable which contains the IP address to be written.
UTI2_psSubnetmask	Pointer to a string variable, which contains the value to be entered from the subnet mask.

Table 61: Return values for UTI2_SetIPConfig

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

UTI2_SetIPDNS

Setting of the IP address of a DNS server in the registry
(if necessary must be saved with UTI2_SaveRegistry)

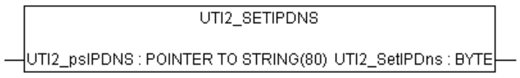


Figure 114:UTI2_SetIPDNS

Table 62: Input variables for UTI2_SetIPDns

Input variables	Meaning
UTI2_psIPDns	Pointer to a string variable which contains the IP address to be written.

Table 63: Return values for UTI2_SetIPDns

ReturnVal	Meaning
1	Write successful
< 0	Write failed
-4	No valid pointer transferred.

UTI2_SetIPGateway

Setting IPGateway address

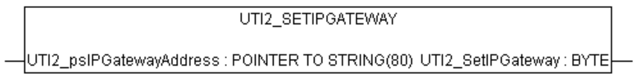


Figure 115:UTI2_SetIPGateway

Table 64: Input variables for UTI2_SetIPGateway

Input variables	Meaning
UTI2_psIPGatewayAddress	Pointer to a string variable, which contains the value to be entered from the gateway address.

Table 65: Return values for UTI2_SetIPGateway

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

UTI2_SetIPWins

Setting of the IP address of a WINS server in the registry
(if necessary must be saved with UTI2_SaveRegistry)

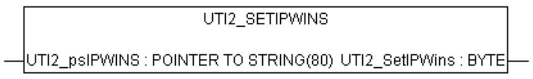


Figure 116:UTI2_SetIPWins

Table 66: Input variables for UTI2_SetIPWins

Input variables	Meaning
UTI2_psIPGatewayAddress	Pointer to a string variable which contains the IP address to be written.

Table 67: Return values for UTI2_SetIPWins

ReturnVal	Meaning
1	Write successful
< 0	Write failed
-4	No valid pointer transferred.

UTI2_SetKeepAliveTime

Sets the KeepAliveTime in seconds.

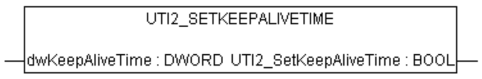


Figure 117:UTI2_SetKeepAliveTime

Table 68: Input variables for UTI2_SetKeepAliveTime

Input variables	Meaning
5 – 500	Meaningful values in seconds

Table 69: Return values for UTI2_SetKeepAliveTime

ReturnVal	Meaning
TRUE	Value valid
FALSE	Value outside of valid range

11.2.5.3 Plc_Prg_Utilities

UTI2_CopyProjectToMmc

Copies a project to MMC/SD.

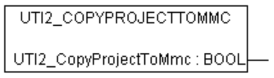


Figure 118:UTI2_CopyProjectToMmc

Table 70: Input variables for UTI2_CopyProjectToMmc

Input variables	Meaning
None	–

Table 71: Return values for UTI2_CopyProjectToMmc

ReturnVal	Meaning
TRUE	Function completed successfully.
FALSE	Errors

UTI2_CopyProjectToUsb

Copies a project to the USB stick.

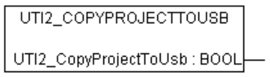


Figure 119:UTI2_CopyProjectToUsb

Table 72: Input variables for UTI2_CopyProjectToUsb

Input variables	Meaning
None	–

Table 73: Return values for UTI2_CopyProjectToUsb

ReturnVal	Meaning
TRUE	Function completed successfully
FALSE	Errors

UTI2_CreateStartupIni

Creates the Startup.ini on disk_sys and disk_mmc.

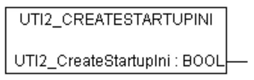


Figure 120:UTI2_CreateStartupIni

Table 74: Input variables for UTI2_CreateStartupIni

Input variables	Meaning
None	–

Table 75: Return values for UTI2_CreateStartupIni

ReturnVal	Meaning
TRUE	Function completed successfully
FALSE	Errors

UTI2_GetPlcVersionList

Display of device version list

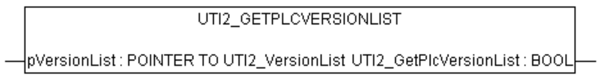

UTI2_GETPLCVERSIONLIST
pVersionList : POINTER TO UTI2_VersionList
UTI2_GetPlcVersionList : BOOL

Figure 121:UTI2_GetPlcVersionList

Table 76: Input variables for UTI2_GetPlcVersionList

Input variables	Meaning
pVersionList	Pointer to UTI2_VersionList

Table 77: Return values for UTI2_GetPlcVersionList

ReturnVal	Meaning
TRUE	Function completed successfully
FALSE	Errors

UTI2_Reboot

Restart of the PLC

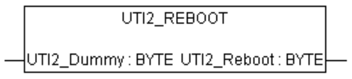


Figure 122:UTI2_Reboot

Table 78: Input variables for UTI2_Reboot

Input variables	Meaning
UTI_Dummy	Set variable to 0.

Table 79: Return values for UTI2_Reboot

ReturnVal	Meaning
1	Dummy return value / Reboot executed afterwards.

UTI2_RemoveProjectFromMmc

Removes the backup project from the MMC/SD

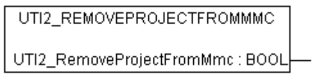


Figure 123:UTI2_RemoveProjectFromMmc

Table 80: Input variables for UTI2_RemoveProjectFromMmc

Input variables	Meaning
None	–

Table 81: Return values for UTI2_RemoveProjectFromMmc

ReturnVal	Meaning
TRUE	Function completed successfully
FALSE	Errors

UTI2_RemoveProjectFromUsb

Removing the backup project from the USB stick

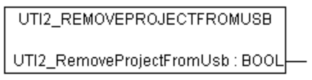


Figure 124:UTI2_RemoveProjectFromUsb

Table 82: Input variables for UTI2_RemoveProjectFromUsb

Input variables	Meaning
None	–

Table 83: Return values for UTI2_RemoveProjectFromUsb

ReturnVal	Meaning
TRUE	Function completed successfully
FALSE	Errors

UTI2_RemoveStartupIni

Removes the Startup.ini file from the disk_sys and the disk_mmc.

UTI2_REMOVESTARTUPINI
UTI2_RemoveStartupIni : BOOL

Figure 125:UTI2_RemoveStartupIni

Table 84: Input variables for UTI2_RemoveStartupIni

Input variables	Meaning
None	–

Table 85: Return values for UTI2_RemoveStartupIni

ReturnVal	Meaning
TRUE	Function completed successfully
FALSE	Errors

UTI2_SaveRegistry

Saves the changes retentively in the Registry.

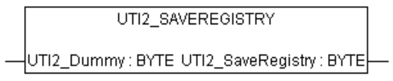


Figure 126:UTI2_SaveRegistry

Table 86: Input variables for UTI2_SaveRegistry

Input variables	Meaning
UTI2_Dummy	Set variable to 0.

Table 87: Return values for UTI2_SaveRegistry

ReturnVal	Meaning
1	Function completed successfully
-1	Errors

11.2.5.4 USER_IP

UIP_AddUserIPAddress

Adds a new IP address to the system.

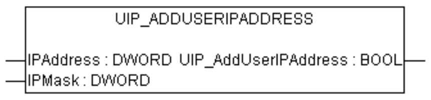


Figure 127:UIP_AddUserIPAddress

Table 88: Input variables for UIP_AddUserIPAddress

Input variables	Meaning
IPAddress	IP address to be added
IPMask	Associated IP screen to be added

Table 89: Return values for UIP_AddUserIPAddress

ReturnVal	Meaning
TRUE	IP address was successfully added.
FALSE	Table full; Address already present in the User IP table, address already present in operating system table

UIP_DeleteUserIPAddress

Deletes an IP address from the system.

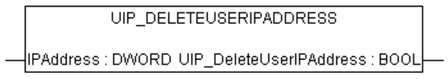


Figure 128:UIP_DeleteUserIPAddress

Table 90: Input variables for UIP_DeleteUserIPAddress

Input variables	Meaning
IPAddress	Identifies the entry to be deleted from the user IP table.

Table 91: Return values for UIP_DeleteUserIPAddress

ReturnVal	Meaning
TRUE	Associated user IP table entry was deleted.
FALSE	Associated user IP table entry was not present.

UIP_GetFirstuserIPAddress

Reads the first user IP address entered from the user IP table.

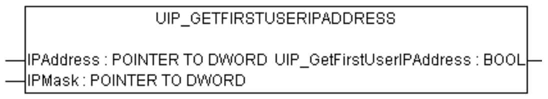

UIP_GETFIRSTUSERIPADDRESS
IPAddress : POINTER TO DWORD
UIP_GetFirstUserIPAddress : BOOL
IPMask : POINTER TO DWORD

Figure 129:UIP_GetFirstuserIPAddress

Table 92: Input variables for UIP_GetFirstuserIPAddress

Input variables	Meaning
IPAddress	Points to wildcard for the user table entry to be determined.
IPMask	Points to wildcard for the user table entry to be determined.

Table 93: Return values for UIP_GetFirstuserIPAddress

ReturnVal	Meaning
TRUE	Valid entry found.
FALSE	The user IP address table does not contain an entry.

UIP_GetNextUserIPAddress

Reads the next user IP address entered from the user IP table.

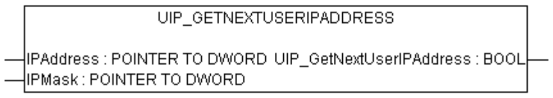

UIP_GETNEXTUSERIPADDRESS
IPAddress : POINTER TO DWORD
UIP_GetNextUserIPAddress : BOOL
IPMask : POINTER TO DWORD

Figure 130:UIP_GetNextUserIPAddress

Table 94: Input variables for UIP_GetNextUserIPAddress

Input variables	**Meaning**
IPAddress	Pointer to DWORD wildcard for the user table entry to be determined.
IPMask	Pointer to DWORD wildcard for the user table entry to be determined.

Table 95: Return values for UIP_GetNextUserIPAddress

ReturnVal	**Meaning**
TRUE	Valid entry found
FALSE	The user IP address table does not contain any other entry.

UIP_GetUserIPAddressCount

Supplies the number of the user IP entries currently present in the user IP table.

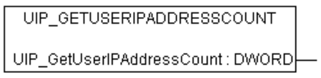


Figure 131:UIP_GetUserIPAddressCount

Table 96: Input variables for UIP_GetUserIPAddressCount

Input variables	**Meaning**
None	–

Table 97: Return values for UIP_GetUserIPAddressCount

ReturnVal	**Meaning**
0 up to maximum number (= 3)	Number of the values currently entered in the user IP address table

UIP_MakeUserIP

Converts the IP string to IP_DWORD (Bigendianness).

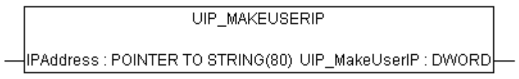


Figure 132:UIP_MakeUserIP

Table 98: Input variables for UIP_MakeUserIP

Input variables	Meaning
IPAdress	Pointer to IP address apostrophe (e.g. 192.168.119.1')

Table 99: Return values for UIP_MakeUserIP

ReturnVal	Meaning
IP address in hexadecimal representation	IP as DWORD (e.g.16#C0A8770B – corresponds to the above IP address 192.168.119.11)
0	Error in the input string or invalid string

12 Browser commands

The PLC browser is a text based PLC terminal monitor. Commands for scanning particular information from the PLC are entered in an entry line and sent to the PLC as a string. The response string is shown in a result window of the browser. This function can be used for diagnosing and debugging. The browser commands available for the XC200 target system are as follows.

Table 100:Browser commands (in alphabetical order)

Attribute ID	Description
?	Get a list of implemented commends
caninfo	Display CAN controller information
canload	Display of the loading of the CAN fieldbus
clearerrorlist	Erase error list
cleareventlist	Delete 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
enabledhcp	Activates the DHCP function of the PLC.
filecopy[1)]	Copy File
filedelete[1)]	Delete 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
getcommunicationport	Read the interface parameters for the TCP/IP communication
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 names
getversion	Display version information
isdhcpenabled	Scanning whether DHCP is activated

Attribute ID	Description
memdisk_sys	Displays the free memory at disk_sys.
pid	Output project ID
pinf	Output project information
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/SD
removestartupini	Erases the Startup.ini 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]
rtsinfo	runtime system information (version, IO drivers)
saveregistry	Accept modifications
saveretain	Save retentive data in the file [file name]
serialProgrammingOn	"Enable serial programming with CoDeSys on XC202 local COM2 port" – enables the local COM as a programming interface. Factory setting. Serial interface in programming mode
serialProgrammingOff	"Disable serial programming with CoDeSys on XC202 local COM2 port" – Disables the local COM as programming interface. Serial interface in programming mode
setcomconfig[1)]	Set the baud rate of the serial interface [setcomconfig 4800,9600,19200, 38400, 57600,115200]
setcommunicationport	Set the interface parameters for the TCP/IP communication
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
tskclear	Clear IEC task information: Cyclecount, accumulated, max. and min. cycle
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 CoDeSys software.
Enter a question mark followed by a space before the command e.g.: ? plcload in the command line of the PLC browser

12.1 Calling browser commands

▶ Activate the “Resources” tab in the CODESYS software 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 other settings to the command if necessary, e.g. baud rate withsetcomconfig, → Table 100.
▶ The command may require additional parameters.
▶ Press the Return button.

The result will be displayed.

12.2 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 or 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 that.

- 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 and settargetname must be supplemented in the command line of the PLC browser, e.g.B. with the Baud rate at setcomconfig, → Table 100. 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.

12.3 Display CPU loading (plcload)

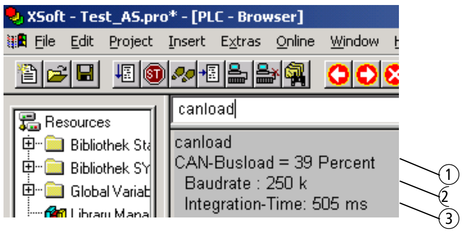
The plcload browser command provides information on the current system loading of the central processing unit.

A utilization of more than 95 percent can cause a failure of the serial and Ethernet communication and/or an impairment of the real-time response.

12.3.1 Display the loading of the CAN bus (canload)

The PLC browser command canload belongs to the “XC200_Util.lib” library. It indicates the loading of the CAN bus.

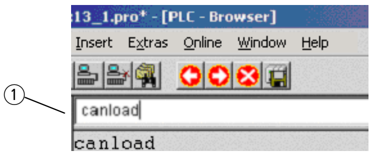
Examples for display:


XSoft - Test_AS.pro* - [PLC - Browser]
File Edit Project Insert Extras Online Window
Resources
Bibliothek Sta
Bibliothek SY
Global Variab
Library Mana
1
2
3

```
canload
canload
CAN-Busload = 39 Percent
  Baudrate : 250 k
  Integration-Time: 505 ms
```

Figure 133: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.



```
canload
canload
```

Figure 134:Loading of the CAN bus with alarm message (example 2)

① Alarm message, → Table 101

Table 101:Possible alarm messages

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

12.3.2 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 for XC-CPU201

```
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”.
On the XC-CPU202 \CONTROL is used instead of \MOELLER.

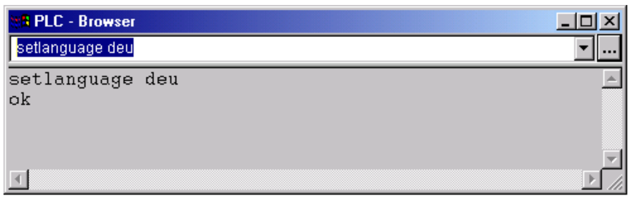
12.3.3 Error and event list after calling browser commands

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.



```
setlanguage deu
ok
```

Figure 135:Browser command "setlanguage"

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:

Modul-ID	Program Type
1	RTS (runtime system)
2	CST (Moeller specific adaption)
3	XIO (XIOC)
4	CAN
5	IEC

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

Modul-ID	Event-ID	Error Message
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

Modul-ID	Event-ID	Error Message
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	Field bus 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 normalized operand
1	338	Floating decimal point: Division by zero
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

13 Appendix

13.1 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 102:Characteristics of the Ethernet cable

	UTP1)	**STP**2)	**SSTP**3)
Transmission medium	Unshielded Twisted Pair	Shielded Twisted Pair	Shielded Twisted Pair
Transfer speed	10 MBit/s 100 MBit/s	10 MBit/s 100 MBit/s	10 MBit/s 100 MBit/s
Surface mounting	Stranded every two cores	Stranded every two cores	Stranded every two cores
	Without screen	with full screen	with full screen, each core pair is additionally screened
Flexibility	Medium	Medium	Medium
Screening	None	Single	double
Topology	Point-to-point	Point-to-point, line, star	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) This cable has a separate internal screen for every conductor pair as opposed to the STP cable. 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 m. If the network expansion is greater, suitable infrastructure components must be used. For this, transceivers, hubs and switches must be considered.

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.

13.2 Properties of the CAN cable

Use only cable that is approved for CAN application, with the following characteristics:

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

The specifications for cable, plugs and bus termination resistor are defined in ISO 11898. Some requirements and specifications for the CAN network are listed below.

In Table 104 standard parameters for the CAN network with fewer than 64 CAN stations are listed. (the table complies with the specifications of ISO 11898.)

The length of the CAN 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 103:Cable cross-section, bus length and number of bus users 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 slaves connected, the ISO 11898 demands a residual ripple of the supply voltage of ≦ 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 104:Standard parameters for CAN network cable according to the ISO 11898

Bus length [m]	**Loop resistance** [mΩ/m]	**Core cross-section** [mm²]	**Bus termination resistor** [Ω]	**Baud rate at 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

13.3 Transparent mode: Text output via RS232 (example)

The example shows a text output via the RS232 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;
    Closeresult: BOOL;
    Coun: DWORD;
    RESET: BOOL;
END_VAR

(*Cycle time: 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;
            ELS
            E
                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
```

13.4 Access to the CPU drives/memory card

13.4.1 SysLibFile.lib library

The SysLibFile library allows you access to the file system of the XC200, the MMC/SD 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 CODESYS programming system under the "SysFile<Function>" search term.

CAUTION

- The PLC may not be switched off when files from the MMC/SD or the USB stick are opened.
- A power failure when a file is opened can destroy the memory card.
- All the open files must be closed before switch off of the voltage.

13.4.2 Modes for opening a file

13.4.2.1 "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.

13.4.2.2 “w“ mode

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

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

13.4.2.3 “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.

13.4.3 Examples of the “SysFile...” functions

The SysFileOpen function 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’);
```

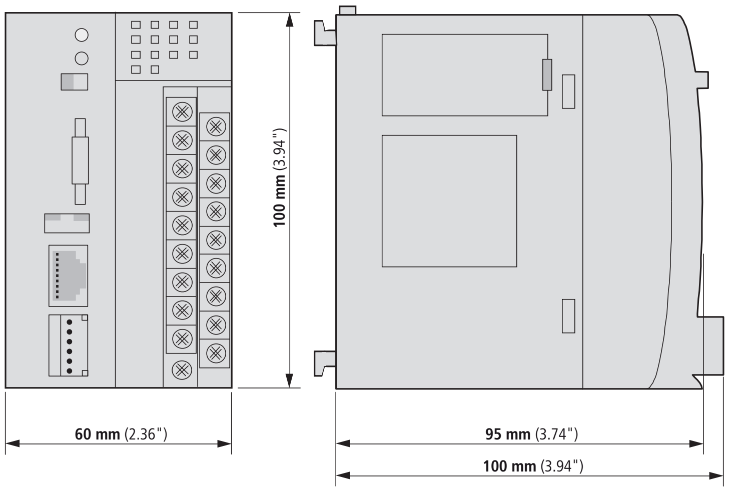
Closing a file

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

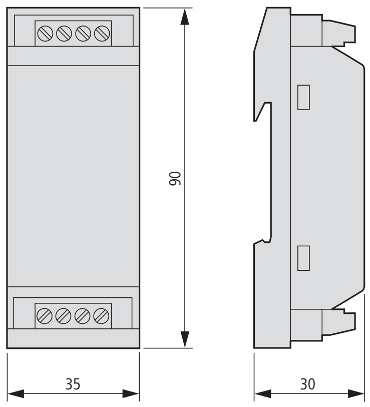
→ On the XC-CPU202 \CONTROL is used instead of \MOELLER.

13.5 Dimensions

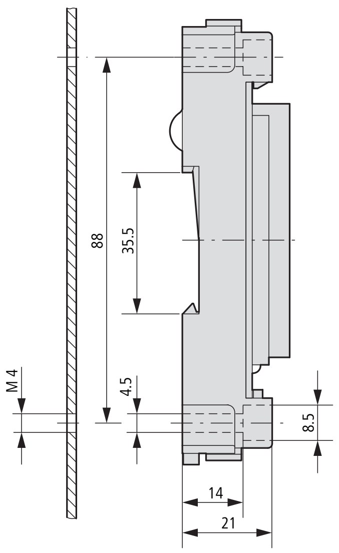
XC-CPU200



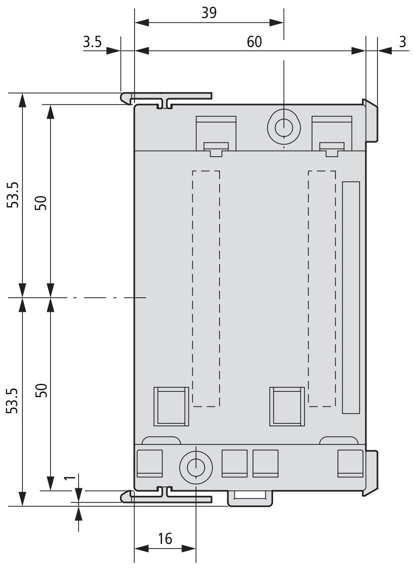
XT-FIL-1 line filter



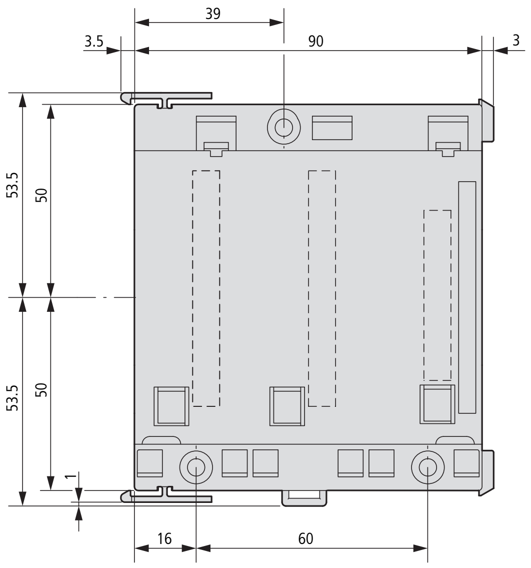
Module rack



XIOC-BP-XC



XIOC-BP-XC1



13.6 Technical Data

			XC-CPU201-EC256-8DI-6DO(-XV), XC-CPU201-EC512-8DI-6DO(-XV) XC-CPU202-EC4M-8DI-6DO-XV
General			
Standards			IEC/EN 61131-2 EN 50178
Ambient temperature		°C	0 - +55
Storage temperature		°C	-25 - +70
Mounting position			Horizontal
Relative humidity, non-condensing (IEC/EN 60068-2-30)		%	10 - 95
Air pressure (in operation)		hPa	795 - 1080
Vibration resistance			5 - 8.4 Hz ±3.5 mm 8.4 - 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 61000-6-4, Class A
Interference immunity			EN 61000-6-2
Battery (lifespan)			Worst case 3 years, typical 5 years
Weight		kg	0.23
Dimensions (W × H × D)		mm	90 × 100 × 100
connection terminals			Plug-in terminal block
Terminal capacity			
Screw terminals			
flexible with ferrule		mm^2	0.5 - 1.5
solid		mm^2	0.5 - 2.5
Spring-cage terminals			
Flexible		mm^2	0.34 - 1.0
solid		mm^2	0.14 - 1.0
Electromagnetic compatibility (EMC)			→ Page 146

			XC-CPU201-EC256-8DI-6DO(-XV), XC-CPU201-EC512-8DI-6DO(-XV) XC-CPU202-EC4M-8DI-6DO-XV
Power supply for the CPU (24 V/0 V)			
Mains failure bridging			
Duration of dip		ms	10
Repetition rate		S	1
Input rated voltage		V DC	24
admissible range		V DC	20.4 - 28.8
Current consumption		A	normally 0.85
Residual ripple		%	≦ 5
Maximum heat dissipation (without local I/O)	P_V	CO	6
TVSS			Yes
Protection against polarity reversal			Yes
External line filter			Part no.: XT-FIL-1
Internal line filter			Yes
Inrush current		$\times I_n$	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 strength			Yes
Potential isolation from supply voltage			No
CPU			
Microprocessor			XC-CPU201…: Risc processor XC-CPU202…: ARM 532MHz
memory			
Program code		kByte	XC-CPU201-EC256…: 512 from operating system version V1.04.01 XC-CPU201-EC512…: 2048 from operating system version V1.04.01 XC-CPU202-EC4M…: 4096
Program data		kByte	XC-CPU201-EC256…: 256 XC-CPU201-EC512…: 512 XC-CPU202-EC4M…: 512
Marker (EC256K/EC512K/EC4M)		kByte	16/16716
Retain data (EC256K/EC512K/EC4M)		kByte	32/32/64
Persistent data (EC256K/EC512K/EC4M)		kByte	32/32/64
Watchdog			Yes
RTC (Real-Time Clock)			Yes
Interfaces			
Multi-Media Card or Secure Digital Card			Yes, optional, order separately

			XC-CPU201-EC256-8DI-6DO(-XV), XC-CPU201-EC512-8DI-6DO(-XV) XC-CPU202-EC4M-8DI-6DO-XV
Ethernet interface			
Baud rate		MBit/s	10/100
Terminal type			RJ45
RS 232 serial interface (without handshake line)			
Baud rate		Bit/s	4800, 9600, 19200, 38400, 57600, 115200
Terminal type			RJ45
potential isolation			No
in the "transparent mode"			
Baud rate		Bit/s	300, 600, 1200, 2400, 4800, 9600, 19200, 38400, 57600, 115200
Character formats			8E1, 8O1, 8N1, 8N2
CAN(open)/easyNet			
Baud rate		Kbits/s	20/50/100/125/250/500/800/1000
Potential isolation			Yes
Device profile			According to DS 301 V4
29 Bit Identifier			XC-CPU201: No XC-CPU202: Yes
PDO type			asyn., cyc., acyc.
Connection			Plug-in spring-loaded terminal block, 6-pole
Bus terminating resistors			With XC-CPU201…: external; with XC-CPU202…: switchable
Module		Count	max. 126
USB interface, V1.1 with XC-CPU201…, V2.0 with XC-CPU202…			
Baud rate (Autochanging)		MBit/s	1.5/12
Potential isolation			No
Power supply for connected devices:			
Rated operating voltage		V DC	5
max. current		A	0.5
Terminal type			Downstream plug
Power supply of local inputs/outputs (24 V_Q/0 V_Q)			
Rated operating voltage		V DC	24
Voltage range		V DC	20.4 - 28.8
Current consumption		A	max. 3 (load dependent)
Potential isolation			
Power supply against CPU voltage			Yes
TVSS			Yes
Protection against polarity reversal			Yes

			XC-CPU201-EC256-8DI-6DO(-XV), XC-CPU201-EC512-8DI-6DO(-XV) XC-CPU202-EC4M-8DI-6DO-XV
Digital inputs			
Input rated voltage		V DC	24, observe polarity
Voltage range		V DC	19.2 - 30
Input current per channel at rated operating voltage			
Functionality: Normal digital input		mA	normally 3.5
Functionality: Fast digital input		mA	normally 7
Heat 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		Count	8
Channels with the same reference potential		Count	8
Of which can be used as			
Interrupt inputs		Count	2
Counter input 32 Bit or		Count	1
Counter input 16 Bit or		Count	2
Incremental encoder input (Track A, B, C)		Count	1
Max. input frequency		kHz	50
Status display			LED
Digital outputs			
Heat dissipation per channel			
QX0.0 and QX0.5		CO	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		Count	6
Channels with the same reference potential		Count	6

			XC-CPU201-EC256-8DI-6DO(-XV), XC-CPU201-EC512-8DI-6DO(-XV) XC-CPU202-EC4M-8DI-6DO-XV
Status display			LED
duty factor		% DF	100
Utilization factor		g	1

Electromagnetic compatibility			
Interference 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 connection)		1 kV
Surge (IEC/EN 61000-4-5)	Digital I/O, asymmetric		0.5 kV
	Analog I/O, asymmetric, connection on the screen		1 kV
	Mains DC, asymmetric		1 kV
	Mains DC, symmetrical		0.5 kV
Cable conducted interference, induced by high frequency fields (previously: radiated RFI) (IEC/EN 61000-4-6)			3 V

13.7 Technical data – Filter

		24 V DC filter XT-FIL-1
General		
Standards		IEC/EN 61131-2 EN 50178
Ambient temperature	°C	0 - +55
Storage	°C	-25 - +70
Mounting position		horizontal/vertical
Relative humidity, non-condensing (IEC/EN 60068-2-30)	%	10 - 95
Air pressure (in operation)	hPa	795 - 1080
Vibration resistance		5 - 8.4 Hz ±3.5 mm 8.4 - 150 Hz ±1.0 g
Mechanical shock resistance		15 g/11 ms
Impact strength		500 g/∅ 50 mm ±25 g
Overvoltage category		II
Pollution degree		2
Degree of protection		IP20
Rated surge voltage	V	850
Emitted interference		EN 61000-6-4, Class A
Interference immunity		EN 61000-6-2
Weight	g	95
Dimensions (W × H × D)	mm	35 × 90 × 30
connection terminals		Screwed terminal
Terminal capacity		
Screw terminals		
flexible with ferrule	mm^2	0.2 - 2.5 (AWG22-12)
solid	mm^2	0.2 - 2.5 (AWG22-12)
power supply		
Input voltage	V DC	24
admissible range	V DC	20.4 - 28.8
Residual ripple	%	≦ 5
TVSS		Yes
Potential isolation		
Input voltage against PE		Yes
Input voltage against output voltage		No
Output voltage to PE		Yes
Output voltage	V DC	24
Output current	A	2.2

Index